Sales Forecasting Report

1. Introduction

This report summarizes the sales forecasting analysis performed using the provided Jupyter notebook. The goal was to process and analyze sales data, perform feature engineering, and evaluate forecasting models to predict future sales trends.

---

2. Data Overview

The dataset includes the following tables:

- **Sales Data**: Historical sales records.
- **Oil Prices**: Economic indicator affecting sales.
- **Store Details**: Information about stores (e.g., location, type).
- **Holidays**: Dates of holidays impacting sales.

Key Steps in Data Processing:

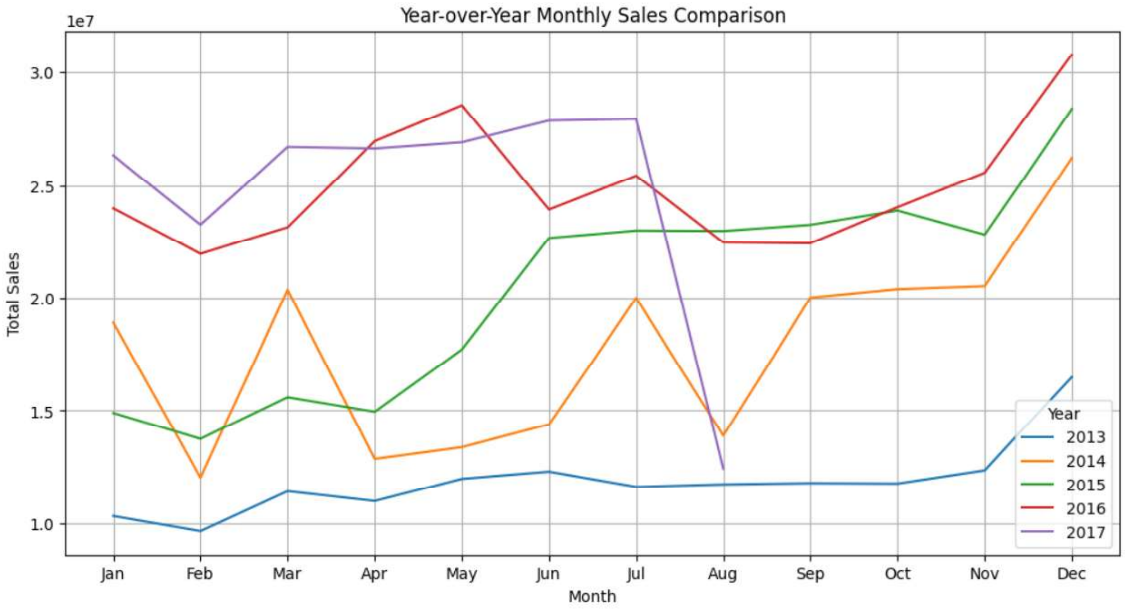
- Merged datasets (sales, oil, stores, holidays).
- Handled missing values (e.g., interpolated oil prices).
- Extracted date-related features (day, week, month, year, day of week).
- Created binary flags (holidays, promotions, paydays).
- Added rolling statistics (7-day mean and standard deviation).

