Time Series Forecasting of Store Sales

Team G12: Shagufta Anjum Ajitesh Nair Varsha Chikkamagaluru Lakshmikanth

I. Abstract

This project applies data mining techniques to forecast retail store sales. We perform comprehensive data exploration to understand patterns in the data, followed by applying statistical, machine learning, and deep learning models to predict sales. The goal is to improve forecasting accuracy, thereby optimizing inventory management in retail operations.

II. Introduction

Inspired by a Kaggle competition, [1], our aim is to predict future sales of thousands of product families across different retail locations based on historical data. The dataset, provided by Corporación Favorita, contains various influencing factors such as promotions, holidays, and macroeconomic indicators. The goal is to develop predictive models to handle the complexities of time series data to provide accurate sales forecasts, thus aiding in inventory management.

III. Motivation

Accurate sales forecasting is crucial in retail for effective inventory management, cost reduction, and improved customer satisfaction [2]. Motivated by these needs, this project investigates time series forecasting methods, leveraging statistical, machine learning, and deep learning models [4]. The integration of external factors, such as oil prices and holidays highlights the importance of exogenous variables in enhancing forecast precision [3].

IV. Related work

Retail sales forecasting has leveraged traditional models like ARIMA and SARIMA to capture trends and seasonality [3]. Machine learning models, including Gradient Boosting and Random Forest, effectively handle non-linear relationships and external factors like promotions, often outperforming statistical methods [4]. Deep learning models, such as LSTMs, N-BEATS [5], and Temporal Convolutional Networks [6] capture complex temporal dependencies with promising results [7]. Recent studies highlight the effectiveness of hybrid approaches that combine traditional and machine learning techniques for enhanced forecasting accuracy [8].

V. Methodology

A. Data Cleaning and Preprocessing

We addressed missing values in oil price data using backward filling method, leveraging the assumption that oil prices on adjacent days are generally similar. We removed all 'transferred' holidays, which are holidays observed on regular workdays to compensate for weekends. Since these do not reflect actual non-working days, including them could introduce biases into the model. Lastly, we merged multiple datasets to create a unified and comprehensive dataset for modeling. Sales data was joined with holiday data, oil prices, and transaction data using appropriate join keys, such as store identifiers and dates.

B. Hypothesis Testing

To assess the impact of external factors such as oil prices and holidays on sales, we conducted hypothesis testing to validate their significance. The null hypothesis for each test stated that there is no relationship between external factor and sales, while the alternative hypothesis proposed a significant impact. Using a significance level (α) of 0.05, the test lead us to reject the null hypotheses, confirming that both oil prices and holidays significantly affect sales.

C. Exploratory Data Analysis

Correlation Analysis: Correlation heat map (Figure 1) shows promotions positively correlates with sales, suggesting promotions boost sales. Regression plots (Figure 2) reveal key correlations: Sales vs. Oil Prices shows a negative trend, suggesting higher oil prices may reduce consumer spending or raise costs, lowering sales. Sales vs. Promotions and Sales vs. Transactions display positive correlations, indicating that promotions effectively drive revenue and higher transaction volumes boost sales. These insights highlight influence of economic factors, promotional strategies on retail sales performance.

Temporal analysis: Daily, weekly and monthly sales patterns were also analyzed to identify impact

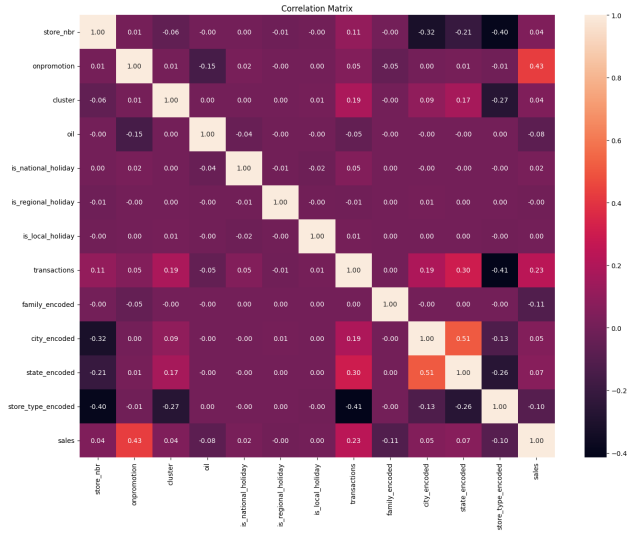

Fig. 1: Correlation Matrix which visually represents the strength and direction of correlation between each of the features and the target variable.

