FORECASTING RETAIL SUCCESS: A DATA-DRIVEN APPROACH TO SALES PREDICTION FOR CORPORACIÓN FAVORITA

MSc BUSINESS ANALYTICS

DEPARTMENT OF OPERATIONS AND INFORMATION MANAGEMENT

SEPTEMBER 2024

ASTON UNIVERSITY

NAME: RANDIVE GAURAV SAMBHAJI

CANDIDATE NO.: 700434

SUPERVISOR: PROF. MOJTABA SAJADI

WORD COUNT: 8756

Declaration

I declare that I have personally prepared this report and that it has not in whole or in part been submitted for any other degree or qualification. Nor has it appeared in whole or in part in any textbook, journal or any other document previously published or produced for any purpose. The work described here is my/our own, carried out personally unless otherwise stated. All sources of information, including quotations, are acknowledged by means of reference, both in the final reference section and at the point where they occur in the text.

Acknowledgments

I would like to express my deepest gratitude to **Prof. Viktor Pekar**, the Program Director of the MSc Business Analytics Program, for his continuous support, guidance, and encouragement throughout my academic journey. His insights and leadership have been instrumental in shaping my understanding of the field.

I would also like to extend my sincere thanks to my dissertation supervisor, **Prof. Mojtaba Sajadi**, for his invaluable guidance, constructive feedback, and unwavering support during the development of this dissertation. His expertise and mentorship have been crucial in navigating the challenges of this project.

Lastly, I would like to thank my family and friends for their patience, understanding, and support throughout this process.

Table of Contents

Table of Tables

Table of Figures

ABSTRACT

This dissertation focusses on forecasting sales for the retail brand Corporación Favorita, which operates 54 stores across multiple regions in Ecuador a country in South America, offering 33 different product families. The dataset, provided by Kaggle, includes a variety of features such as store-level sales data, daily oil prices, and information on holidays and events. The time series spans from January 1, 2013, to August 31, 2017, with the goal of predicting sales for the subsequent 16 days after the final date in the training data.

Holidays and events, which have a big impact on consumer behaviour, were included as important factors along with oil prices, which are essential because Ecuador's economy depends so heavily on oil. The analysis investigates how fluctuations in oil prices affect sales patterns across different product families and how holidays and events affect sales. The size and complexity of the dataset required extensive preprocessing and transformation, particularly to tailor features for various machine learning and time series models.

We explored a variety of forecasting models, including machine learning models like Long Short-Term Memory (LSTM), Random Forest Regressor and XGBoost and traditional time series methods like ARIMA, AR, and MA.

The insights generated from this analysis can help Corporación Favorita optimise its manufacturing and supply chain processes by providing a reliable demand forecast. By anticipating sales, the company can better manage inventory levels, promotions, and resource allocation, resulting in improved operational efficiency.

1. INTRODUCTION

1.1 Background and Context

Sales forecasting is a key element of retail management that enables businesses to anticipate consumer demand, optimise inventory, and streamline supply chain operations. Corporación Favorita, a leading grocery retailer in Ecuador, operates 54 stores and offers 33 different product families. Because of its large-scale operations and diverse product offerings, the company requires accurate sales forecasts to improve its operational efficiency.

The dataset used in this study spans from January 1, 2013, to August 31, 2017, and contains valuable information such as store sales data, holidays, promotions, and external economic factors like daily oil prices. Incorporating these variables into sales forecasting models presents multiple challenges due to seasonality, holidays, and the oil-dependent nature of Ecuador's economy (Ibarra, 2014). Machine learning and time series analysis have shown enormous promise in tackling these challenges, enabling more accurate sales forecasts.

1.2 Problem Statement

This dissertation aims to predict sales for the top 5 product families in each store over the next 16 days, based on the sales history and other influencing factors such as holidays and oil prices. Holidays and events have been observed to significantly impact retail sales, often causing demand surges or dips depending on the type of event (Adhikari & Agrawal, 2013). Additionally, fluctuations in oil prices can affect consumer spending in an oil-dependent economy like Ecuador, introducing another layer of complexity into the sales forecasting process (Ibarra, 2014).

This research aims to develop predictive models that integrate these variables to produce reliable sales forecasts, which can be used by Corporación Favorita to optimise inventory management, marketing strategies, and supply chain operations.

1.3 Research Questions and Objectives

The following research questions form the basis of this study:

1. How do holidays and events affect sales across different product families and stores?

2. In an oil-dependent economy like Ecuador, what is the relationship between oil prices and sales?

3. Which stores excel in sales, and how can we enhance their performance?

4. Based on projected sales, how can we manage the supply chain for top-selling products more efficiently?

5. What patterns of sales correlations exist between stores, and how can these correlations help explain regional sales trends?

The primary objective is to develop robust forecasting models to predict sales for the top five product families while also identifying key trends and patterns that can enhance Corporación Favorita's operations.

1.4 Significance of the Study

Accurate sales forecasting is crucial for improving operational efficiency in retail, particularly for managing inventory and the supply chain. For Corporación Favorita, anticipating fluctuations in sales—especially during holidays and periods of oil price volatility—can help prevent stockouts, reduce excess inventory, and improve overall resource allocation.

Additionally, this study will provide insights into the correlation between store sales patterns, potentially revealing regional trends and operational inefficiencies. These findings are relevant not only for Corporación Favorita but also contribute to the broader academic literature on retail sales forecasting, particularly in the context of developing economies (Adhikari & Agrawal, 2013). Machine learning models have gained attention in retail forecasting due to their ability to handle large datasets and complex variables (Brockwell & Davis, 2016).

2. LITERATURE REVIEW

2.1 Sales Forecasting in Retail

Sales forecasting plays a vital role in retail management by enabling businesses to optimise their inventory, manage supply chains, and plan for promotional activities. Retailers like Corporación Favorita, which operates in diverse markets with varying product demands, require accurate forecasts to maintain operational efficiency. Traditional statistical models like ARIMA (AutoRegressive Integrated Moving Average) have been widely used in time series forecasting due to their ability to model linear relationships and capture trends in historical data (Box, Jenkins, & Reinsel, 2015). However, the retail sector has increasingly turned to machine learning models, which can handle large datasets, complex relationships, and external factors, making them more suitable for the dynamic nature of retail sales (Brockwell & Davis, 2016).

2.2 Impact of Holidays and Events on Sales

One of the primary challenges in retail sales forecasting is accounting for the impact of holidays and events, which often cause significant sales fluctuations. Special occasions such as national holidays, religious festivals, and local events can lead to sharp increases in consumer spending on certain product categories, while other categories may see reduced sales due to store closures or shifts in consumer behaviour (Ferreira et al., 2016).

For example, studies on retail sales during the Christmas holiday period have shown that certain product categories, such as electronics and clothing, experience significant sales spikes, while others, like groceries, tend to see more steady demand (Zhang & Liu, 2020). Similarly, in Ecuador, cultural and religious holidays such as Carnival and Holy Week (Semana Santa) drive variations in demand, particularly for food and beverages. These variations make it essential to incorporate holiday data into sales forecasting models to capture the seasonality and sudden shifts in demand (Adhikari & Agrawal, 2013).

Event-related sales variations are not limited to holiday seasons. Events like promotions and discount days also play a significant role in altering consumer purchasing behaviour. For example, Black Friday and Cyber Monday have been shown to significantly alter sales volumes, leading to inventory challenges for retailers that do not adequately forecast demand (Choi, Hui, & Bell, 2021). Failure to

account for these factors can lead to inaccurate forecasts and, consequently, lost sales opportunities or overstock issues.

2.3 The Effect of External Economic Factors on Sales

In addition to holidays and events, external economic factors, such as oil prices, can influence retail sales, particularly in oil-dependent economies like Ecuador. Oil price fluctuations affect consumer purchasing power, especially for lower- and middle-income consumers, who may adjust their spending habits in response to changes in fuel prices (Ibarra, 2014). Villanueva (2019) found that rising fuel and transportation costs in Ecuador led to a decline in consumer spending on non-essential goods as a result of oil price increases.

Incorporating oil price data into sales forecasting models is particularly important for understanding shifts in consumer demand. When oil prices rise, retailers may see a drop in sales for higher-priced or luxury goods, while demand for essential products like food may remain more stable. Machine learning models, which can handle multiple external factors, are well-suited for integrating economic indicators like oil prices into sales predictions (Rana, Goel, & Tiwari, 2021).

2.4 Machine Learning Models for Sales Forecasting

The growing complexity of retail environments, coupled with large datasets containing diverse features (such as promotions, holidays, and oil prices), has led to an increasing reliance on machine learning models for sales forecasting. Machine learning models, including Random Forest, Gradient Boosting Machines (GBM), and XGBoost, have shown superior performance over traditional time series models in several studies (Chandola, Banerjee, & Kumar, 2009).

The Random Forest Regressor is perfect for this job because it can handle both linear and non-linear relationships. This lets it easily capture the complex dependencies between things like oil prices, holidays, and sales (Breiman, 2001). Random forests are also resistant to overfitting, which is especially useful when working with big datasets, because they take the average of the outcomes of several decision trees, which lowers the variation.

XGBoost, an optimised implementation of gradient boosting, has become one of the most popular models in machine learning competitions due to its ability to handle large datasets and its efficient performance (Chen & Guestrin, 2016). XGBoost's strength lies in its ability to model non-linear relationships between features and sales outcomes, while also accounting for seasonality, trends, and external factors like oil prices. This makes it ideal for a retail forecasting scenario where external shocks and special events are common.

ARIMA models, while effective for time series data with strong trends and seasonality, are limited in their ability to incorporate external factors like oil prices and holidays. However, they remain useful when combined with machine learning models in a hybrid approach, where ARIMA captures the underlying time series trend and machine learning models handle external influences (Makridakis, Spiliotis, & Assimakopoulos, 2018).

2.5 Model Selection and Justification

Given the nature of the dataset and the complexity of the forecasting task, selecting appropriate models was crucial for generating accurate predictions. The diversity of factors influencing sales, including holidays, events, and oil prices, made it necessary to adopt multiple modelling approaches.

- **ARIMA (AutoRegressive Integrated Moving Average) Model**: ARIMA continues to be widely used in time series forecasting due to its ability to model both the trend and autocorrelation in data. Its relevance remains strong in fields like retail sales forecasting, where the model captures underlying patterns in sales without the need for external predictors (Rizvi, 2024). Furthermore, recent applications emphasize its reliability in time series with strong autocorrelations, making it a solid baseline for comparison with machine learning models (Hands-on Cloud, 2023)

- **Random Forest Regressor**: In 2023, Random Forest remains a preferred method for handling both linear and non-linear interactions, especially in retail environments where data is influenced by external factors like promotions and holidays. Its robustness to overfitting and

ability to manage large datasets make it an effective tool for sales forecasting. Recent studies highlight its capability to accurately capture complex patterns within the data, making it suitable for dynamic retail scenarios (Patil et al., 2023)

- **XGBoost**: XGBoost continues to show superior performance in handling large-scale, multi-dimensional datasets, especially in retail forecasting tasks were external variables like oil prices impact sales. Its gradient boosting framework makes it particularly adept at capturing non-linear relationships, and recent advancements in 2023 highlight its use in optimizing retail sales forecasts (Rane et al., 2023)

This multi-model approach allowed for a detailed exploration of different techniques, with ARIMA serving as the baseline for comparison. Because machine learning models like Random Forest and XGBoost are flexible, they let complicated factors like holidays, sales, and oil prices be included. This made the sales forecast for Corporación Favorita more accurate and reliable.

3. METHODOLOGY

3.1. Data Overview

The datasets used in this project provide a detailed representation of the sales activities of Corporación Favorita, along with external factors such as oil prices and holiday events. Properly merging and preprocessing these datasets was essential to ensuring accurate model predictions. This section details the primary dataset's characteristics and elucidates their transformation for analysis.

3.1.1 Train Dataset

The Train dataset contains 3,000,888 entries and includes six columns: `id`, `date`, `store_nbr`, `family`, `sales`, and `onpromotion`. Machine learning models for sales forecasting use this dataset, which spans from January 1, 2013, to August 31, 2017. The key features are:

Column	Data Type	Description
id	int64	Unique identifier for each record.
date	object	Date of the transaction, formatted as an object.
store_nbr	int64	Store number, identifying each store uniquely.
family	object	Product family, representing different categories of products sold.
sales	float64	Sales values, including fractional units (e.g., 1.5 kg of cheese).
onpromotion	int64	Number of items on promotion during the sales period.

Table 3.1.1 Train Dataset Overview

This dataset occupies 137.4 MB of memory. We applied various preprocessing steps, such as handling missing values, outlier detection, and merging with external datasets like oil prices and holidays.

3.1.2 Test Dataset

The Test dataset contains 28,512 entries and five columns:

Column	Data Type	Description
id	int64	Unique identifier for each record.
Date	object	Date of the transaction, formatted as an object.
Store_nbr	int64	Store number, identifying each store uniquely.
Family	object	Product family, representing different categories of products sold.
Onpromotion	int64	Number of items on promotion during the sales period.

Table 3.1.2 Test Dataset Overview

Unlike the train dataset, the test dataset excludes the `sales` column since it is used to make predictions. The key features are similar to those in the train dataset, excluding `sales`. After preprocessing, we merged this dataset with external data like oil prices and holidays for validation.

The test dataset occupies 1.1 MB of memory and was preprocessed similarly to the train dataset, with the same transformations applied to ensure compatibility during model training and testing.

3.1.3 Oil Price Dataset

The Oil Price dataset provides daily oil prices, which are a significant external factor influencing consumer behaviour in Ecuador, an oil-dependent economy. This dataset contains 1,218 entries and two columns:

Column	Data Type	Description
date	object	Date of the oil price record.
dcoilwtico	float64	Oil price per barrel in USD.

Table 3.1.3 Oil Price Dataset Overview

Several missing values were found in the oil price data, particularly in the `dcoilwtico` column. These missing values were filled using linear interpolation, a method that estimates missing data points by considering the linear trend from neighbouring values (Little & Rubin, 2019). This dataset was later merged with both the train and test datasets using the `date` column, ensuring that each sales record had the corresponding oil price for that day.

The oil price dataset occupies 19.2 KB of memory.

3.1.4 Holiday and Events Dataset

The Holiday and Events dataset contains 350 entries and provides crucial information about national and regional holidays and events, which often cause significant fluctuations in sales patterns. The dataset includes the following columns:

Column	Data Type	Description
date	object	Date of the holiday or event.
type	object	Type of holiday/event (e.g., national holiday, regional event).
locale	object	Locale where the holiday applies (e.g., national, regional, or local).
locale_name	object	Specific region or city affected by the holiday or event.
description	object	Description of the holiday or event.
transferred	bool	Indicates if the holiday was transferred to another day.

Table 3.1.4 Holidays and Events Dataset Overview

The holidays and events were initially stored as categorical data, which is not suitable for machine learning models. To address this, we applied one-hot encoding, transforming each unique holiday or event into its own binary column, where 1 indicates that the day is a holiday/event and 0 indicates a

normal day (Ferreira, Lee & Simchi-Levi, 2016). This change allowed the model to more effectively account for the effect of holidays on sales.

This dataset occupies 14.1 KB of memory.

3.1.5 Transactions Dataset

The **Transactions dataset** provides a record of daily transactions for each store. It contains information about the number of transactions completed in each store on a given date, making it highly correlated with the stores overall sales volume. This dataset is useful for understanding store-level sales patterns and consumer behaviour over time. However, it was used solely for **Exploratory Data Analysis (EDA)** and was not merged or transformed for model training purposes.

Column	Data Type	Description
date	object	Date of the transaction, formatted as an object.
Store_nbr	int64	Store number, uniquely identifying each store.
Transactions	int64	Total number of transactions for that store on the given date.

Table 3.1.5 Transactions Dataset Overview

This dataset provides valuable insights into the number of customers entering stores and purchasing products, as well as helping to identify store-level trends and patterns. Transaction volume is directly correlated with the sales figures, making it a useful feature for exploration. The direct correlation between transaction volume and sales figures makes it a valuable feature for exploratory analysis. However, the transaction data was not incorporated into the final models or preprocessing steps.

3.1.6 Merging and Preprocessing

3.1.6.1 Oil Price Integration

To analyse the impact of oil prices on sales, we first preprocessed the oil price data by filling in missing values using linear interpolation. Following that, the `dcoilwtico` column was renamed to `oil_price` for clarity. The next step was to merge the processed oil price data with the training dataset using the

`date` column. This merge ensured that each sales record was associated with the correct oil price, allowing for a thorough analysis of the relationship between oil prices and sales trends (Villanueva, 2019).

Additionally, the `date` column in both the train and test datasets was converted from object format to datetime64. This allowed for easier manipulation of time-based features and made it possible to perform temporal analyses, such as weekly and monthly patterns (Hyndman & Athanasopoulos, 2018).

3.1.6.2 Holiday and Events Integration

We transformed the holiday data using one-hot encoding, transforming each unique holiday into a binary column. For each date, a 1 in the corresponding column indicated that the day was a holiday, and a 0 indicated a normal day. This transformation was essential for enabling machine learning models to recognise how different holidays and events affected sales. After transforming the holiday dataset, the encoded features were merged with the train and test datasets based on the `date` column.

3.1.6.3 Feature Engineering for Time-Specific Attributes

We developed additional time-specific features to further improve the model's predictive power:

Features	Description
dayOfWeek	Represents the day of the week (e.g., Monday, Tuesday) as a numerical value. Retail sales often vary depending on the day, with weekends typically seeing higher sales.
Month	Represents the month of the year, capturing any seasonality effects (e.g., higher sales in December due to the holidays).
WeekOfYear	Represents the week number in the year, which can capture both seasonal and promotional trends (Makridakis, Spiliotis & Assimakopoulos, 2018).

Table 3.1.6.3.1 Time Specific Feature Dataset Overview

We appended these additional features to both the train and test datasets, completing the preprocessing stage. The final datasets now contained 111 columns, as well as the following data types:

Column	Data Type	Description
date	datetime64	Date of the transaction, converted to datetime format for time-based analysis.
Store_nbr	int64	Unique identifier for each store.
Family	object	Product family, representing different product categories.
Sales	float64	Sales values, including fractional units.
Onpromotion	int32	Number of items on promotion.
Holiday/Event 103 Columns	int32	Binary columns indicating whether each day was a holiday or event (0 or 1).
dayOfWeek	int64	Day of the week, represented numerically (e.g., 0 for Monday).
Month	int64	Month of the year (1 for January, 2 for February, etc.).
WeekOfYear	Uint32	Week of the year, capturing seasonal sales trends.

