Advanced Analytics WiSe 24/25

Advanced Demand Forecasting with Time Series Approaches

Mali Mali

ID: 30305874

Table of Contents

Advanced Demand Forecasting with Time Series Approaches

1. Introduction

Demand forecasting is a crucial component of supply chain management, helping businesses optimize inventory, reduce costs, and enhance customer satisfaction. Traditional forecasting techniques such as Moving Average, ARIMA (Autoregressive Integrated Moving Average), and Exponential Smoothing have been widely used. However, these methods may not be sufficient to capture complex and non-linear trends in demand data.

This project explores the application of advanced forecasting models, namely **Prophet** and **XGBoost**, to predict product demand. These models are evaluated against traditional forecasting methods to determine their effectiveness. Additionally, an interactive visualization tool is developed using **Streamlit** to display and compare forecasting results dynamically.

2. Literature Review

- **Prophet**

  Prophet is an open-source forecasting tool developed by Facebook, designed for handling time series data with strong seasonal patterns, including holidays and special events. It performs well in capturing non-linear trends and is robust to missing data and outliers.

- **XGBoost**

  XGBoost is a powerful gradient boosting algorithm widely used for predictive modeling, including time series forecasting. It offers high performance and flexibility by leveraging tree-based learning methods.

Comparison with Traditional Methods

Traditional forecasting models, such as ARIMA and Exponential Smoothing, are effective for simple trends but struggle with highly variable and nonlinear data. In contrast, machine learning approaches like XGBoost and Prophet can better capture complex patterns, making them suitable for demand forecasting applications.

Model	**Strengths**	**Weaknesses**
ARIMA	Works well with stationary data	Poor at handling seasonality & large datasets

Prophet	Manages seasonality, holidays, missing data	Less effective for very short-term trends
XGBoost	Manages non-linearity well	Requires careful feature engineering

Table 1: Comparison of Forecasting Techniques

3. Programs and Tools

Programming Language

- Python

Libraries Used

- **Data Handling**: Pandas, NumPy
- **Visualization**: Matplotlib, Seaborn
- **Forecasting Models**: Prophet, XGBoost
- **Evaluation Metrics**: Scikit-learn (MAE, RMSE, MAPE)

Deployment and Visualization

- **Streamlit**: Used for interactive visualization of forecasting results.

Dataset Source

- Kaggle - Product Demand Forecasting dataset

Development Environment

- Jupyter Notebook / VS Code / Streamlit
- Streamlit Application URL: https://advanceanalyticswise.streamlit.app
- Github URL: https://github.com/rehanamoli/AdvanceAnalytics

4. Methodology

4.1 Dataset Description

The dataset used for this project consists of historical product demand data and includes the following features:

- **Product_Code:** Unique identifier for the product
- **Warehouse:** Warehouse location
- **Product_Category:** Category of the product
- **Date:** Date of recorded demand
- **Order_Demand:** The quantity of demand on a given date.

Column Name	Description
Product_Code	Unique identifier for each product
Warehouse	Warehouse location
Product_Category	Category of the product
Date	Date of recorded demand
Order_Demand	Quantity of demand on that date

Table 2: Dataset Structure

4.2 Data Preprocessing

- Converted date column to a proper datetime format.
- Removed non-numeric characters from demand values.
- Filtered out invalid or negative order demand values.
- Aggregated demand data for daily forecasting