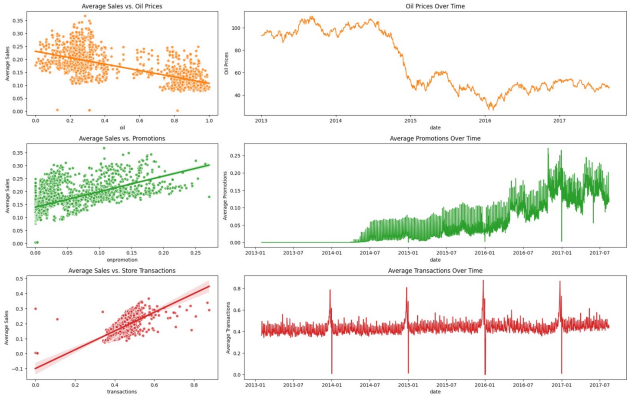

Fig. 2: Teft column shows correlation plots of average sales versus oil prices (negative correlation), promotions (positive correlation), and store transactions (strong positive correlation). The right column shows the oil prices, average promotions and average store transactions vary over time from 2013 to 2017.

of temporal components. These sales patterns reveal that sales increase year-over-year, with peaks in November and December due to holiday shopping, and higher sales on weekends compared to weekdays.

Holidays Impact: Analyzing holiday data through filtering and categorizing revealed that events like Labor Day, Christmas, and Independence Day significantly boost consumer spending. National holidays generally have a higher impact on sales compared to regional or local ones. Additionally, 2016 earthquake also caused an increase in sales.

D. Analyzing Trends and Seasonality

Understanding trends and seasonality is crucial for forecasting sales data, as these components represent long-term patterns and recurring cycles. To identify these patterns, we used following techniques:

1) **Simple Moving Average**: Smooths out short-term fluctuations to show longer-term trends.
2) **Exponential Moving Average**: Like SMA, but gives more weight to recent observations, making it more responsive to dynamic changes.
3) **Hodrick-Prescott Filter**: Separates the data into a trend and a cyclical component by minimizing the variance of the difference between the observed data and the estimated trend, with a penalty for changes in the trend's growth rate.
4) **Additive Decomposition:** Breaks the time series into trend, seasonality, and residuals such that these components can be added to obtain the original series.
5) **Seasonal-Trend Decomposition using LOESS** (STL): Decomposes a time series into seasonal, trend, and residual components using LOESS, a technique for smoothing data using locally weighted regression.
6) **Fourier Transform**: Converts the time series from the time domain into the frequency domain, allowing us to decompose the data into a spectrum of sinusoidal components from which we can identify dominant frequencies that corresponded to recurring cycles.

Figure 3 shows trends identified by various methods, highlighting clear seasonality with periodic sales spikes, especially around year-end. Figure 4 displays STL decomposition, revealing annual peaks during high-demand times like Christmas, with residuals indicating irregular, short-term sales anomalies.

E. Feature Engineering

We used several techniques to generate meaningful features from existing data to capture additional temporal relationships such as seasonality and trends, as such features might improve the predictive power of the models. Below, we detail the techniques used:

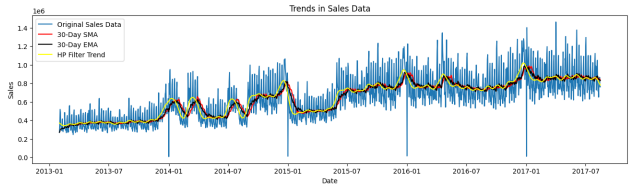

Fig. 3: The blue line represents the original sales data and three smoothing techniques are applied: a 30-Day Simple Moving Average (red line), a 30-Day Exponential Moving Average (black line), and a Hodrick-Prescott Filter trend (yellow line). The overall trend indicates a gradual increase in sales with notable seasonal fluctuations.