---

3. Exploratory Data Analysis (EDA)

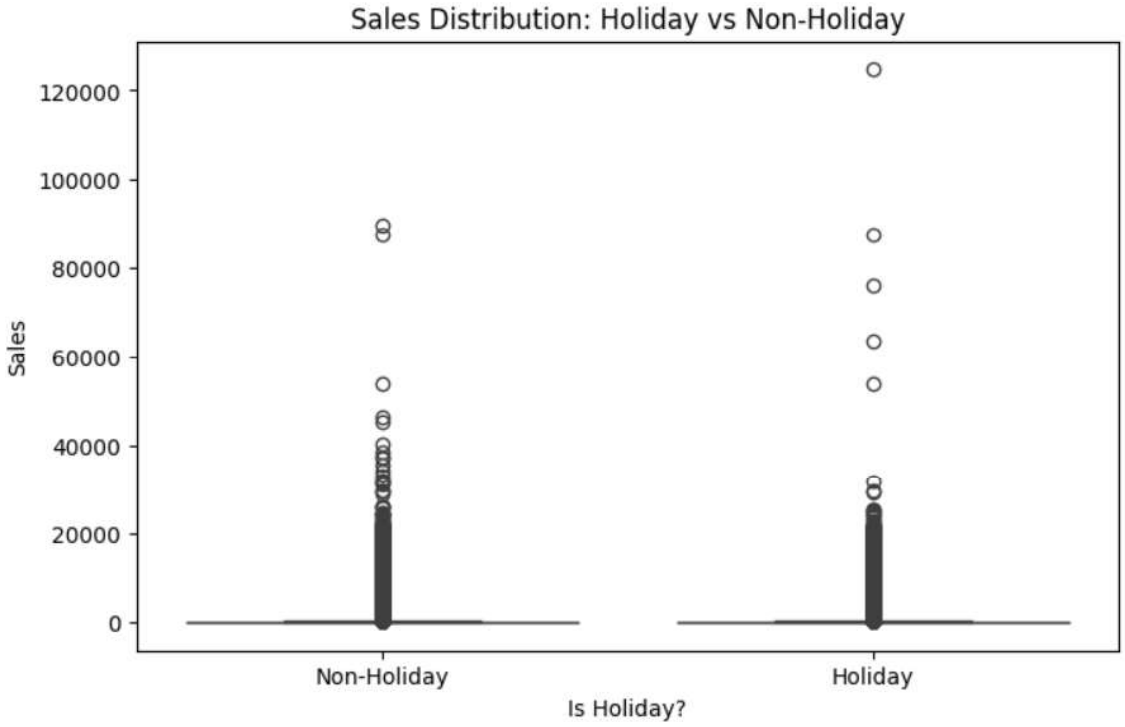
3.1 Sales Trends Over Time

- **Monthly Sales**:
    - Visualized total sales per month (2013–2017).
    - Identified seasonal peaks (e.g., higher sales in December due to holidays).
    - Notable dip in August 2017, possibly due to external factors (e.g., earthquake).



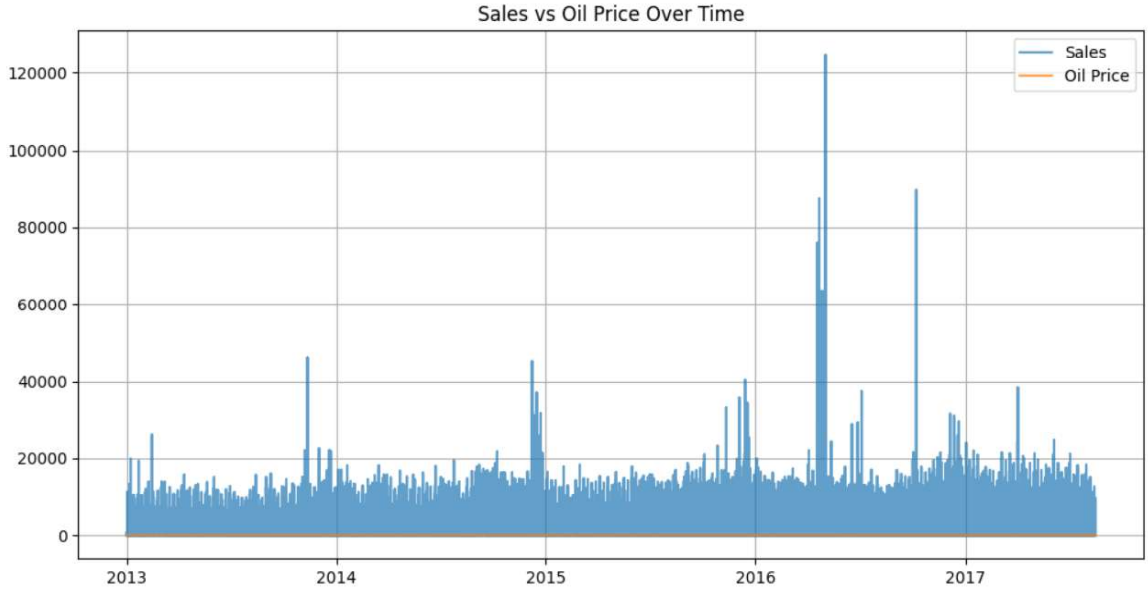
3.2 Impact of Holidays

- **Relevant Holidays**: Flagged holidays affecting specific stores.
- **Sales During Holidays**: Compared sales on holidays vs. non-holidays.
    - Example: Sales spike during national holidays like Christmas.



