EXPLAINABLE DEMAND FORECASTING FOR RETAILERS USING GRAPH NEURAL NETWORKS AND TEMPORAL FUSION TRANSFORMERS

MAKSATBEK KYZY AIZHAN

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

UNIVERSITI MALAYA
KUALA LUMPUR

2026

EXPLAINABLE DEMAND FORECASTING FOR RETAILERS USING GRAPH NEURAL NETWORKS AND TEMPORAL FUSION TRANSFORMERS

MAKSATBEK KYZY AIZHAN

Research Report submitted in Fulfillment of the requirements for the degree of Master of Artificial Intelligence (Coursework)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE
UNIVERSITI MALAYA
KUALA LUMPUR

2026

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Maksatbek kyzy Aizhan Passport No: PE0583904

Matric No: 23115588

Name of Degree: Master of Artificial Intelligence (Coursework)

Title of Research Report: Explainable Demand Forecasting for Retailers using Graph Neural Network and Temporal Fusion Transformers

Field of Study: Deep Learning

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the Universiti Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date: 10.1.2026

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation

EXPLAINABLE DEMAND FORECASTING FOR RETAILERS USING GRAPH NEURAL NETWORKS AND TEMPORAL FUSION TRANSFORMERS

Abstract

Keywords: Graph Neural Networks, Temporal Fusion Transformers, Demand forecasting, Retail, Explainable AI, Time Series Data

[TAJUK LAPORAN PENYELIDIKAN/DISERTASI/TESIS]

ABSTRAK

(Please delete this part): Abstract should not be more than 500 words in both versions; Bahasa Malaysia and English, typed in single paragraph.

Use *abstract title* for heading and *abstract text* for body of your abstract.

Kata kunci: Maximum of five (5) keywords.

Acknowledgements

I would like to express my sincere gratitude to Associate Professor Ts. Dr. Aznul Qalid Md Sabri, for supervision of this project, which involved extensive work with time-series analysis and prediction.

I am deeply grateful for steadfast support to my husband, Elzar, and to my parents, Marat and Turar, for life courage.

Table of Contents

UNIVERSITI MALAYA 2
ORIGINAL LITERARY WORK DECLARATION 2
EXPLAINABLE DEMAND FORECASTING FOR RETAILERS USING GRAPH NEURAL NETWORKS AND TEMPORAL FUSION TRANSFORMERS
Abstract 4
Acknowledgements 6
Table of Contents 7
List of Figures 8
List of Tables 9
List of Symbols and Abbreviations 10
List of Appendices 11
CHAPTER 1: Introduction 1
CHAPTER 2: Problem Statement 3
CHAPTER 3: Research Questions and Research Objectives 5
CHAPTER 4: Literature Review 6
CHAPTER 5: Methodology 13
Figure 5.1 Methodology Workflow 13
5.1 Statistical Exploratory Data Analysis 14
5.2 Model Development 14
5.2.1 Baseline Model 14
5.2.2 Graph Neural Network 15
5.2.3 Temporal Fusion Transformer 16
5.2.4 GNN-TFT Hybrid 18
5.3 Explainability and Feature Importance 20
5.4 Optimizations 20
5.4.1 Optimizations for GNN 21
5.4.2 Optimizations for TFT 21
5.5 Dataset to be used 22
Feature 22
Description 22
5.6 Evaluation Metrics 25
CHAPTER 6: Results 27
6.1 Temporal Fusion Transformer 27
6.2 Spatio-Temporal Graph Neural Network (STGNN) 34
6.4 Hybrid Model 44
Parameters 48
Performance in Metrics 48
References 50
Appendix 51

List of Figures

Figure 5.1 Project Methodology Workflow

Figure 5.2 Graph NN

Figure 5.3. Temporal Fusion Transformer Architecture (Sukel, Maarten (2023))

Figure 5.4: Hybrid architecture GNN and TFT (Sukel, Maarten (2023))

Figure 5.5.1 Promotion, sales effect on daily, monthly and yearly sales

Figure 5.5: Stores per city

Figure 5.6 Heatmap of yearly sales per product

Figure 6.1: TFT Model forecast for Automotive in Store 1

Figure 6.2: TFT Model forecast for Automotive in all stores

Figure 6.3: TFT-Instrinsic XAI Variable Importances for Automotive in Store 1

Figure 6.4: TFT Model-Instrinsic XAI Input for Automotive in Store 1

Figure 6.5: TFT Model MAE comparison

Figure 6.6: STGNN Model forecast for Cleaning in store 1

Figure 6.7: STGNN Model forecast for Cleaning in all stores

Figure 6.8: Store Graph close look

Figure 6.9: Product-Family Graph closer look

Figure 6.10: GNN explanation input - Node 2 (store 1, Beauty product)