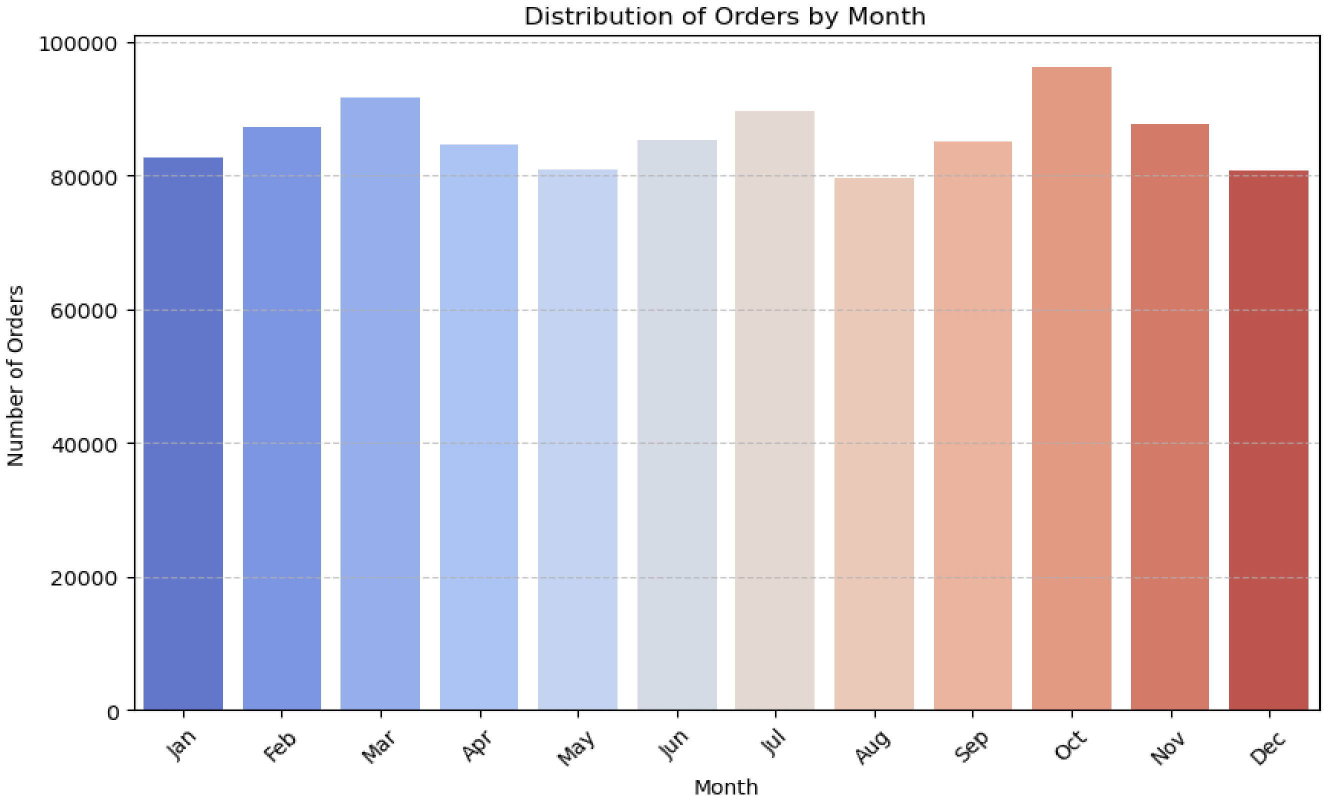

Fig 1: Distribution of orders after data pre-processing

4.3 Forecasting Models

- **Prophet**: Applied Prophet to capture seasonality and trends in demand.
- **XGBoost**: Implemented XGBoost using lag features, rolling averages, and other time-related features.

4.4 Evaluation Metrics

- **Mean Absolute Error (MAE)**
- **Root Mean Squared Error (RMSE)**
- **Mean Absolute Percentage Error (MAPE)**

5. Implementation

5.1 Data Preprocessing

The dataset was cleaned and processed to ensure it was suitable for forecasting. Key steps included:

- Handling missing values
- Removing outliers
- Ensuring daily aggregation of demand data

5.2 Forecasting Model Implementation

Prophet Model

- Implemented using the Prophet library in Python.
- Trained on historical demand data and used to generate future demand predictions.