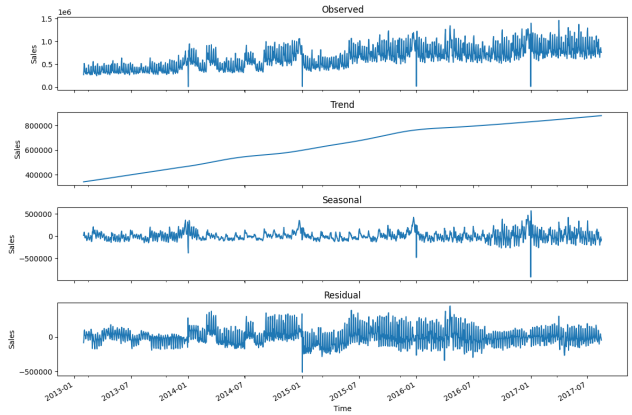

Fig. 4: Decomposition of sales data into trend, seasonality and residual components using STL with LOESS. An upward trend and annual seasonality can be observed.

1) **Date-based features**: We derived a variety of features from date column, such as *'Day of the Week'*, *'Day of the Month'*, *'Is Weekend'*, *'Month'*, *'Year'*, etc. to model various short-term and long-term trends in sales and unique patterns due to changing consumer behavior.
2) **Lagged Sales**: These are created by shifting sales data by a fixed intervals. A lag of 1 day captures day-to-day variability, and lags of 7, 14, and 30 days capture weekly, bi-weekly, and monthly patterns, respectively, which are important for understanding periodicity in sales. By providing these lagged values, model could learn from past trends to make future predictions.
3) **Lagged Oil Prices**: Increased oil prices might lead to higher transportation costs, which could influence product pricing and sales with a delay. Changes in oil prices might also affect disposable income and consumer spending patterns. Lags of 1, 7, and 30 days for oil prices can help capture delayed effects on sales.
4) **Lagged Holidays**: Consumers may shop more in preparation for the holiday, and there could be reduced sales after the holiday due to prior stocking up. Sales might show delayed effects surrounding holidays, which can be captured by lags in the holidays.
5) **Rolling Averages**: These are average sales calculated over a moving window of data to smooth out daily fluctuations and capture longer-term trends. We used the rolling average of sales over periods of 7, 14, 30 days.
6) **Seasonal Components**: The seasonality and trend components of sales data described in the *Section D* extracted using decomposition techniques can be used as features. Trend accounts for long-term movements in sales, while seasonality accounts for recurrent patterns due to cycles. By explicitly providing these components, the model doesn't have to infer them.

Creation of lagged and rolling features inherently leaves missing values at the beginning of the dataset where there is insufficient historical data to compute these features. To address this, we imputed them the median, which ensures that the imputed values are representative of each group's distribution.

Additionally, categorical variables like store number and product family were label-encoded to convert them into numerical format for the models.

F. Feature Selection

We used several techniques to select the optimal features to use for training the model. Below, we detail the methods used:

1) **Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)** : These are used to find optimal lag features. ACF measures the correlation of a time series with its own lagged values, while PACF isolates the direct correlation at each lag by removing the influence of intermediate lags.
   We analyzed ACF and PACF plots for each product family to identify significant lags corresponding to significant spikes in autocorrelation, based on a threshold (±0.6). Figure 5

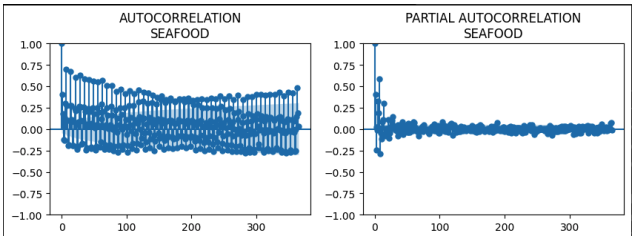

Fig. 5: ACF and PACF plots for the 'Seafood' family, highlighting significant lags and seasonal patterns in the time series. The dots represent correlation values and the shaded region indicates significance levels.