Table 3.1.6.3.2 Transformed Dataset Overview

This dataset, now fully preprocessed, was ready for analysis and prediction.

3.2 Exploratory Data Analysis (EDA)

We conducted Exploratory Data Analysis (EDA) to understand the underlying trends and patterns in the sales data. We used visual and statistical analyses to uncover key insights into how external factors like oil prices and holidays impacted sales.

3.2.1 Understanding Sales Trends and Seasonality

Through EDA, I examined the trend and seasonality of sales data, key elements of time series analysis (Hyndman & Athanasopoulos, 2018). Visualising the sales data over time, I observed distinct peaks and troughs corresponding to holiday periods and promotions, which aligned with expected retail behaviour. These visualisations helped reveal:

- The long-term trends indicate whether sales have been generally increasing or decreasing over the years.
- Sales exhibit seasonality, reaching their peak during major holidays and weekends. Seasonal peaks were particularly evident during national holidays like Carnival and Christmas.

To better understand the stationarity of the sales data, I applied the Augmented Dickey-Fuller (ADF) test, which is a standard statistical test for checking stationarity in time series data (Greene, 2012). The test revealed that while there was some degree of seasonality in the data, the sales trends were not stationary, which necessitated data transformations like differencing or detrending before applying time series models such as ARIMA (Box, Jenkins, & Reinsel, 2015).

3.2.2 Dependency of Sales on Oil Prices

Given Ecuador's economic reliance on oil, I also explored the relationship between oil prices and sales volumes. I plotted oil price trends alongside sales data to identify any potential correlation between rising oil prices and changes in consumer behaviour. A moderate correlation was identified, with sales of certain non-essential products dropping during periods of higher oil prices (Villanueva, 2019). This trend reflects how external economic factors such as oil price fluctuations can influence consumer purchasing power and retail demand (Rana, Goel, & Tiwari, 2021).

3.2.3 Analysis of Transactions

The transaction data helped to identify store-level performance differences and sales trends. Visualisations of daily transactions over time revealed strong correlations with the overall sales data, especially during holiday periods and promotional events. For instance, the number of transactions generally increased significantly during major holidays and promotional periods (e.g., Black Friday or Christmas). This correlation reinforced the decision to include holiday data as an important feature in the modelling process.

Through the transaction data, we observed:

- Peak days, coinciding with holiday events or promotions, witness exceptionally high transaction volumes.
- Certain stores consistently have higher transaction volumes than others, indicating patterns in their performance. These patterns were particularly useful in identifying top-performing stores.

Despite not transforming or merging the transaction data into the final dataset for modelling, it offered crucial insights during the EDA process. These insights helped pick out the important features and gave us ideas for the modelling phase's preprocessing and feature engineering steps.

3.3 Data Slicing for Scalability

Given the large size of the dataset (33 product families across 54 stores), it became computationally challenging to run models on the entire dataset due to hardware limitations (8 GB of RAM). We implemented a data slicing strategy based on the following approach to address this challenge:

3.3.1 First Approach: Forecasting Sales for Top 5 Product Families

In the first approach, the goal was to predict the **total sales** for top 5 selling product families for the next 16 days, irrespective of store. We transformed the data accordingly by aggregating sales across all product families and stores. We used this aggregate dataset to forecast the total expected sales for the top 5 selling products, aiding in the management of high-level operations such as staffing and inventory allocation.

3.3.2 Second Approach: Sales Prediction for Each Store and Product Family

In the second approach, the focus shifted to predicting sales for each product family in each store for the next 16 days. This required further segmentation of the dataset based on store and product family. We created separate datasets for each of the top 5 selling product families through EDA, enabling more focused and accurate predictions. This segmentation allowed the model to capture the specific seasonality's and trends of individual product families across different stores.

3.4 Model Training and Evaluation

3.4.1 ARIMA as a Baseline Model

To establish a performance benchmark, I used the ARIMA (AutoRegressive Integrated Moving Average) model as the baseline model for time series forecasting. ARIMA is a widely used model for time series data, capable of capturing trends, seasonality, and autocorrelation in stationary datasets (Box, Jenkins, & Reinsel, 2015). Since the sales data exhibited non-stationary behaviour, I applied differencing to achieve stationarity before fitting the ARIMA model (Hyndman & Athanasopoulos, 2018). The baseline ARIMA model provided a reference point for evaluating the performance of more advanced machine learning models.

3.4.2 Machine Learning Models

Given the complexity of the dataset and the inclusion of external factors such as holidays and oil prices, I explored several machine learning models in addition to ARIMA.

- Breiman (2001) selected the Random Forest Regressor due to its robustness in handling large datasets and its ability to capture both linear and non-linear relationships between features. Random Forest's ensemble approach made it particularly effective at reducing overfitting, ensuring that the model generalised well to unseen data.

- We also employed XGBoost, a powerful gradient boosting algorithm. XGBoost is highly effective in handling large, complex datasets and can model intricate dependencies between features (Chen & Guestrin, 2016). Its ability to automatically handle missing data and provide feature importance metrics was particularly useful in this project, where external variables such as oil prices and holidays played a significant role.
- Hochreiter and Schmidhuber (1997) introduced the Long Short-Term Memory (LSTM) model, which is highly effective in handling time series data with long-term dependencies. LSTM's ability to retain information over extended periods made it suitable for capturing complex sales patterns, particularly for stores with strong seasonality or fluctuating trends. Its architecture helped model sequential dependencies in the sales data, providing more accurate predictions for the top product families across multiple time frames.

3.5 Model Evaluation

After training the models, I evaluated their performance using standard time series forecasting metrics, including root mean squared error (RMSE) and mean absolute error (MAE) (Hyndman & Athanasopoulos, 2018). The significance of RMSE lies in its ability to penalise larger errors more heavily, thereby minimising large deviations in sales predictions. MAE provided a more intuitive measure of average prediction error, making it easier to communicate the model's performance to stakeholders.

We selected the model with the highest R-square, lowest RMSE, and MAE to make the final 16-day sales forecast. We then adjusted this forecast based on upcoming holidays and expected fluctuations in oil prices, giving Corporación Favorita actionable insights to guide inventory management, staffing, and overall operations.

4. DATA ANALYSIS

4.1 Exploratory Data Analysis (EDA) for Transactions

Exploratory Data Analysis (EDA) is an essential step in uncovering the patterns, trends, and relationships present in the dataset. This analysis provides valuable insights into sales behaviour, transactions, and external factors such as holidays and oil prices. This section will cover several visualisations and analyses to explore how these parameters affect sales.

4.1.1 Store-Level Transaction Patterns

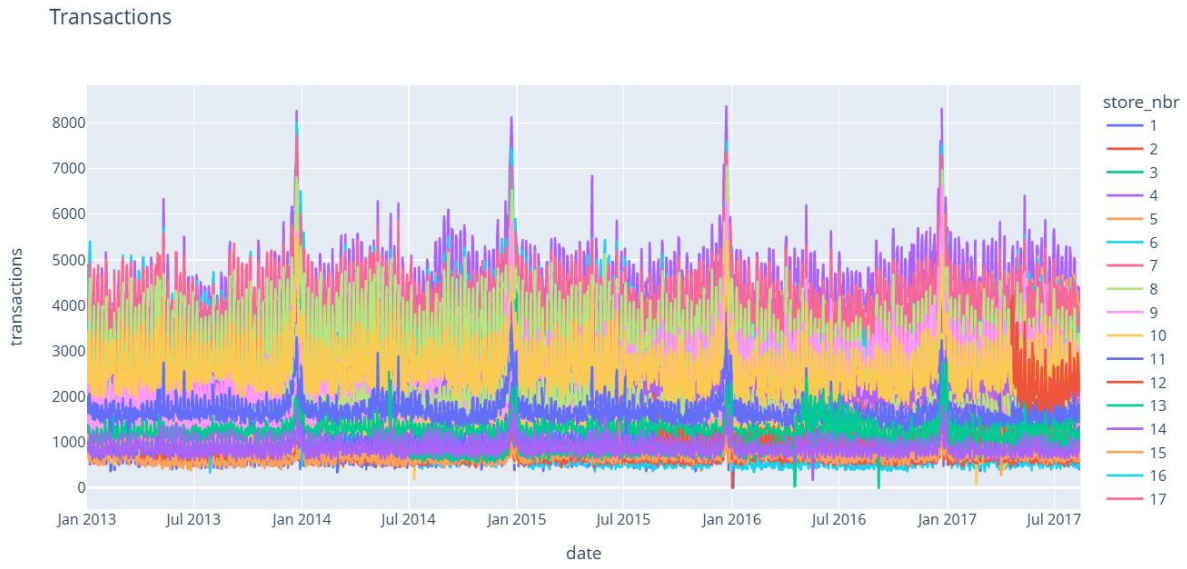

Fig. 4.1.1 Store-Level Transaction Patterns

- We analysed the daily transaction volume across 54 stores between January 2013 and August 2017 to understand overall store performance and identify seasonal peaks.

- Seasonal Peaks: The interactive graph (Figure 4.1.1) highlights peaks in transactions during holiday periods like Christmas, Black Friday, and New Year, reflecting the increased consumer activity during festive seasons. These spikes are crucial indicators of when store operations and inventory management need adjustments to handle higher footfall.

- Store Performance Variation: Different stores show varying transaction volumes, likely due to differences in location, size, and the demographics of the regions they serve. Such insights suggest the need to account for store-level differences when creating predictive models for sales.

- The importance of understanding store-specific transaction behaviour is highlighted in studies like Ferreira, Lee, and Simchi-Levi (2016), which show that store-level analysis helps optimise operations and promotions during peak periods.

4.1.2 Monthly Transaction Patterns

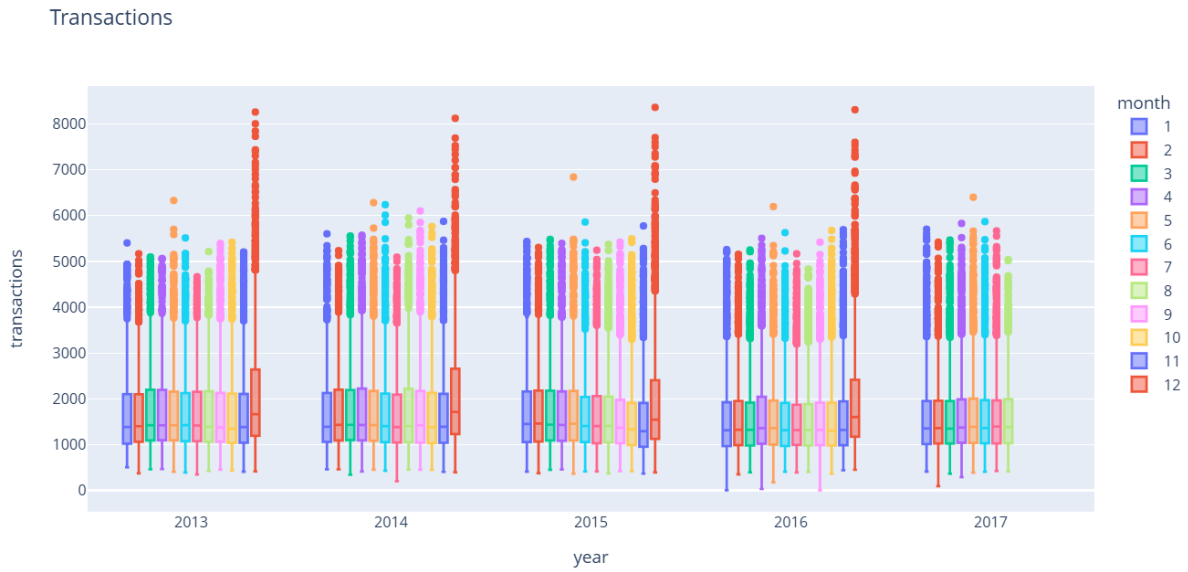

Fig. 4.1.2 Monthly Transaction Pattern

- A boxplot of transactions over months for the years 2013–2017 (Figure 2) reveals that certain months consistently show higher transaction volumes, particularly towards the end of each year.

- Year-end Increases: The boxplot reveals consistent increases in transaction volume in November and December, driven by holiday shopping and promotions. This trend has been widely documented in retail studies (Choi, Hui, & Bell, 2021), which point to spikes in consumer demand during key shopping seasons.

- Annual Consistency: Despite some fluctuations between years, the general pattern of high year-end sales remains consistent, reaffirming the importance of seasonality in sales prediction models.

- This analysis highlights the need to incorporate month-of-year effects into the forecasting model to capture annual seasonality.

4.1.3 Average Monthly Transactions

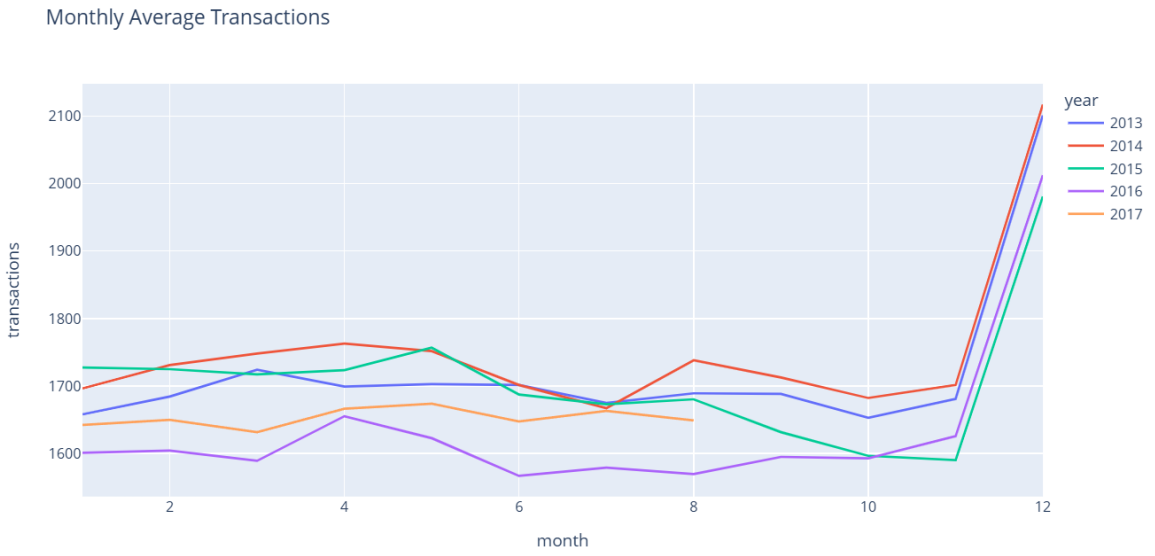

Fig. 4.1.3 Monthly Average Transactions

- The monthly average transactions (Figure 4.1.3) offer insights into the cyclical nature of consumer behaviour.

- Seasonality Across Years: The graph shows a clear upward trend in average transactions from October to December in all years, reinforcing the high demand seen during holiday periods. Similarly, there is a mid-year dip (May-August), reflecting a slowdown in shopping activity, likely driven by fewer major holidays and regional economic factors.

- December Peak: Each year sees a sharp spike in transactions in December, providing evidence of the strong holiday effect in driving retail sales (Adhikari & Agrawal, 2013). This highlights the need to capture seasonal peaks in any sales forecasting model.

4.1.4 Correlation Between Transactions and Sales

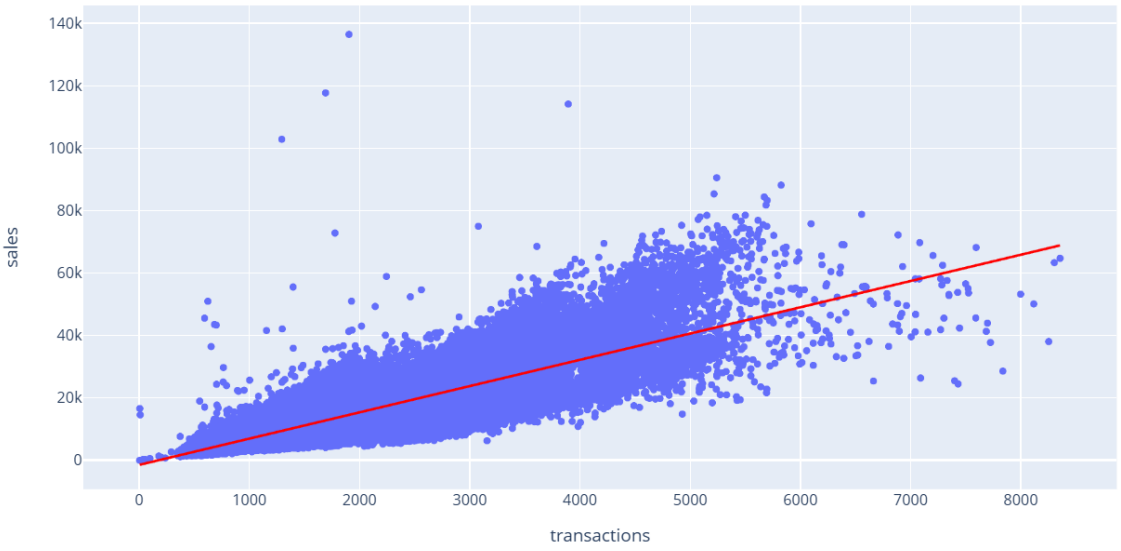

Fig. 4.1.4 Correlation Between Transactions and Sales

- The scatter plot with a fitted regression line (Figure 4.1.4) demonstrates the correlation between daily transactions and sales.

- Strong positive correlation: There is a clear linear relationship between the number of transactions and total sales, confirming the intuitive understanding that more transactions lead to higher sales. Studies by Rana, Goel, and Tiwari (2021) have demonstrated similar correlations in retail environments, where transactional data is a significant predictor of sales volume.

- Outliers: A few outliers show unusually high transactions that did not correspond to high sales. Factors such as promotional discounts or the sale of lower-priced items during specific events could influence these instances.

- Given the strength of this relationship, transaction volume is a critical feature to include in the predictive models.

