Store Sales Forecasting

Business-Driven Time Series Forecasting

Business Problem

INABILITY TO ANTICIPATE DEMAND SPIKES LEADING TO LAST-MINUTE OPERATIONAL DECISION MAKING

REVENUE IS LOST DURING PEAK PERIODS DUE TO DELAYED STAFFING AND INVENTORY DECISIONS

DECISION-MAKING REMAINS REACTIVE, INCREASING RISK AND OPERATIONAL INEFFICIENCY

Business Goal & KPIs

Enable proactive data-driven retail operation by accurately forecasting demand at the store-department level allowing the business to anticipate sales spikes optimize staffing and inventory decisions, and maximize revenue and profit during peak and holiday periods.

Forecast accuracy improvements, lost sales reduction, reactive action reduction, forecast adoption rate.

KPIs & Metrics

Forecast Accuracy requirements: 85% or above

Metric of evaluation: MAPE (Mean Average Precision Error)

Lost Sales reduction: Decrease lost sales during peak periods by 15 to 20%

Reaction Decision reduction: reduce last-minute operational decision makings by 15%

Project Scope

Focus on store and department level demand forecasting for peak periods, promotional events and holidays.

Constraints: Limited historical data of previous 3 years in all stores.

Design Steps

- **Data Acquisition & Storage:**Ingested historical sales, store, and product datasets from Kaggle (CSV) and organized them in a structured local repository to ensure version control and reproducibility.
- **Database Ingestion:**Designed PostgreSQL schemas and tables, loaded CSV data via SQL scripts, and validated ingestion through row counts, null checks, and duplicate detection.

Design Steps

Data Cleaning & Transformation: Standardized column names and data types, handled missing and inconsistent records using SQL and Python, and prepared clean, analysis-ready tables.

EDA: Analyzed trends, seasonality, and anomalies using Python (Pandas, Matplotlib, Seaborn) to identify key demand drivers and guide feature engineering.

Design Steps

Feature Engineering & Aggregation:
Engineered derived metrics and weekly/monthly aggregates in SQL, storing transformed feature tables in PostgreSQL for forecasting.

Forecasting & Analysis Preparation:
Prepared SQL-derived datasets for Python-based forecasting models, implemented train/validation/test splits, and automated feature refresh workflows.

Design Steps

Reporting & Visualization: Designed dashboards in Tableau, Power BI, or Streamlit connected to PostgreSQL and automated summary exports for management and operations teams

Documentations & Governance: Documented database schemas, ETL pipelines, validation checks, and EDA insights to ensure transparency, reproducibility, and future scalability.

Feature Engineering

Trained models per store-department combination

Prophet emerged as top-performing model

Hyperparameter tuning focused on stability

Feature table

Feature Category	Feature Name	Time Context	Business Rationale
Temporal	Sales (t-1)	1 week lag	Captures short-term demand momentum
Temporal	Sales (t-2)	2 week lag	Accounts for delayed customer behavior
Temporal	Sales (t-4)	4 week lag	Captures monthly seasonality
Temporal	Sales (t-12)	12 week lag	Captures annual seasonal patterns
Temporal	Rolling Mean (4 weeks)	Historical window	Smooths weekly demand fluctuations
Temporal	Rolling Mean (12 weeks)	Historical window	Captures medium-term trend
Temporal	Rolling Std Dev (4 weeks)	Historical window	Measures short-term volatility
Temporal	Rolling Std Dev (12 weeks)	Historical window	Measures long-term demand instability
Event-Based	Holiday Flag	Event week	Identifies demand spikes due to holidays
Event-Based	Holiday Lead Indicator	1–2 weeks before	Captures pre-holiday demand buildup
Event-Based	Holiday Lag Indicator	1–2 weeks after	Captures post-holiday demand decay
External Driver	CPI	Weekly	Reflects inflation impact on purchasing power
External Driver	Fuel Price	Weekly	Proxy for customer mobility & store visits
External Driver	Unemployment Rate	Weekly	Indicator of consumer confidence
Interaction	Store × Department	Static	Captures localized demand behavior

Model Development

•**Candidate Models:**

1. **ARIMA**: Captures autoregressive trends and moving averages
2. **SARIMA**: Extends ARIMA with seasonal effects (weekly/annual patterns)
3. **Prophet**: Handles multiple seasonality, holidays, and trend changes automatically

• **Modeling Granularity**: Forecasting performed at **Store × Department level** to capture local demand patterns