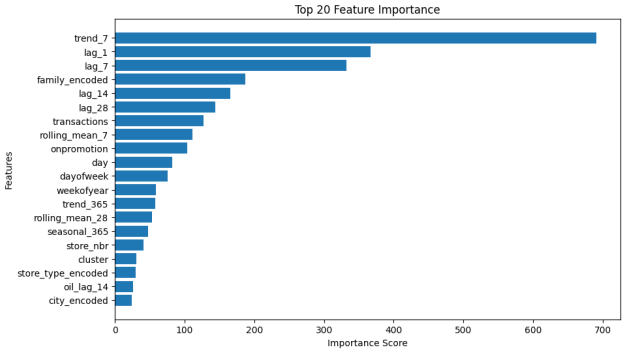

Fig. 6: The top 20 features that are most highly correlated with the sales data, extracted using LightGBM.

shows the the ACF and PACF plots for the "Seafood" family with high autocorrelations at lags such as 1, 7, 14, and 28 days. An analysis was performed across all families to determine the most meaningful lag periods that capture patterns and seasonality, which were used for lag feature engineering described in *Section E*.

2) **LightGBM Feature Importance**: LightGBM is a gradient boosting framework that can rank features based on their contribution to the model's predictive performance. By training an initial LightGBM model, we extracted feature importance scores. This method can handle high-dimensional data, non-linear relationships, and correlated features, which are common in time series. Figure 6 shows some of the highest correlation features ordered by their importance scores.
3) **Correlation matrix**: We used a correlation matrix plot (Figure 1) to analyze feature relationships and identify redundancies. Using the pairwise correlation coefficients between numerical variables, we identified features strongly correlated with sales or to each other. Features with excessive multicollinearity, which can distort model performance and interpretability, were removed. From the remaining features, the ones with highest correlation (positive or negative) to sales were chosen.

Combining the above methods, we selected a final set of 25 features that showed the strongest predictive performance and relevance to sales forecasting.

G. Models Used

Statistical Models:

- **ARIMA** (Autoregressive Integrated Moving Average) [9]: Used for univariate time series forecasting to capture trends and seasonality while accounting for autocorrelation in the data.
- **SARIMAX** (Seasonal AutoRegressive Integrated Moving Average with Exogenous Regressors) [9]: An extension of ARIMA that incorporates both seasonal and non-seasonal patterns, as well as external factors to capture complex trends and recurring cycles.
- **VECM** (Vector Error Correction Model) [10]: A statistical model for multivariate time-series with cointegrated variables, capturing short-term dynamics and long-term relationships for interdependent data, making it ideal for time-series scenarios influenced by trends.
- **VAR** (Vector Autoregression) [10]: Designed for multivariate time series, it can model the interdependencies among multiple variables, making it suitable for time series where the features influence each other.
- **Prophet** [16]: A robust forecasting tool developed by Facebook, it excels in handling irregular time series with strong seasonality, providing interpretable trend and seasonality components.

Machine Learning Models:

- **Linear Regression**: A simple and interpretable model that assumes a linear relationship between the features and target variable. For time series, it can be used to model trends by incorporating lagged values and seasonality indicators. It can be a good baseline model, but struggles with non-linear trends.