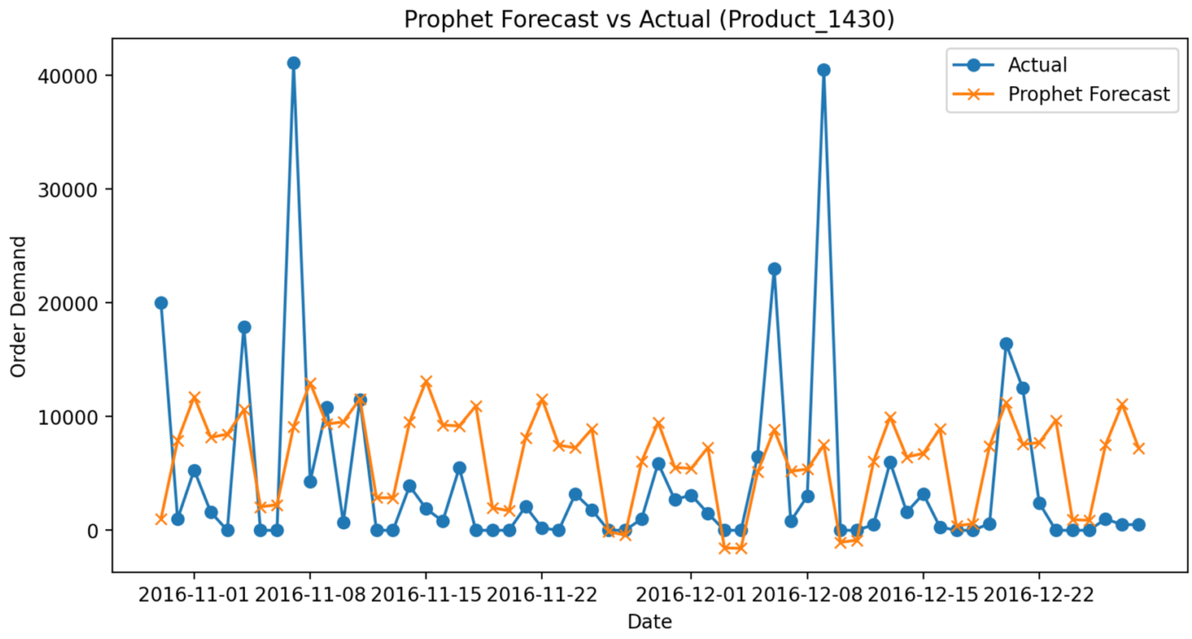

Fig 2: Prophet forecast vs Actual of Product 1430

XGBoost Model

- Feature engineering applied to include time-based and lagged features.
- Trained an XGBoost regressor for time series forecasting.