4.1.5 Weekly Transaction Patterns

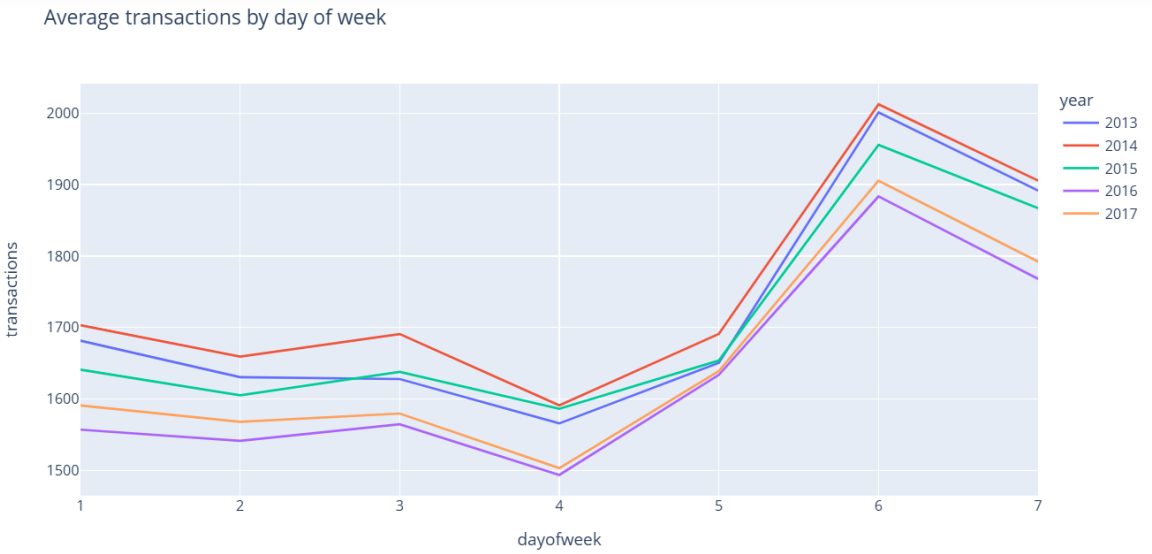

Fig. 4.1.5 Average transactions by day of week

- The final EDA chart (Figure 4.1.5) illustrates the average transactions by day of the week over the years.

- Weekend Effect: The analysis reveals significantly higher transactions on Saturdays and Sundays, likely driven by consumer shopping habits. Mid-week days (Monday to Wednesday) report fewer transactions, a pattern that aligns with typical weekly retail cycles (Zhang & Liu, 2020). This insight is critical when engineering features based on the day of the week.

- Consistency Across Years: This pattern is consistent across all five years, indicating that day-of-week trends are reliable predictors of short-term sales fluctuations.

4.3 Exploratory Data Analysis (EDA): Oil Prices

The oil price dataset is a key external factor for sales analysis, particularly given Ecuador's status as an oil-dependent economy. In this section, we explore how fluctuations in oil prices influence consumer spending behaviour and sales across product families.

4.3.1 Oil Price Trends

Fig. 4.3.1 Oil Price Trend

- The first chart (Figure 4.3.1) shows the daily oil price trends from January 2013 to August 2017, with the red line representing actual oil prices (`dcoilwtico`) and the blue line representing interpolated oil prices. The interpolation was necessary to handle missing values in the original dataset.

- Oil Price Volatility: Between 2013 and mid-2014, oil prices hovered above $90 per barrel. However, a steep decline followed, with prices dropping to below $50 per barrel by early 2016. The trend stabilises slightly in 2016 and 2017, with oil prices fluctuating between $40 and $60. This sharp decline in oil prices coincided with economic challenges for oil-dependent nations like Ecuador, impacting consumer purchasing power (Villanueva, 2019).

- Interpolation of Missing Data: The blue interpolated line fills in the gaps in the dataset where there are no recorded oil prices. Linear interpolation ensures that the missing values do not distort the analysis of trends and their relationship with sales (Little & Rubin, 2019).

4.3.2 Oil Price Impact on Transactions and Sales

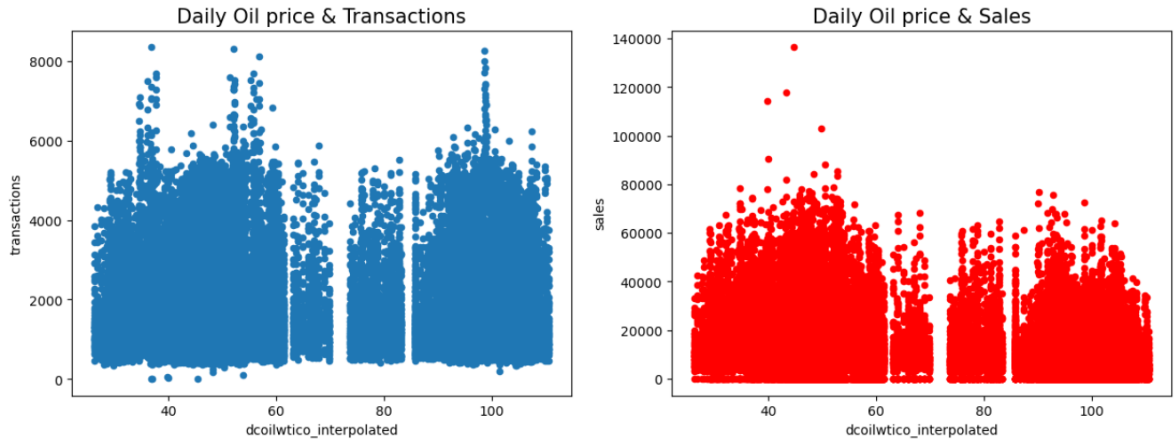

Fig. 4.3.2 Oil Price Impact on Transactions and Sales

- Figure 4.3.2 shows scatter plots illustrating the relationship between oil prices and both daily transactions (left) and sales (right). These plots help visualise how oil price fluctuations affect store activity and overall sales.

- No immediate trend: At first glance, the scatter plots do not show a clear relationship between oil prices and transactions/sales. There are clusters of points across different oil price ranges, making it difficult to draw immediate conclusions. We would have been wrong if we concluded that there is no relationship based on this simple output.

- Clusters Around Oil Prices: Upon closer inspection, there appear to be two distinct clusters of transaction and sales data, roughly divided by an oil price of $70. Below $70, we see higher sales and transaction volumes, while above $70, activity appears to be lower. This finding suggests that consumer spending is more constrained when oil prices are higher. This aligns with earlier studies indicating that rising oil prices often reduce disposable income, leading to decreased spending on non-essential goods (Villanueva, 2019).

4.3.3 Oil Price and Product Family Sales

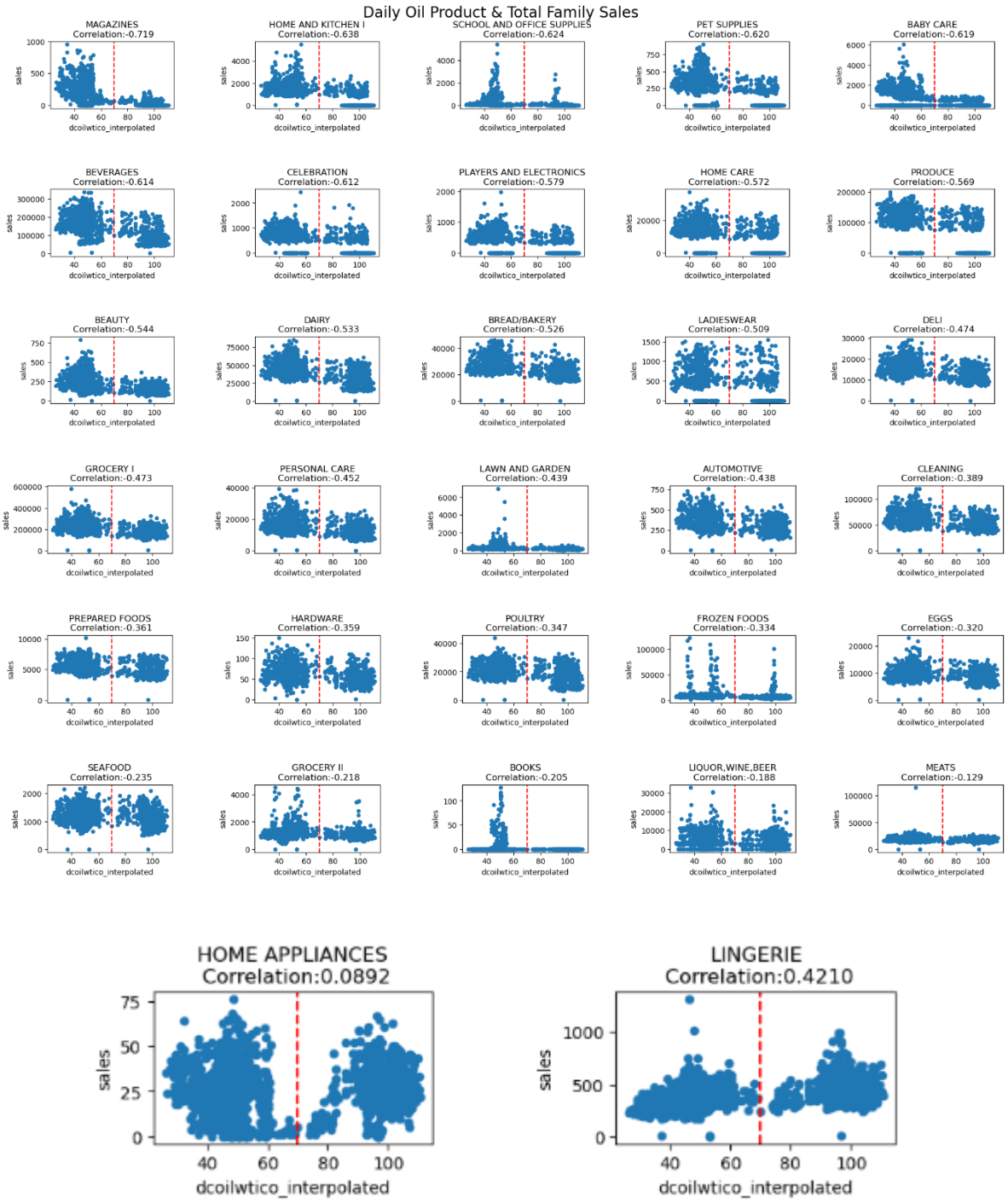

Fig. 4.3.3 Oil Price and Product Family Sales

- Figure 4.3.3 provides a detailed analysis of the correlation between oil prices and sales for different product families. The red vertical line at $70 highlights the threshold where patterns in consumer spending shift. The correlations between oil prices and sales are displayed for each product family, revealing which goods are most sensitive to oil price fluctuations.

- Product Families with Strong Negative Correlations: Several product families, such as magazines (-0.719), home and kitchen (-0.638), and beverages (-0.614), show strong negative correlations with oil prices. This suggests that as oil prices increase, sales in these categories decrease significantly. Non-essential items like magazines and beverages are particularly susceptible to changes in consumer budgets during times of economic uncertainty.

- Moderate to Weak Negative Correlations: Categories like frozen foods (-0.334) and eggs (-0.320) show weaker but still negative correlations. These are essential goods, meaning that while sales drop as oil prices rise, they are less affected than luxury or non-essential items.

- Product Families with Positive Correlations: Interestingly, certain categories like lingerie (0.421) and home appliances (0.089) show positive correlations. For these goods, sales may increase even as oil prices rise, possibly due to promotional events or consumer behaviour changes during times of economic volatility (Zhang & Liu, 2020).

4.4 Insights from EDA

4.4.1 EDA for Transactions:

After analysing the above transactions data, we realised that the data shows patterns and explains the customer behaviour depending on the time of the year, time of the week and holidays. Hence, we can conclude that we have answer research question 1 and learnt the pattern which are given below:

1. **Strong Seasonality**: Both monthly and weekly transaction data reveal clear seasonal patterns, with peaks around holidays and weekends. These insights will be incorporated into the forecasting models as seasonality components.

2. **Transaction-Sales Relationship**: A strong positive correlation between transactions and sales indicates that transaction data is a reliable predictor of sales volume.

4.4.2 Key Insights from Oil Price Analysis

After analysing the oil price data, we realise that the oil prices certainly affect the product family sales. Some are affected positively, while some are negative affected. Hence, we can conclude that the sales are affected by oil price. Here we answered research question 2 focusing on oil price and sales relationship.

The analysis of oil prices yields several critical insights for sales forecasting:

1. **Oil Price Threshold Effect**: The $70 oil price threshold seems to delineate two distinct behaviours in consumer spending. Below $70, sales and transactions are higher, while above $70, they decline. This finding suggests a non-linear relationship between oil prices and sales, necessitating the inclusion of oil price data as a variable in forecasting models.

2. **Differential Impact Across Product Families**: Oil prices have varying effects across product families. Luxury and non-essential goods show stronger negative correlations, while essential goods are less affected. These patterns need to be considered when creating product-family-specific sales forecasts.

3. **Correlation with External Economic Conditions**: The overall pattern of oil prices affecting sales reinforces the importance of incorporating macroeconomic variables into the predictive models. The models can account for external economic shocks that impact consumer behaviour, improving their forecasting accuracy.

4.5 Model Training and Forecasting Approach

4.5.1 Challenges in Handling the Full Dataset

Before diving into the model training, it is essential to address the challenges posed by the full dataset, which consists of 33 product families and 54 stores, each with distinct trends and seasonality patterns.

- Each product family has unique trends and seasonality: As each product family and store exhibits different sales patterns, applying a single forecasting model to the entire dataset may yield inaccurate results. Seasonality and trends vary significantly across different products and locations, making it difficult for a single model to generalise effectively.

- System Constraints: Due to computational limitations (e.g., limited RAM and processing power), training a model on the entire dataset was impossible. Handling large amounts of time series data simultaneously also risked running into memory bottlenecks. This challenge is not uncommon in large-scale forecasting scenarios, as noted in literature by Bengio et al. (2012), where resource constraints limit the ability to scale up model training.

4.5.2 Business Problem 1: Forecasting Sales for Top 5 Product Families

We decided to focus on the top 5 selling product families to address both business and technical constraints. We could make more accurate predictions and solve various business problems efficiently by training separate models for these products.

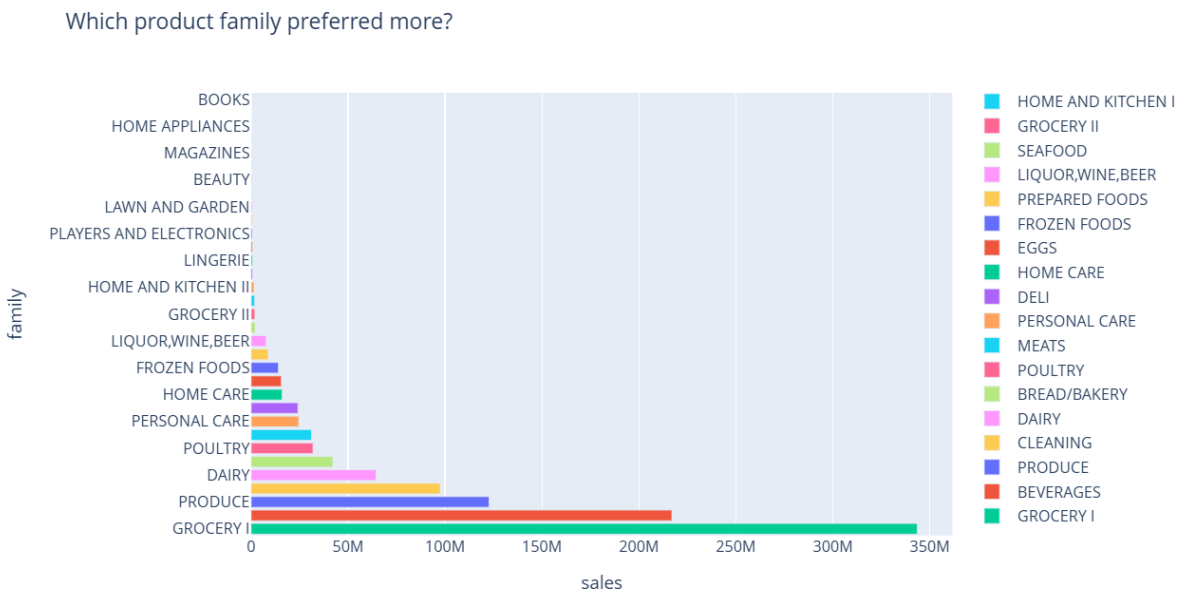

Fig. 5.2.1 Top selling Products

- Selection of Top 5 Product Families: Using sales volume data, I identified the top 5 selling product families, GROCERY I, BEVERAGES, PRODUCE, CLEANING and DAIRY. These categories account for the majority of the company's revenue, making them a priority for forecasting efforts. This step was driven by business priorities, as optimising supply chains and resource allocation for top-selling products can maximise profitability and minimise risk (Choi et al., 2021).

- Data Segmentation: We created a separate dataframe for each of these top-selling products, excluding store-specific information for the first business problem. This reduced the complexity of the dataset and enabled more accurate forecasts for each product.

```
Grocery_df = data[data['family']=='GROCERY I']
```

```
Beverage_df = data[data['family']=='BEVERAGES']
```

```
Produce_df = data[data['family']=='PRODUCE']
```

```
Cleaning_df = data[data['family']=='CLEANING']
```

```
Dairy_df = data[data['family']=='DAIRY']
```

Fig. 5.2.2 Slicing Data

4.5.2.1 Model Selection: ARIMA for Overall Product Family Sales

- Rationale for ARIMA: We selected the ARIMA (AutoRegressive Integrated Moving Average) model for the initial forecasting task, which involved predicting the overall sales for each top product family across all stores. ARIMA is a well-established model for time series forecasting, capable of handling trends and seasonality in non-stationary data (Box et al., 2015). The model is particularly effective for capturing the underlying patterns of a single time series, such as the overall sales trend of a specific product family.

- Business Value: This forecast helps the company optimise manufacturing processes by anticipating demand for each product family over the next 16 days. By providing accurate demand forecasts, the business can adjust production schedules, preventing overproduction or stockouts, and thus improving overall profitability (Adhikari & Agrawal, 2013).

- Evaluation: We trained the ARIMA model on the segmented product family datasets, capturing key trends and seasonal patterns. Performance metrics such as mean absolute error (MAE) and root mean squared error (RMSE) were used to evaluate the model's accuracy (Hyndman & Athanasopoulos, 2018).

