Predicting Retail Sales: Final Deliverable

Zhiqi Ma
Ruoheng Du
Xiaojiang Wu
Yijian Liu

Agenda

Introduction

Data

Architecture

Product Categorization

Models

Previous Result Comparison

Results I – XGBoost

Results II – Chronos

Results III – Prophet

Results IV – LightGBM

Summary

Further Improvement




Introduction

- This project focuses on forecasting retail sales using time series modeling techniques. Given the complex and seasonal nature of retail sales, we analyze historical daily/weekly sales data to identify trends and seasonal patterns.

Problem

Sales forecasting in online retail faces significant challenges due to complex purchasing behaviors, fluctuating market dynamics, and a large variety of products with different sales patterns.

Objective

Evaluate the predictive performance of statistical, machine learning, and foundation models on retail sales forecasting, both directly and after product clustering using unsupervised techniques such as KMeans.

From:

Raw transactional data of over 1 million records from the Online Retail II dataset, covering purchases from a UK-based non-store retailer

To:

Predictive models and assessment of forecasting accuracy across original and KMeans-clustered product categories.

Value Creation

Enhanced Forecast Accuracy
Supports a deeper understanding of model capabilities in complex retail environments, improving forecasting reliability.

Cluster-Aided Forecasting
Leverages clustering to enhance the interpretability of sales patterns and improve the performance evaluation of predictive models across differentiated product groups.

Data-Driven Decision Making
Supports informed strategies for inventory planning, marketing strategy, and policy making.

Timeline

Gather raw retail sales data across 2 years. Clean, aggregate, and transform data for analysis.

Train different time series models and test different hyperparameters.

Summarize key findings and model performance. Provide recommendations for improving forecasting accuracy.

Identify trends, seasonality, and anomalies. Visualize key patterns using ACF, PACF, PSD, and decomposition.

Split data into training, validation, and testing sets. Compute MAPE and analyze error distributions.




Data

Data

Source: UCI Online Retail II

Type: Time-Series

Instances: 1,067,371 time-stamped transactions records at minute level from Dec 2009 to Dec 2011

Features: Invoice number, stock code, product description, quantity, invoice date, unit price, customer ID, country

Missing Values: some missing customer ID and product description; some dates with 0 sales

Feature Type: Mixed (categorical and numerical attributes)

	A	B	C	D	E	F	G	H
1	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
2	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	12/1/09 7:45	6.95	13085	United Kingdom
3	489434	79323P	PINK CHERRY LIGHTS	12	12/1/09 7:45	6.75	13085	United Kingdom
4	489434	79323W	WHITE CHERRY LIGHTS	12	12/1/09 7:45	6.75	13085	United Kingdom
5	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	12/1/09 7:45	2.1	13085	United Kingdom
6	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	12/1/09 7:45	1.25	13085	United Kingdom
7	489434	22064	PINK DOUGHNUT TRINKET POT	24	12/1/09 7:45	1.65	13085	United Kingdom

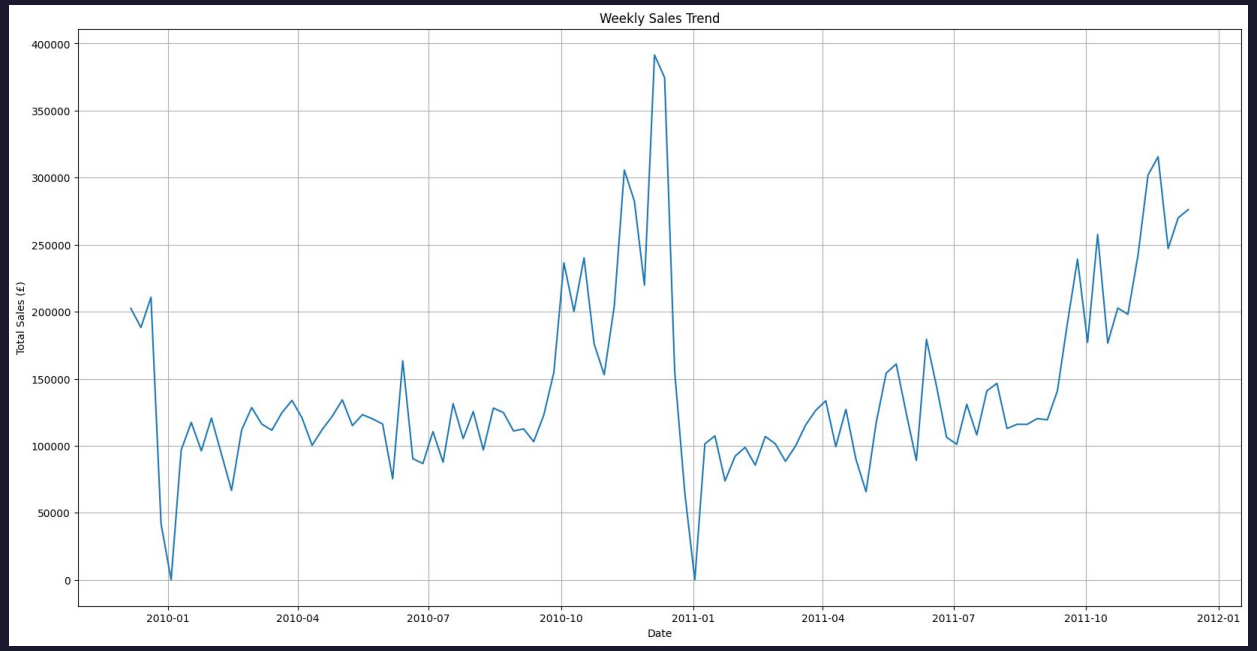
Data Cleaning & Preprocessing

- **Missing Value Handling:** Removed rows with missing product descriptions and missing customer IDs before modeling.
- **Negative and Cancelled Transactions:** Identified and removed negative quantity and unit price transactions; matched and netted out valid cancellations records.
- **Duplicate Removal:** Dropped fully duplicated records across all fields to eliminate recounting sales.
- **Product Record Filtering:** Excluded administrative entries, gift vouchers, and non-salable items through manual inspection of Stock Code and Product Description.

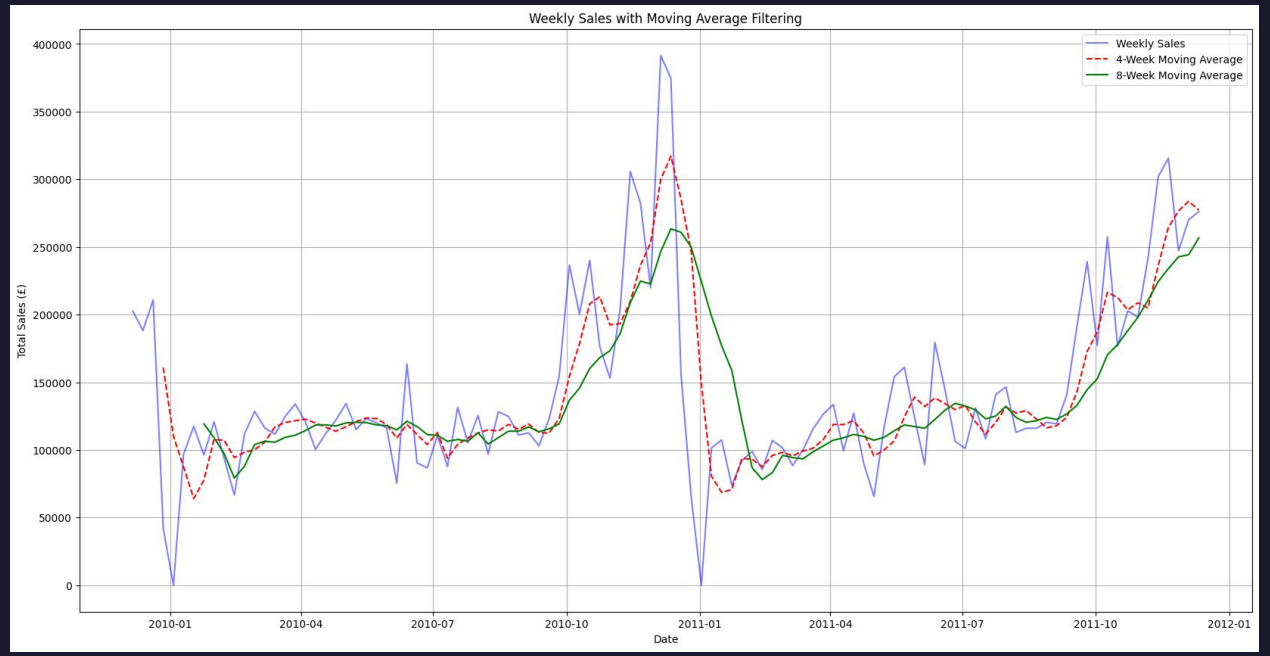
  Text Normalization: Standardized product descriptions to lowercase, removed special characters, and cleaned corrupted text entries for clustering purpose.
- **Feature Engineering:** Created a new "Sales" feature by UnitPrice x Quantity for use in forecasting models.

Exploratory Data Analysis

Weekly Long-Term Sales Patterns



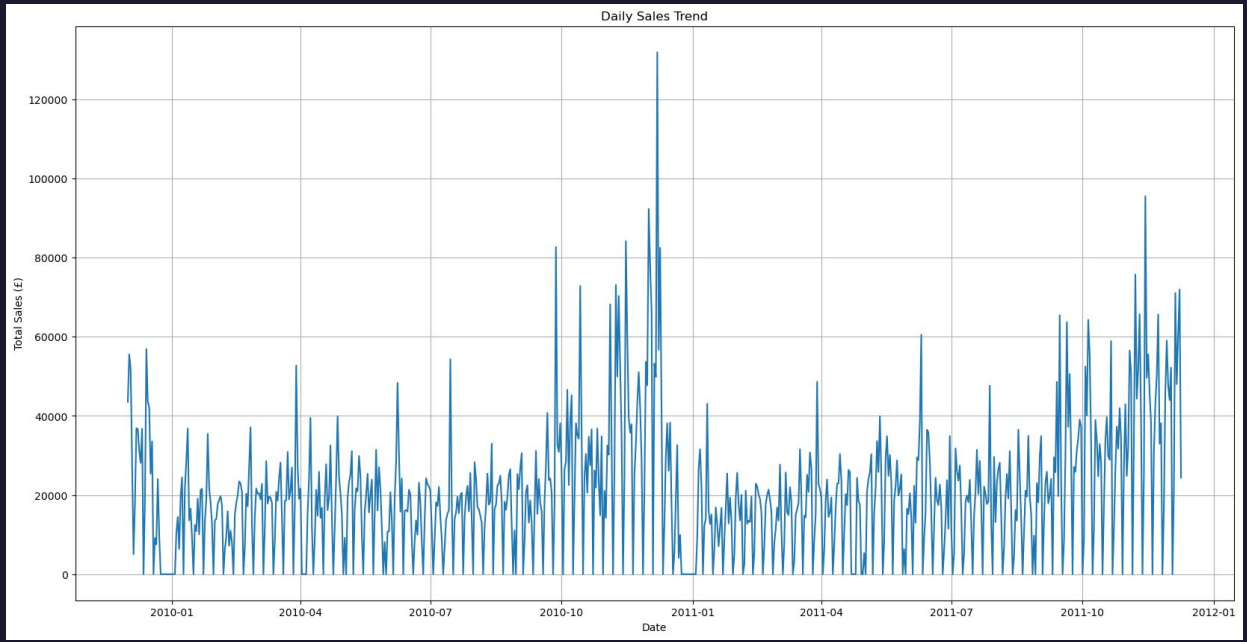
- Weekly sales exhibit strong seasonality with clear spikes during the year-end holiday periods and occasional sharp drops, indicating irregular large-scale sales events or data volatilities.



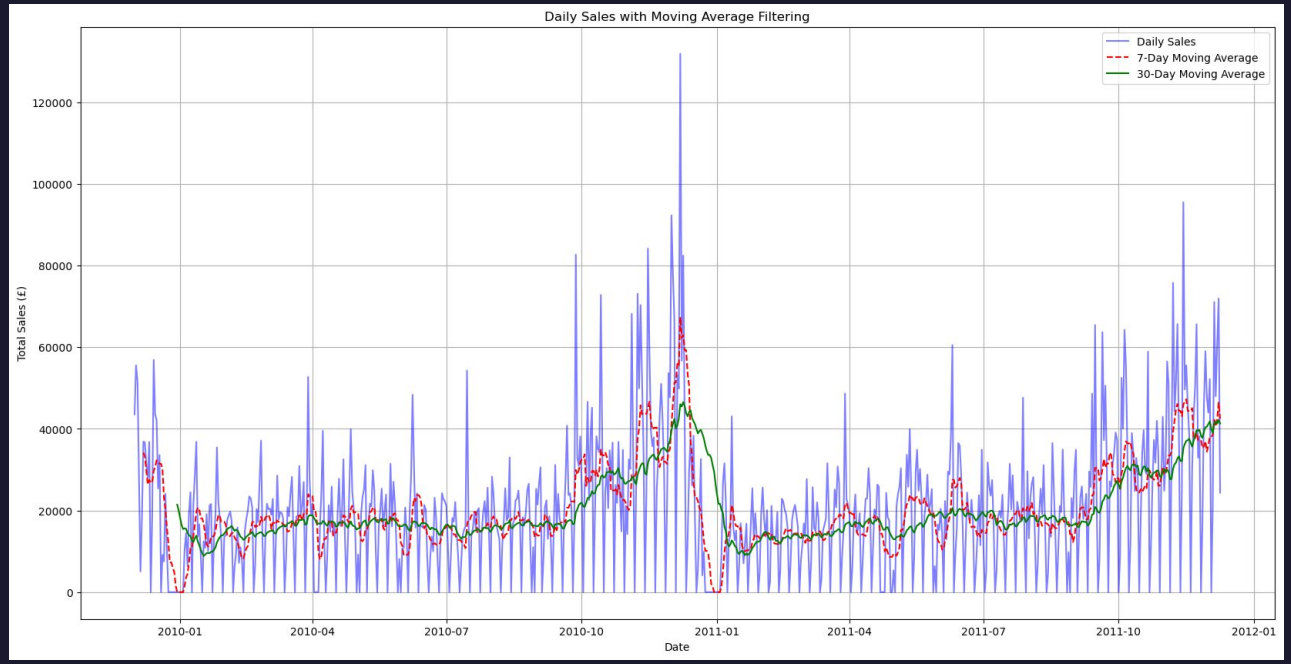
- Applying 4-week and 8-week moving averages smooths out short-term volatility, revealing an overall upward sales trend with recurring seasonal peaks around the end of each year.

