Sales Forecasting and Optimization

Team

Mahmoud Sabry Ahmed Hussein

Mohamed Samy Rizk Abuzaid

Huda Magdy Abdel Wahab Abdel Razzaq

Belal Khamis Qutb Hassan

Marwa Kotb Saad Kotb

Contents

Sales Forecasting and Optimization

---

1. Project Overview

1.1 Objective and Importance

Sales forecasting plays a central role in modern business operations, especially in retail where inventory levels, staffing, marketing campaigns, and supply chain logistics are tightly linked to anticipated customer demand. This project aims to develop a robust, data-driven machine learning system to forecast sales volumes in advance — allowing businesses to better align resources, reduce losses, and maximize profits.

The key goal of this project is to accurately **predict future sales** across different retail stores and product categories by leveraging **historical sales records, promotions, holiday effects, oil prices, and store metadata**. Accurate forecasts will provide tangible business benefits, including:

- Reducing **overstocking** and **stockouts**.
- Improving **promotional planning**.
- Enhancing **demand-driven logistics and marketing**.
- Enabling **strategic pricing** and **staffing decisions**.

This project is not just a modeling task; it incorporates the **full data science lifecycle**, from data collection to deployment, incorporating MLOps practices to ensure scalability, maintainability, and reproducibility.

2. Data Collection, Exploration, and Preprocessing

2.1 Data Source and Description

We used a dataset from Kaggle titled:
"Store Sales - Time Series Forecasting (Merged Dataset)"
Dataset Link

This dataset is a **merged compilation** of multiple sources from a forecasting competition. It integrates:

- Historical **sales data**
- **Promotions**
- **Store metadata** (store type, location, cluster)
- **Holiday and event data**
- **Oil prices**

- **Transaction data**

Key Columns:

COLUMN	DESCRIPTION
DATE	The date of the transaction
STORE_NBR	Store identifier
FAMILY	Product category/type (e.g., dairy, cleaning)
SALES	Sales value (can be fractional due to weights, e.g., 1.5 kg cheese)
ONPROMOTION	Number of items on promotion that day
CITY, STATE, TYPE, CLUSTER	Store metadata
HOLIDAY_TYPE, TRANSFERRED	Holiday metadata with transfer information
DCOILWTICO	Oil price on that date (macroeconomic indicator)
TRANSACTIONS	Number of purchase transactions on that date (missing in test data)

This multidimensional dataset provides not only time series sales data but also **external variables** (oil, holidays), which are essential for building **multivariate forecasting models**.

2.2 Data Exploration (EDA - Exploratory Data Analysis)

Purpose of EDA:
EDA helps uncover patterns, outliers, trends, seasonality, and relationships in data. Before modeling, this step ensures we understand how variables interact and behave across time.

Key Observations:

- **Sales Trend**: We observed a general upward trend in sales over time. This suggests positive business growth or an increase in consumer demand.
- **Seasonality**: Sales increase significantly during holidays (e.g., Christmas, New Year), confirming seasonal patterns.
- **Promotions Impact**: Products on promotion showed significantly higher sales.
- **Holiday Impact**: While most holidays increased sales (due to demand spikes), some led to dips — likely due to store closures or reduced consumer movement.
- **Store Differences**: Sales volumes varied widely across store clusters and locations.

Visualizations Used:

- **Line charts** for time-based trends.
- **Boxplots** to observe sales distribution across store types and families.
- **Heatmaps** for correlation analysis between features.
- **Bar plots** to compare promotional vs. non-promotional periods.

Each chart was accompanied by interpretation to derive meaningful insights and identify feature candidates for modeling.

2.3 Preprocessing and Feature Engineering

This step transforms raw data into a format suitable for machine learning by cleaning, encoding, normalizing, and creating useful features.
1. Handling Missing Values

- **Oil Prices and Holidays** had missing values.
- Used **forward fill** for time series fields and dropped rows with unimportant missing metadata.

2. Outlier Detection

- Sales outliers were visualized via boxplots and line charts.
- Outliers during promotions or holidays were retained (likely valid), while others were winsorized or removed.

3. Feature Engineering

This involves creating new, informative variables to improve model performance:

FEATURE	DESCRIPTION	USE
DAY, WEEK, MONTH, YEAR	Extracted from date	Time-aware features
IS_HOLIDAY, IS_WEEKEND	Flags	Captures temporal events
LAG_SALES_T1, LAG_SALES_T7, LAG_SALES_T30	Previous sales at various lags	Time-series memory
ROLLING_MEAN_7, ROLLING_STD_7	7-day rolling statistics	Captures trends/smoothing
PROMO_FLAG	If product was on promotion	Promotion modeling
OIL_TREND_BIN	Oil prices bucketed	External macro condition

These features help the model learn **patterns of temporal dynamics** and **external drivers of sales**.