Forecast for the top five selling products:

1. GROCERY I:

Model Description

			Model Type
Model ID	sales	Model_1	ARIMA(17,0,18)

Fig. 5.2.1.1 Arima model for Grocery I

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Model Fit statistics: R-squared	Model Fit statistics: MAE	Ljung-Box Q(18): Statistics	Ljung-Box Q(18): DF	Ljung-Box Q(18): Sig.	Number of Outliers
sales-Model_1	3	.757	.757	17403.450	.	0	.	0

Fig. 5.2.1.2 ARIMA Model Statistics for Grocery I

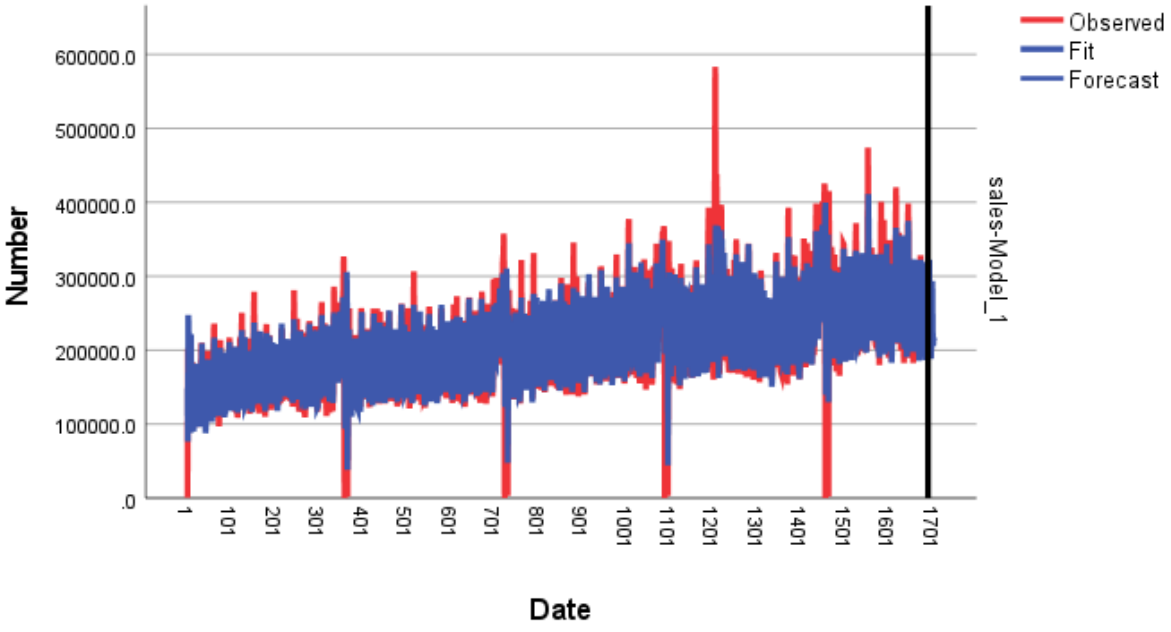

Fig. 5.2.1.3 ARIMA Forecast Chart

date	Predicted_sales_Model_1_A
2017/08/16	227897.1
2017/08/17	224586.4
2017/08/18	251334.4
2017/08/19	299301.4
2017/08/20	322373.6
2017/08/21	259169.3
2017/08/22	223004.1
2017/08/23	210622.4
2017/08/24	188613.0
2017/08/25	206989.2
2017/08/26	271698.9
2017/08/27	293620.5
2017/08/28	226977.1
2017/08/29	205503.6
2017/08/30	217059.3
2017/08/31	212705.6

Fig. 5.2.1.4 Grocery I Predicted Values

2. BEVERAGES:

Model Description

			Model Type
Model ID	totalsales	Model_1	ARIMA(16,1,16)

Fig. 5.2.2.1 ARIMA model for Beverages

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Model Fit statistics: R-squared	Model Fit statistics: MAE	Ljung-Box Q(18): Statistics	Ljung-Box Q(18): DF	Ljung-Box Q(18): Sig.	Number of Outliers
totalsales-Model_1	1	.690	.871	13302.191	.	0	.	0

Fig. 5.2.2.2 ARIMA model Statistics for Beverages

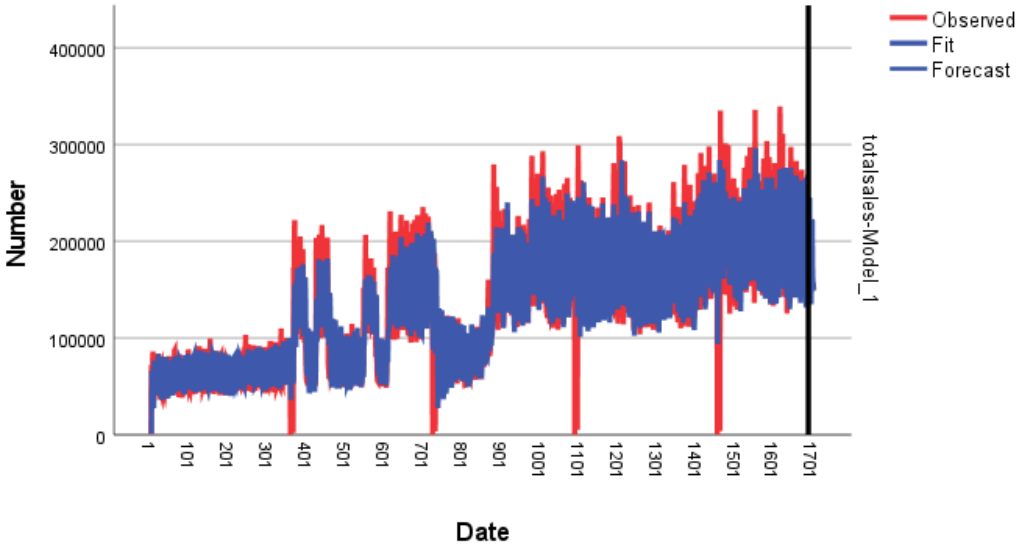

Fig. 5.2.2.3 ARIMA Forecast Chart

Date	Predicted_totalsales_Model_1
16.08.2017	158797
17.08.2017	145348
18.08.2017	176140
19.08.2017	230433
20.08.2017	245440
21.08.2017	183925
22.08.2017	152810
23.08.2017	150624
24.08.2017	134916
25.08.2017	157204
26.08.2017	216590
27.08.2017	223043
28.08.2017	172690
29.08.2017	152577
30.08.2017	157423
31.08.2017	149171

Fig. 5.2.2.4 Beverages Predicted Sales

3. PRODUCE:

Model Description

			Model Type
Model ID	sales	Model_1	ARIMA(12,0,14)

Fig. 5.2.3.1 ARIMA model for Produce

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Model Fit statistics: R-squared	Model Fit statistics: MAE	Ljung-Box Q(18): Statistics	Ljung-Box Q(18): DF	Ljung-Box Q(18): Sig.	Number of Outliers
sales-Model_1	3	.929	.929	8622.475	.	0	.	0

Fig. 5.2.3.2 ARIMA model statistics for Produce

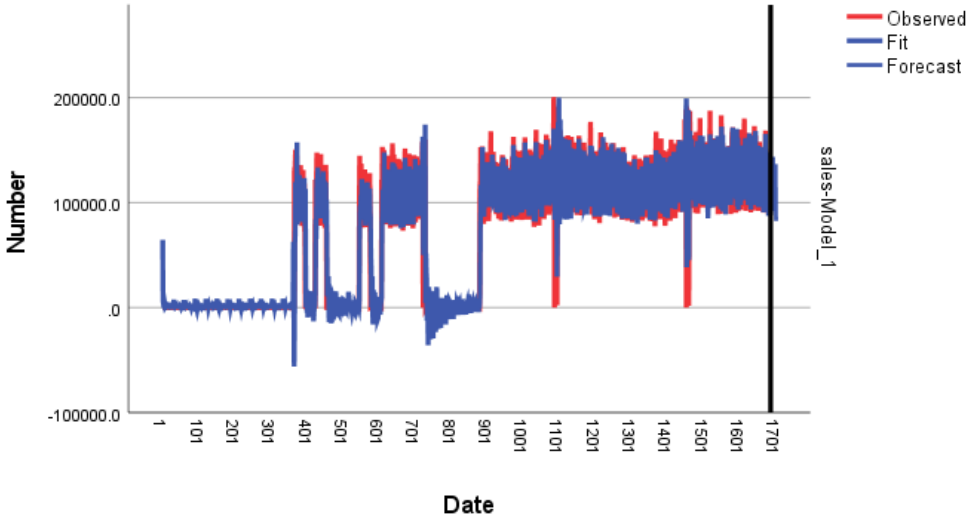

Fig. 5.2.3.3 ARIMA Forecast Chart

date	Predicted_sales_Model_1
2017/08/16	142703.4
2017/08/17	88077.0
2017/08/18	112417.7
2017/08/19	124627.7
2017/08/20	136205.2
2017/08/21	114375.8
2017/08/22	119389.5
2017/08/23	143608.0
2017/08/24	92037.9
2017/08/25	110795.5
2017/08/26	126503.1
2017/08/27	135842.1
2017/08/28	110253.9
2017/08/29	113845.5
2017/08/30	136713.0
2017/08/31	82550.3

Fig. 5.2.3.4 Produce Predicted Sales

4. CLEANING PRODUCTS:

Model Description

			Model Type
Model ID	sales	Model_1	ARIMA(10,0,16)

Fig. 5.2.4.1 ARIMA model for Cleaning Product

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Model Fit statistics: R-squared	Model Fit statistics: MAE	Ljung-Box Q(18): Statistics	Ljung-Box Q(18): DF	Ljung-Box Q(18): Sig.	Number of Outliers
sales-Model_1	4	.859	.859	3049.421	.	0	.	0

Fig. 5.2.4.2 ARIMA model statistics for Cleaning products

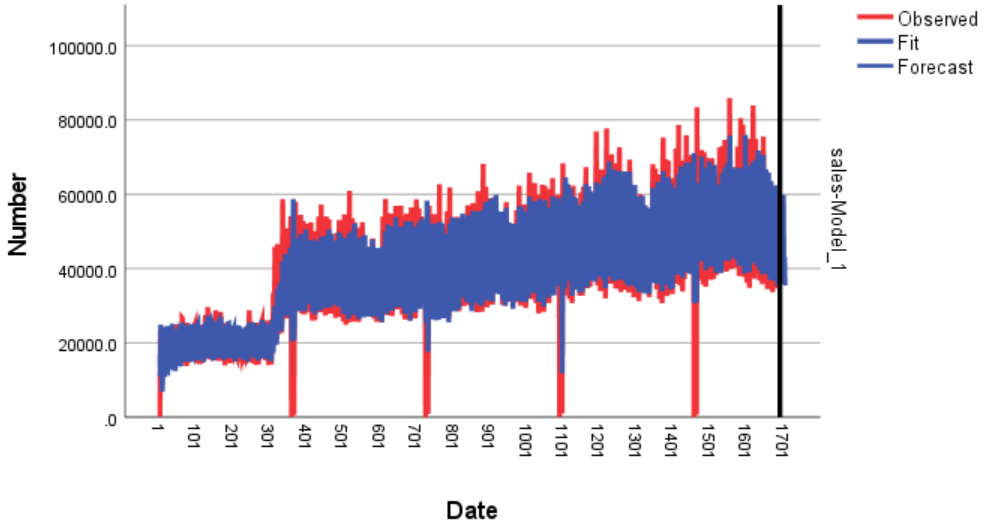

Fig. 5.2.4.3 ARIMA Forecast Chart

date	Predicted_sales_Model_1
2017/08/16	44431.2
2017/08/17	36601.5
2017/08/18	43594.3
2017/08/19	52708.8
2017/08/20	59518.1
2017/08/21	44979.7
2017/08/22	41810.4
2017/08/23	44673.4
2017/08/24	35744.2
2017/08/25	40898.6
2017/08/26	53541.6
2017/08/27	59814.9
2017/08/28	43107.6
2017/08/29	39345.7
2017/08/30	43212.6
2017/08/31	35482.4

Fig. 5.2.4.4 Cleaning Product Predicted Sales

5. DAIRY PRODUCTS:

Model Description

			Model Type
Model ID	sales	Model_1	ARIMA(14,1,14)

Fig. 5.2.5.1 ARIMA model for Dairy Products

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Model Fit statistics: R-squared	Model Fit statistics: MAE	Ljung-Box Q(18): Statistics	Ljung-Box Q(18): DF	Ljung-Box Q(18): Sig.	Number of Outliers
sales-Model_1	3	.765	.854	3199.251	.	0	.	0

Fig. 5.2.5.2 ARIMA model statistics for Dairy Products

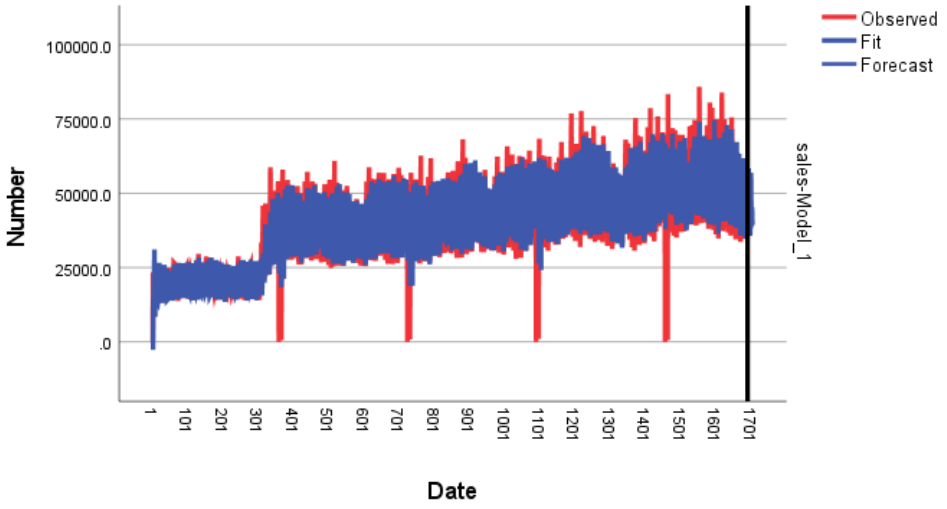

Fig. 5.2.5.3 ARIMA Forecast Chart

date	Predicted_sales_Model_1_A
2017/08/16	45567.5
2017/08/17	39403.4
2017/08/18	43887.6
2017/08/19	51611.5
2017/08/20	58381.4
2017/08/21	45601.1
2017/08/22	43300.2
2017/08/23	44376.5
2017/08/24	35721.6
2017/08/25	36599.8
2017/08/26	49444.4
2017/08/27	56959.8
2017/08/28	42640.1
2017/08/29	42226.7
2017/08/30	45336.1
2017/08/31	39338.1

Fig. 5.2.5.4 Dairy Products Predicted Sales

- **Insights:**

  In the above section, we have successfully forecasted the total sales for the top 5 selling product families: GROCERY I, BEVERAGES, PRODUCE, CLEANING, and DAIRY. The predicted sales for these product families provide valuable insights that can help the business optimise manufacturing processes. For example, to avoid potential shortages, the business could implement strategies such as producing 10% more than the forecasted demand for each product family. These strategies, however, are subject to the decisions made by stakeholders and leadership.

  This forecasting model also sets the stage for more research, like figuring out how much money will be made and other financial metrics, if it is given the right financial datasets and data. This would allow the business to not only plan for product demand but also project revenues and costs associated with these sales forecasts.

4.5.3 Business Problem 2: Forecasting Sales by Store for Top 5 Product Families

In the second approach, the aim was to predict store-specific sales for each of the top 5 selling product families. This allowed for more granular forecasting, which would support the optimisation of supply chains and enable the company to track store performance.

4.5.3.1 Segmentation and Feature Engineering for Store-Specific Forecasting

- Dataframe Segmentation: We segmented the dataset by store and product family, creating 5 new dataframes for each top-selling product family. This breakdown was crucial, as sales patterns vary significantly from one store to another based on location, promotions, and local events. The main dataset was sliced, and 5 data frames were framed based on the top 5 selling products. For instance, the Grocery_data data frame contained only data related to the Grocery category.

```
Grocery_Data = data[data['family']=="GROCERY I"]
```

Fig. 5.3.1.1 Slicing Data for Grocery I

- Feature Engineering: Additional features were added earlier in the main dataset, including holiday data, store-specific variables, and time-based features such as day of the week and month. As

noted by Zhang & Liu (2020), such external factors significantly impact sales patterns and need to be included in forecasting models. The figure shows the features that were included for each of the top 5 selling products in the data frame.

```
data.columns
```

```
Index(['store_nbr', 'sales', 'onpromotion', 'Batalla de Pichincha',
       'Black Friday', 'Cantonizacion de Cayambe',
       'Cantonizacion de El Carmen', 'Cantonizacion de Guaranda',
       'Cantonizacion de Latacunga', 'Cantonizacion de Libertad',
       ...
       'Traslado Batalla de Pichincha', 'Traslado Fundacion de Guayaquil',
       'Traslado Fundacion de Quito', 'Traslado Independencia de Guayaquil',
       'Traslado Primer Grito de Independencia', 'Traslado Primer dia del ano',
       'Viernes Santo', 'dayOfWeek', 'Month', 'WeekOfYear'],
```

Fig. 5.3.1.2 Features of top 5 selling products data individually

4.5.3.2 Model Selection, Training and Results

- To train the model and predict the next 16 days sales, we currently have the datasets for the top 5 selling products ready and transformed.

- We choose multiple models like LSTM (Long Short-Term Memory), Random Forest and XGBoost (Extreme Gradient Boosting) depending on the data and need. The models were first trained and evaluated, and then the best model was selected to perform predictions on the target dataset.

- LSTM (Long Short-Term Memory): LSTM is a type of recurrent neural network (RNN) designed to handle time series data with long-term dependencies. Its ability to retain past information over extended periods made it a suitable choice for capturing complex, store-specific sales patterns (Hochreiter & Schmidhuber, 1997). LSTM was particularly helpful in identifying long-term seasonality effects driven by holidays and promotions.

- Random Forest: The Random Forest Regressor was also trained on this segmented dataset. Random Forests are well-suited for handling high-dimensional datasets and complex feature

interactions, such as the effect of holidays, store location, and promotions on sales (Breiman, 2001). Its ensemble nature ensures a robust prediction even when dealing with non-linear relationships.

- XGBoost (Extreme Gradient Boosting): XGBoost, a popular machine learning model known for its efficiency and performance, was chosen as a competing model. XGBoost can effectively handle missing data, nonlinear relationships, and complex interactions between features (Chen & Guestrin, 2016). It was particularly beneficial for this multi-dimensional dataset with external variables like holidays and oil prices influencing sales patterns.