Exploratory Data Analysis

Daily Long-Term Sales Patterns



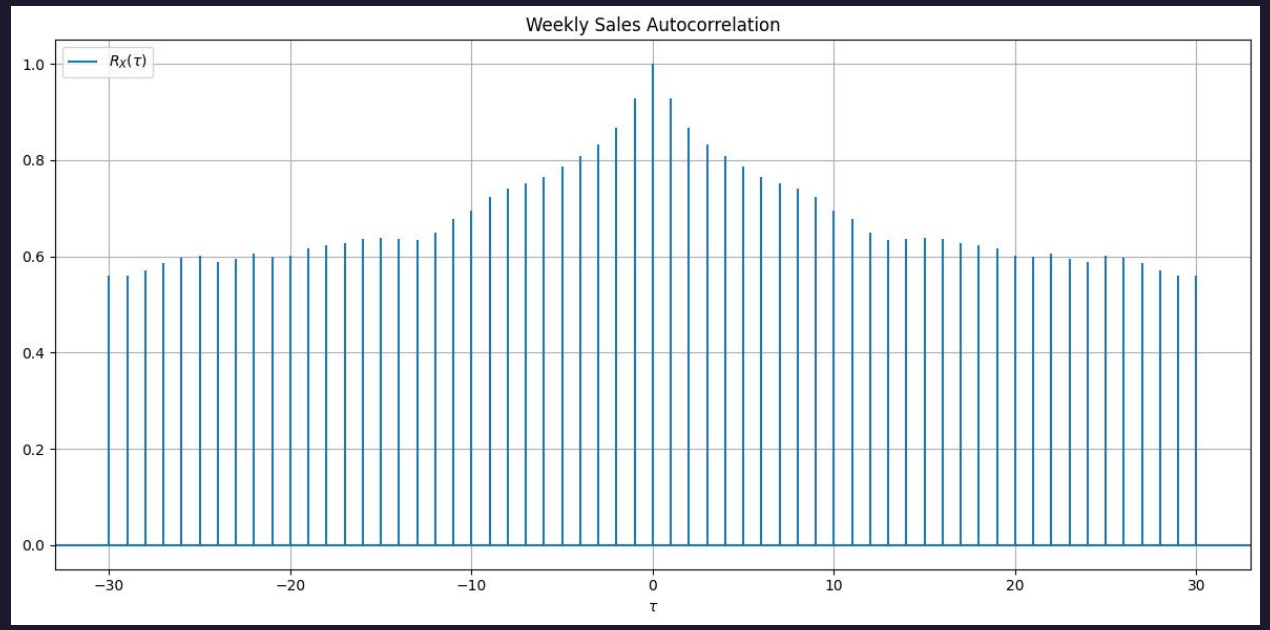
- Daily sales exhibit high volatility with frequent sharp spikes and dips, suggesting irregular large transactions or special events impacting sales patterns.



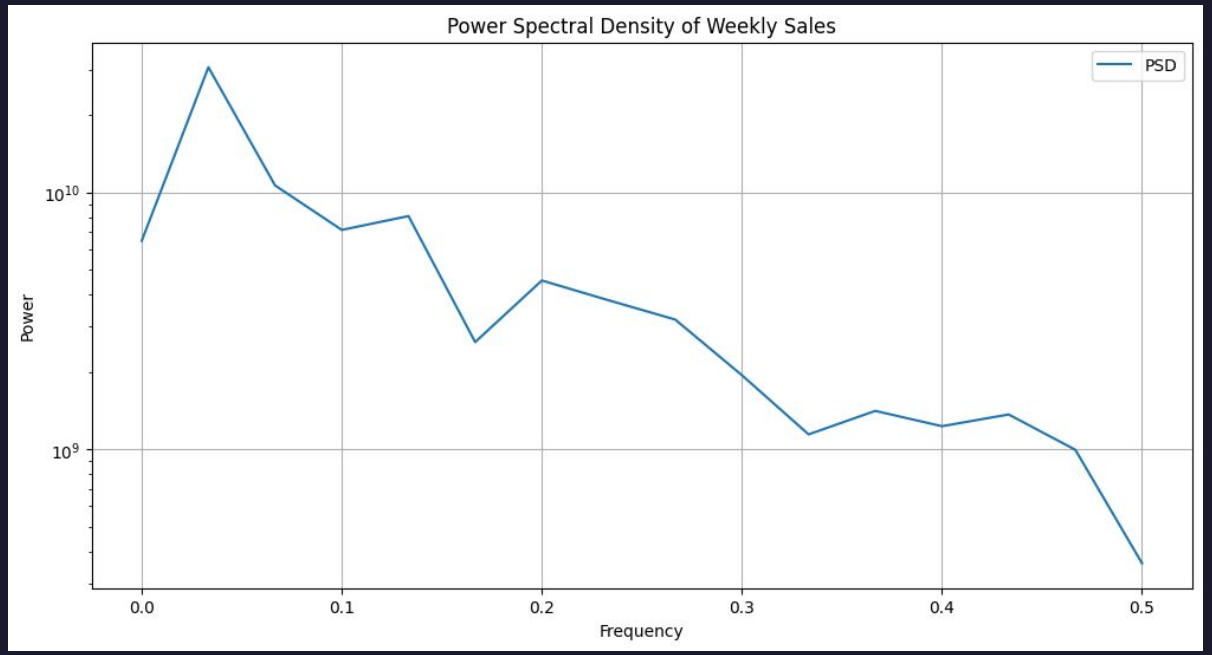
- Smoothing with 7-day and 30-day moving averages reveals an gradual upward trend throughout the year and sudden drop in the next year's beginning, punctuated by seasonal peaks corresponding to holiday sales periods.

Exploratory Data Analysis

Periodic Patterns in Weekly Retail Sales - Autocorrelation & PSD



- Autocorrelation plot of weekly sales shows a gradual decline from the peak at lag 0, indicating strong persistence and long-term dependencies in sales behavior over time.

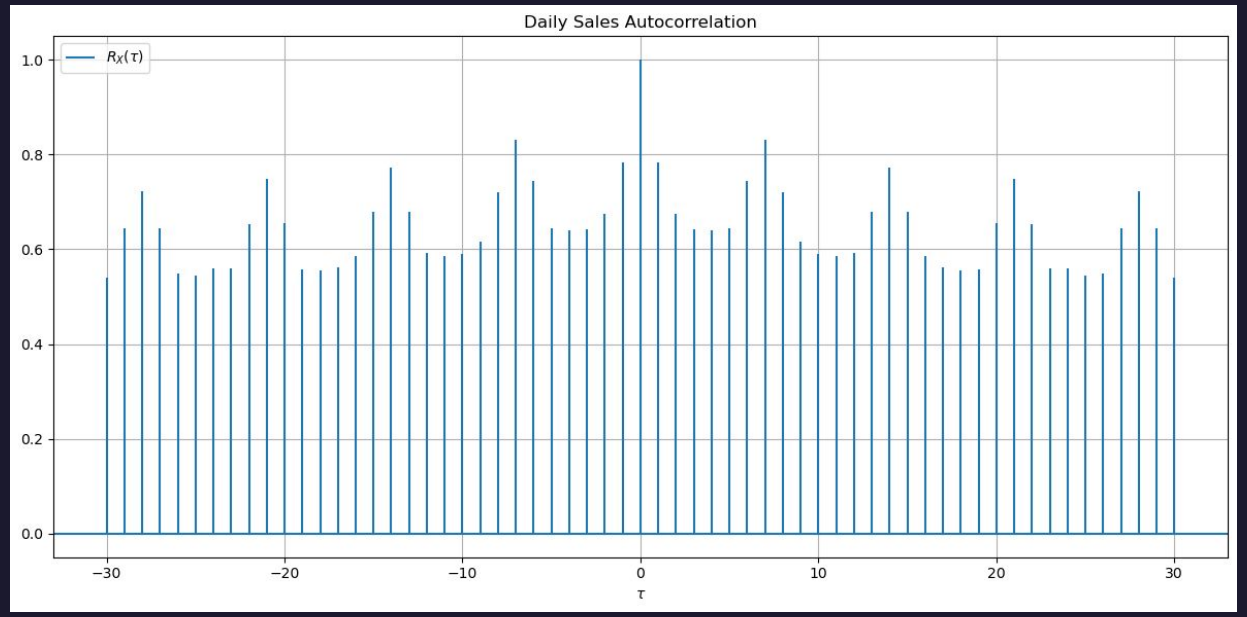


- PSD plot highlights the dominance of low-frequency components, suggesting most variations in weekly sales are driven by long-term rather than short-term fluctuations.

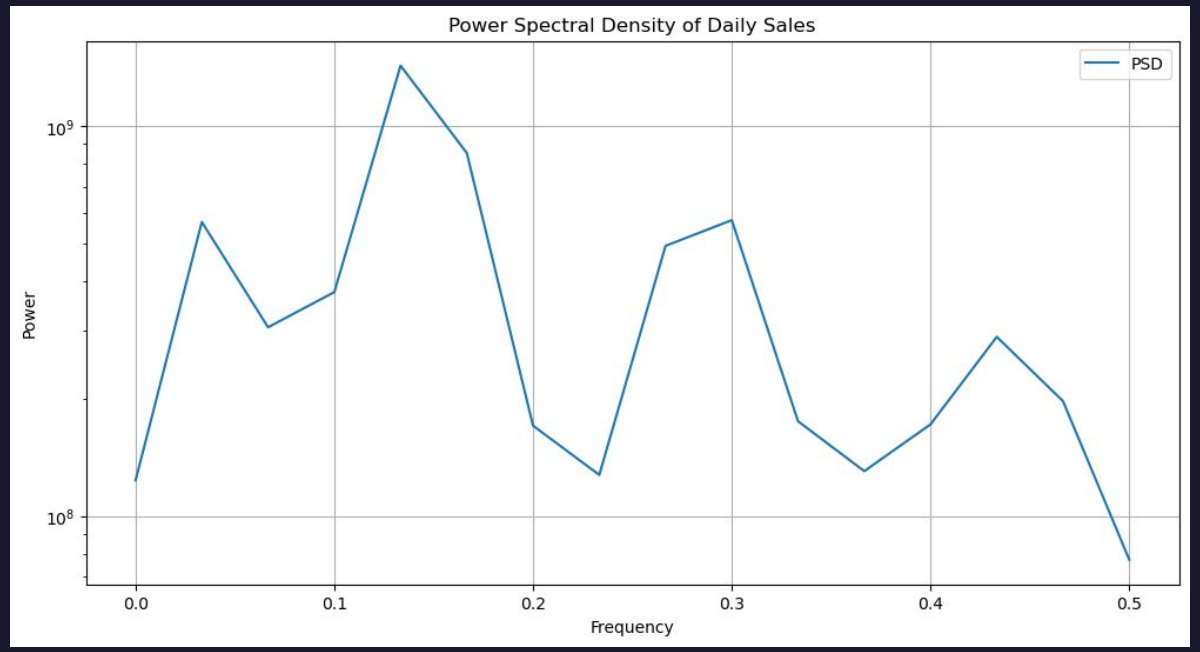
*: Notice that no statistical models that require data to be stationary like GLM are used in modeling, so autocorrelation and PSD plots are only shown for illustration purposes. We'll not handle stationarity here.

Exploratory Data Analysis

Periodic Patterns in Daily Retail Sales - Autocorrelation & PSD



- Daily autocorrelation plot shows clear periodic patterns, indicating strong short-term seasonality in daily sales, possibly driven by weekly or promotional cycles.