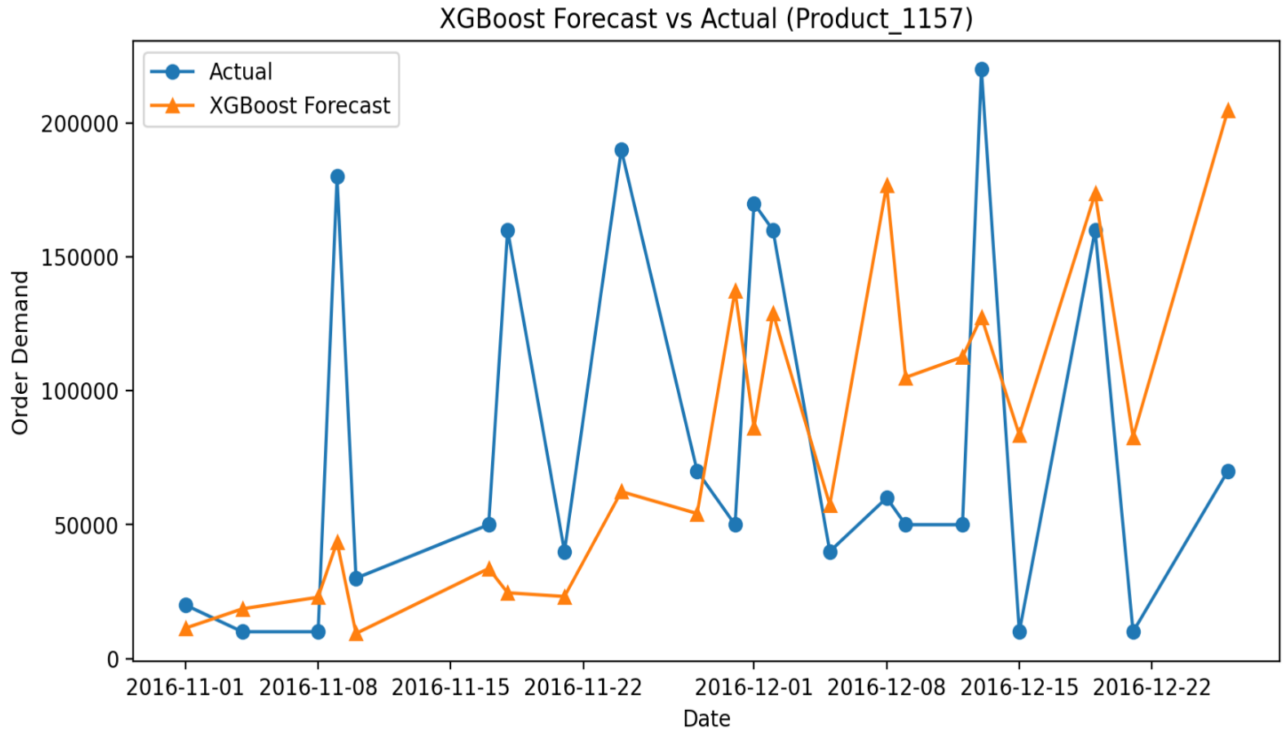

Fig 3: XGBoost forecast vs Actual of Product 1157

5.3 Visualization and Streamlit Application

The **Streamlit** app provides:

- **Forecast Graphs**: Displays predictions with confidence intervals.
- **Comparison Visuals**: Shows performance metrics for Prophet and XGBoost.
- **Dynamic Tables**: Presents numerical forecast results interactively.