Figure 6.11: GNN explanation with SHAP

Figure 6.12: GNN explanation with Custom Edge Mask: Top neighbors

Figure 6.13: GNN Multiple Architecture MAE Comparison

List of Tables

Table 3.1: Research Questions and Objectives

Table 4.1: Graph Results by Aktas, M. Y., and Ji, T. (2024)

Table 5.1. Dataset features

Table 6.1: TFT Input Data Category Details

Table 6.2: Model-Agnostic Method: Permutation Importance Results

Table 6.3: Multiple TFT Architecture Performance Results

Table 6.4: Shapley Feature Mean Absolute Values

Table 6.5: GNN Comparative Performance Results

List of Symbols and Abbreviations

TFT	:	Temporal Fusion Transformers
GNN	:	Graph Neural Networks
DNN	:	Deep Neural Networks
RNN	:	Recurrent Neural Networks
DC	:	Distribution Center
ANN	:	Artificial Neural Networks
ARIMA	:	Autoregressive Integrated Moving Average
SKU	:	Stock Keeping Unit
XAI	:	Explainable AI
MAE	:	Mean Absolute Error
LSTM	:	Long Short Term Memory
CNN	:	Convolutional Neural Network
TiDE	:	Time-Series Dense Encoder
MQRNN	:	Multi-Horizon Quantile Recurrent Forecaster
VAR	:	Vector Autoregressive
GCN	:	Graph Convolution Network
SHAP	:	SHAP (SHapley Additive exPlanations)
LIME	:	Local Interpretable Model-agnostic Explanations
RE	:	Relational Encoder

List of Appendices

CHAPTER 1: INTRODUCTION

In today's fast-paced retail sector, accurately forecasting product demand is essential for ensuring operational efficiency. Precise demand predictions enable retailers to optimize inventory management, stock replenishment, and human resources. Supply chains are becoming more complex, particularly with the involvement of third-party logistics providers, making advanced forecasting tools increasingly critical.

The combination of artificial intelligence and data analytics has opened up new ways to tackle these challenges. Models such as Graph Neural Networks, Temporal Fusion Transformers, and hybrid GNN-TFT approaches can uncover complex relationships and patterns that traditional methods may miss. This research aims to develop and compare the effectiveness of GNN, TFT, and hybrid GNN-TFT models. Alongside a focus on predictive accuracy, this project highlights the importance of making models interpretable, so that AI-generated insights are transparent and easily understood for real-world decision-making.

As ML models become increasingly sophisticated, interpreting how these algorithms arrive at their predictions presents new challenges. This has led to continued research on novel methods for improving explainability, which are typically classified as either model-specific or model-agnostic approaches. Explainable AI encompasses processes and tools designed to make the reasoning behind AI and machine learning decisions clear and accessible to humans.

In retail, explainable AI is essential for earning the trust of users and stakeholders. Transparency ensures that all parties involved in decision-making can understand how conclusions are reached, while fairness guarantees that algorithmic predictions remain unbiased across various product categories. Building trust also requires assessing the level of confidence retail managers can place in the AI system's recommendations for real-world scenarios. Ultimately, interpretability allows stakeholders to comprehend the rationale behind model outputs, supporting actionable insights and informed choices.

CHAPTER 2: Problem Statement

Precise demand forecasting plays a vital role in retail by impacting inventory management, store restocking, supply chain performance, customer satisfaction, and overall sales. However, this task is complicated by several challenges, particularly as retailers expand, operate through multiple channels (both online and physical stores), and depend on complex logistics networks. Key challenges in retail demand forecasting include:

- Overestimating demand, which can result in overproduction, high holding costs, and waste if products expire or go out of style.
- Difficulty in predicting budgets for investments, estimating returns, and determining the profitability of operations.
- Uncertainty in forecasting operating costs, stock requirements, and optimal order quantities for each season.
- The need to align orders with seasonal trends and consumer preferences to ensure in-demand products are stocked while minimizing excess inventory.
- Challenges in introducing new products without overshadowing existing ones, while still ensuring new stock will be sold.
- Greater forecasting difficulty due to expansion across regions and sales channels, given the diversity of consumer behaviors and logistics complexities.
- Delays in sales flow and the challenge of modeling complex relationships between spatial (location-based), item-similarity, and temporal (time-based) factors.

- The complexity of advanced forecasting models, which can make them difficult for decision-makers to interpret and trust.