Results of models:

1. Training Model to forecast **GROCERY I**:

Model	**R-square**	**Mean Squared Error (RMSE)**	**Mean Absolute Error (MAE)**
Random Forest Model	0.87	994	554
Long Short-Term Memory (LSTM)	0.72	1643	1038
XGBOOST	0.90	869	512

Table 5.3.2.1 Model Comparison for Grocery I

- **Random Forest Model:**

```
Mean Absolute Error (MAE): 553.8988140224476
R-squared: 0.87039948440032
Mean Sqaure Error: 993.8131497950411
```

Fig. 5.3.2.1 Accuracy of Random Forest Model for Grocery I

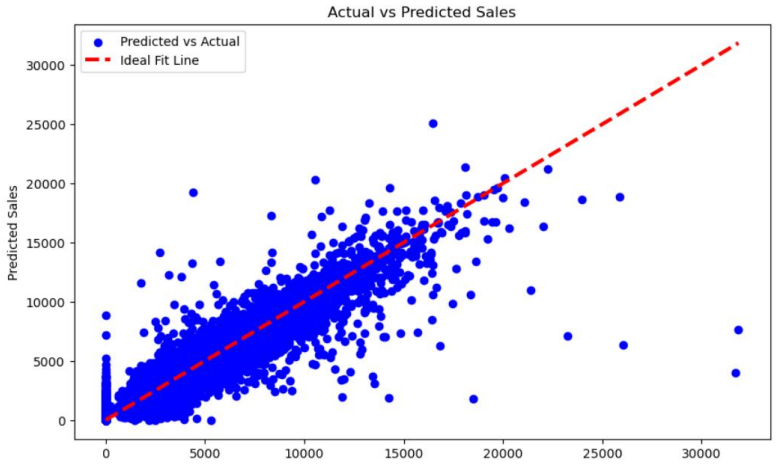

Fig. 5.3.2.2 Scatter Plot of Actual and Predicted Sales for Grocery I

- **Long Short-Term Memory (LSTM) Model:**

```
MAE: 1038.0091870127458, RMSE: 1642.9175325987999, R-squared: 0.725149925548705
```

Fig. 5.3.2.3 LSTM Model Accuracy for Grocery I

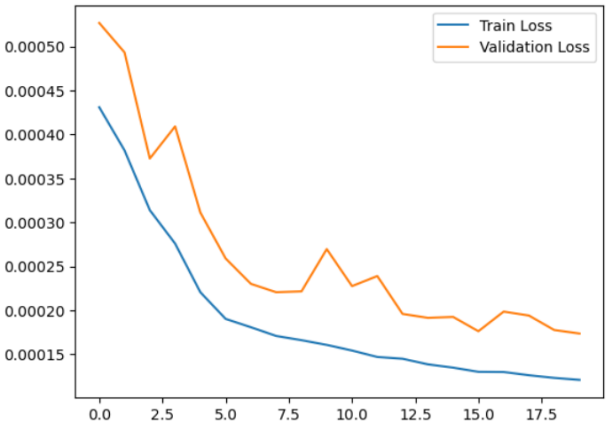

Fig. 5.3.2.4 LSTM model chart for Grocery I

- **XGBoost Model:**

```
Root Mean Squared Error (RMSE): 869.5542430782918
R-squared: 0.900781967935906
Root Absolute Error: 512.2697263342977
```

Fig. 5.3.2.5 XGBoost Model Accuracy for Grocery I

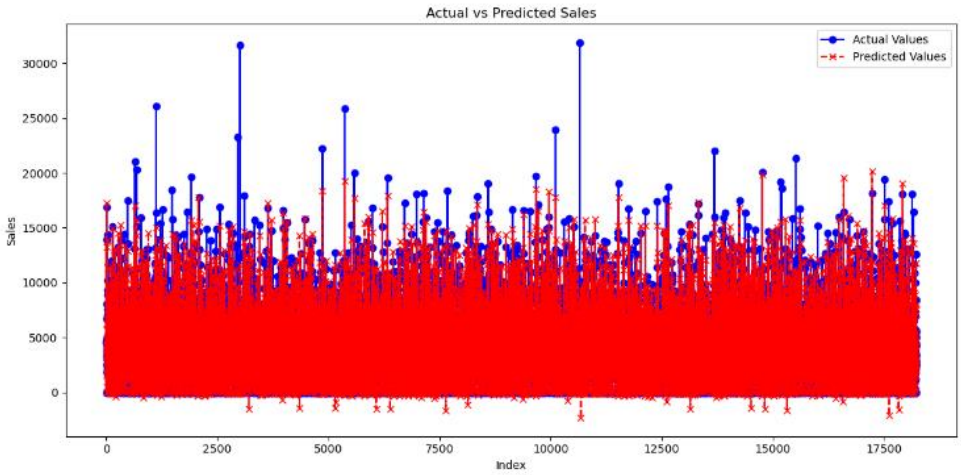

Fig. 5.3.2.6 Actual and Predicted sales by XGBoost Model for Grocery I

- As XGBoost model has performed the best, I have selected this model to forecast sales for the next 16 days.

	store_nbr	predicted_sales
date		
2017-08-16	37	3695.957031
2017-08-16	36	3678.981445
2017-08-16	35	1925.132568
2017-08-16	40	5053.300293
2017-08-16	41	3168.760010
...	...	...
2017-08-31	17	3721.531494
2017-08-31	15	2359.229492
2017-08-31	20	4286.153320
2017-08-31	18	2189.252441
2017-08-31	19	3100.937500

Fig. 5.3.2.7 Predicted Sales for Grocery I for next 16 days by XGBoost Model

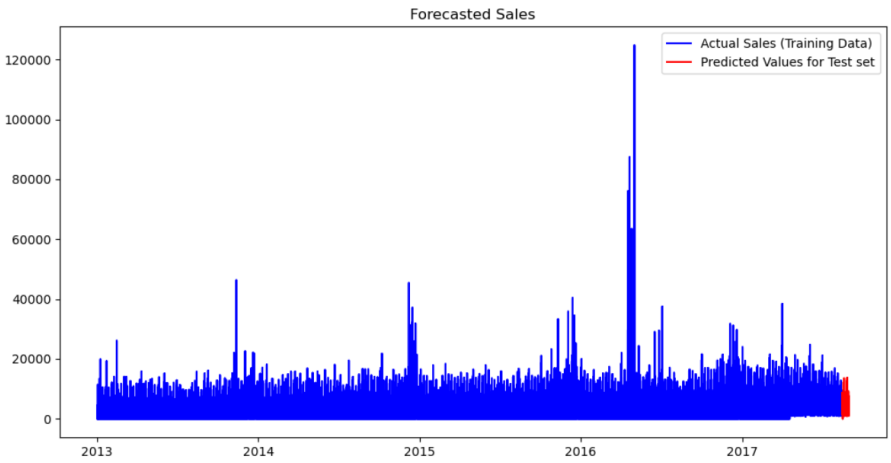

Fig. 5.3.2.8 Actual Sales and Predicted Sales by XGBoost Model for Grocery I

2. Training Model to forecast **BEVERAGES**:

Model	R-square	Mean Squared Error(RMSE)	Mean Absolute Error (MAE)
Random Forest Model	0.85	872	512
Long Short-Term Memory (LSTM)	0.72	1451	974
XGBOOST	0.89	752	468

Table 5.3.2.2 Model Comparison for Beverages

- **Random Forest Model:**

```
Mean Absolute Error (MAE): 512.1654861109033
R-squared: 0.8536091907622138
Mean Squared Error:872.6847766954794
```

Fig. 5.3.2.9 Accuracy of Random Forest Model for Beverages

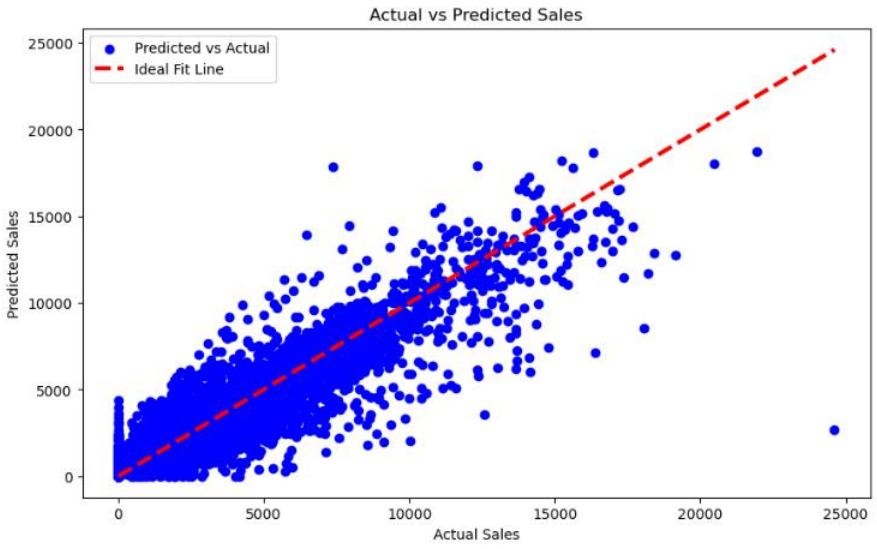

Fig. 5.3.2.10 Scatter Plot of Actual and Predicted Sales for Beverages

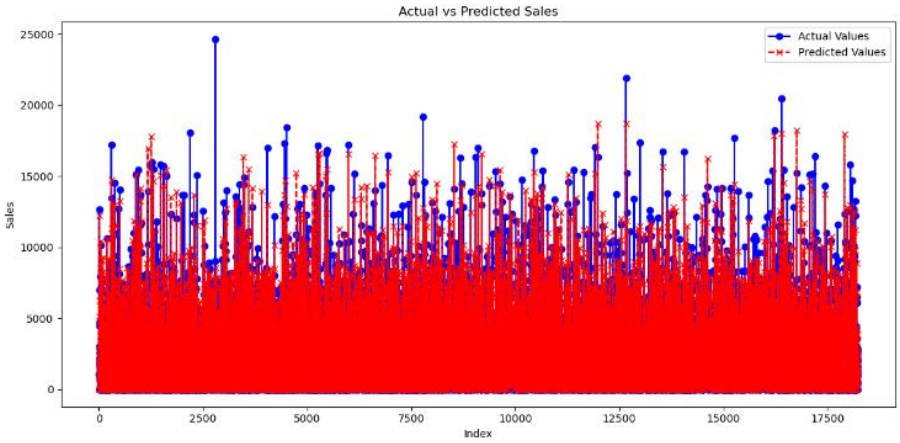

Fig. 5.3.2.11 Actual and Predicted sales by XGBoost Model for Beverages

- **Long Short-Term Memory (LSTM) Model:**

```
MAE: 974.9942561310784, RMSE: 1451.6077962174809, R-squared: 0.7209068289397895
```

Fig. 5.3.2.12 LSTM Model Accuracy for Beverages

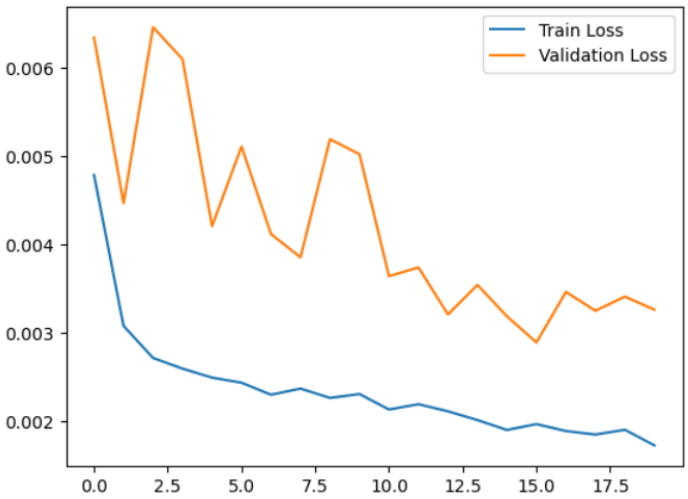

Fig. 5.3.2.13 LSTM model chart for Beverages

- **XGBoost Model:**

```
Root Mean Squared Error (RMSE): 752.0230651418168
R-squared: 0.8912920431714444
Root Absolute Error: 467.85679178301734
```

Fig. 5.3.2.14 XGBoost Model Accuracy for Grocery I

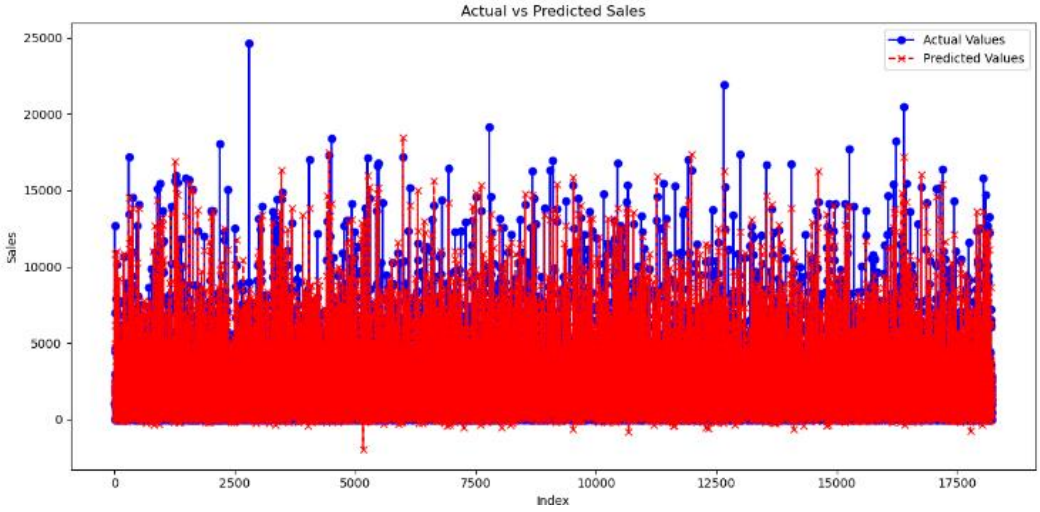

Fig. 5.3.2.15 Actual and Predicted sales by XGBoost Model for Beverages

- As XGBoost model has performed the best, I have selected this model to forecast sales for the next 16 days.

date	store_nbr	predicted_sales
2017-08-16	1	2166.784424
2017-08-16	2	2607.734375
2017-08-16	3	7409.901367
2017-08-16	4	2383.450684
2017-08-16	5	1673.129639
...	...	...
2017-08-31	50	2929.020752
2017-08-31	51	3484.655762
2017-08-31	52	3060.183838
2017-08-31	53	1792.419556
2017-08-31	54	1626.453857

Fig. 5.3.2.16 Predicted Sales for Beverages for next 16 days by XGBoost Model

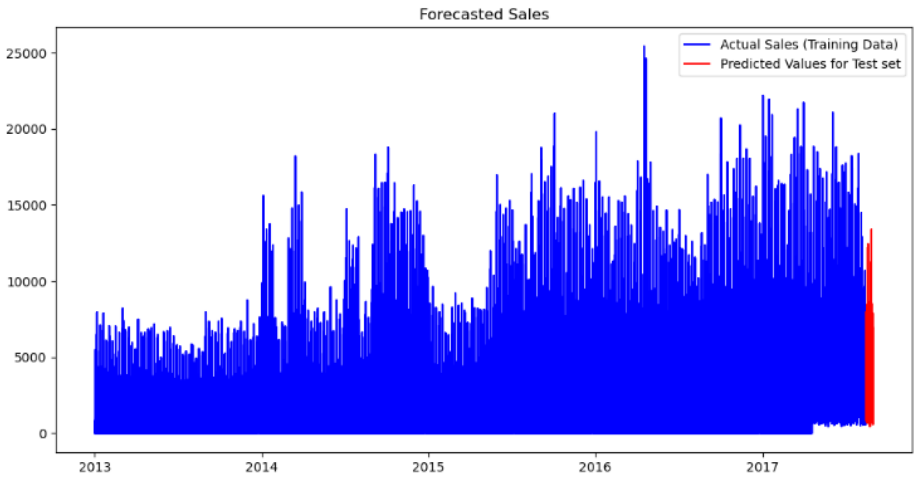

Fig. 5.3.2.17 Actual Sales and Predicted Sales by XGBoost Model for Beverages

3. Training Model to forecast **PRODUCE PRODUCTS**:

Model	**R-square**	**Mean Squared Error(RMSE)**	**Mean Absolute Error (MAE)**
Random Forest Model	0.55	702	1422
Long Short-Term Memory (LSTM)	0.57	1598	1018
XGBOOST	0.72	1121	644

Table 5.3.2.3 Model Comparison for Produce Product

- **Random Forest Model:**

```
Mean Absolute Error (MAE): 702.5151194716599
R-squared: 0.5558720162942536
Mean Square Error: 1422.155286278861
```

Fig. 5.3.2.18 Accuracy of Random Forest Model for Produce Product

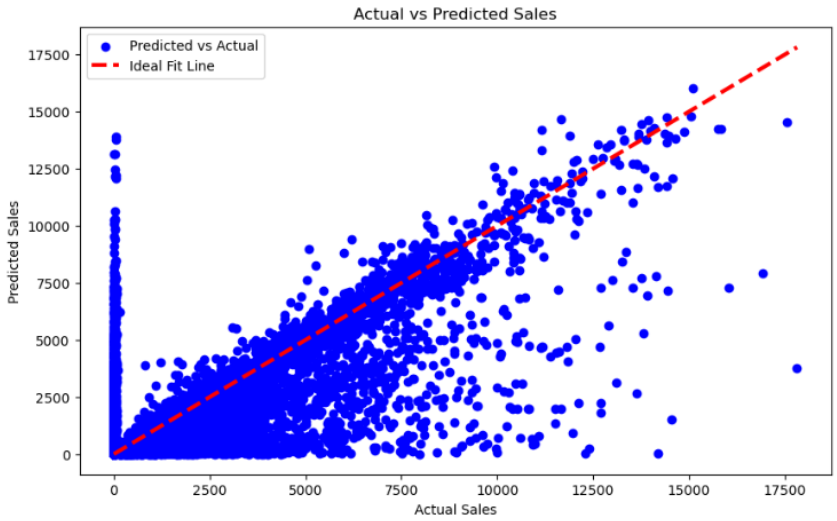

Fig. 5.3.2.19 Scatter Plot of Actual and Predicted Sales for Produce Product

- **Long Short-Term Memory (LSTM) Model:**

```
MAE: 1018.755586216459, RMSE: 1598.112373414184, R-squared: 0.5754233216952491
```