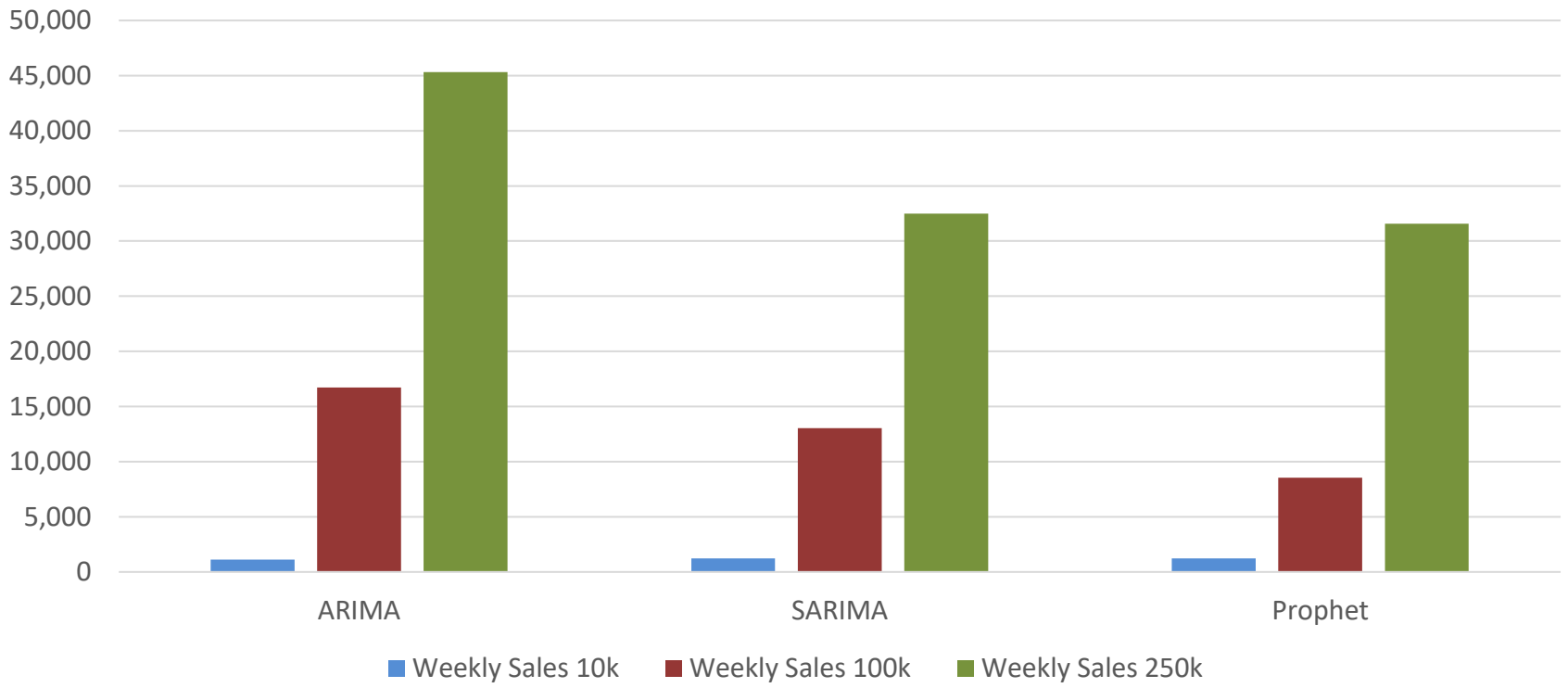
•Time-series split (train/test by chronological order)

• **Evaluation Metrics**: **MAPE**

•**Model Selection Rationale**

•Compare candidate models based on **accuracy, stability,** Choose model that **balances predictive performance with interpretability**

Model Performances (MAPE values)



Evaluation Results

Significant error reduction vs naïve baseline

ARIMA: 1,128–45,331 units/week

SARIMA: 1,227–32,488 units/week

Prophet: 1,225–31,569 units/week

SARIMA and Prophet outperform ARIMA, especially for higher-volume departments.

Consistent performance on high-volume departments

Prophet shows the **lowest absolute error** for 100k–250k weekly sales, making forecasts more reliable for large stores.

Lower variance during peak holiday periods

Prophet and SARIMA maintain more stable predictions during high-demand weeks compared to ARIMA, reducing risk in inventory planning.

Business Impact

REDUCED OVERSTOCK AND STOCK-OUT RISK

INCREASED SALE ACCUMULATION ACROSS STORES

LESS REACTIVE OPERATIONAL DECISION BEING MADE ACROSS THE STORES