- **Random Forest** [13]: An ensemble method that averages the predictions of multiple decision trees and reduces overfitting. For time series forecasting, it can effectively capture non-linear relationships between lagged features and external variables and works well for datasets with a mix of categorical and continuous features.
- **Support Vector Machine** (SVM): Aims to optimize classification boundaries through hyperplane construction in a high-dimensional space. For time series forecasting, SVMs with a kernel trick (e.g., RBF) can model non-linear patterns in the data but are sensitive to feature scaling.
- **XGBoost** [11](Extreme Gradient Boosting): A tree-based ensemble model that optimizes gradient boosting algorithms. It is highly efficient and effective for capturing non-linear relationships in time series data by learning from lagged features and external covariates. Its regularization techniques (L1 and L2) help prevent overfitting.
- **LightGBM** [12] (Light Gradient Boosting Machine): A gradient boosting framework designed for efficiency and scalability. It splits trees leaf-wise, which often results in better accuracy than level-wise approaches like XGBoost, especially for datasets with large feature spaces. For time series forecasting, it is well-suited for handling large datasets and features generated from lagged observations, trends, and seasonality.

Deep Learning Models:

- **Recurrent Neural Network** (RNN) [17]: A neural network designed to handle sequential data by maintaining a hidden memory state that evolves over time. Through recurrent connections, it propagates information from previous timesteps to influence current predictions. It may struggle with long-term memory due to vanishing gradients.
- **Long Short-Term Memory** (LSTM) [15]: A specialized form of RNNs designed to model both short-term and long-term dependencies in sequential data. It uses gated mechanisms to control which information to store, update, or discard, enabling it to capture temporal dependencies across varying time scales.
- **Temporal Fusion Transformer** (TFT) [18]: A neural network architecture tailored for multi-horizon forecasting that combines attention mechanisms with interpretable components to model both static and temporal covariates. It offers both performance and explainability for time-series tasks.
- **Temporal Convolutional Network** (TCN) [6]: A convolutional neural network architecture designed for analyzing time series data by employing a hierarchy of temporal convolutions, dilated convolutions, and pooling layers to effectively process sequential information.
- **N-beats** (Neural Basis Expansion Analysis for Interpretable Time Series forecasting) [5]: A deep learning architecture for time series forecasting that uses basis expansion to learn complex patterns, offering both performance and interpretability without relying on time-series-specific components. However, it may require large datasets, significant computational resources, and extensive hyperparameter tuning.

VI. Empirical Results

A. Model Training

1) **Splitting the data**: For time series data, splitting data into training and test sets must account for temporal structure to avoid data leakage, ensuring future observations are never used to predict past or present ones. Instead of shuffling the data, dataset is split chronologically. We chose the split ratio 80-20 (80% used for training and validation, 20% used for testing).
2) **Scaling the data**: Before fitting the models, we applied the Standard Scaler to standardize the features, ensuring that all have a mean of 0 and a standard deviation of 1. This step is particularly beneficial for models like LightGBM and Neural Networks, which are sensitive to feature magnitudes. To prevent data leakage, the scaler was fit only on the training data, and the same transformation was subsequently applied to the test data for consistency.
3) **Time Series Cross-Validation** (TSCV) was used for validation to evaluate our model performance for various hyperparameters while accounting for the temporal dependencies inherent in the data. TSCV ensures that model is trained on past data and tested on future, unseen observations, thus preventing data leakage. Each iteration of TSCV incrementally expands training set to include more historical

data, while test set remains a non-overlapping segment of future observations such that model is trained on progressively larger histories.

B. Model Evaluation

For evaluation, we chose the Root Mean Squared Logarithmic Error (RMSLE) metric as it aligns well with the characteristics of sales data. Sales data exhibit occasional spikes or dips, and RMSLE effectively mitigates the influence of these outliers by applying a logarithmic transformation, which reduces the impact of large absolute errors and ensures that the model's performance is not overly penalized for extreme values. Additionally, RMSLE inherently penalizes under-predictions more heavily than over-predictions, which is particularly relevant in sales forecasting where underestimating demand can lead to missed revenue opportunities.

The RMSLE formula is defined as:

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(y_i+1) - \log(\hat{y}_i+1))^2}$$

where y_i is the actual observed value and $\hat{y}_i$ is the predicted value for the i-th data point. The addition of 1 within the logarithmic function ensures that zero values, common in sales data, are handled gracefully. RMSLE ranges from 0 to positive infinity, with lower values indicating better model performance.