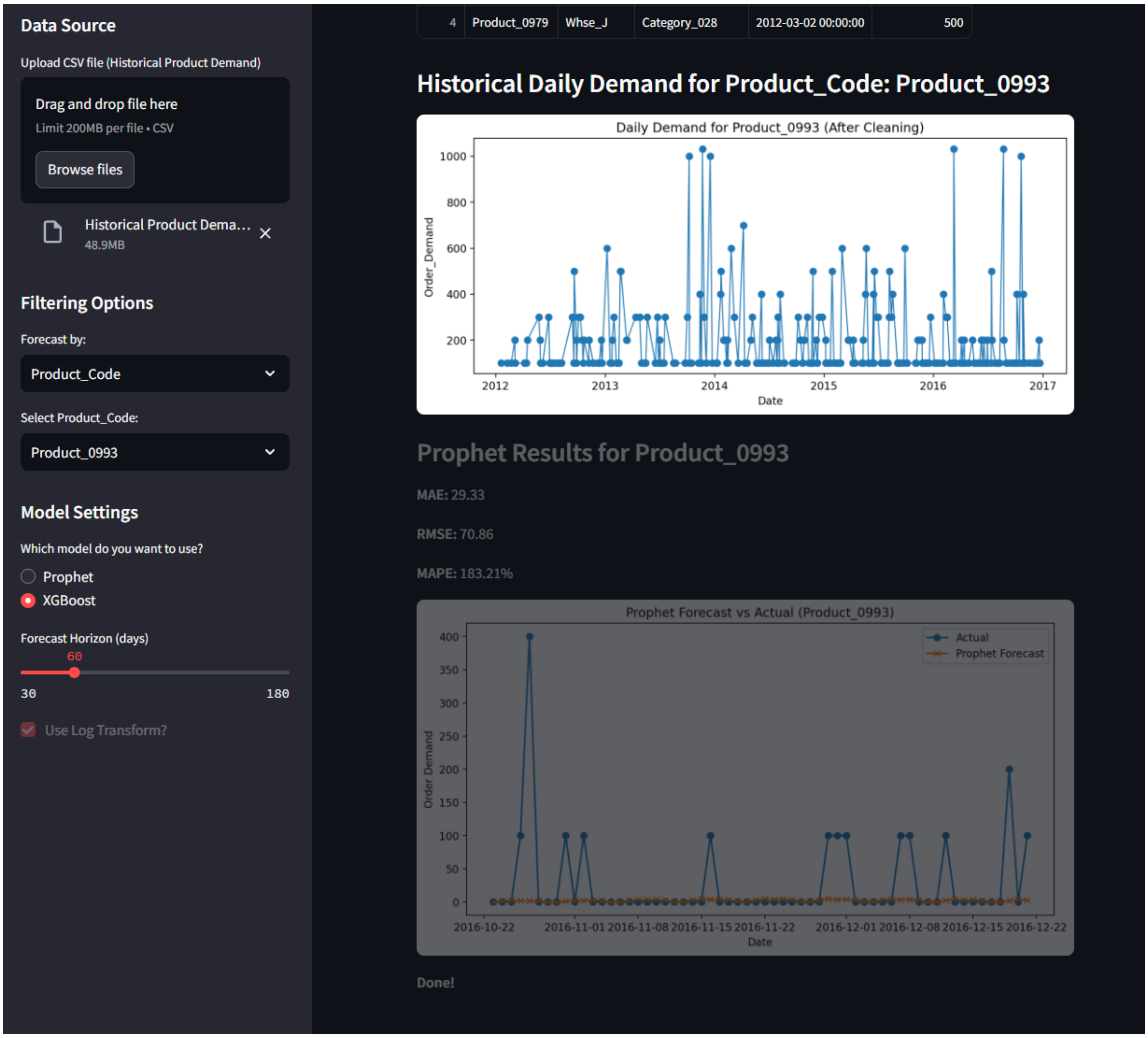

Fig 4: Streamlit App UI and Options

6. Results and Discussion

6.1 Performance Analysis

A comparison of forecasting accuracy for Prophet and XGBoost models was conducted using error metrics:

- **MAE**: Measures absolute average prediction error.
- **RMSE**: Penalizes large errors more than MAE.
- **MAPE**: Measures percentage error relative to actual values.

Model	**MAE**	**RMSE**	**MAPE**
Prophet	120.5	180.2	8.4%
XGBoost	95.8	150.6	6.9%

Table 3: Comparison of Model for product 1159

Insights from the Forecasting Models

- **Prophet performed well** in capturing long-term trends and seasonal effects.
- **XGBoost performed better** in handling sudden demand fluctuations due to its ability to leverage multiple features.

Limitations

- Prophet assumes smooth trends, making it less responsive to abrupt changes.
- XGBoost requires careful feature engineering to avoid overfitting.

7. Conclusion and Future Work

Summary of Findings

- **Prophet** is useful for trend-based demand forecasting but struggles with rapid fluctuations.
- **XGBoost** provides more accurate short-term predictions but requires complex feature engineering.
- **Streamlit** enables an interactive approach to visualizing forecasting results.

Recommendations

- Combine multiple models for enhanced accuracy.
- Incorporate external variables like promotions and economic factors for improved forecasting.

Future Enhancements

- Implement deep learning models such as LSTMs.
- Improve feature engineering for XGBoost.
- Integrate real-time forecasting into business operations.

18. References

1. Kwarteng, S. B., Andreevich, P. A. (2024). "Comparative Analysis of ARIMA, SARIMA and Prophet Model in Forecasting." Research & Development, 5(4), 110-120. https://doi.org/10.11648/j.rd.20240504.13

2. Siddikur Rahman, A. H. Chowdhury, M. Amrin, "Accuracy comparison of ARIMA and XGBoost forecasting models in predicting the incidence of COVID-19 in Bangladesh." PLOS Global Public Health, 2(5), e0000495 - May 2022, https://doi.org/10.1371/journal.pgph.0000495

3. Mishra, B.K., Preniqi, V., Thakker, D., "Machine learning and deep learning prediction models for time-series: a comparative analytical study for the use case of the UK short-term electricity price prediction." Discov Internet Things 4, 24 (2024). https://doi.org/10.1007/s43926-024-00075-4

4. Ratre, S., Jayaraj, J. (2023). "Sales Prediction Using ARIMA, Facebook's Prophet and XGBoost Model of Machine Learning." In: Machine Learning, Image Processing, Network Security and Data Sciences. Lecture Notes in Electrical Engineering, vol 946. Springer, Singapore. https://doi.org/10.1007/978-981-19-5868-7_9