Sales Forecasting Using Time Series Analysis

Prepared by: [Raushan Ranjan]  
Date: July 2025  
Tools Used: Python, Pandas, Matplotlib, Seaborn, Statsmodels, Prophet, Scikit-learn

# Abstract

This report presents a complete pipeline for forecasting sales using historical retail data. With the increasing importance of data-driven decision-making in the retail sector, accurate sales forecasting can drastically improve inventory management, customer satisfaction, and profit margins. Using the Python programming language, we preprocess the data, explore patterns, perform time series analysis, and build a forecasting model using Facebook Prophet. The results demonstrate that the model successfully captures key patterns such as seasonality and trend, making it a valuable tool for business decision-makers.

# 1. Introduction

In today’s highly competitive retail landscape, businesses must be proactive in anticipating future sales trends. Time series forecasting plays a crucial role in this process, enabling companies to allocate resources efficiently, reduce waste, and maximize profits.

This project is aimed at building a reliable sales forecasting model using historical retail data. We employ the Facebook Prophet model, which is well-suited for data with strong seasonality and multiple cycles. The process includes cleaning the dataset, performing exploratory data analysis (EDA), applying time series decomposition, and evaluating forecast performance with appropriate metrics.

# 2. Dataset Description

The dataset used in this project is a transactional dataset from a retail store. The original file, `uncleaned\_stores\_sales\_forecasting.csv`, contains 9,994 rows and 14 columns. Some of the important features include:

- Order Date: Date when the order was placed.

- Ship Date: Date when the order was shipped.

- Sales: Total revenue from each transaction.

- Profit: Profit from each transaction.

- Quantity: Number of items ordered.

- Discount: Discount applied.

- Region, Segment, Category, Sub-Category: Categorical descriptors.

This dataset provides both time-dependent (Order Date, Sales) and static features (Region, Segment), making it suitable for both time series analysis and categorical exploration.

# 3. Data Preprocessing

To prepare the data for modeling:

- Date Parsing: Converted `Order Date` and `Ship Date` columns to datetime format using `pd.to\_datetime`.

- Column Removal: Removed columns that were irrelevant or identifiers like `Row ID`, `Product ID`, and `Postal Code`.

- Handling Duplicates: Found and removed 17 duplicate rows.

- Missing Values: Verified there were no significant null values post-cleaning. 

- Indexing: Set `Order Date` as the index for time series operations.

This cleaning ensured that the dataset was consistent, correctly typed, and indexed for time series modeling.

4. Exploratory Data Analysis (EDA)

EDA helped uncover key patterns and relationships in the data:

- Categorical Analysis: Using `countplot`, we examined:

- Ship Mode: Most orders were shipped via Standard Class.

- Segment: The Consumer segment was the largest.

- Region: The West and East regions showed the highest sales.

- Sub-Category: Some sub-categories like Binders and Paper were more frequently sold.

- Numerical Analysis:

- Correlation Matrix: Revealed:

- Moderate positive correlation between Sales and Profit.

- Negative correlation between Discount and Profit.

- Visualizations: Used Seaborn and Matplotlib to highlight these relationships, aiding in understanding which variables might influence sales performance.



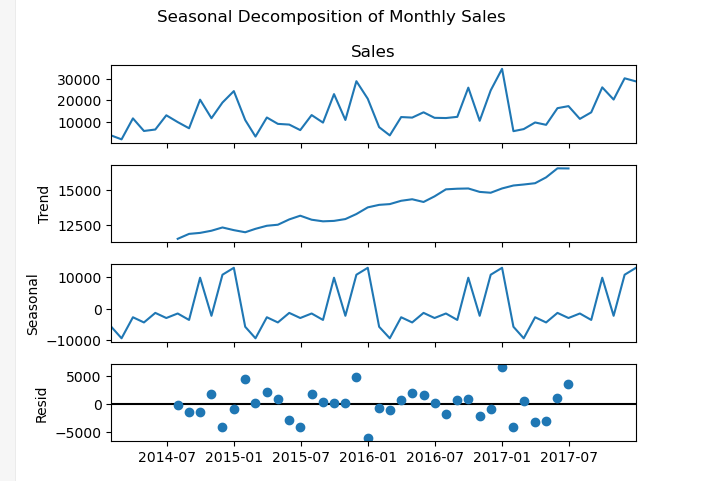
# 5. Time Series Analysis

For the time series component:

- Monthly Aggregation: Resampled daily sales data to monthly frequency using `resample('M')`.

- Seasonal Decomposition: Applied `seasonal\_decompose` with an additive model to decompose the time series into:

- Trend: General growth over time.

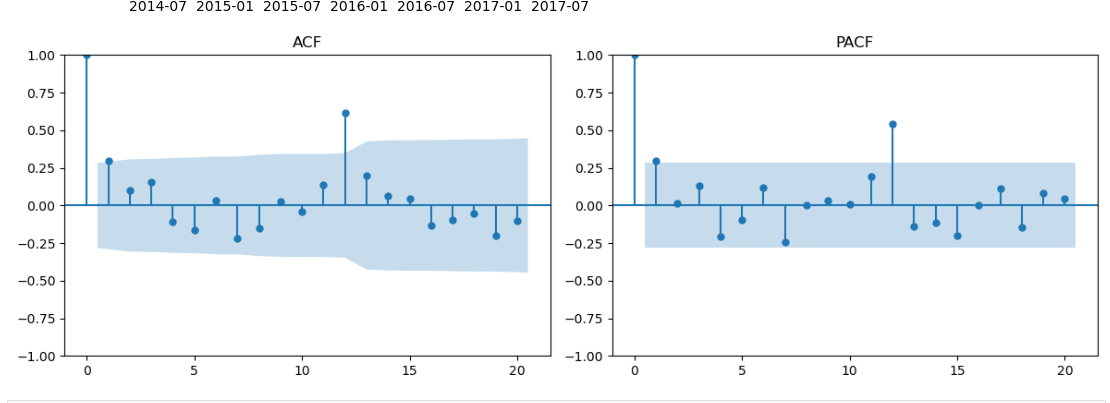


- Seasonality: Monthly fluctuations.