C. Hyperparameter Tuning

Grid Search was employed to identify the best hyperparameters for the machine learning and deep learning models used. By exhaustively searching over a predefined grid of hyperparameter values, grid search evaluates all possible combinations to find the optimal one. Time-series cross-validation was used during the search to evaluate each parameter combination on multiple folds (between 3 and 5), preserving the temporal structure of the data and reducing the risk of overfitting to a single split. RMSLE scoring metric was utilized to tailor the optimization process. The resulting tuned models demonstrated significant improvements in predictive performance compared to their default settings.

D. Results and Discussion

See Table I for RMSLE obtained for each model.

XGBoost has the best performance with lowest RMSLE of 0.0163, followed by LightGBM with 0.0389.

Type	Model	Test RMSLE
Statistical	ARIMA	3.1719
	SARIMAX	0.4718
	VECM	0.3088
	VAR	2.2050
	Prophet	0.6101
Machine Learning	Linear Regression	0.0793
	Random Forest	0.6182
	SVM RBF Kernel	2.1368
	XGBoost	0.0163
	LightGBM	0.0389
Deep Learning	RNN	0.0592
	LSTM	0.5544
	TFT	0.4970
	TCN	0.6624
	N-beats	2.2333

TABLE I: RMSLE scores of predictive models categorized by model type.

Statistical models: VECM and SARIMAX show moderate performance due to their ability to handle seasonal patterns and cointegration relationships. ARIMA and VAR show poor performance, which could be due to their inability to capture complex non-linear patterns and handle multiple variables.

Machine Learning Models: XGBoost emerges as the overall best model, likely due to good handling of non-linear relationships and gradient boosting's ability to reduce bias and variance. LightGBM also performs well for similar reasons, and Random Forest performs moderately. Linear Regression does moderately due to underfitting. SVM struggles, possibly due to ineffective parameter tuning.

Deep Learning Models: RNN shows good results, likely due to its natural ability to handle sequential data and memory of past values. LSTM, TFT, and TCN show moderate performance, but worse than RNN, and N-beats does not perform up to the mark. This could be a result of overfitting, insufficient data or ineffective hyperparameter tuning.

VII. Conclusion

This project applies statistical, machine learning, and deep learning models to forecast retail sales, achieving promising results with models like XGBoost and LightGBM. Feature engineering, including lagged variables and seasonal components, proved crucial in enhancing performance. Future work could explore hybrid models that combine traditional statistical methods with machine learning or deep learning techniques, leveraging their complementary strengths for improved accuracy and interpretability.

REFERENCES

[1] Kaggle, "Store Sales - Time Series Forecasting," [Online]. Available: https://www.kaggle.com/competitions/store-sales-time-series-forecasting/overview. Accessed: Oct. 7, 2024.

[2] ma, Shaohui & Kolassa, Stephan & Fildes, Robert. (2018). Retail forecasting: research and practice. 10.13140/RG.2.2.17747.22565.

[3] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed., OTexts, 2018.

[4] Makridakis S, Spiliotis E, Assimakopoulos V. Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLoS One. 2018 Mar 27;13(3):e0194889. doi: 10.1371/journal.pone.0194889. PMID: 29584784; PMCID: PMC5870978.

[5] Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting.

[6] Bai, S., Kolter, J. Z., & Koltun, V. (2018). "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling." arXiv preprint arXiv:1803.01271.

[7] Brownlee, J. (2017). Deep Learning for Time Series Forecasting. Machine Learning Mastery.

[8] Smyl, S. (2020). A Hybrid Method of Exponential Smoothing and Recurrent Neural Networks for Time Series Forecasting. International Journal of Forecasting, 36(1), 75-85.

[9] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd ed.)

[10] Lütkepohl, H. (2005). New Introduction to Multiple Time Series Analysis. Springer Science & Business Media.

[11] Chen, T., & Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.

[12] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." Advances in Neural Information Processing Systems (NIPS), 30, 3149-3157.