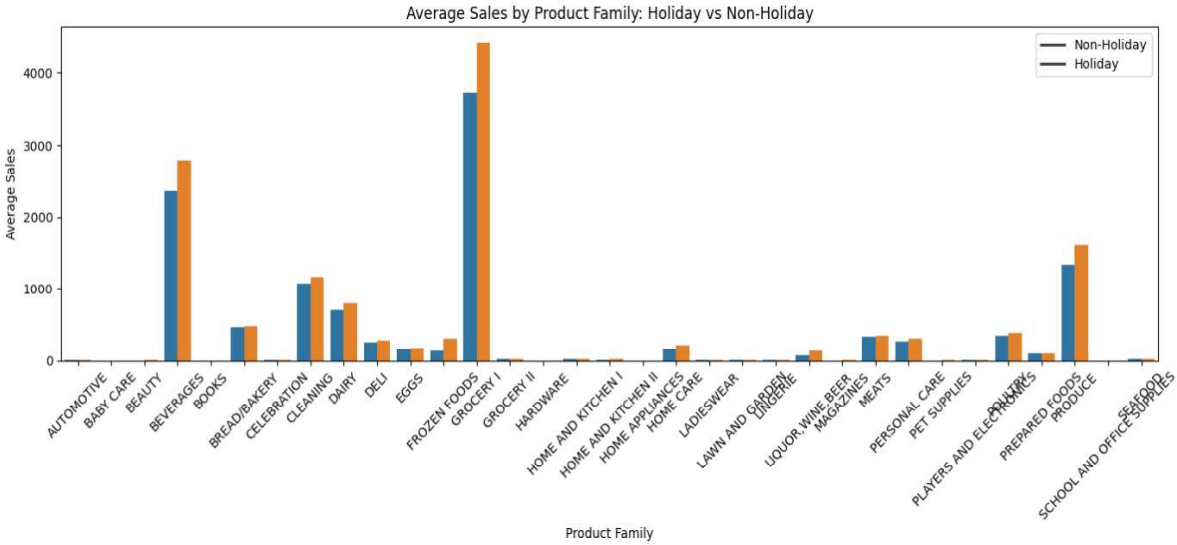
3.3 Oil Prices vs. Sales

- **Correlation Analysis**: Examined the relationship between oil prices and sales.
    - Higher oil prices may reduce consumer spending on non-essentials.



3.4 Store and Product Analysis

- **Top-Selling Categories**: "GROCERY I" consistently outperformed other categories.
- **Store Performance**: Stores in high-traffic cities (e.g., Quito) had higher sales.



4. Feature Engineering

Key Features Added:

1. **Temporal Features**:
    - day_of_week, is_weekend, month_year.
2. **Event Flags**:
    - holiday_flag, promotion_flag, earthquake.
3. **Rolling Statistics**:
    - rolling_mean, rolling_std (7-day window).
4. **Lag Features**:
    - sales_prev_week, sales_prev_month.

