SALES FORECASTING FOR RETAIL STORES

A Project Report Submitted by

Manaswini

Data Science Project

ABSTRACT

This project presents a comprehensive approach to forecasting retail store sales using time-series analysis and machine learning techniques. Accurate sales forecasts enable retailers to optimize inventory, reduce stockouts, and improve profitability. The project demonstrates data preprocessing, feature engineering, and model comparison (ARIMA/Prophet, XGBoost, and LSTM). Evaluation metrics such as MAE and RMSE are used to determine model performance. The final deliverable includes recommendations and a dashboard-ready forecasting pipeline.

INTRODUCTION

Sales forecasting is a critical business function that predicts future demand for products at store or SKU level. Modern retailers rely on robust forecasting models to make informed inventory and procurement decisions. This project focuses on building and evaluating multiple forecasting models on historical sales data and identifying the best-performing approach for retail use-cases.

PROBLEM STATEMENT

Retailers face the challenge of accurately predicting sales due to seasonality, promotions, holidays, and external factors like weather. The objective is to create a predictive model that captures temporal patterns and external influences to produce reliable short-term sales forecasts.

OBJECTIVES

- To preprocess and analyze retail sales time-series data.
- To engineer useful time-based and external features (lags, rolling statistics, promotions, holidays).
- To build and compare ARIMA/Prophet, XGBoost, and LSTM models for forecasting.
- To evaluate models using MAE and RMSE and recommend the best approach.
- To outline a deployment strategy for integration into retail operations.

LITERATURE REVIEW

Time-series forecasting has evolved from classical statistical models like ARIMA to powerful machine learning and deep learning approaches. Prophet (by Facebook) simplifies trend and seasonality modeling. Gradient boosting methods (XGBoost) handle feature-rich datasets, while LSTM networks capture long-range temporal dependencies. Studies show hybrid approaches often outperform single-model baselines in retail forecasting tasks.

METHODOLOGY

The methodology comprises data collection, preprocessing, exploratory analysis, feature engineering, model development, evaluation, and deployment planning. 1. Data Collection: Use retail sales datasets (e.g., Kaggle Retail Sales Forecasting or Walmart dataset). 2. Preprocessing: Handle missing values, aggregate sales at the required granularity, convert dates to datetime objects, and create train/test splits. 3. Feature Engineering: Create lag features, rolling means, day-of-week, month, promo flags, and holiday indicators. 4. Modeling: Fit ARIMA/Prophet for baseline; use XGBoost for feature-driven modeling; develop LSTM for sequence learning. 5. Evaluation: Compute MAE and RMSE; perform cross-validation for robustness.

DATASET DESCRIPTION

A typical dataset used in this project contains the following columns: Date, Store_ID, Item_ID (optional), Sales, IsHoliday, Promotion, and other external variables such as Temperature and Events. For demonstration, many public datasets provide multiple years of daily/weekly sales, which are suitable for training forecasting models.

IMPLEMENTATION STEPS

The following steps outline an implementation pipeline (code snippets are illustrative):

• **1. Data Loading & Inspection**

Load CSV data, parse dates, and inspect missing values and basic statistics.

• **2. Preprocessing**

Aggregate sales, fill or drop missing entries, and encode categorical variables.

• **3. Feature Engineering**

Create lag features (t-1, t-7), rolling windows (7-day mean), and calendar features.

• **4. Modeling**

Train ARIMA/Prophet for baseline; prepare feature matrix for XGBoost; scale and shape sequences for LSTM.

• **5. Evaluation**

Predict on test set and compute MAE, RMSE, and visualize actual vs predicted.

• **6. Deployment**

Save model artifacts and prepare a dashboard (e.g., Streamlit) for stakeholders.

CODE SNIPPETS (ILLUSTRATIVE)

Example: Creating lag features (pandas) data['lag_1'] = data['Sales'].shift(1) data['rolling_7'] = data['Sales'].rolling(window=7).mean().shift(1) # Example: XGBoost training import xgboost as xgb model = xgb.XGBRegressor(n_estimators=500, learning_rate=0.05) model.fit(X_train, y_train) # Example: LSTM (Keras) from tensorflow.keras.models import Sequential from tensorflow.keras.layers import LSTM, Dense model = Sequential([LSTM(64, input_shape=(timesteps, features)), Dense(1)]) model.compile(optimizer='adam', loss='mse') model.fit(X_seq_train, y_seq_train, epochs=30, batch_size=32)

MODEL EVALUATION

As an example, the following evaluation results were observed on a hold-out test set (illustrative values): • ARIMA: MAE = 210.5, RMSE = 320.7 • Prophet: MAE = 198.3, RMSE = 305.2 • XGBoost: MAE = 145.0, RMSE = 220.4 • LSTM: MAE = 130.6, RMSE = 205.8 The LSTM model achieved the best RMSE in this example, indicating stronger capability to capture temporal dynamics when sufficient data and tuning are available.

RESULTS SUMMARY

Model	MAE	RMSE
ARIMA	210.5	320.7
Prophet	198.3	305.2
XGBoost	145.0	220.4
LSTM	130.6	205.8

DISCUSSION

Model choice depends on data availability, interpretability needs, and computational resources. Tree-based methods like XGBoost provide competitive performance and are easier to interpret with SHAP values, while LSTM models can capture complex temporal dependencies but require more data and tuning. Combining models through ensembling or blending can yield further improvements.

DEPLOYMENT PLAN

1. Export model artifacts (pickle for XGBoost, SavedModel for LSTM). 2. Create a REST API (Flask/FastAPI) to serve forecasts. 3. Build a lightweight dashboard (Streamlit/Plotly Dash) for visualization and refresh schedules. 4. Monitor model drift and retrain periodically using rolling windows.

CONCLUSION AND FUTURE SCOPE

This project outlines robust methods for retail sales forecasting, demonstrating both statistical and machine learning approaches. For practical deployment, feature engineering, external regressors (holidays, weather), and continuous monitoring are essential. Future work includes building an automated retraining pipeline, integrating real-time sales streams, and exploring advanced transformer-based time-series models.

REFERENCES

1. Taylor, S. J., & Letham, B. (2018). Forecasting at scale (Prophet).
2. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system.
3. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory.
4. Kaggle Retail Datasets: Various public datasets for sales forecasting.