4. Encoding Categorical Variables

- Used **One-Hot Encoding** for family, city, type, and holiday_type.
- Avoided Label Encoding due to risk of imposing artificial order.

5. Scaling and Normalization

- Used **StandardScaler** to standardize features like oil prices and lag statistics for algorithms like XGBoost.

3. Advanced Analysis, Feature Selection, and Visualization

3.1 Time Series Analysis

To ensure valid time series modeling, we performed:

- **Stationarity Tests**: Used **Augmented Dickey-Fuller (ADF)** to check stationarity. Time series had trends and were non-stationary; handled through differencing for ARIMA and by using trend-aware models like Prophet and XGBoost.
- **Seasonal Decomposition**: Using statsmodels to separate **trend**, **seasonal**, and **residual** components.
- **Autocorrelation Analysis**: ACF/PACF plots helped us determine lag dependencies and cyclic behavior.

3.2 Feature Importance and Correlation

- Calculated **Pearson** and **Spearman** correlation matrices.
- Analyzed **Mutual Information** to understand non-linear associations.
- Feature importance was also derived from XGBoost (tree-based impurity scores).

3.3 Enhanced Visualizations

- Used **Plotly** and **Seaborn** for interactive insights.
- Visualized **sales trends by family, store type, and region**.
- Interactive dashboards allowed dynamic filtering by product and date ranges.

4. Forecasting Model Development and Optimization

4.1 Models Built

We built and compared:

- **Facebook Prophet**:
  - Great for time series with clear trends and seasonality.
  - Automatically detects holidays and trends.
  - Handles missing data and outliers natively.
- **XGBoost Regressor**:
  - Gradient boosting model; handles non-linear relationships and high-dimensional features.
  - Beneficial for multivariate time series with many engineered features.
  - Requires careful feature design (lags, date parts).

4.2 Model Training & Evaluation

- Time-based **train-test split** ensured no data leakage.
- Used **TimeSeriesSplit** cross-validation.
- Evaluated using:
  - **MAE**: Mean Absolute Error (robust to outliers).
  - **RMSE**: Root Mean Squared Error (penalizes large errors).
  - **MAPE**: Mean Absolute Percentage Error (interpretable in %).

MODEL	RMSE	MAE	MAPE
PROPHET	342.5	212.3	19.7%
XGBOOST	289.1	188.7	15.4%

XGBoost outperformed Prophet, due to its capacity to learn from engineered features and external variables.

5. Deployment and MLOps

5.1 Streamlit App Deployment

Built a user-friendly web app using **Streamlit** where business users can:

- Select store, product type, and future date.
- Get **forecasted weekly sales**.
- View charts comparing predictions with past performance.

5.2 MLOps Practices

- **MLflow**: Logged model versions, parameters, and metrics.
- **DVC**: Tracked data changes and feature versions for reproducibility.
- **Docker**: Used for containerized deployment (optional step).
- Future integrations with **cloud platforms (AWS, Azure)** are possible.

5.3 Monitoring & Retraining

- Designed mechanisms to log performance metrics weekly.
- Alerts set up for **model drift** detection.
- Retraining strategy involves weekly data refresh and retraining every quarter.

6. Business Impact

Key Business Outcomes:

- **Inventory Optimization**: Forecasts help maintain optimal stock levels, avoiding overstocking and understocking.
- **Marketing Efficiency**: Timing promotions based on demand peaks.
- **Revenue Growth**: Data-backed decisions improve profitability.
- **Customer Satisfaction**: Ensures product availability during peak demand.

7. Challenges and Solutions

CHALLENGE	SOLUTION
HIGH VARIANCE ACROSS STORES	Store-level segmentation
IRREGULAR SEASONALITY	Lag features and holiday flags
SPARSE DATA IN CERTAIN CATEGORIES	Data aggregation and smoothing
REAL-TIME PREDICTION DEMANDS	API deployment via Streamlit/Flask

8. Future Improvements

- Add **macroeconomic indicators** (e.g., inflation, unemployment).
- Integrate **competitor pricing** if available.
- Experiment with **DeepAR**, **LSTM**, and **Transformer-based** forecasting.

- Build **multi-step forecasts** and confidence intervals.

9. Conclusion

This project demonstrates a comprehensive end-to-end data science pipeline, from raw data to deployed forecasting system. By integrating classical time series techniques with machine learning and modern MLOps tools, we created a scalable and impactful system for sales forecasting in retail. The methodology used here is extendable to many other demand forecasting problems in retail, supply chain, and finance.