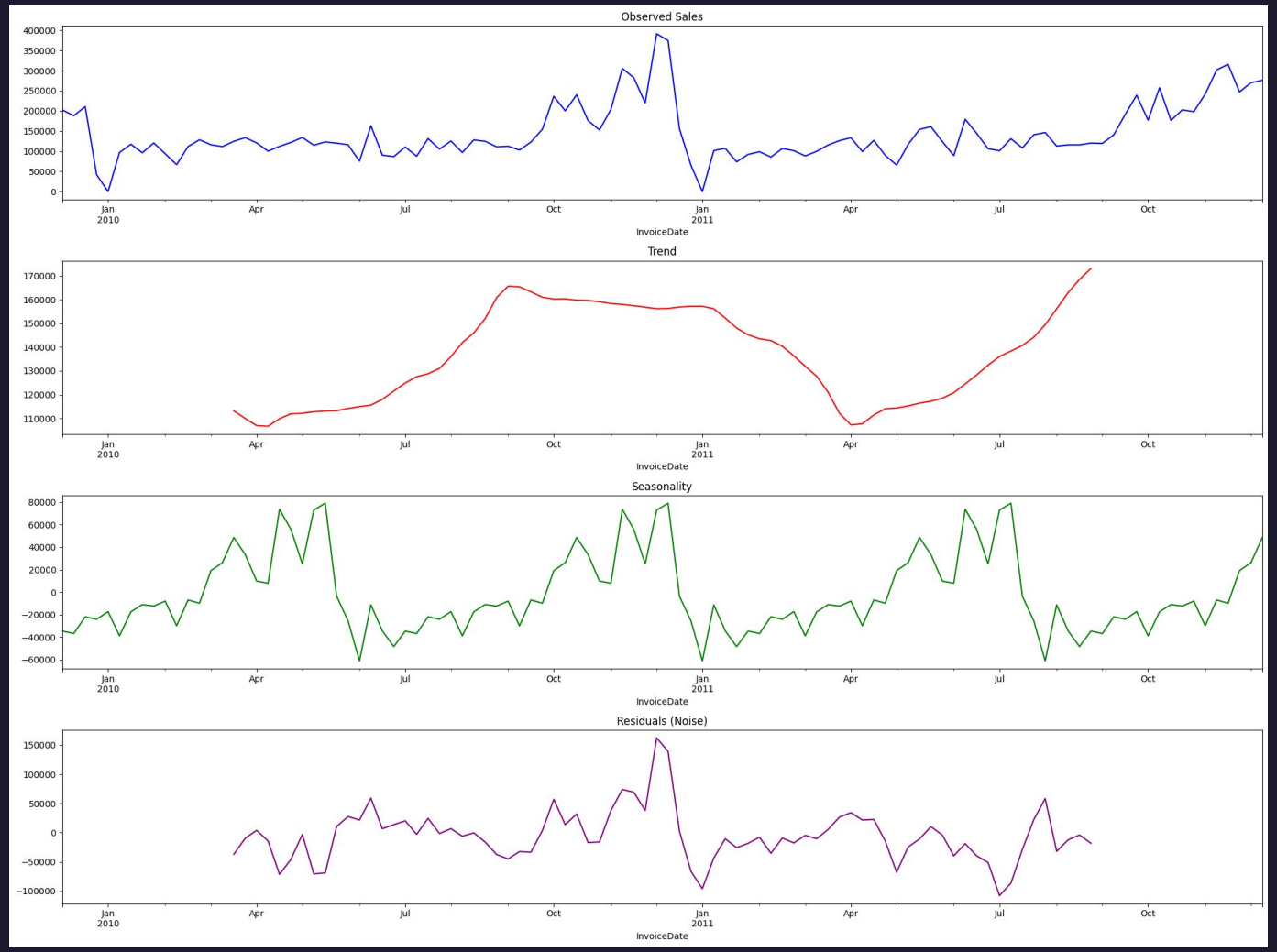


- Daily PSD plot reveals dominant low-frequency components, suggesting that long-term trends and seasonal effects play a significant role in the daily sales variability.

*: Notice that no statistical models that require data to be stationary like GLM are used in modeling, so autocorrelation and PSD plots are only shown for illustration purposes. We'll not handle stationarity here.

Exploratory Data Analysis

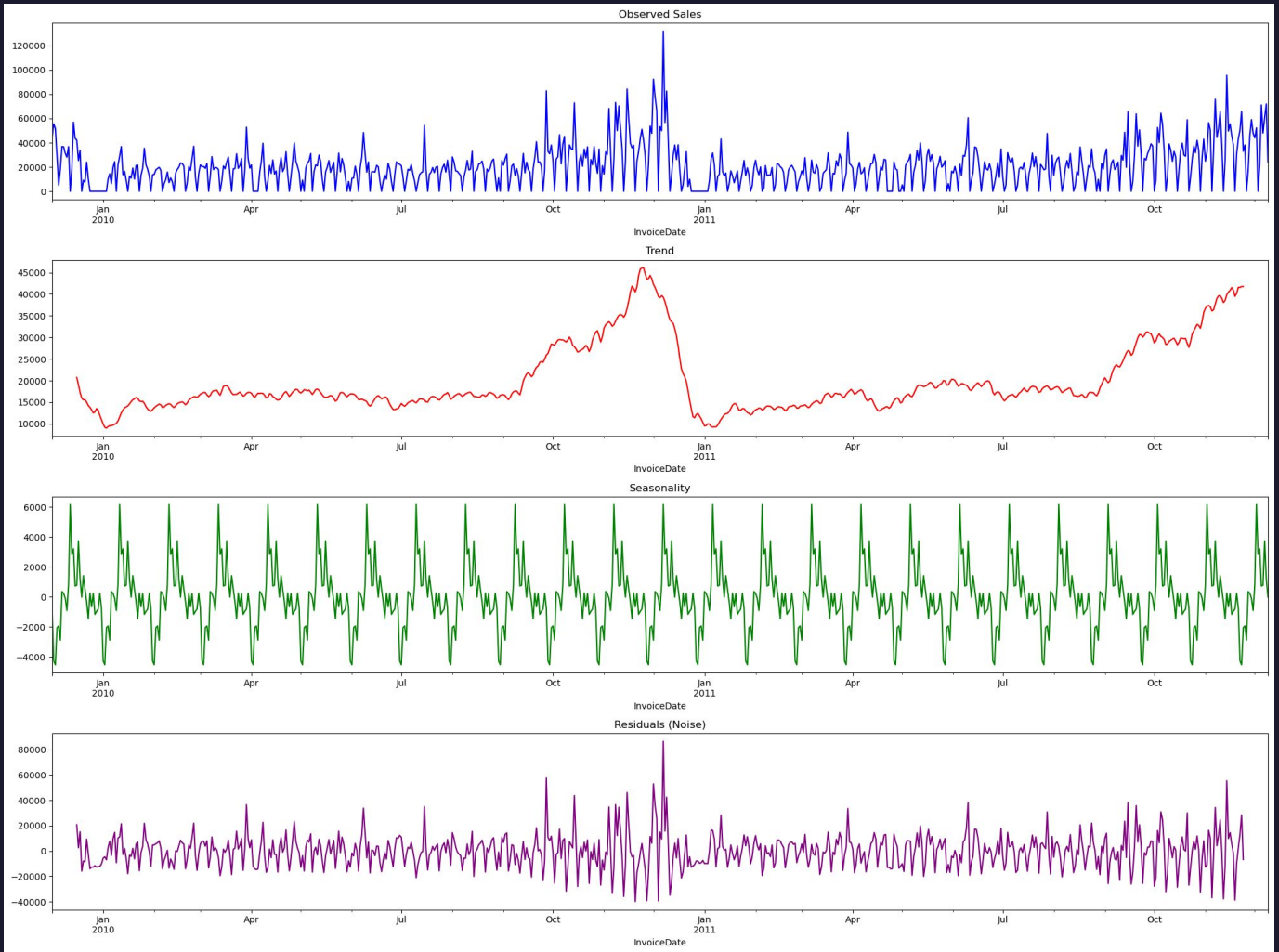
Weekly Time Series Decomposition



- Decomposition of weekly sales reveals a clear upward long-term trend throughout the year, especially noticeable towards the end of 2011, indicating strong business growth over time.
- Seasonal component shows recurring weekly fluctuations, with stronger positive deviations around holiday seasons, suggesting the impact of periodic consumer behaviors.
- Residuals remain relatively centered around zero, although with some periods of higher volatility, indicating the presence of irregular events or short-term anomalies not captured by the trend or seasonality.

Exploratory Data Analysis

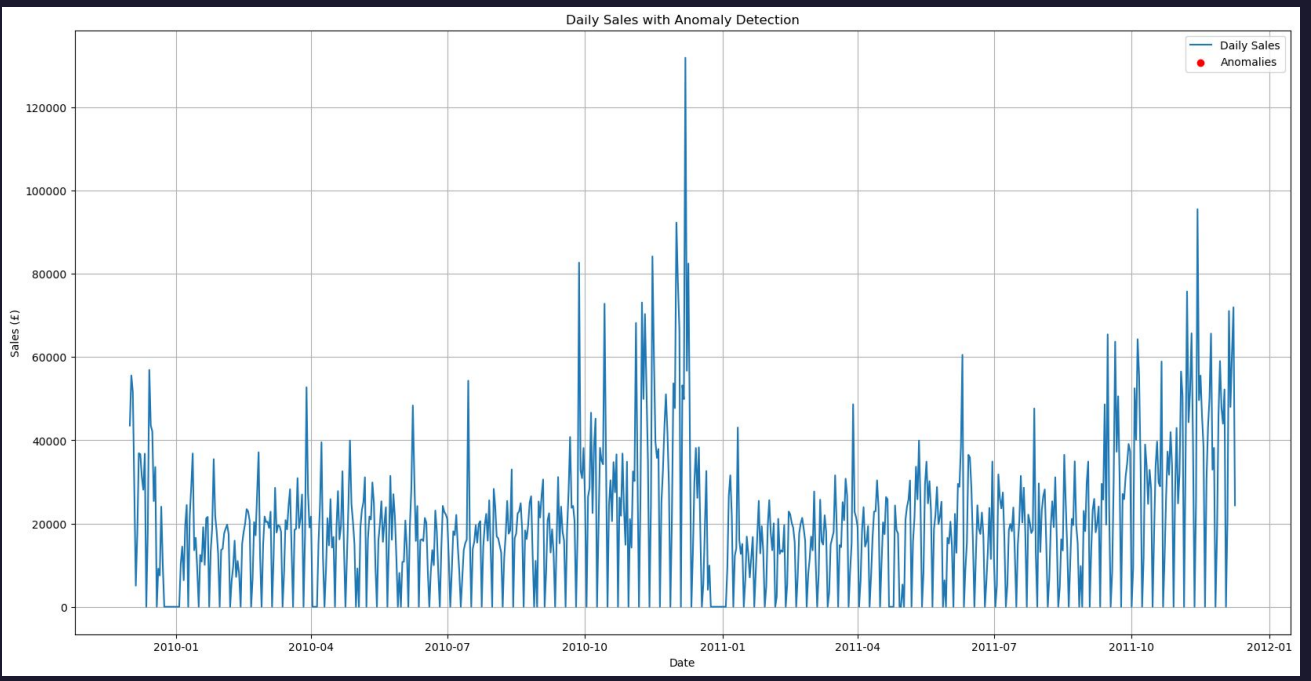
Daily Time Series Decomposition

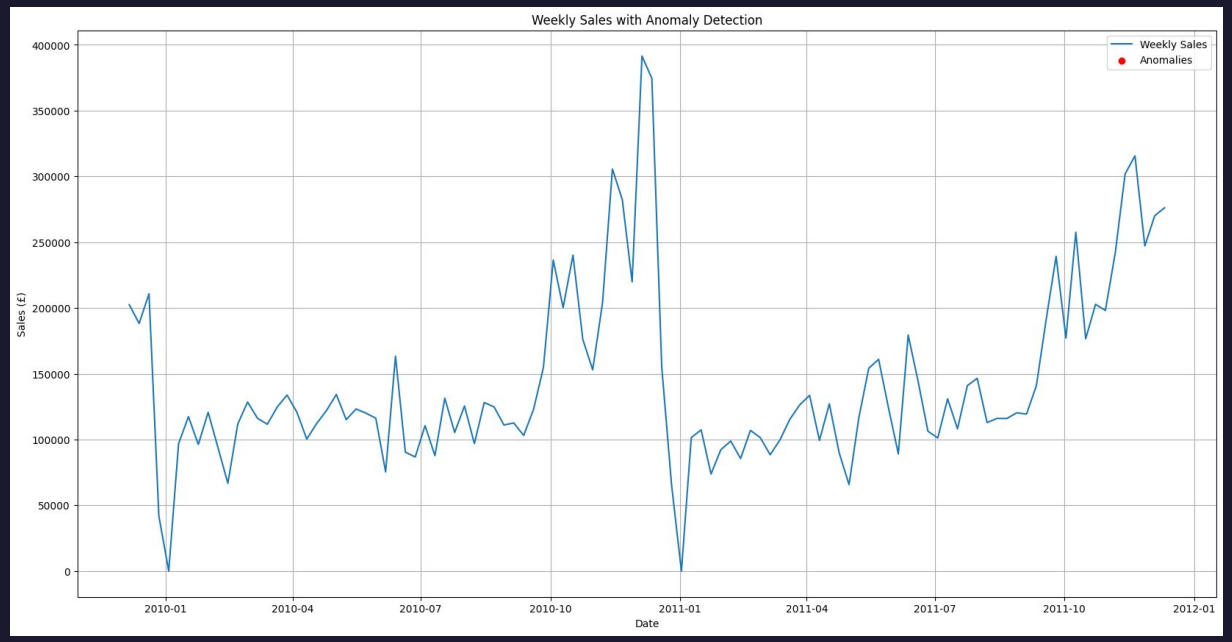


- Decomposition of daily sales reveals a gradually increasing long-term trend, particularly accelerating after mid-2011, reflecting steady business growth.
- Seasonal component shows strong and regular weekly patterns, with sharp periodic peaks that align with expected consumer buying cycles across weeks.
- Residuals exhibit higher volatility during certain periods, notably around the end of 2010, suggesting the presence of irregular spikes possibly due to large promotional events, holidays, or data anomalies.