[13] Breiman, L. (2001). "Random Forests." Machine Learning, 45(1), 5-32

[14] Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time Series Analysis: Forecasting and Control (5th ed.). John Wiley & Sons.

[15] Hochreiter, S., & Schmidhuber, J. (1997). "Long Short-Term Memory." Neural Computation, 9(8), 1735-1780.

[16] Taylor, S. J., & Letham, B. (2018). "Forecasting at scale." The American Statistician, 72(1), 37-45.

[17] Elman, J. L. (1990). "Finding structure in time." Cognitive Science, 14(2), 179-211.

[18] Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). "Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting." International Journal of Forecasting, 37(4), 1748-1764.

APPENDIX

This appendix contains some additional data visualizations created during the data exploration phase.

Fig. 7: Sales Trends by Temporal Components: Highlights sales patterns across years, months, weeks, and days, showing significant peaks during weekends, end-of-month periods, and Q4 holiday seasons.

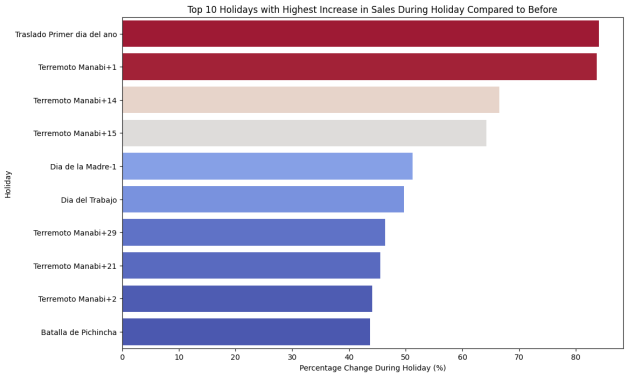

Fig. 8: Top 10 Holidays that had the largest increase in sales during the holiday as compared to before. Many of these are holidays due to the Manabi Earthquake. Other important holidays are New Year, Mother's Day and Labor Day. These are all national holidays, which have the greatest impact on sales overall.