Fig. 5.3.2.20 LSTM Model Accuracy for Produce Product

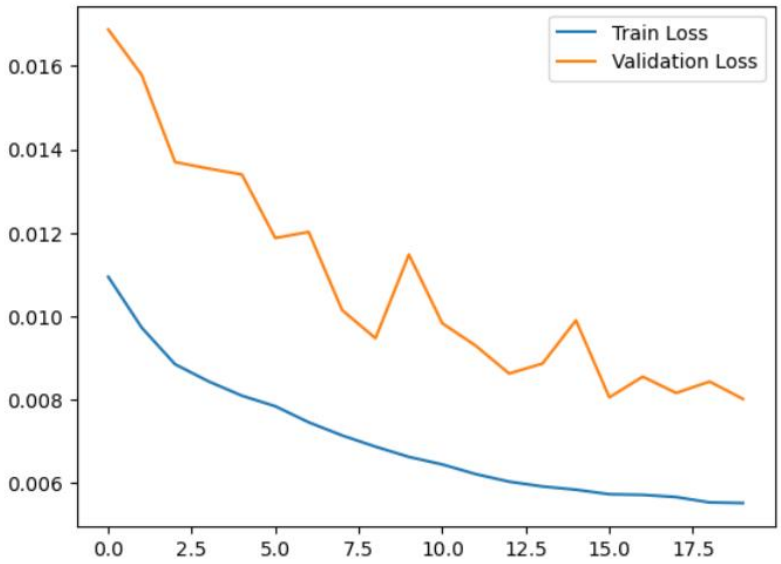

Fig. 5.3.2.21 LSTM model chart for Produce Product

- **XGBoost Model:**

```
Root Mean Squared Error (RMSE): 1121.4452984914465
R-squared: 0.7238339505779814
Root Absolute Error: 643.6993857684004
```

Fig. 5.3.2.22 XGBoost Model Accuracy for Produce Product

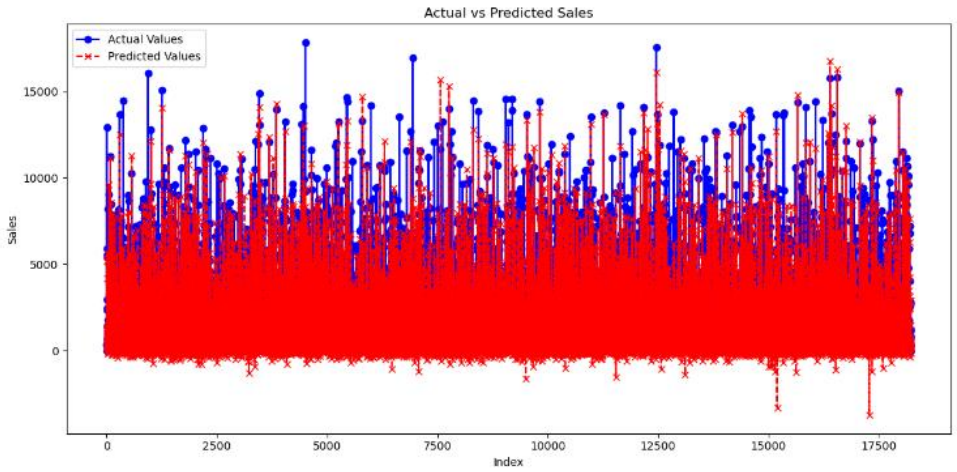

Fig. 5.3.2.23 Actual and Predicted sales by XGBoost Model for Produce Product

- As XGBoost model has performed the best, I have selected this model to forecast sales for the next 16 days.

date	store_nbr	predicted_sales
2017-08-16	1	3636.619385
2017-08-16	2	3163.784424
2017-08-16	3	10120.309570
2017-08-16	4	6268.323242
2017-08-16	5	5144.593750
...	...	...
2017-08-31	50	1501.730469
2017-08-31	51	3603.262207
2017-08-31	52	3258.220459
2017-08-31	53	1439.808105
2017-08-31	54	440.473389

Fig. 5.3.2.24 Predicted Sales for Produce Product for next 16 days by XGBoost Model

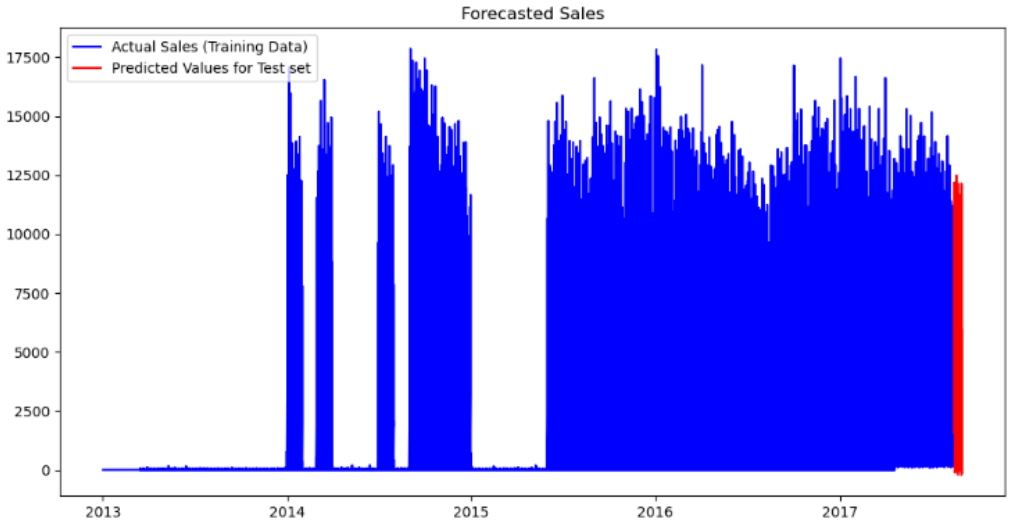

Fig. 5.3.2.25 Actual and Predicted sales by XGBoost Model for Produce Product

4. Training Model to forecast **CLEANING PRODUCTS**:

Model	R-square	Mean Squared Error(RMSE)	Mean Absolute Error (MAE)
Random Forest Model	0.81	313	181
Long Short-Term Memory (LSTM)	0.52	550	346
XGBOOST	0.85	280	163

Table 5.3.2.4 Model Comparison for Cleaning Product

- **Random Forest Model:**

```
Mean Absolute Error (MAE): 181.29695899158256
R-squared: 0.8174279414447347
MSE: 313.53122466257435
```

Fig. 5.3.2.26 Accuracy of Random Forest Model for Cleaning Product

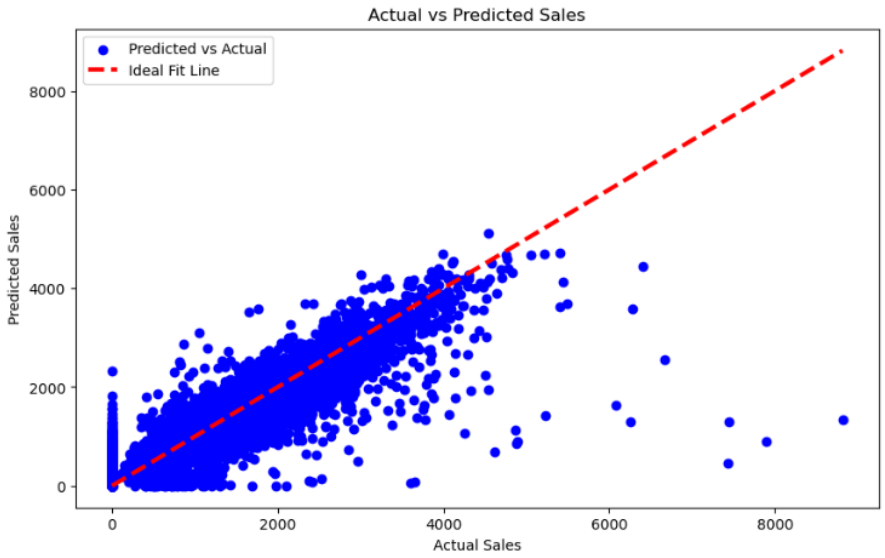

Fig. 5.3.2.27 Scatter Plot of Actual and Predicted Sales for Cleaning Product

- **Long Short-Term Memory (LSTM) Model:**

```
MAE: 346.1950889674038, RMSE: 550.0523735449259, R-squared: 0.5184102704239659
```

Fig. 5.3.2.28 LSTM Model Accuracy for Cleaning Product

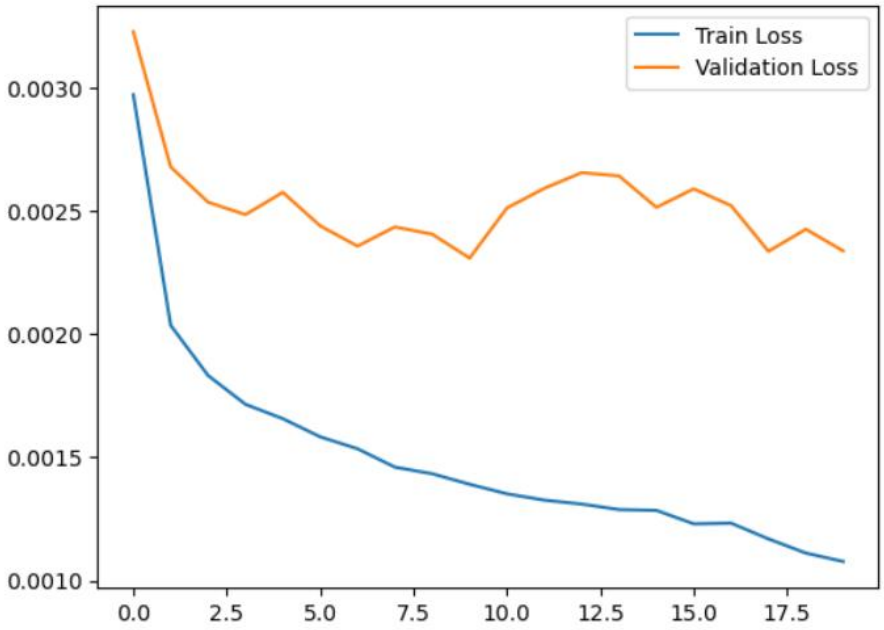

Fig. 5.3.2.29 LSTM model chart for Cleaning Product

- **XGBoost Model:**

```
Root Mean Squared Error (RMSE): 280.40515974017904
R-squared: 0.853969115273415
Root Absolute Error: 163.1966738616287
```

Fig. 5.3.2.30 XGBoost Model Accuracy for Cleaning Product

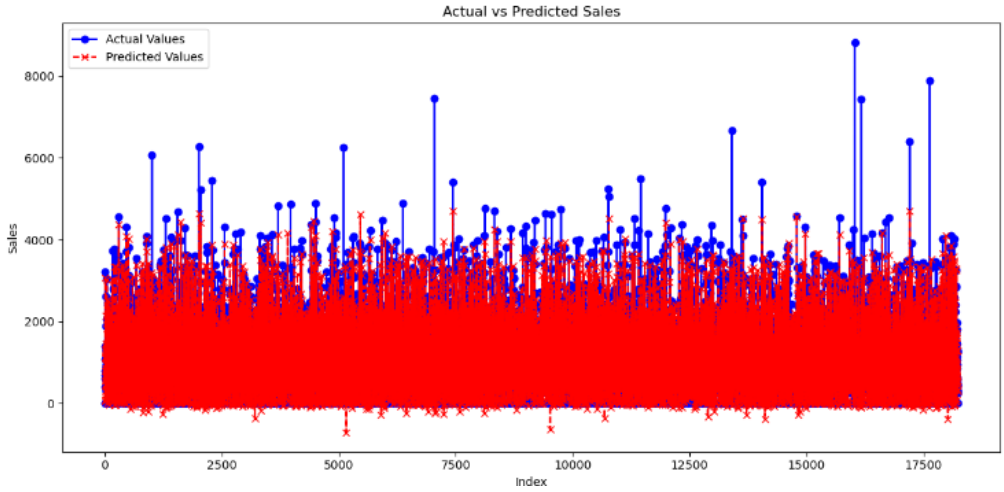

Fig. 5.3.2.31 Actual and Predicted sales by XGBoost Model for Cleaning Product

- As XGBoost model has performed the best, I have selected this model to forecast sales for the next 16 days.

date	store_nbr	predicted_sales
2017-08-16	1	837.974976
2017-08-16	2	950.516052
2017-08-16	3	1931.981201
2017-08-16	4	802.698425
2017-08-16	5	885.976074
...	...	...
2017-08-31	50	1169.448975
2017-08-31	51	1036.421631
2017-08-31	52	1149.588501
2017-08-31	53	937.570190
2017-08-31	54	751.831238

Fig. 5.3.2.32 Predicted Sales for Cleaning Product for next 16 days by XGBoost Model

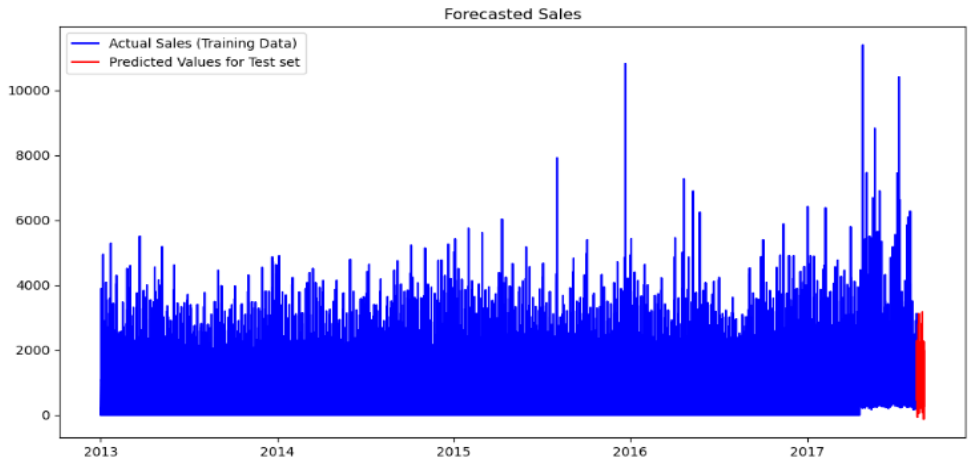

Fig. 5.3.2.33 Actual and Predicted sales by XGBoost Model for Cleaning Product

5. Training Model to forecast **DAIRY PRODUCTS**:

Model	**R-square**	**Mean Squared Error(RMSE)**	**Mean Absolute Error (MAE)**
Random Forest Model	0.86	246	153
Long Short-Term Memory (LSTM)	0.64	438	297
XGBOOST	0.91	190	127

Table 5.3.2.5 Model Comparison for Dairy Products

- **Random Forest Model:**

```
Mean Absolute Error (MAE): 153.05953829284795
R-squared: 0.8615269466803362
MSE: 246.50212234898692
```

Fig. 5.3.2.34 Accuracy of Random Forest Model for Dairy Product

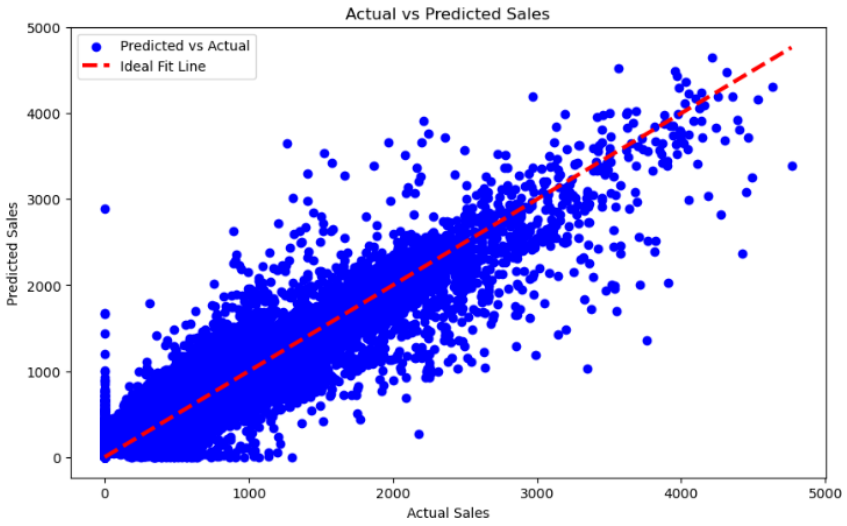

Fig. 5.3.2.35 Scatter Plot of Actual and Predicted Sales for Dairy Product

- **Long Short-Term Memory (LSTM):**

```
MAE: 297.5100846485663, RMSE: 438.1090345944414, R-squared: 0.64941101103331
```

Fig. 5.3.2.36 LSTM Model Accuracy for Dairy Product

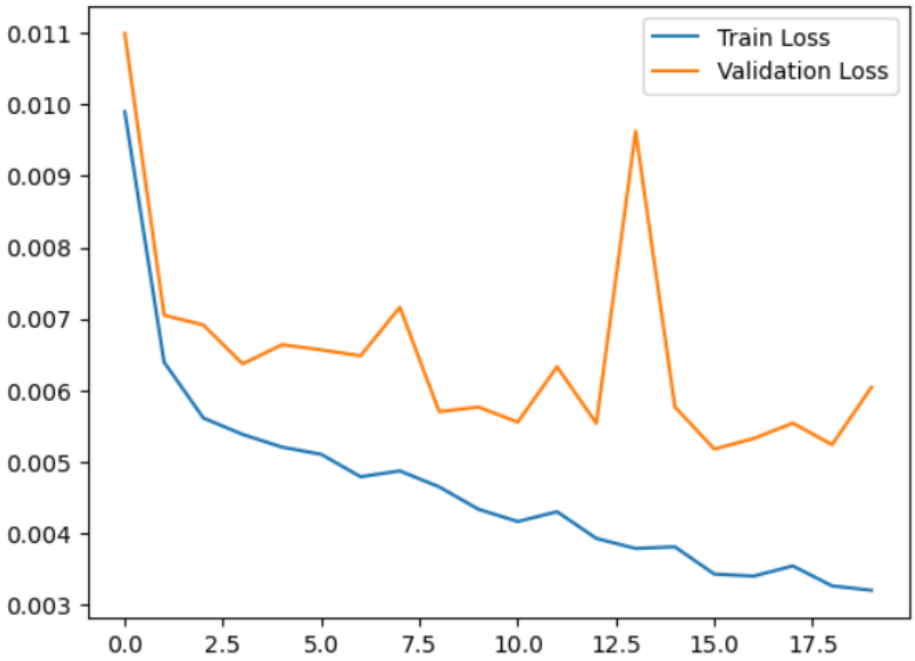

Fig. 5.3.2.37 LSTM model chart for Dairy Product

- **XGBoost Model:**

```
Root Mean Squared Error (RMSE): 190.09615344641358
R-squared: 0.9176486731426451
Root Absolute Error: 127.37799849153859
```