Exploratory Data Analysis

Daily & Weekly Sales Data Anomaly Detection



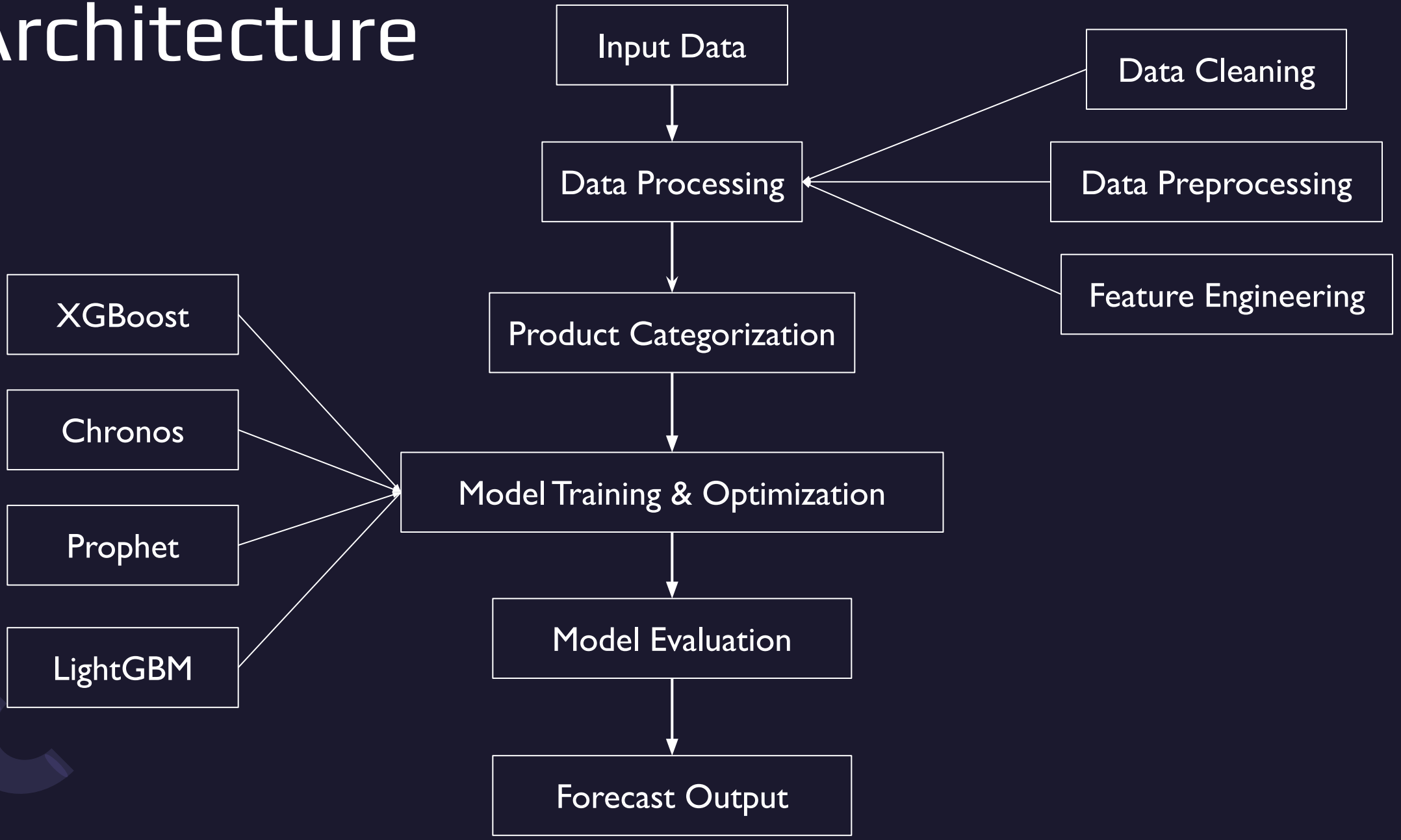


- Daily anomaly detection using 7-day rolling mean +/- 3 std method.
- Weekly anomaly detection using 4-week rolling mean +/- 3 std method.
- No significant anomalies were detected, suggesting relatively stable electricity usage patterns.




Architecture

Architecture






Product Categorization

Categorization Methods Comparison

Objective: Categorize products based on their sales patterns to enable more targeted modeling.

Methods Compared:

- **KMeans Clustering** (k=2 to k=10)
- **Agglomerative Hierarchical Clustering**
- **DBSCAN**

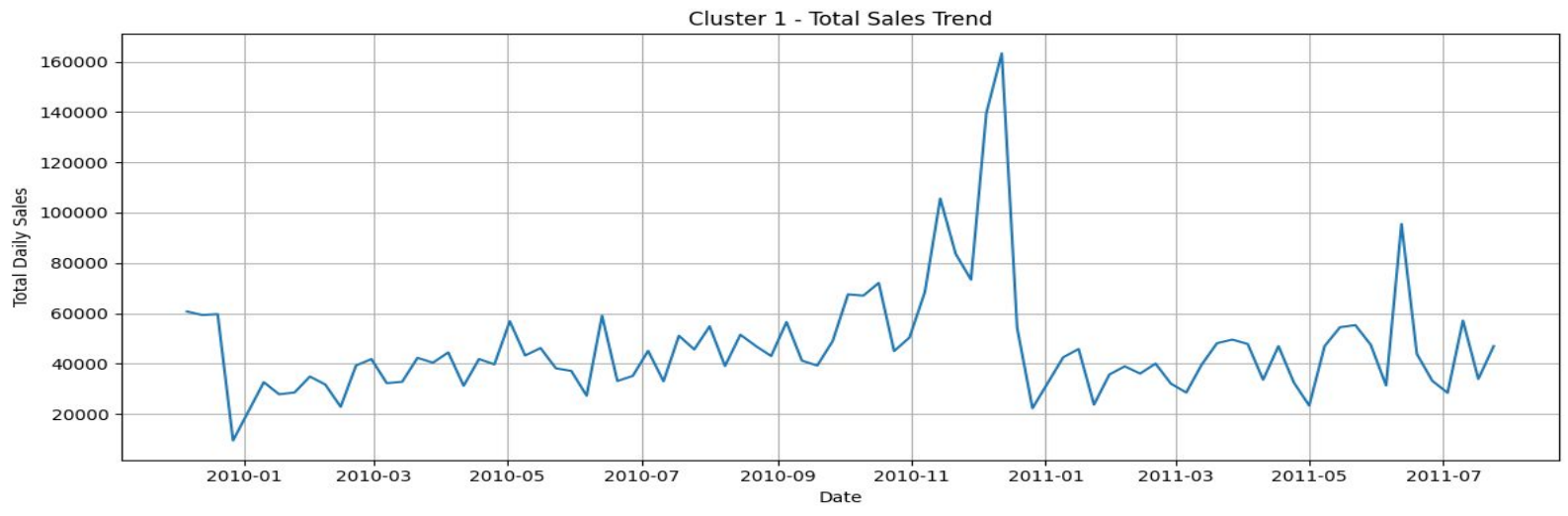
Evaluation Metric: Silhouette Score

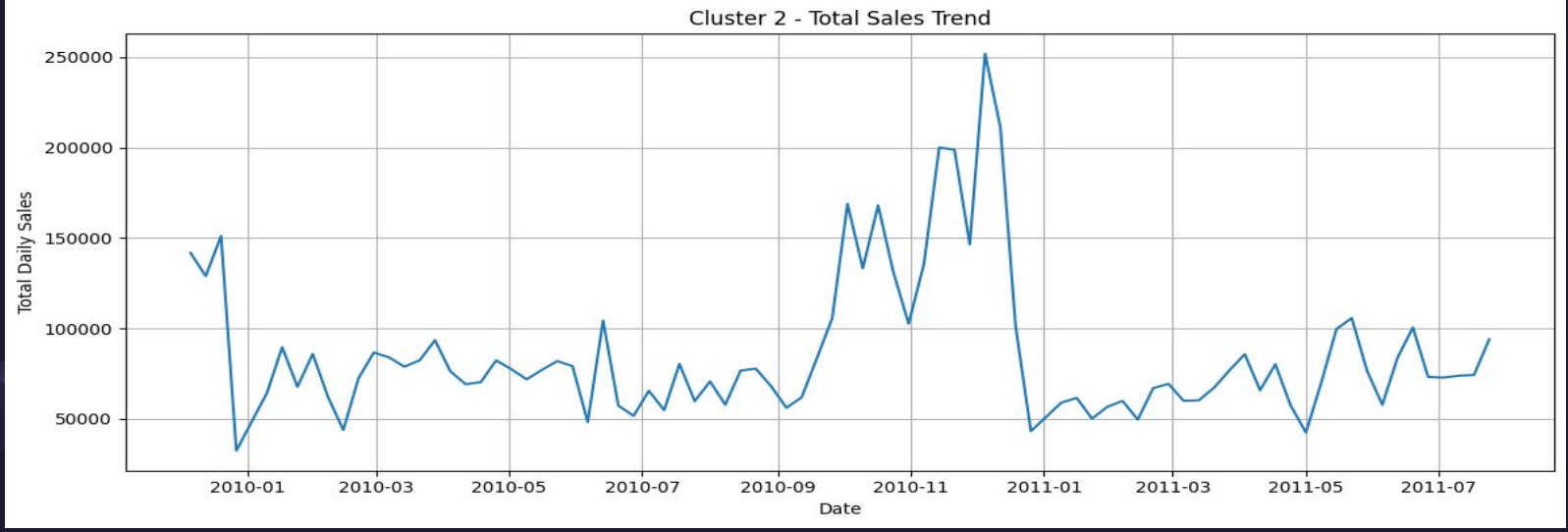
Summary of Results:

- **Best Silhouette Score:**
  - KMeans (k=2): **0.834**
  - Agglomerative (k=2): **0.833**
- **KMeans (k=2)** was selected.
- **Reason:** Slightly higher silhouette score, simpler structure, faster clustering, and better stability for downstream forecasting.

	method	k	silhouette
0	kmeans	2	0.834409
9	agglomerative	2	0.833363
11	agglomerative	4	0.816141
10	agglomerative	3	0.816091
1	kmeans	3	0.792991
18	dbscan	2	0.781888
2	kmeans	4	0.692337
3	kmeans	5	0.688980
7	kmeans	9	0.643391
8	kmeans	10	0.641955
4	kmeans	6	0.615573
5	kmeans	7	0.615563
6	kmeans	8	0.599671
14	agglomerative	7	0.530514
13	agglomerative	6	0.530501
12	agglomerative	5	0.529487
17	agglomerative	10	0.524521
16	agglomerative	9	0.523955
15	agglomerative	8	0.523885

Kmeans (k=2) Categorization Result





Setup:

- KMeans clustering with k=2
- Based on weekly sales patterns

Counts:

- Cluster 1: 130
- Cluster 2: 4673




Models

Models

XGBoost

XGBoost (Extreme Gradient Boosting) is a scalable and efficient implementation of gradient boosted decision trees. In this project, we utilized XGBoost for weekly total sales forecasting at the cluster level. It is particularly well-suited for time series forecasting when the temporal patterns can be captured through engineered features, such as lagged values and date-related attributes.

- **Feature Engineering**
  - Created lag features (e.g., past 7 weeks' sales) to capture autoregressive dependencies.
  - Extracted date features like day of week, month, and day of month to model seasonality and calendar effects.
- **Model Training**
  - Applied a rolling train-validation split (60%-20%-20%) to fine-tune hyperparameters such as n_estimators (number of trees), learning_rate, and max_depth.
  - Selected the best configuration based on validation MAPE (Mean Absolute Percentage Error).

Models

Amazon Chronos

Chronos is a family of pretrained time series forecasting (foundation) models based on T5 architecture, trained on 84B data.

To use Chronos model, **AutoGluon-TimeSeries (AG-TS)** is the best choice, which provides a robust and easy way to use Chronos through the familiar TimeSeriesPredictor API. Roughly speaking, we can:

- Use Chronos in **zero-shot** mode to make forecasts without any dataset-specific training
  - only prediction length, train data, and model version needed as parameters
- **Fine-tune** Chronos models on custom data to improve the accuracy
  - can use default fine tuning setting, or manually set hyperparameters, including learning rate, batch size, fine tune steps, etc.; also allowed to control fine tune time limit
  - can choose to fine-tune using CPU or GPU
- **Handle covariates & static features** by combining Chronos with a tabular regression model
  - can add covariate regressors to be combined with univariate Chronos model to predict target

Models

Prophet

Prophet is a time series data forecasting process. Its foundation is an additive model that takes into account seasonality on a daily, weekly, or annual basis, trends, and holiday effects. It is robust to missing data and trend shifts, but it can perform best with time series that have strong seasonalities and trends. In this project, we focused on using prophet to predict the weekly total sales and also tried K Means to do clustering based on the time series data of products before training.

When setting the prophet model, we mainly model following components

- Seasonality
  - tune the binary parameters like 'add_weekly' and 'add_monthly' to automatically determine the necessity of each seasonality based on validation performance. Tune 'fourier_order' for each components and tune 'seasonality_prior_scale' and 'seasonality_mode' for the whole seasonality part.
- Trend
  - Tune 'changepoint_prior_scale' to balance model flexibility
- Holiday
  - Use binary parameter 'add_holiday' to automatically determine the necessity of the built-in UK holiday. If added, use the 'holiday_prior_scale' to tune its flexibility.