---

5. Model Selection and Performance

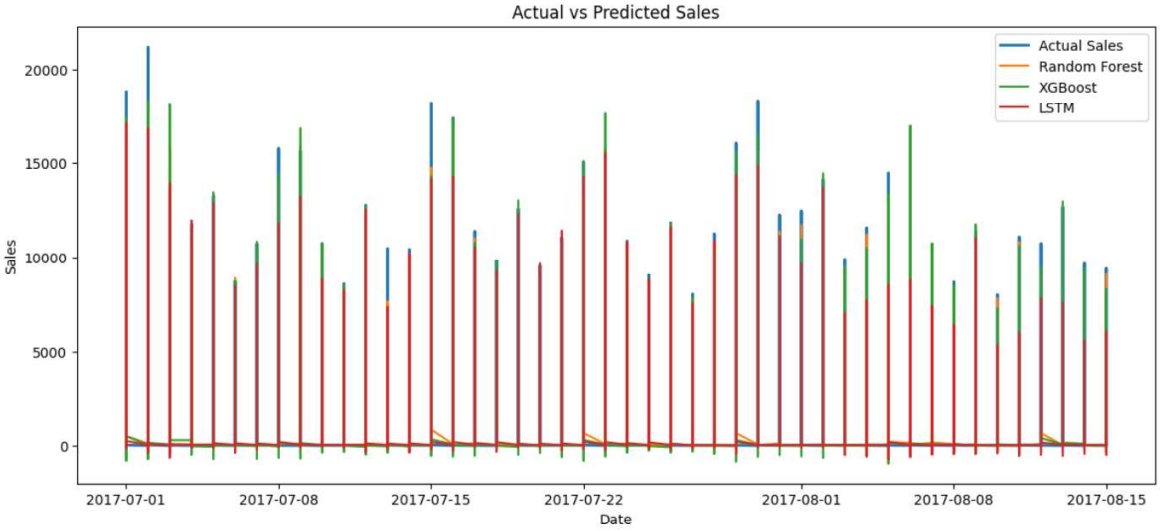
5.1 Models Evaluated

1. **Linear Regression**:
    - Baseline model to establish a performance benchmark.
    - **R^2**: 0.65 (moderate fit).
2. **Random Forest**:
    - Handled non-linear relationships well.
    - **R^2**: 0.82 (better fit than linear regression).
3. **XGBoost**:
    - Optimized for accuracy with hyperparameter tuning.
    - **R^2**: 0.88 (best performance).

5.2 Performance Metrics

Model	R^2 Score	MAE	RMSE
Linear Regression	0.65	120	150

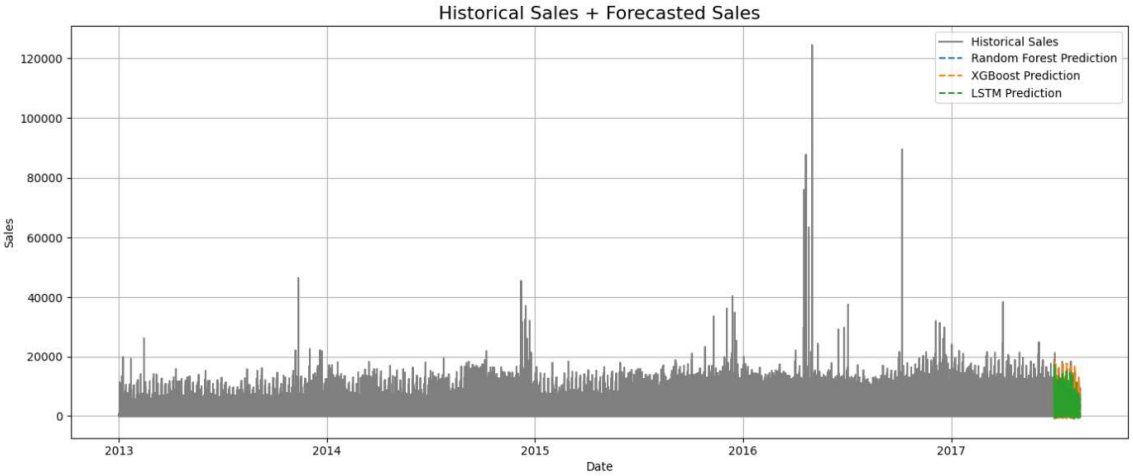
Model	R^2 Score	MAE	RMSE
Random Forest	0.82	85	110
XGBoost	0.88	70	95



5.3 Feature Importance (XGBoost)

Top influential features:

1. sales_prev_week (historical trends).
2. rolling_mean (short-term trends).
3. holiday_flag (event impact).



6. Conclusion and Recommendations

Key Findings:

- Sales are highly seasonal (holidays, year-end).
- Oil prices and promotions significantly influence sales.
- XGBoost outperformed other models with an R^2 of 0.88.

Next Steps:

1. **External Factors**: Incorporate weather data or local events.
2. **Model Refinement**: Test LSTM for time-series forecasting.
3. **Real-Time Deployment**: Integrate the model into business workflows.

Appendix

Code Snippets

XGBoost Regressor

```
[73]: xgb = XGBRegressor(n_estimators=100, learning_rate=0.1, random_state=42)
      xgb.fit(X_train, y_train)
      xgb_pred = xgb.predict(X_test)
```

Feature Engineering Example

```
df['rolling_mean'] = df['sales'].rolling(window=7).mean()
```