Fig. 5.3.2.38 XGBoost Model Accuracy for Dairy Product

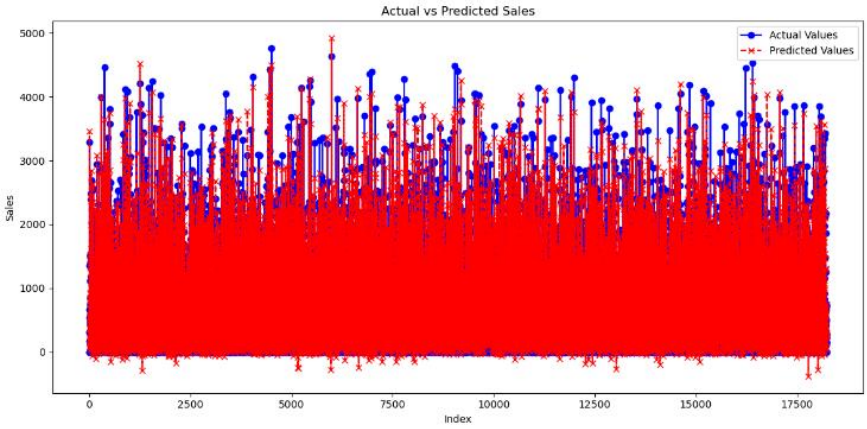

Fig. 5.3.2.39 Actual and Predicted sales by XGBoost Model for Dairy Produce

- As XGBoost model has performed the best, I have selected this model to forecast sales for the next 16 days.

date	store_nbr	predicted_sales
2017-08-16	1	839.869080
2017-08-16	2	831.690063
2017-08-16	3	2109.257812
2017-08-16	4	773.087952
2017-08-16	5	546.965210
...	...	...
2017-08-31	50	875.162231
2017-08-31	51	1163.939819
2017-08-31	52	835.581970
2017-08-31	53	501.653473
2017-08-31	54	129.895523

Fig. 5.3.2.40 Predicted Sales for Dairy Product for next 16 days by XGBoost Model

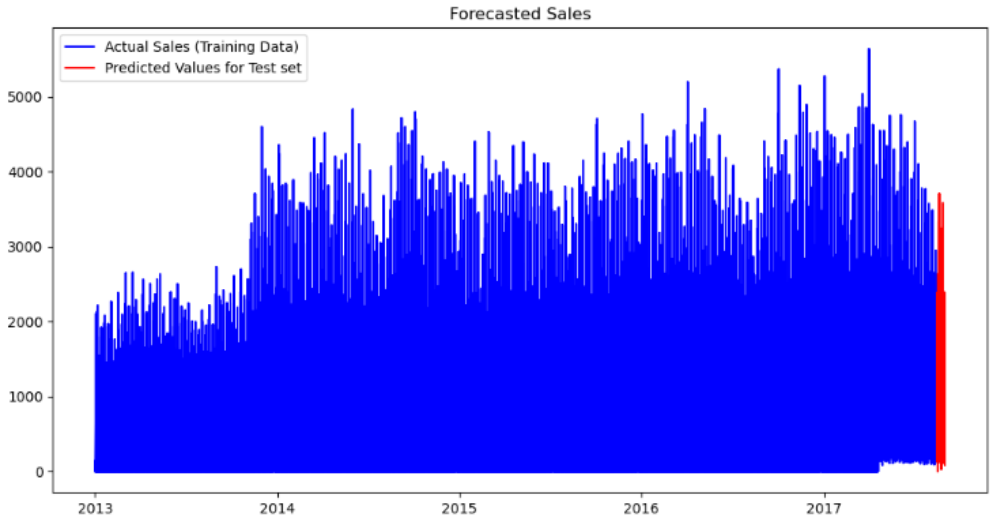

Fig. 5.3.2.41 Actual and Predicted sales by XGBoost Model for Dairy Products

- **Insights**:

  The above models have helped us to forecast sales for top 5 selling products for next 16 days for each store, considering the holidays, events, and time specific features. Now, we know the demand of each store for top 5 selling products.

4.5.3.3 Model Tuning and Selection

- Hyperparameter Tuning: All three models (LSTM, Random Forest, XGBoost) were fine-tuned using cross-validation and grid search to identify the best hyperparameters for each product family and store-specific dataset. The results you see above are after hyperparameter tuning the models. Key metrics such as RMSE and MAE were used to evaluate model performance (Hyndman & Athanasopoulos, 2018).
- Hyperparameter Tuning Random Forest Regressor Model:

```
rf_model = RandomForestRegressor(n_estimators=100,random_state=42)
rf_model.fit(X_train,y_train)
```

Before Tuning

```
rf_model = RandomForestRegressor(n_estimators=150,random_state=42)
rf_model.fit(X_train,y_train)
```

After Tuning

- Hyperparameter Tuning LSTM Model:

```
# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
```

Before Tuning

```
# Train the model
history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))
```

After Tuning

- Hyperparameter Tuning XGBoost Model:

```
# Initialize the XGBoost model
xgboost_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100, random_state=42)
```

Before Tuning

```
# Initialize the XGBoost model
xgboost_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=150, random_state=42)
```

After Tuning

- Final Model Selection: After comparing the performance of all three models, the one with the best results (lowest RMSE and MAE) i.e. XGBoost model was selected to forecast the sales for the next 16 days. The model was used to generate forecasts for each product family at each store, helping the company predict store-level sales patterns.

4.5.4 Business Value: Supply Chain Optimisation and Store Performance Tracking

The forecasts generated by the second approach provide several operational advantages:

- Supply Chain Optimisation: By predicting the sales volume for each product family at the store level, the company can better manage inventory levels and optimise supply chain logistics. Stores expecting high sales can be better stocked, while stores with lower forecasts can adjust their inventory to avoid excess stock. This insight is crucial for managing product distribution efficiently (Ferreira et al., 2016).

- Store Performance Analysis: In addition to predicting sales, the forecasts also serve as a tool for tracking store performance. By comparing the actual sales with forecasted values, the company can identify underperforming stores and take corrective action, such as adjusting marketing efforts or changing store layouts.

4.5.5 Store Performance for Top 5 Selling Products

- **GROCERY I:**

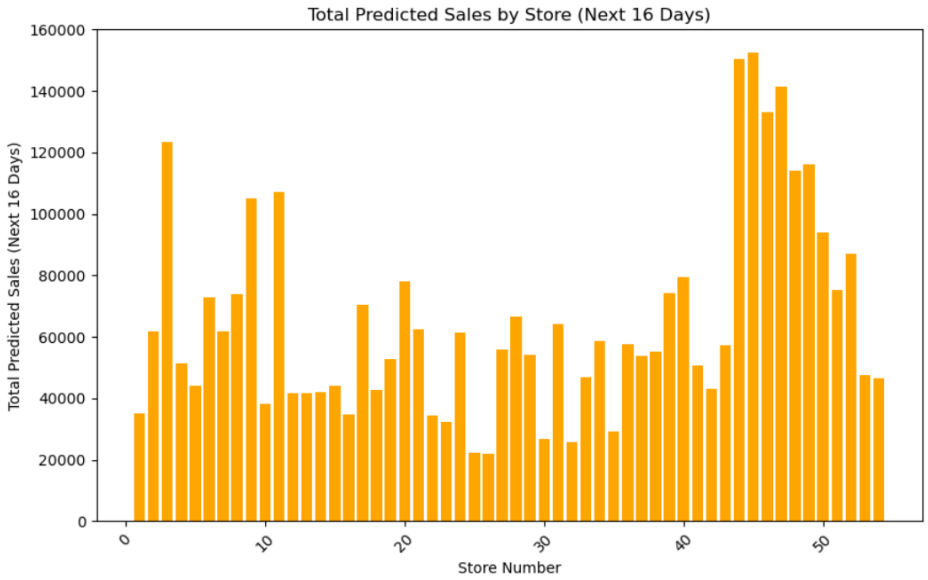

Fig. 5.4.1 Store Performance for Grocery I

- Analysis: The plot shows that store 3, 9, 11, 44, 45, 46, 47, and 48 should be expecting high Grocery Sales for the next 16 days.

- **BEVERAGES:**

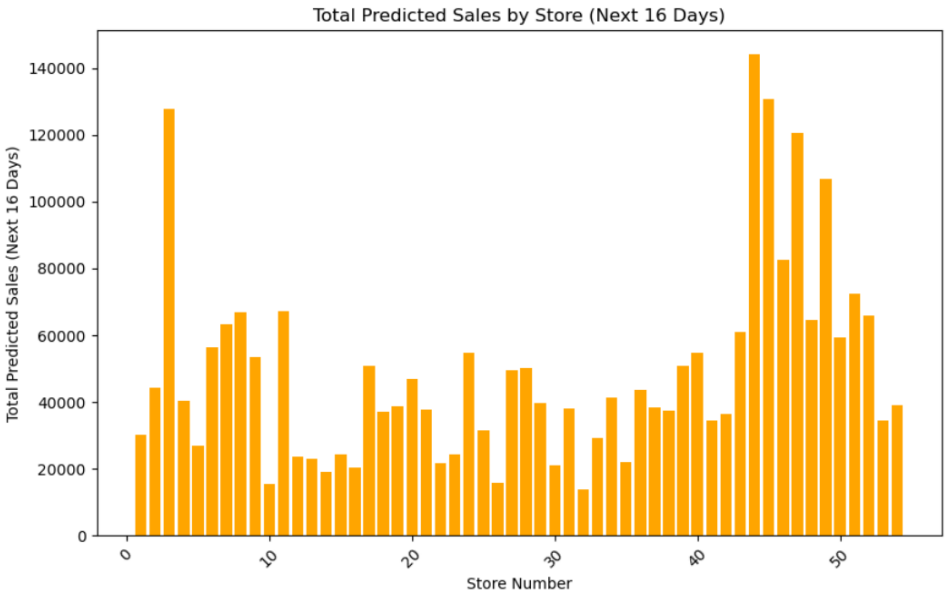

Fig. 5.4.2 Store Performance for Beverages

- Analysis: The plot shows that stores 3, 44, 45, 47, and 49 should expect high Beverage sales.

- **PRODUCE PRODUCT**:

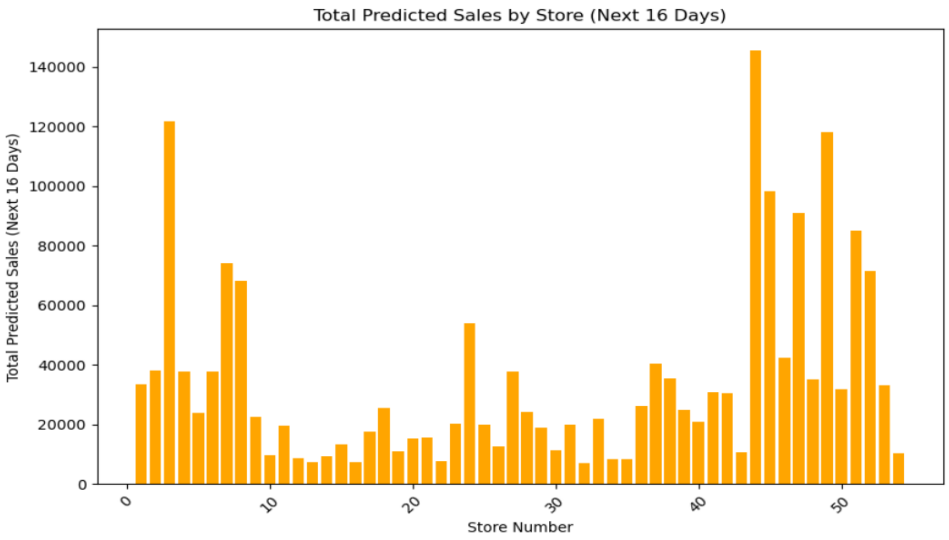

Fig. 5.4.3 Store Performance for Produce Products

- Analysis: The plot shows that stores 3, 44, 45, 47, and 49 should expect high Produce product sales.

- **CLEANING PRODUCT:**

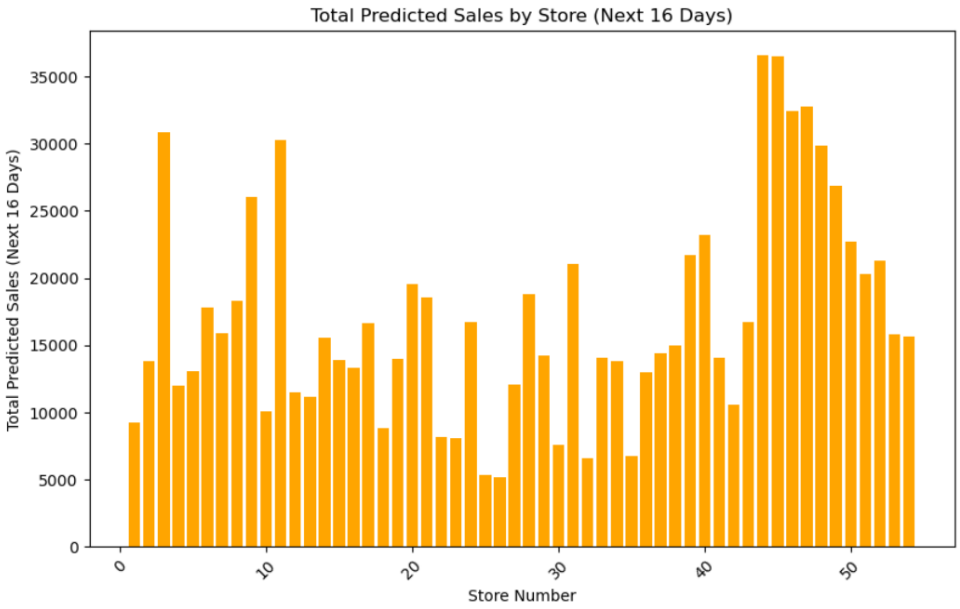

Fig. 5.4.4 Store Performance for Cleaning Products

- Analysis: The plot shows that stores 3, 9, 11, 44, 45, 47, 48, and 49 should expect high Cleaning Product sales.

- **DAIRY PRODUCT:**

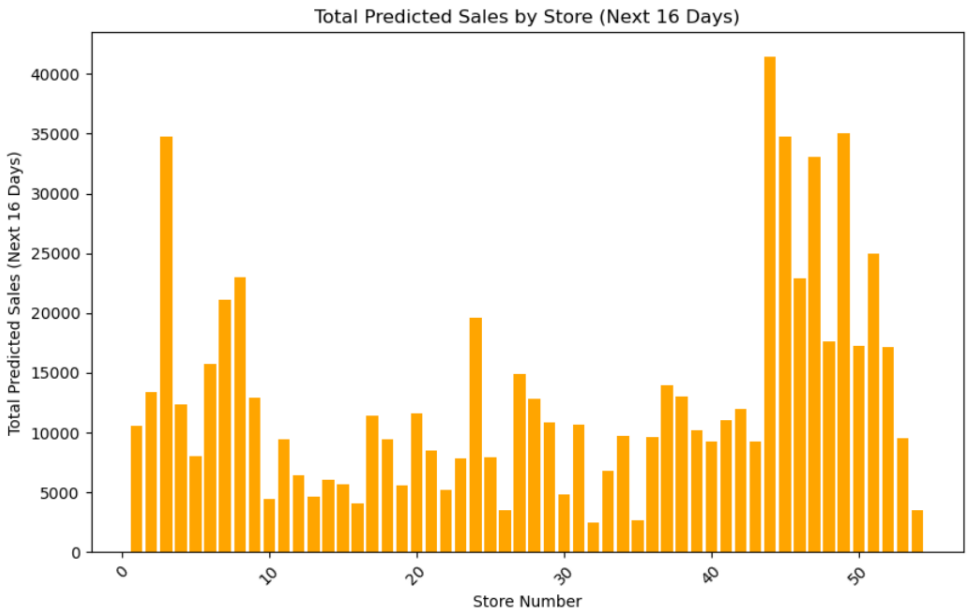

Fig. 5.4.5 Store Performance for Dairy Products

- Analysis: The plot shows that stores 3, 44, 45, 47, and 49 should expect high Dairy Product sales.

- **Insights:**

The overall top performing stores are 3, 44, 45, 47, and 49. We must align the supply with each store's demand now that we have a clear understanding of the demand for product families across various stores. To further refine this process, having access to additional datasets—such as store locations, transportation costs, and transportation methods—would enable more precise and effective supply chain optimization. Incorporating these factors would allow for more accurate analysis and decision-making regarding logistics, helping to minimise transportation costs and improve overall efficiency in product distribution.

The above analysis helps us to reach the conclusion that we have answered research question 3,4 & 5, were we aimed to find out the top performing stores and the correlation between their sales. Now, we have the list of top performing stores which will help the business to better optimise supply and focus on low performing stores.

5. CONCLUSION & RECOMMENDATIONS

CONCLUSION

This dissertation focused on building accurate sales forecasting models for **Corporación Favorita**, a large Ecuadorian retail chain. By leveraging a variety of data sources, including sales, holidays, and oil prices, the study aimed to provide reliable sales predictions for the top-selling product families, both at the store and aggregate levels.

Through **Exploratory Data Analysis (EDA)**, it was observed that significant external factors such as holidays and oil prices strongly influence sales patterns, demonstrating the need for careful data preprocessing and feature engineering. Incorporating these insights into the forecasting models improved the accuracy of predictions.

The two-pronged approach to forecasting—first predicting total sales for the top-selling product families and then predicting store-level sales—allowed for detailed and actionable insights. The use of models like **ARIMA**, **LSTM**, **Random Forest**, and **XGBoost** proved effective in capturing both linear and non-linear trends in the data.

From a business perspective, Corporación Favorita can benefit from the findings of this study by optimising their supply chain management, adjusting their manufacturing processes, and making informed decisions about inventory allocation. Forecasting demand at both the manufacturing and distribution levels can help achieve this. aggregate and store levels. levels, the company can prevent shortages, minimise overstocking, and ensure operational efficiency during peak seasons.

The study's findings show how important it is to use both traditional time series methods and more advanced machine learning methods together to make forecasting more accurate in complicated retail settings. To improve supply chain optimisation, future work could refine these models by incorporating additional datasets, such as transportation costs, products under the product families, and store locations.

RECOMMENDATIONS

- **Product-Level Forecasting**: The dataset utilised in this analysis was deficient in comprehensive sub-product details within each product category. The availability of product-level data would provide more accurate demand predictions for each specific product. This would allow Corporación Favorita to optimise manufacturing quantities with greater precision, minimising waste and enhancing inventory management.

- **Automation of the Forecasting Process**: Utilising modern servers and processing capabilities, the forecasting model may be automated. Utilising loops, the dataset could be systematically categorised by product family, and models would be trained and assessed. The optimal model for each category will be chosen to forecast sales for the subsequent 15-16 days, therefore obviating the necessity for manual interventions.

- **Enhanced Decision-Making**: Automating and optimising the forecasting process at the product level would enable the organisation to more effectively react to market trends and demand changes. This would improve operational efficiency and offer significant data for decision-making at both product and supply chain levels.