Models

LightGBM

Light Gradient Boosting Machine (LightGBM), developed by Microsoft, is a scalable, and memory-efficient gradient boosting framework. It builds an ensemble of decision trees in stages, with each new tree correcting previous errors. Unlike other methods like XGBoost, LightGBM grows trees leaf-wise by selecting the leaf with the highest split gain, leading to faster convergence.

Just like XGBoost, LightGBM also requires feature engineering, such as lag features to handle temporal dependencies (1 day ago, 7 days ago) and calendar-based features (month, day of week, is_weekend) to capture the cyclical patterns in the data.

LightGBM shares many parameters with traditional tree-based and boosting models but also introduces unique ones to boost efficiency and control overfitting. Key parameters include num_leaves (maximum leaves per tree), learning_rate (shrinks feature weights to slow learning), feature_fraction (randomly selects a subset of features per iteration), and min_data_in_leaf and max_depth help prevent overfitting by limiting leaf size and tree depth.

Evaluation Metric

Mean Absolute Percentage Error (MAPE)

MAPE measures the average percentage error between actual and predicted values, making it useful for evaluating the accuracy of a forecasting model.

Formula

$$MAPE = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{Y_t - \hat{Y}_t}{Y_t} \right| \times 100\%$$

where:

- Y_t = actual value at time t
- $\hat{Y}_t$ = predicted value at time t
- n = total number of observations




Previous Result Comparison

Previous Result Comparison

Chronos forecasting with previous team's data & categorization setting

Category	Home & Storage	Fashion & Accessories	Kitchen & Dining	Lights & Decorations	Toys & Gifts	Weighted MAPE	Improvement
Percentage	31.8000%	12.4000%	25.9000%	13.5000%	16.5000%		
PREVIOUS	**31.1200%**	**36.6800%**	**33.0500%**	**36.0600%**	**37.6300%**	**34.0815%**	
zero-shot bolt_tiny	32.9582%	32.5504%	32.1771%	**35.5352%**	39.0039%		
zero-shot bolt_mini	32.2490%	30.6645%	32.2588%	39.6002%	37.5858%		
zero-shot bolt_small	32.0139%	31.1872%	30.0368%	37.7398%	**36.2885%**		
zero-shot bolt_base	**31.9659%**	**30.5643%**	**29.9891%**	39.2078%	37.2806%		
fine-tuned best *	**29.5886%**	**30.5643%**	**28.6320%**	**32.8461%**	**34.3786%**	**30.7215%**	**3.3600%**

Observation: Slight increase in MAPE after both zero-shot and fine-tuned Chronos forecasting compared with the previous team's model results.

Reason: The previous team didn't do necessary data cleaning → the data contains lots of useless information and noise → seriously impacts the forecasting accuracy

Conclusion: **Revisit data preprocessing** and **redesign product categorization**

*: Here fine-tuned best means fine-tuning using the best zero-shot model preset, for example, since zero-shot bolt_base achieves the best MAPE for Home & Storage category's data, we still fine-tune using bolt_base for this category's data.




Results I – XGBoost

Results I

XGBoost Model Setup

Overview

- Model: XGBoost Regressor
- Target: Weekly total sales per product cluster
- Categorization: KMeans (k=2) clustering on product sales patterns

Feature Engineering

- Lag Features: Sales from previous 1 to 7 weeks - Capture short-term sales dynamics
- Date-based Features: Day of week, Month, Day - Capture seasonality and calendar effects

Hyperparameter Tuning (based on lowest validation MAPE)

- n_estimators: [100, 200, 300]
- learning_rate: [0.1, 0.05, 0.01]
- max_depth: [3, 5, 10]

Results I

XGBoost Best Model Performance

Train:Validation:Test: 6:2:2 chronologically

Best Model Configuration (Fine-Tuned):

- Kmeans categorization: k=2
- Parameters:

```
{'n_estimators': 200, 'learning_rate': 0.01, 'max_depth': 3}
```

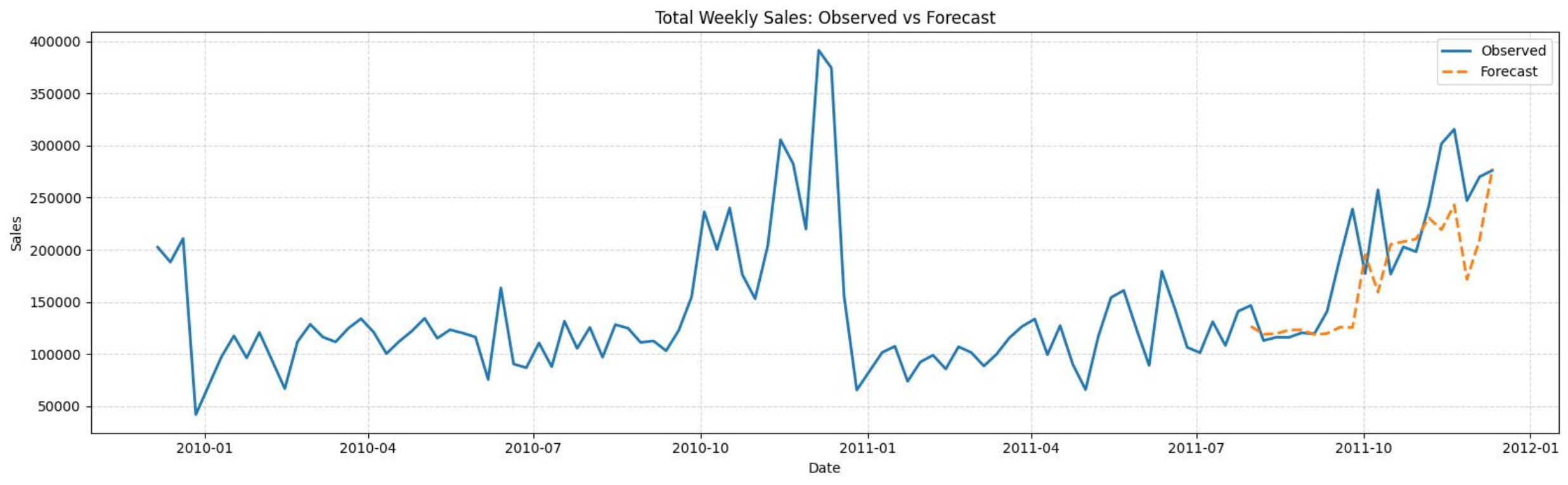
Best Global MAPE: **17.97%**

3-Test Period Results:

- Test Period I MAPE: 9.83%
- Test Period II MAPE: 23.11%
- Test Period III MAPE: 20.58%

Results I

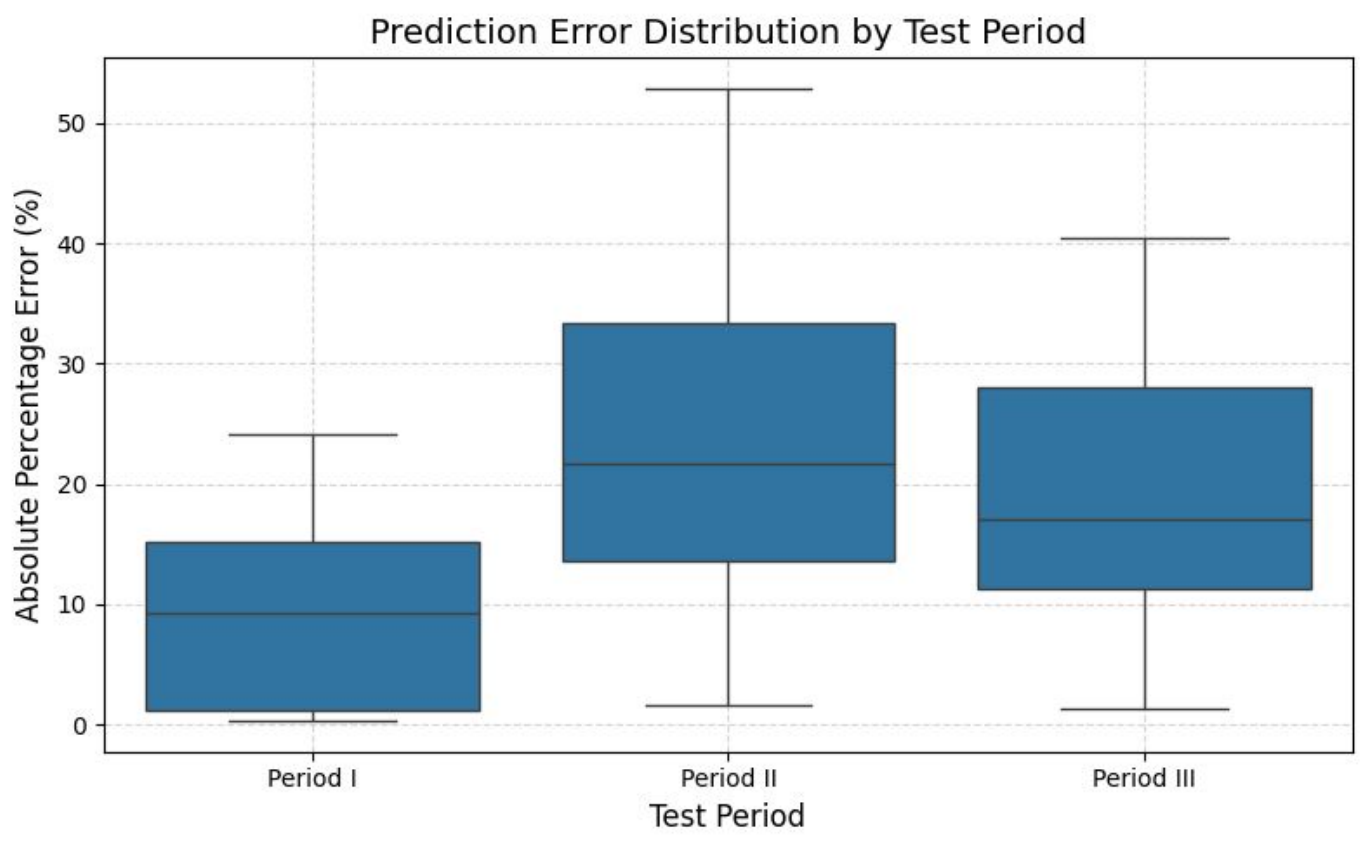
XGBoost Best Model Performance



- The blue line represents the observed (true) weekly sales.
- The orange line represents the forecasted sales during the test period.
- Overall, the model captures the general increasing trend but tends to slightly underpredict peak sales weeks.

Results I

XGBoost Best Model Performance



- Period I exhibited the lowest error dispersion, indicating the model fit better at the start of the test set.
- Periods II and III showed greater variability, likely reflecting increased volatility in product sales.

Results I

XGBoost Summary

XGBoost (Kmeans k = 2)	Overall Test MAPE	Test Period I MAPE	Test Period II MAPE	Test Period III MAPE
Best Config	**17.97%**	**9.83%**	**23.11%**	**20.58%**
Previous Team	34.08%	/	/	/

- ➔ The best-performing model achieved **an overall test MAPE of 17.97%**, which **reduced the overall MAPE by around 16.11%** compared to the previous team.
- ➔ This highlights the **effectiveness** of our **improved data cleaning**, **product categorization**, and hyperparameter optimization steps.




Results II – Chronos

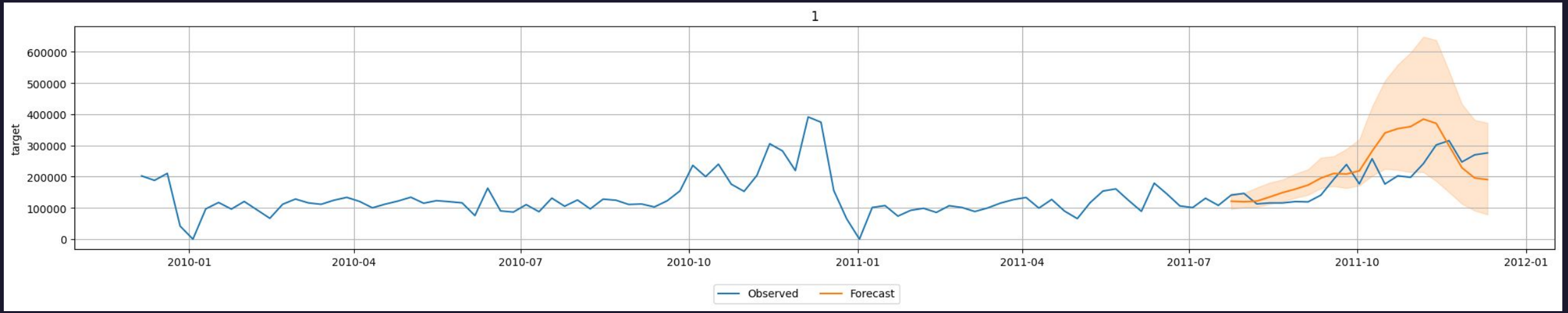
Results II

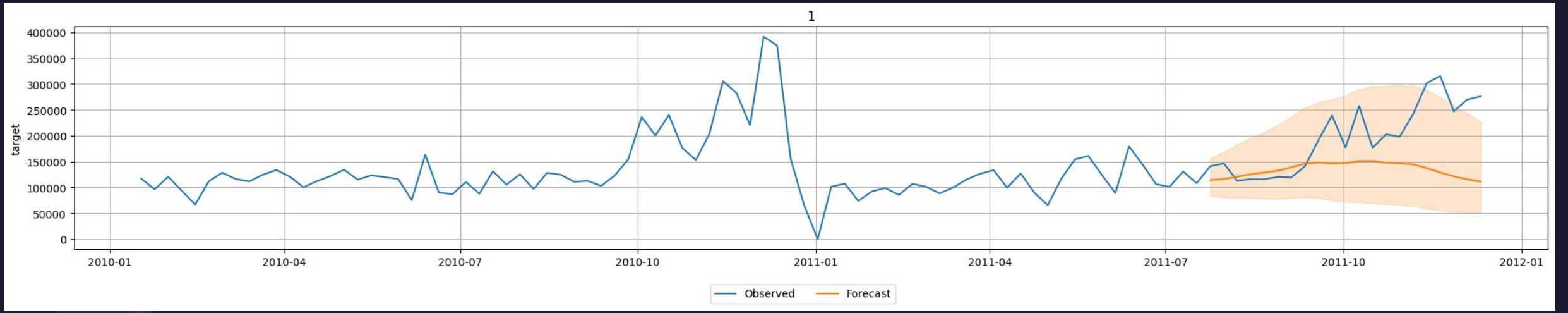
Chronos forecasting using cleaned data, weekly sales prediction