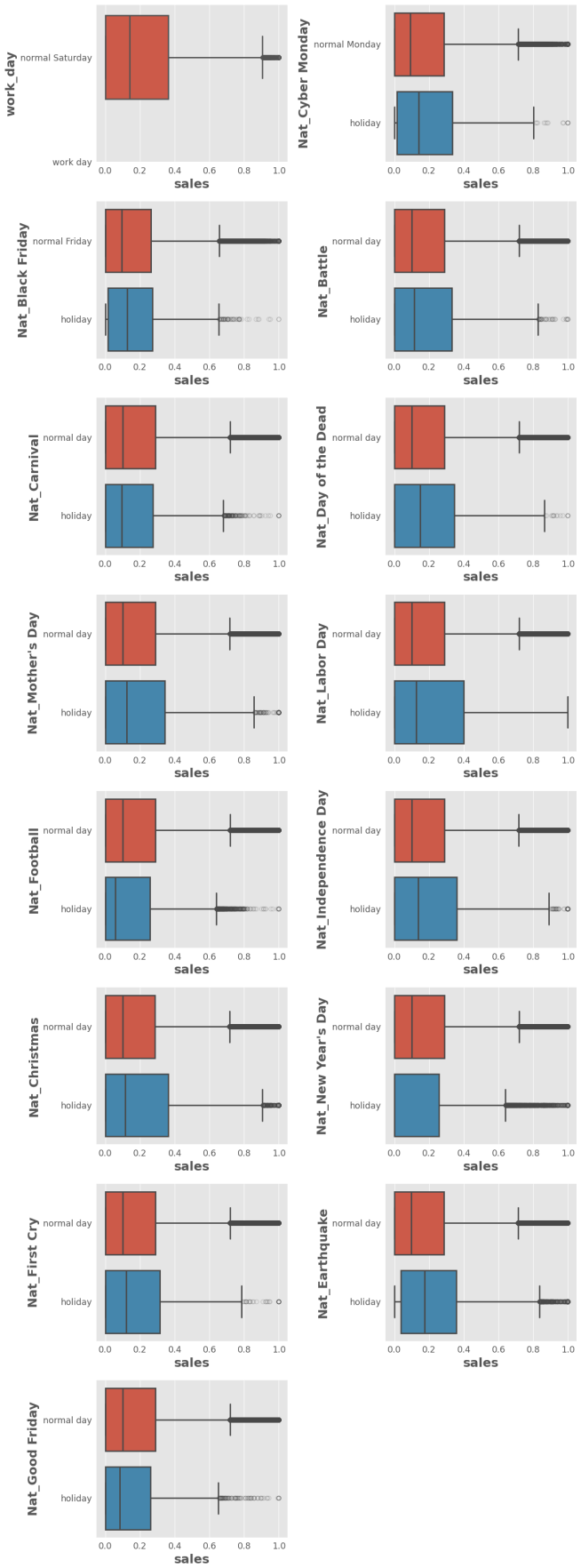

Fig. 9: Box Plots Comparing Sales on National Holidays and Normal Days Across Events: e.g., higher peaks on Black Friday versus regular Fridays

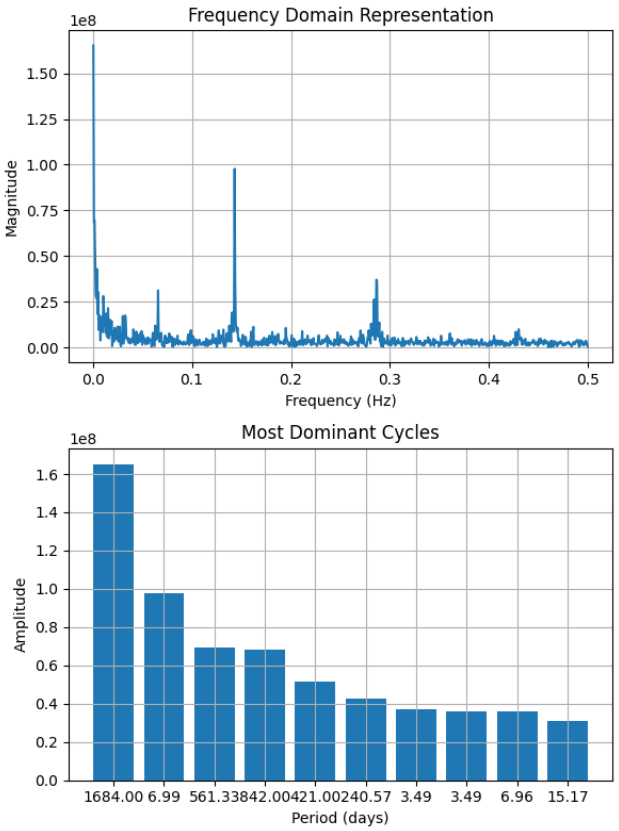

Fig. 11: The top figure shows the sales data in the frequency domain after applying the Fourier transform. The spikes with highest amplitude correspond to the strongest cycles, which are plotted in the figure below after converting back to the time domain.

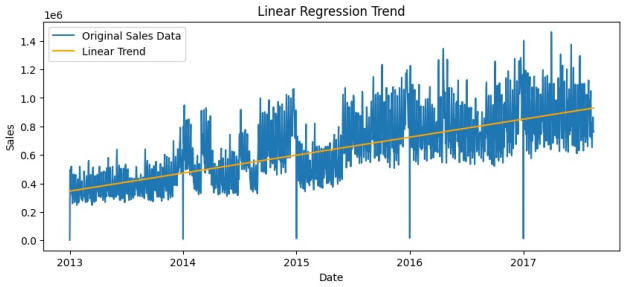

Fig. 10: Linear Regression Trend Analysis for Sales Data: This plot illustrates the original sales data (blue) and the linear regression trend line (yellow) fitted over time, highlighting a gradual upward sales trajectory from 2013 to 2017.