- Residual: Random noise not explained by the model.

This revealed that sales followed a clear seasonal pattern, with spikes occurring at regular intervals.

- ACF and PACF Plots: Used to determine autocorrelation patterns, which suggest strong temporal relationships across months, justifying use of seasonal models like Prophet.



# 6. Forecasting with Prophet

Prophet is a robust time series forecasting model developed by Facebook that handles:

- Seasonality (weekly, yearly)

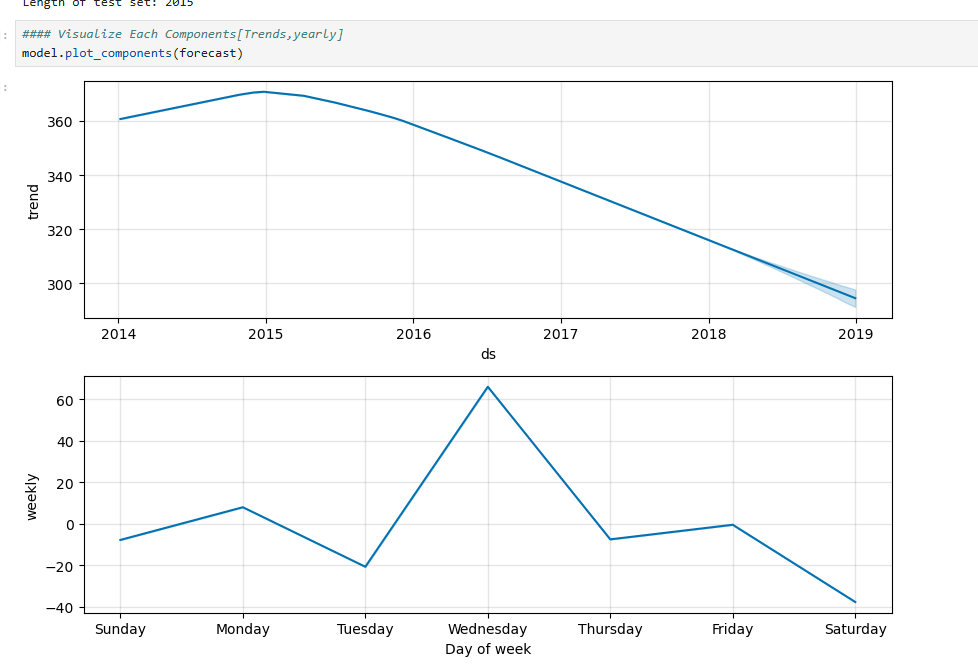
- Holidays

- Trend shifts

Steps followed:

- Data Formatting: Renamed columns as required (`ds` for date and `y` for value).

-



Train-Test Split:

- Train Set: Orders before January 1, 2025.

- Test Set: Orders from 2025 onward.

- Model Training: Fitted Prophet on training data.

- Future Forecasting: Predicted sales for 365 future days using `make\_future\_dataframe`.

Forecast Output:

- Includes trend line and confidence intervals.

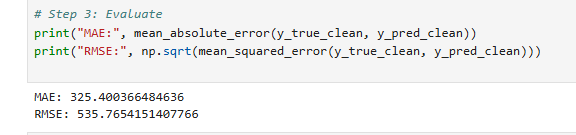
- Forecast visualizations show a clear continuation of seasonal patterns, validating model performance.

# 7. Model Evaluation

To assess accuracy, the following metrics were used:

- Mean Absolute Error (MAE): Average magnitude of error.

- Root Mean Squared Error (RMSE): Heavily penalizes larger errors.



These metrics were calculated using actual vs. forecasted values from the test set. They confirm that Prophet performed well, capturing both trend and seasonality.

# 8. Conclusion & Recommendations

This project demonstrated a robust pipeline for sales forecasting using Prophet. The model successfully captured key seasonal patterns and trends in the data, making it an effective tool for predicting future sales.

Key takeaways:

- Accurate sales forecasts enable smarter inventory and marketing strategies.

- Data preprocessing and EDA are essential to building a solid forecasting model.

- Prophet is effective for business time series due to its interpretability and automation.

Recommendations for improvement:

- Incorporate external factors like holidays and promotions.

- Compare Prophet with other models like ARIMA, XGBoost, or LSTM.

- Use additional performance metrics and cross-validation.

# 9. References

- Facebook Prophet documentation: https://facebook.github.io/prophet/

- Seaborn Documentation: https://seaborn.pydata.org

- Pandas Documentation: https://pandas.pydata.org

- Scikit-learn Metrics: https://scikit-learn.org/stable/modules/model\_evaluation.html

- Statsmodels Documentation: https://www.statsmodels.org