PREVIOUS Weighted MAPE	34.0815%	Improvement
zero-shot bolt_tiny (worst)	28.7973%	**5.2842%**
zero-shot bolt_mini	24.4255%	**9.6560%**
zero-shot bolt_small	19.1217%	**14.9598%**
zero-shot bolt_base (best)	**14.2683%**	19.8132%
fine-tuned bolt_tiny	28.5000%	**5.5815%**
fine-tuned bolt_mini	24.0416%	**10.0399%**
fine-tuned bolt_small	17.5394%	**16.5421%**
fine-tuned bolt_base	14.2683%	**19.8132%**

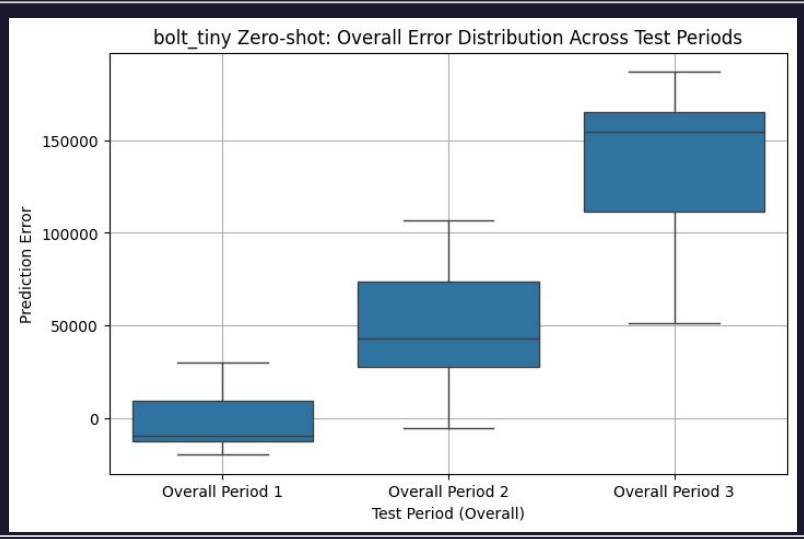
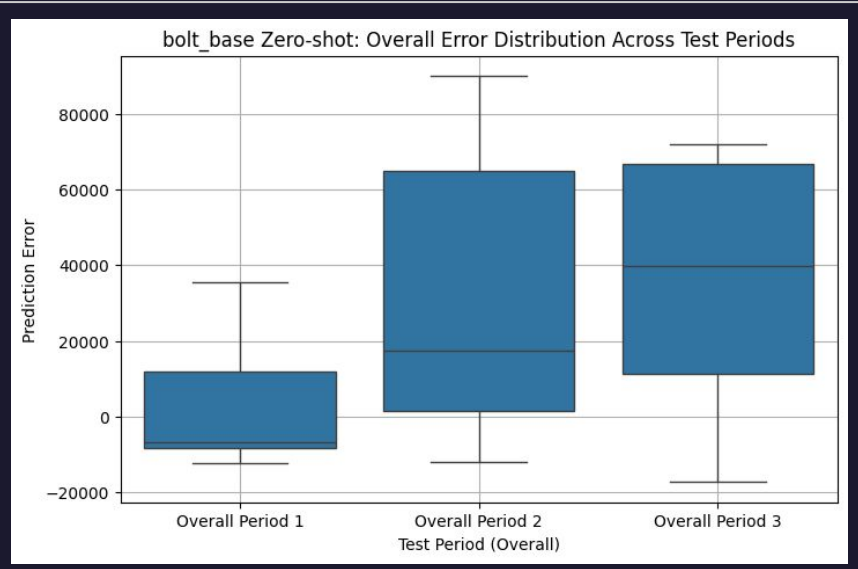
- Performance improves for all Chronos models compared with the previous team's result.
- **Best-performing model**: Chronos[bolt_base] achieves the lowest overall MAPE of 14.2683%, with a MAPE improvement of 19.8132%.
- **Worst-performing model**: Chronos[bolt_tiny] shows the highest MAPE of 28.7973%, with a MAPE improvement of 5.2842%.

Chronos forecasting using cleaned data, weekly sales prediction – best (zero-shot bolt_base) and worst (zero-shot bolt_tiny) model – Actual vs. Forecasted Plot





Chronos forecasting using cleaned data, weekly sales prediction – best (zero-shot bolt_base) and worst (zero-shot bolt_tiny) model – Error Distribution Plot & 3-period test MAPE

Model	Error Distribution Plot	Overall MAPE for Period 1, 2, 3
zero-shot bolt_tiny (worst)		13.22%, 23.53%, 49.65%
zero-shot bolt_base (best)		11.59%, 16.04%, 15.18%

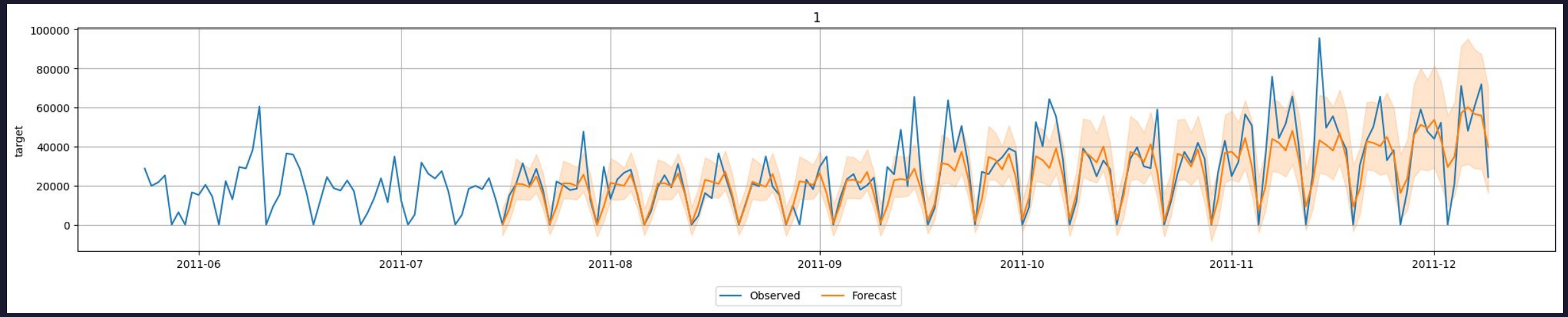
Results II

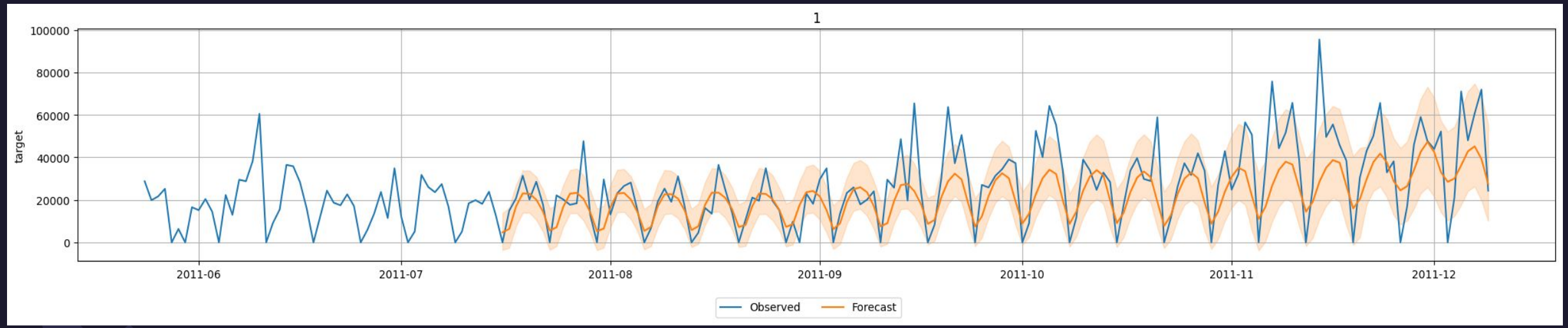
Chronos forecasting using cleaned data, daily sales prediction

PREVIOUS Weighted MAPE	34.0815%	Improvement
zero-shot bolt_tiny (worst)	29.4251%	**4.6564%**
zero-shot bolt_mini	29.0504%	**5.0311%**
zero-shot bolt_small	28.8359%	**5.2456%**
zero-shot bolt_base	27.9666%	**6.1149%**
fine-tuned bolt_tiny	27.4432%	**6.6383%**
fine-tuned bolt_mini (best)	**24.9931%**	**9.0884%**
fine-tuned bolt_small	26.9026%	**7.1789%**
fine-tuned bolt_base	26.5394%	**7.5421%**

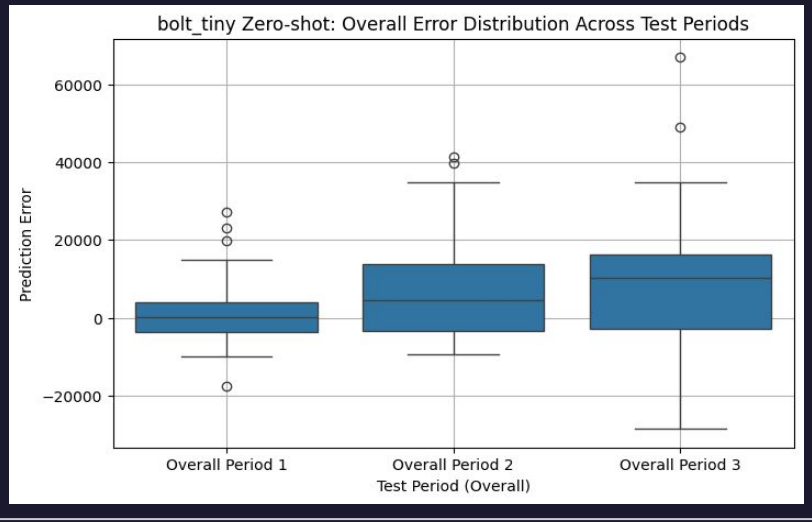
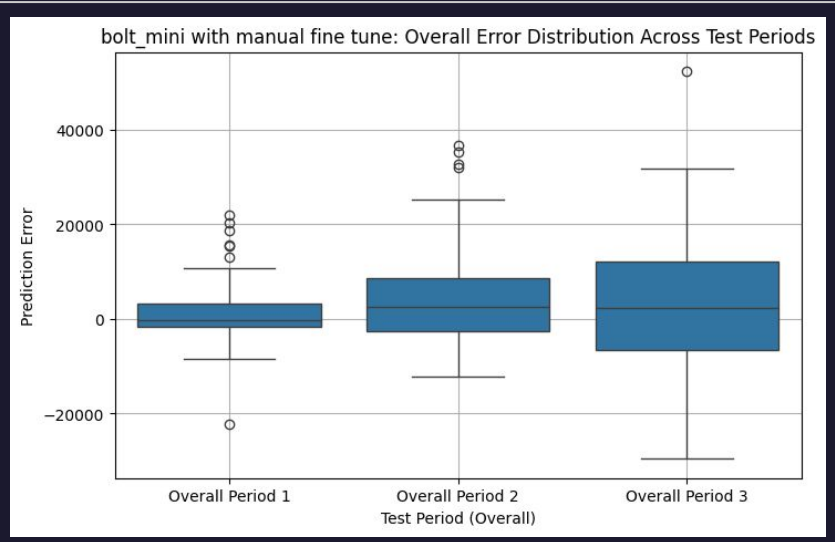
- Again, performance improves for all Chronos models compared with the previous team's result.
- **Best-performing model**: ChronosFineTuned[bolt_mini] achieves the lowest overall MAPE of 24.9931%, with a MAPE improvement of 9.0884%.
- **Worst-performing model**: Chronos[bolt_tiny] shows the highest MAPE of 29.4251%, with a MAPE improvement of 4.6564%.