6.REFERENCES

- Adhikari, R. & Agrawal, R. K., 2013. An introductory study on time series modelling and forecasting. arXiv preprint arXiv:1302.6613.
- Azzalini, A. and Scarpa, B., 2012. Data Analysis and Modelling of Holidays Impact on Retail Sales. Springer-Verlag.
- Bengio, Y., 2012. Practical recommendations for gradient-based training of deep architectures. In Neural networks: Tricks of the trade (pp. 437-478). Springer, Berlin, Heidelberg.
- Bengio, Y., Courville, A. & Vincent, P., 2012. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence, 35(8), pp. 1798-1828.
- Box, G.E.P., Jenkins, G.M. and Reinsel, G.C., 2015. Time Series Analysis: Forecasting and Control. 5th ed. Hoboken, NJ: Wiley.
- Breiman, L., 2001. Random Forests. Machine Learning, 45(1), pp. 5-32.
- Brockwell, P.J. and Davis, R.A., 2016. Introduction to Time Series and Forecasting. Springer.
- Chandola, V., Banerjee, A. and Kumar, V., 2009. Anomaly Detection: A Survey. ACM Computing Surveys, 41(3), pp.1-58.
- Chen, T. & Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 785-794.
- Choi, T.-M., Hui, C.-L. & Bell, D. R., 2021. Distribution-free inventory strategies for holiday seasons and their effects on consumer demand. European Journal of Operational Research, 295(1), pp. 241-253.
- Duan, J., Cao, Q., and Xu, J., 2019. Sales Forecasting in Retail: A Machine Learning Approach. Journal of Retail Analytics, 14(3), pp.212-225.
- Ferreira, K. J., Lee, B. H. & Simchi-Levi, D., 2016. Analytics for an online retailer: Demand forecasting and price optimization. Manufacturing & Service Operations Management, 18(1), pp. 69-88.
- Hands-on Cloud, 2023. *Using the ARIMA model and Python for Time Series forecasting*.
- Hochreiter, S. & Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp. 1735-1780.
- Hyndman, R.J. and Athanasopoulos, G., 2018. Forecasting: Principles and Practice. 2nd ed. OTexts.
- Ibarra, D., 2014. Oil Price Fluctuations and Economic Impact in Ecuador: A Time Series Analysis. Journal of Latin American Economics, 10(2), pp.112-125.

- Kotsiantis, S.B., 2013. Data Preprocessing for Supervised Learning. International Journal of Data Mining & Knowledge Management Process (IJDKP), 1(2), pp.1-13.
- Lazzeri, F., 2020. Machine Learning for Time Series Forecasting with Python. Wiley.
- Little, R. J. & Rubin, D. B., 2019. Statistical Analysis with Missing Data. 3rd ed. Hoboken, NJ: John Wiley & Sons.
- Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., & Király, F. J., 2019. sktime: A Unified Interface for Machine Learning with Time Series. *arXiv preprint arXiv:1909.07872*.
- Makridakis, S., Spiliotis, E. and Assimakopoulos, V., 2018. Statistical and Machine Learning Forecasting Methods: Concerns and Ways Forward. PLOS ONE, 13(3), pp.1-26.
- Patil, K., Sawant, A., Rane, S., 2023. *Times Series Sales Forecasting using ARIMA Model*. IRJMETS, [online]
- Rana, A., Goel, R., & Tiwari, P., 2021. Transaction-based retail sales forecasting using machine learning techniques. Journal of Retail Analytics, 45(2), pp. 53-68.
- Rizvi, M.F., 2024. *Time Series Sales Forecasting using ARIMA Model*. International Journal of Recent Advances in Engineering Science & Technology.
- Villanueva, A., 2019. Oil Price Fluctuations and Consumer Spending in Oil-Dependent Economies: Evidence from Ecuador. Latin American Economic Review, 28(2), pp. 117-134.
- Zhang, X. & Liu, L., 2020. Retail sales forecasting using machine learning models: A case study of holiday effects. Journal of Retailing and Consumer Services, 56(1), pp. 35-45.
- Zhao, H., Zhang, J., and Liu, Y., 2020. Machine Learning Approaches for Sales Forecasting in Retail. International Journal of Data Science, 5(1), pp.100-115.

7. APPENDICES

7.1 Trained models snippets for Business Problem 2: Forecasting Sales by Store for Top 5 Product Families.

1. Grocery I:

- Random Forest Regressor Model:

```
X = data.drop(columns=['sales'])
y = data['sales']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)

rf_model = RandomForestRegressor(n_estimators=150,random_state=42)
rf_model.fit(X_train,y_train)

y_pred = rf_model.predict(X_test)

mae = mean_absolute_error(y_test,y_pred)
r2 = r2_score(y_test, y_pred)
rme = mean_squared_error(y_test,y_pred,squared=False)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"R-squared: {r2}")
print(f"Mean Sqaure Error: {rme}")

# Plot actual vs. predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=3, label='Ideal Fit Line')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs Predicted Sales')
plt.legend()
plt.show()
```

```
Mean Absolute Error (MAE): 553.8988140224476
R-squared: 0.87039948440032
Mean Sqaure Error: 993.8131497950411
```

- Long Short-Term Memory Model (LSTM):

```
X = data.drop('sales',axis=1)
y = data['sales'].values

# Scale the features and target
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.reshape(-1, 1))

# Function to create sequences for LSTM
def create_sequences(X, y, time_steps=7):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:i + time_steps])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

# Create sequences with a time step of 7 days
time_steps = 7
X_seq, y_seq = create_sequences(X_scaled, y_scaled, time_steps)

# Split into training and test sets
train_size = int(0.8 * len(X_seq))
X_train, X_test = X_seq[:train_size], X_seq[train_size:]
y_train, y_test = y_seq[:train_size], y_seq[train_size:]

# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, activation='relu', return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(LSTM(units=50, activation='relu'))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

# Make predictions
y_pred_scaled = model.predict(X_test)
y_pred = scaler_y.inverse_transform(y_pred_scaled)
y_test_actual = scaler_y.inverse_transform(y_test)

# Calculate error metrics
mae_lstm = mean_absolute_error(y_test_actual, y_pred)
rmse_lstm = np.sqrt(mean_squared_error(y_test_actual, y_pred))
r2_lstm = r2_score(y_test_actual, y_pred)

print(f'MAE: {mae_lstm}, RMSE: {rmse_lstm}, R-squared: {r2_lstm}')

# Plot loss history
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
```

- XGBoost Model:

```
# Split the data into features (X) and target (y)
X = data.drop(columns=['sales'])  # All columns except 'sales' are features
y = data['sales']  # Target variable is 'sales'

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the XGBoost model
xgboost_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=150, random_state=42)

# Train the model
xgboost_model.fit(X_train, y_train)

# Make predictions
y_pred = xgboost_model.predict(X_test)

# Evaluate the model: Calculate the Root Mean Squared Error (RMSE) and R-squared
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
rae = mean_absolute_error(y_test, y_pred)

print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared: {r2}")
print(f"Root Absolute Error: {rae}")

# Optionally, print the first few actual vs predicted values
comparison_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).reset_index(drop=True)
print(comparison_df.head())
```

```
Root Mean Squared Error (RMSE): 869.5542430782918
R-squared: 0.900781967935906
Root Absolute Error: 512.2697263342977
```

2. Beverages:

- Random Forest Model:

```
X = data.drop(columns=['sales'])
y = data['sales']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)

rf_model = RandomForestRegressor(n_estimators=150,random_state=42)
rf_model.fit(X_train,y_train)

y_pred = rf_model.predict(X_test)

mae = mean_absolute_error(y_test,y_pred)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test,y_pred, squared=False)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"R-squared: {r2}")
print(f'Mean Squared Error:{mse}')

# Plot actual vs. predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=3, label='Ideal Fit Line')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs Predicted Sales')
plt.legend()
plt.show()
```

```
Mean Absolute Error (MAE): 512.1654861109033
R-squared: 0.8536091907622138
Mean Squared Error:872.6847766954794
```

- Long Short-Term Memory (LSTM) model:

```
# Selecting relevant features for training
features = ['store_nbr', 'onpromotion', 'dayOfWeek', 'Month', 'WeekOfYear']
X = df[features]
y = df['sales'].values

# Scale the features and target
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.reshape(-1, 1))

# Function to create sequences for LSTM
def create_sequences(X, y, time_steps=7):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:i + time_steps])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

# Create sequences with a time step of 7 days
time_steps = 7
X_seq, y_seq = create_sequences(X_scaled, y_scaled, time_steps)

# Split into training and test sets
train_size = int(0.8 * len(X_seq))
X_train, X_test = X_seq[:train_size], X_seq[train_size:]
y_train, y_test = y_seq[:train_size], y_seq[train_size:]

# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, activation='relu', return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(LSTM(units=50, activation='relu'))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

# Make predictions
y_pred_scaled = model.predict(X_test)
y_pred = scaler_y.inverse_transform(y_pred_scaled)
y_test_actual = scaler_y.inverse_transform(y_test)

# Calculate error metrics
mae_lstm = mean_absolute_error(y_test_actual, y_pred)
rmse_lstm = np.sqrt(mean_squared_error(y_test_actual, y_pred))
r2_lstm = r2_score(y_test_actual, y_pred)

print(f'MAE: {mae_lstm}, RMSE: {rmse_lstm}, R-squared: {r2_lstm}')

# Plot loss history
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
```

- XGBoost Model:

```
X = data.drop(columns=['sales'])  # All columns except 'sales' are features
y = data['sales']  # Target variable is 'sales'

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the XGBoost model
xgboost_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=150, random_state=42)

# Train the model
xgboost_model.fit(X_train, y_train)

# Make predictions
y_pred = xgboost_model.predict(X_test)

# Evaluate the model: Calculate the Root Mean Squared Error (RMSE) and R-squared
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
rae = mean_absolute_error(y_test, y_pred)

print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared: {r2}")
print(f"Root Absolute Error: {rae}")

# Optionally, print the first few actual vs predicted values
comparison_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).reset_index(drop=True)
print(comparison_df.head())
```

```
Root Mean Squared Error (RMSE): 752.0230651418168
R-squared: 0.8912920431714444
Root Absolute Error: 467.85679178301734
```

3. Produce:
   - Random Forest Model:

```
X = data.drop(columns=['sales'])
y = data['sales']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)

rf_model = RandomForestRegressor(n_estimators=100,random_state=42)
rf_model.fit(X_train,y_train)

y_pred = rf_model.predict(X_test)

mae = mean_absolute_error(y_test,y_pred)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test,y_pred,squared=False)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"R-squared: {r2}")
print(f'Mean Square Error: {mse}')

# Plot actual vs. predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=3, label='Ideal Fit Line')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs Predicted Sales')
plt.legend()
plt.show()
```

```
Mean Absolute Error (MAE): 702.5151194716599
R-squared: 0.5558720162942536
Mean Square Error: 1422.155286278861
```

- Long Short-Term Memory (LSTM) model:

```
X = data.drop('sales',axis=1)
y = data['sales'].values

# Scale the features and target
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.reshape(-1, 1))

# Function to create sequences for LSTM
def create_sequences(X, y, time_steps=7):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:i + time_steps])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

# Create sequences with a time step of 7 days
time_steps = 7
X_seq, y_seq = create_sequences(X_scaled, y_scaled, time_steps)

# Split into training and test sets
train_size = int(0.8 * len(X_seq))
X_train, X_test = X_seq[:train_size], X_seq[train_size:]
y_train, y_test = y_seq[:train_size], y_seq[train_size:]

# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, activation='relu', return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(LSTM(units=50, activation='relu'))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

# Make predictions
y_pred_scaled = model.predict(X_test)
y_pred = scaler_y.inverse_transform(y_pred_scaled)
y_test_actual = scaler_y.inverse_transform(y_test)

# Calculate error metrics
mae_lstm = mean_absolute_error(y_test_actual, y_pred)
rmse_lstm = np.sqrt(mean_squared_error(y_test_actual, y_pred))
r2_lstm = r2_score(y_test_actual, y_pred)

print(f'MAE: {mae_lstm}, RMSE: {rmse_lstm}, R-squared: {r2_lstm}')

# Plot loss history
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
```

- XGBoost Model:

```
X = data.drop(columns=['sales'])  # All columns except 'sales' are features
y = data['sales']  # Target variable is 'sales'

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the XGBoost model
xgboost_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=150, random_state=42)

# Train the model
xgboost_model.fit(X_train, y_train)

# Make predictions
y_pred = xgboost_model.predict(X_test)

# Evaluate the model: Calculate the Root Mean Squared Error (RMSE) and R-squared
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
rae = mean_absolute_error(y_test, y_pred)

print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared: {r2}")
print(f"Root Absolute Error: {rae}")

# Optionally, print the first few actual vs predicted values
comparison_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).reset_index(drop=True)
print(comparison_df.head())
```

```
Root Mean Squared Error (RMSE): 1121.4452984914465
R-squared: 0.7238339505779814
Root Absolute Error: 643.6993857684004
```

4. Cleaning:
   - Random Forest Model:

```
X = data.drop(columns=['sales'])
y = data['sales']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)

rf_model = RandomForestRegressor(n_estimators=100,random_state=42)
rf_model.fit(X_train,y_train)

y_pred = rf_model.predict(X_test)

mae = mean_absolute_error(y_test,y_pred)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test,y_pred,squared=False)
print(f"Mean Absolute Error (MAE): {mae}")
print(f"R-squared: {r2}")
print(f'MSE: {mse}')

# Plot actual vs. predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=3, label='Ideal Fit Line')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs Predicted Sales')
plt.legend()
plt.show()
```

```
Mean Absolute Error (MAE): 181.29695899158256
R-squared: 0.8174279414447347
MSE: 313.53122466257435
```

   - Long Short-Term Memory (LSTM) model:

```
X = data.drop('sales',axis=1)
y = data['sales'].values

# Scale the features and target
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.reshape(-1, 1))

# Function to create sequences for LSTM
def create_sequences(X, y, time_steps=7):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:i + time_steps])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

# Create sequences with a time step of 7 days
time_steps = 7
X_seq, y_seq = create_sequences(X_scaled, y_scaled, time_steps)

# Split into training and test sets
train_size = int(0.8 * len(X_seq))
X_train, X_test = X_seq[:train_size], X_seq[train_size:]
y_train, y_test = y_seq[:train_size], y_seq[train_size:]

# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, activation='relu', return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(LSTM(units=50, activation='relu'))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

# Make predictions
y_pred_scaled = model.predict(X_test)
y_pred = scaler_y.inverse_transform(y_pred_scaled)
y_test_actual = scaler_y.inverse_transform(y_test)

# Calculate error metrics
mae_lstm = mean_absolute_error(y_test_actual, y_pred)
rmse_lstm = np.sqrt(mean_squared_error(y_test_actual, y_pred))
r2_lstm = r2_score(y_test_actual, y_pred)

print(f'MAE: {mae_lstm}, RMSE: {rmse_lstm}, R-squared: {r2_lstm}')

# Plot loss history
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
```

- XGBoost Model:

```
X = data.drop(columns=['sales'])  # All columns except 'sales' are features
y = data['sales']  # Target variable is 'sales'

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the XGBoost model
xgboost_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=150, random_state=42)

# Train the model
xgboost_model.fit(X_train, y_train)

# Make predictions
y_pred = xgboost_model.predict(X_test)

# Evaluate the model: Calculate the Root Mean Squared Error (RMSE) and R-squared
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
rae = mean_absolute_error(y_test, y_pred)

print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared: {r2}")
print(f"Root Absolute Error: {rae}")

# Optionally, print the first few actual vs predicted values
comparison_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).reset_index(drop=True)
print(comparison_df.head())
```

```
Root Mean Squared Error (RMSE): 280.40515974017904
R-squared: 0.853969115273415
Root Absolute Error: 163.1966738616287
```

5. Dairy:
   - Random Forest Model:

```
X = data.drop(columns=['sales'])
y = data['sales']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)

rf_model = RandomForestRegressor(n_estimators=100,random_state=42)
rf_model.fit(X_train,y_train)

y_pred = rf_model.predict(X_test)

mae = mean_absolute_error(y_test,y_pred)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test,y_pred,squared=False)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"R-squared: {r2}")
print(f'MSE: {mse}')

# Plot actual vs. predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=3, label='Ideal Fit Line')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs Predicted Sales')
plt.legend()
plt.show()
```

```
Mean Absolute Error (MAE): 153.05953829284795
R-squared: 0.8615269466803362
MSE: 246.50212234898692
```

- Long Short-Term Memory (LSTM) model:

```
X = data.drop('sales',axis=1)
y = data['sales'].values

# Scale the features and target
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.reshape(-1, 1))

# Function to create sequences for LSTM
def create_sequences(X, y, time_steps=7):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:i + time_steps])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

# Create sequences with a time step of 7 days
time_steps = 7
X_seq, y_seq = create_sequences(X_scaled, y_scaled, time_steps)

# Split into training and test sets
train_size = int(0.8 * len(X_seq))
X_train, X_test = X_seq[:train_size], X_seq[train_size:]
y_train, y_test = y_seq[:train_size], y_seq[train_size:]

# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, activation='relu', return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(LSTM(units=50, activation='relu'))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

# Make predictions
y_pred_scaled = model.predict(X_test)
y_pred = scaler_y.inverse_transform(y_pred_scaled)
y_test_actual = scaler_y.inverse_transform(y_test)

# Calculate error metrics
mae_lstm = mean_absolute_error(y_test_actual, y_pred)
rmse_lstm = np.sqrt(mean_squared_error(y_test_actual, y_pred))
r2_lstm = r2_score(y_test_actual, y_pred)

print(f'MAE: {mae_lstm}, RMSE: {rmse_lstm}, R-squared: {r2_lstm}')

# Plot loss history
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
```

s

- XGBoost Model:

```
X = data.drop(columns=['sales'])  # All columns except 'sales' are features
y = data['sales']  # Target variable is 'sales'

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the XGBoost model
xgboost_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=150, random_state=42)

# Train the model
xgboost_model.fit(X_train, y_train)

# Make predictions
y_pred = xgboost_model.predict(X_test)

# Evaluate the model: Calculate the Root Mean Squared Error (RMSE) and R-squared
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
rae = mean_absolute_error(y_test, y_pred)

print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared: {r2}")
print(f"Root Absolute Error: {rae}")

# Optionally, print the first few actual vs predicted values
comparison_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).reset_index(drop=True)
print(comparison_df.head())
```

```
Root Mean Squared Error (RMSE): 190.09615344641358
R-squared: 0.9176486731426451
Root Absolute Error: 127.37799849153859
```