Chronos forecasting using cleaned data, daily sales prediction – best (fine-tuned bolt_mini) and worst (zero-shot bolt_tiny) model – Actual vs. Forecasted Plot





Chronos forecasting using cleaned data, daily sales prediction – best (zero-shot bolt_base) and worst (zero-shot bolt_tiny) model – Error Distribution Plot & 3-period test MAPE

Model	Error Distribution Plot	Overall MAPE for Period 1, 2, 3
zero-shot bolt_tiny (worst)		25.80%, 31.73%, 30.66%
fine-tuned bolt_mini (best)		24.19%, 26.49%, 24.27%




Results III – Prophet

Results III

Prophet forecasting using cleaned data, weekly sales prediction

PREVIOUS Weighted MAPE	34.0815%	Improvement
Prophet (No Clustering)	10.65%	23.4315%
Prophet (No Clustering and removing Outliers)	17.34%	16.7415%
Prophet (Clustering:K Means k = 2)	12.61%	21.4715%
Prophet (Clustering: Agglomerative (k=2))	20.80%	13.2814%

- All models have increase in performance compared with previous team's model.
- The best model is the regular Prophet with no clustering and no further processing.It achieves overall MAPE of 10.65% with the improvement of 23.4315%.The corresponding parameter is:

```
Best Parameters Found:
changepoint_prior_scale          0.01
seasonality_prior_scale           1.0
holidays_prior_scale              1.0
seasonality_mode             additive
yearly_fourier                       6
weekly_fourier                      10
monthly_fourier                     10
use_holidays                     False
add_weekly                        True
add_monthly                      False
```

- The worst model is prophet with agglomerative clustering (k=2). It achieves Overall MAPE of 20.80% with the improvement of only 13.2814%. The optimal hyperparameters found:

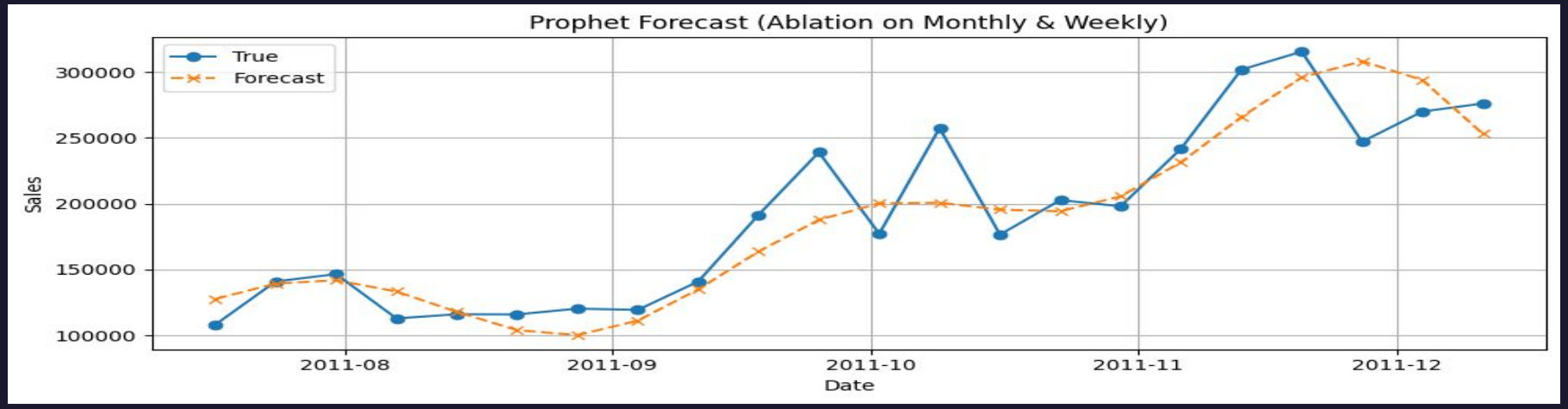
```
Best params for Cluster 1:
    {   'changepoint_prior_scale': 0.01,
    'seasonality_prior_scale': 0.1,
    'holidays_prior_scale': 1.0,
    'seasonality_mode': 'additive',
    'yearly_fourier': 7,
    'weekly_fourier': 10,
    'monthly_fourier': 5,
    'use_holidays': True,
    'add_weekly': True,
    'add_monthly': False}
```

```
Best params for Cluster 2:
    {   'changepoint_prior_scale': 0.1,
    'seasonality_prior_scale': 1.0,
    'holidays_prior_scale': 0.1,
    'seasonality_mode': 'multiplicative',
    'yearly_fourier': 10,
    'weekly_fourier': 10,
    'monthly_fourier': 5,
    'use_holidays': False,
    'add_weekly': True,
    'add_monthly': True}
```

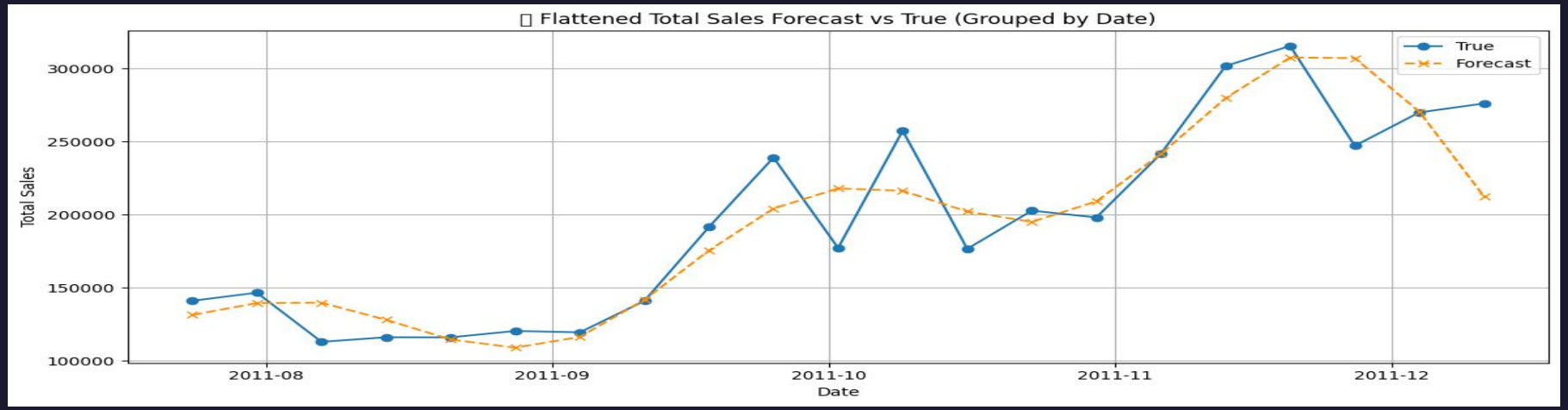
Results III

Prophet forecasting using cleaned data, weekly sales prediction – best (Prophet (No Clustering)) and worst (Prophet (Clustering: Agglomerative (k=2))) model – Actual vs. Forecasted Plot of test set

Best Model:
Prophet (No Clustering)

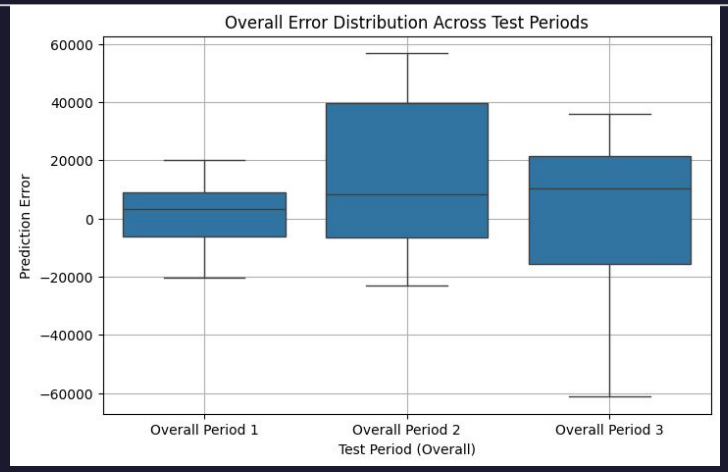
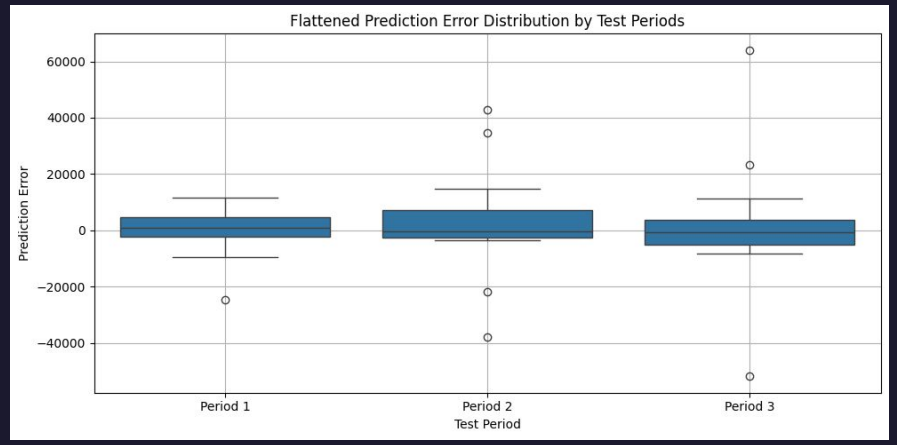


Worst Model:
Prophet (Clustering: Agglomerative (k=2))



Results III

Prophet

Model	Error Distribution Plot	Overall MAPE for Period 1, 2, 3
Prophet (No Clustering) (Best)		Overall MAPE for Test Period 1: 9.49% Overall MAPE for Test Period 2: 12.89% Overall MAPE for Test Period 3: 9.73%
Prophet (Clustering: Agglomerative (k=2)) (Worst)		Overall MAPE for Test Period 1: 18.75% Overall MAPE for Test Period 2: 18.48% Overall MAPE for Test Period 3: 25.18%




Results IV – LightGBM

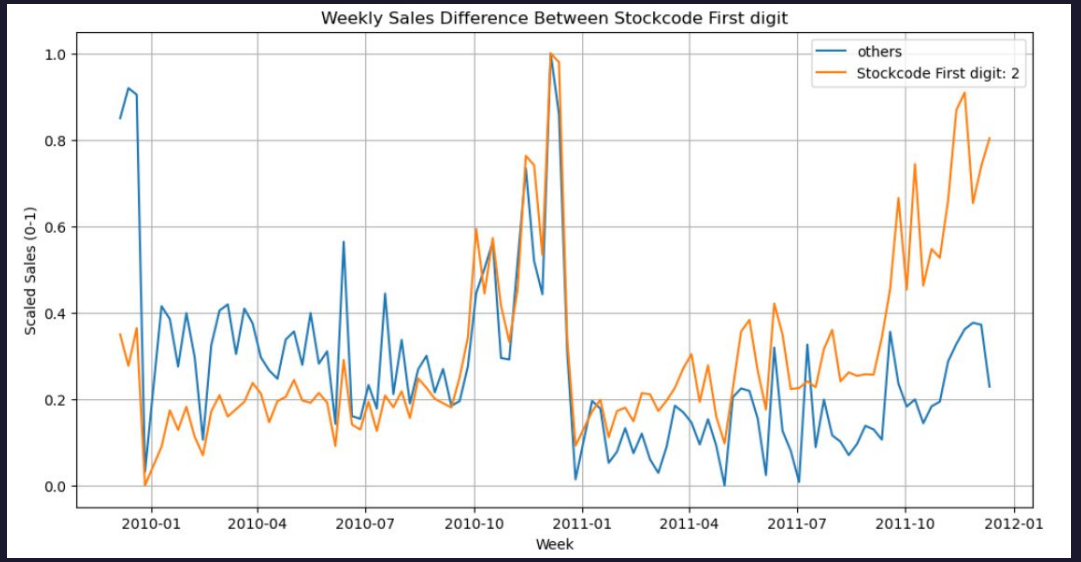
Results IV

LightGBM

Overall Model Performance

Model	MAPE	MAPE improvement from previous group best model(34.0815%)
aggregated model	26.720%	+7.3615%
one model for all	18.034%	+16.0475%

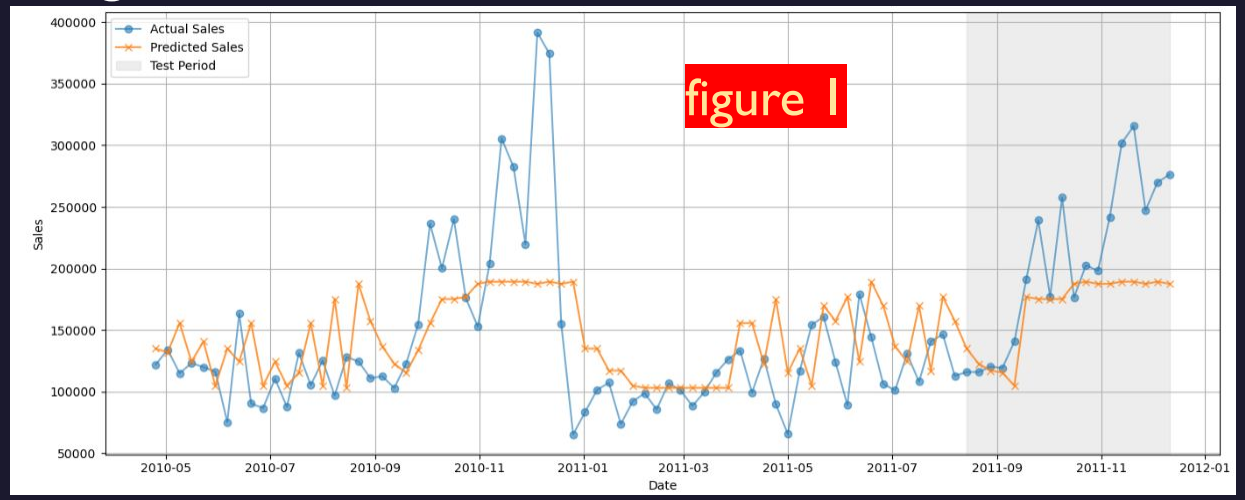
Sales trend between two different product type defined base on stock code initial.



Both models performed better than the previous group overall.
Dividing the product into two types base on their stock code initial and training the models separately did not perform better than a single model, even through these two type of product present visible difference in trend as shown in right figure.

Results IV

LightGBM



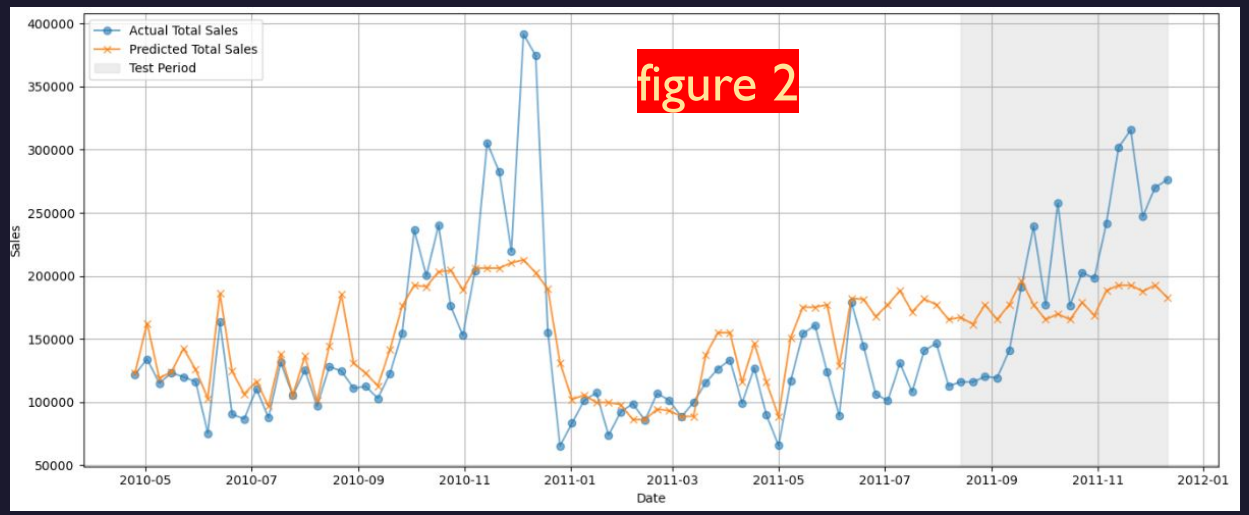
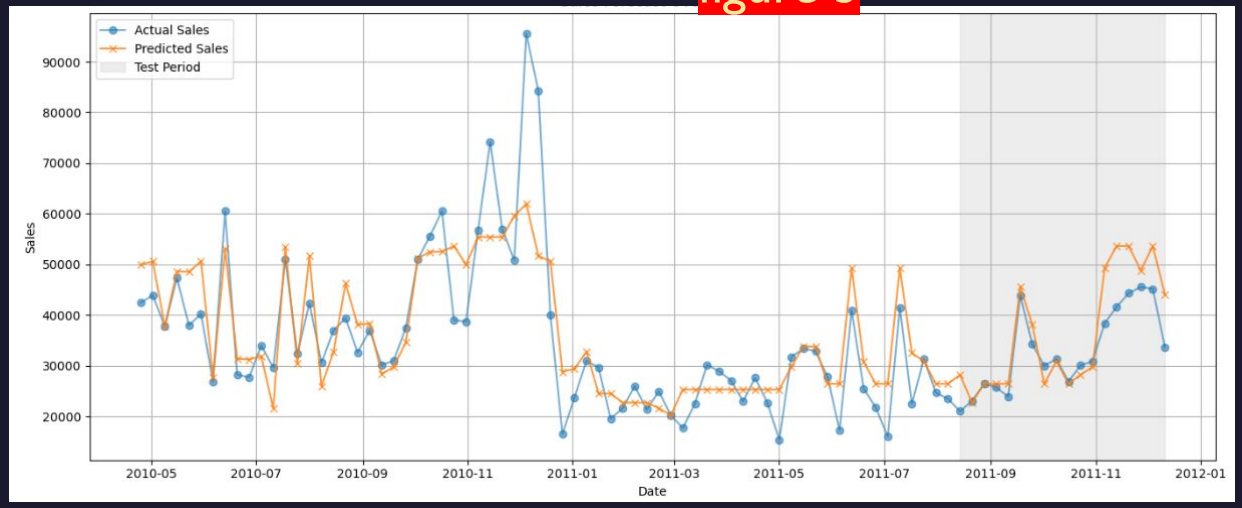


figure 3



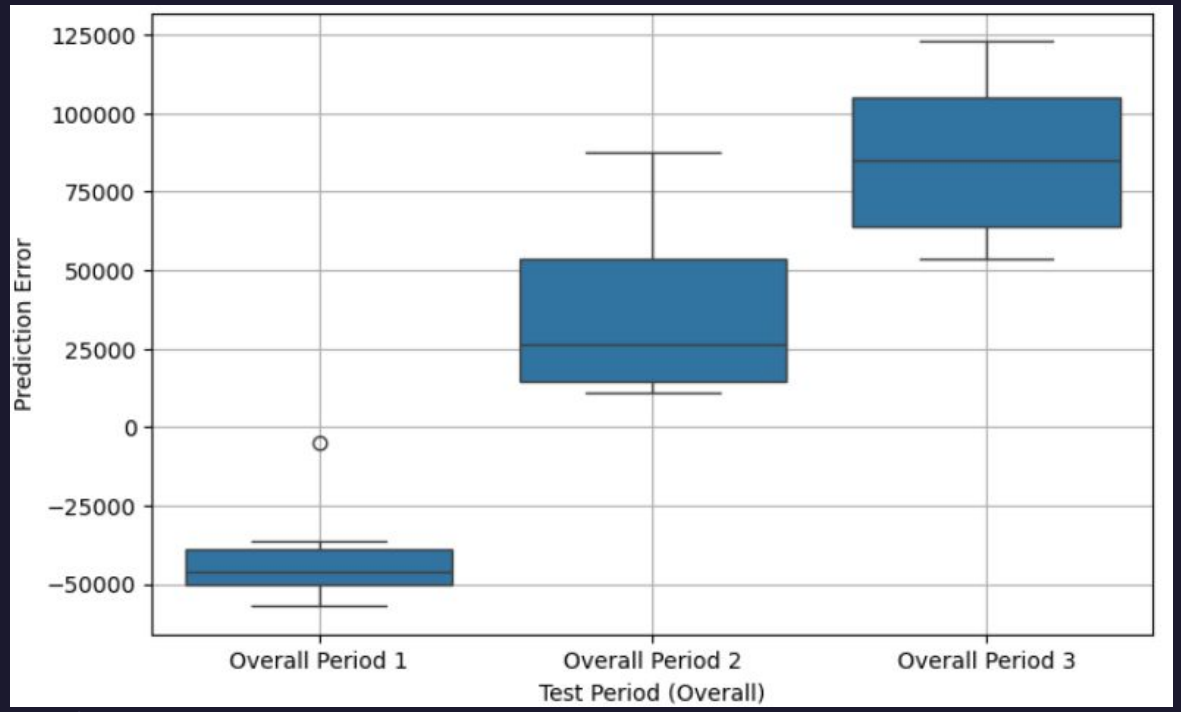
Overall both models do not capture the upward trend of sales in the later period very well. However, for the model with stock initial other than 2, the prediction results are significantly better than the overall results. This may be because compared with the product with stock code initial 2, the sales of these products have more distinct trend and seasonality therefore easier to be captured by the model.

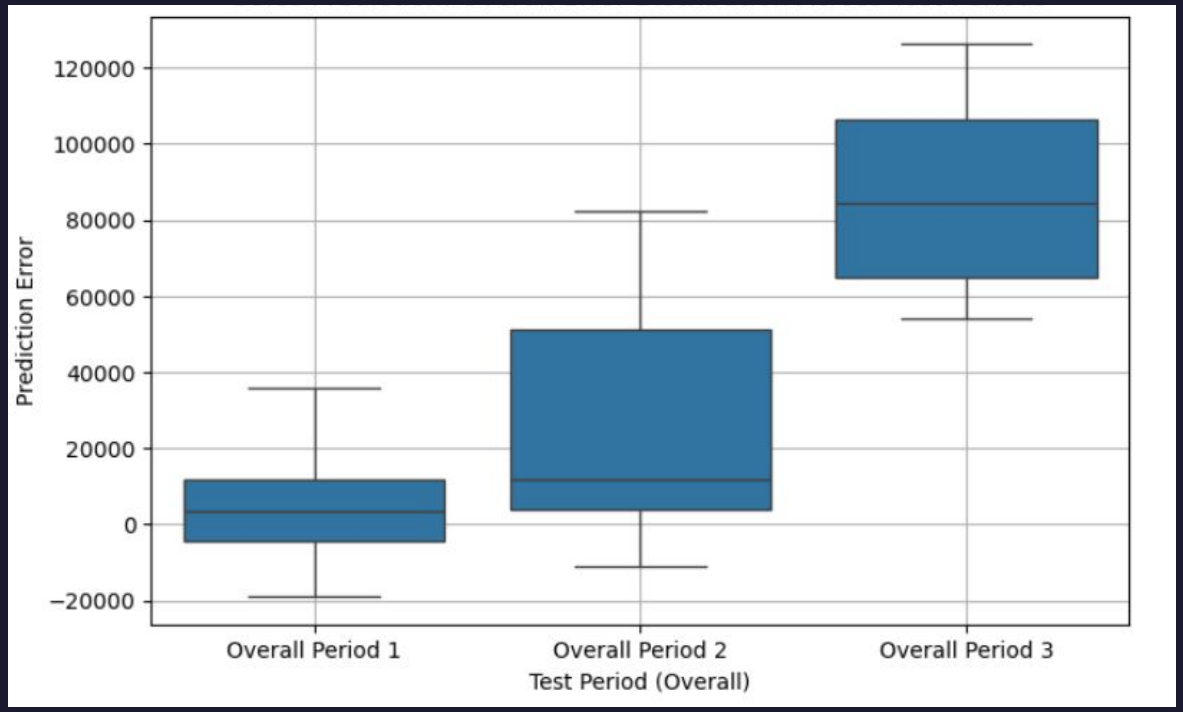
figure 1: prediction on all product by unitary model; figure 2: prediction on all product by aggregated model
figure 3: prediction on product with stock code initial other than 2

Results IV

LightGBM

Error Distribution Plot for unitary model(left) and aggregated model(right)





For all models, dividing the MAPE into three parts shows that they all struggle more with capturing sudden sales increases at more distant time.

Summary

Summary

1. Best Model: Prophet
    a. Achieved the lowest overall MAPE (10.65%) on the test set, demonstrating strong capability in forecasting weekly sales after appropriate feature engineering and parameter tuning.
    b. easy to use, interpretable, and robust; captures trend, seasonality, and holiday effects well; Need some domain knowledge to determine what to include in the model; performs reliably with limited data but may underperform on highly complex patterns; Time consuming to search for the optimal parameters in large parameter grids.
2. Other Models:
    a. XGBoost: provided a fast and scalable solution. Achieved competitive performance, but was slightly less effective in modeling long-term seasonality compared to Prophet and Chronos.
    b. Chronos: easy to use, fast, and efficient foundation model; very powerful forecasting capability; fine-tuning does not require domain knowledge as Prophet; may need sufficient data points to fully show its power, especially for fine-tuning.
    c. LightGBM trained quickly and easy to deploy but performed the worst among the four models. It needs more advanced feature engineering and fine-tuning to improve.
3. Key Takeaways:
    a. Proper product categorization and data cleaning substantially improved forecasting accuracy.
    b. Different models present trade-offs between accuracy, interpretability, and training efficiency.
    c. Incorporating historical lags and date-related features was crucial across all models to capture seasonality and trends effectively.

Further Improvement

Further Improvement

1. Categorization
   a. Incorporate additional product features such as sales volatility or product segmentation information.
   b. Evaluate clustering quality using both internal metrics (e.g., Silhouette Score) and external business validation.
2. XGBoost
   a. Further tuning of advanced hyperparameters may improve the model's ability to generalize.
   b. Incorporating engineered features like moving averages, rolling standard deviations, and holiday indicators could help capture more complex sales dynamics.
   c. Explicitly creating seasonality-related features (e.g., Fourier terms) may improve prediction performance.
3. Chronos
   a. Consider adding covariate analysis, such as holiday and weather variables, as mentioned when we introduce how to use this model in the model design section.
   b. Switch to using GPU rather than CPU for model fine-tuning, and also change hyperparameters, then check if performance will improve.
   c. Collect more data for Chronos to learn to fully make use of foundation model’s capabilities.

Further Improvement

4. Prophet
    a. Even though we have already taken a lot of parameters into consideration, for 'changepoint_prior_scale', 'seasonality_prior_scale' and 'holiday_prior_scale' we have not explored a lot of values since searching in the large parameter grid will be time consuming. So future improvements could explore more values for the hyperparameter.
    b. It is worth trying if 'Logistic' trend could improve the performance since we only use the default version of the trend
5. LightGBM
    a. Include more time-based features, like month-end indicators, and select lag values based on evidence from relevant research rather than only experience.
    b. Tune not just model hyperparameters but also optimize lag selections and calendar features for better performance.
    c. Use different LightGBM models with customized parameters and features for different product types.

Team

Zhiqi Ma

MSDS Student

Ruoheng Du

MSDS Student

Xiaojiang Wu

MSDS Student

Yijian Liu

MSDS Student