These issues can be efficiently managed by treating them as a time-series forecasting task, aiming to predict future product demand based on past sales data, seasonal trends, and additional contextual variables. By approaching retail demand forecasting in this way and applying advanced models such as GNN, TFT, and their hybrid GNN-TFT variant, it becomes possible to tackle many of the identified challenges and substantially enhance forecasting accuracy, ultimately supporting better business decisions and outcomes.

These models are particularly well-suited for managing volatile and seasonal sales patterns, which are common in industries that experience various product life cycles and distinct seasonal trends (Spring-Summer, Fall-Winter). They are capable of capturing autocorrelations as well as linear and non-linear dynamics within intricate sales data. Additionally, the models excel at representing both spatial (across different stores or regions), item similarity and temporal (across time) dependencies, effectively addressing challenges such as delayed sales flow and the integration of spatial-temporal features and item similarity. Additionally, they are able to capture both long-term patterns and short-term changes in demand at the same time, offering thorough insights for retail forecasting.

CHAPTER 3: Research Questions and Research Objectives

This study aims to explore and evaluate the performances of modern machine learning models specifically, GNN, TFT and hybrid for predicting retail demand, while also examining ways to make these models more interpretable and accurate.

Table 3.1: Research Questions and Objectives

Research Questions	Research Objectives
Which approach yields higher accuracy in retail demand forecasting: GNN, TFT, or a hybrid GNN-TFT model?	To evaluate and compare the forecasting accuracy of GNN, TFT, and hybrid GNN-TFT models.
Which features within the retail dataset have the greatest and least impact on the predictive accuracy of these models?	To interpret model predictions by applying both model-specific and model-agnostic explainable AI (XAI) methods.
What architectural improvements are most effective for optimizing the performances of models?	To find and implement architectural improvements, aiming to maximize predictive performance and operational value of models.

CHAPTER 4: LITERATURE REVIEW

Demand forecasting typically involves several key processes such as

1. Data collection and integration of historical sales, promotional campaigns, pricing, inventory records, product metadata, customer demographics, and external variables (such holidays, weather).
2. Cleaning the data, handling missing values, extracting seasonality and trend features, and constructing additional features.
3. Analyse demand patterns, trends, and anomalies across stores, products, and regions.
4. Selecting and training forecasting models using historical data and engineered features.
5. Evaluation of model performance using evaluation metrics.
6. Translating forecasts into actionable insights for inventory replenishment, logistics planning, and promotional strategies.

While every process in demand forecasting is important, this study focuses on the modeling stage, specifically on GNN, TFT and hybrid due to their unique abilities to handle the complex, high-dimensional, and relational characteristics of contemporary retail data.

GNNs are powerful for modeling relationships and dependencies between products, stores, or customer segments, which are often represented as graphs. This is essential for environments where products influence each other.

TFTs excel at capturing temporal patterns, handling multiple exogenous variables, and providing multi-horizon forecasts, which are critical for seasonality and promotion-heavy retail environments.

Retail demand is influenced not only by a product's own history but also by the relationships between items, stores, and customers. GNNs are uniquely suited to capture these dependencies, leveraging graph structures (e.g., co-purchase, sales similarity, or geographical proximity) (Kozodoi, N., Zinovyeva, L., and Valentin (2023)).

TFTs handle long-term dependencies, seasonality, and exogenous variables, enabling accurate forecasting far into the future (e.g., for entire seasons) (Bryan Lim and Sercan Ö. Arık (2021)).

GNNs and TFTs scale to large, high-dimensional datasets common in retail, outperforming traditional models that cannot efficiently handle such complexity (Kozodoi, N., Zinovyeva, L., and Valentin (2023)).

Although both GNNs and TFTs are powerful, their "black-box" nature makes explainability a challenge. Literature notes that standard explainability methods (e.g., LIME, SHAP) struggle with time dependencies and feature interactions in these architectures, motivating research into improved interpretability (Bryan Lim, Sercan (2021)).

An important advancement in this area comes from Lalou, P., Ponis, S.T., & Efthymiou, O.K. (2020), who examined demand forecasting across a network of 129 stores located in various countries. Their findings highlight the potential of data analytics and statistical programming to enhance decision-making processes for store restocking, inventory optimization at distribution centers and retail sites, and the organization of human and automated workforce schedules within logistics centers.

Sukel, Maarten, and Rudinac (2023) emphasize that demand prediction is essential for accurate stock replenishment and inventory management, enabling retailers to minimize waste and maintain competitiveness. They note that forecasting demand is complex due to the numerous variables that affect customer behavior over time.While conventional approaches typically use tabular data from past sales and seasonal patterns, the authors advocate for incorporating visual and textual product features, which strongly influence purchasing decisions. Additionally, they point out that geographic factors also play a significant role in shaping demand.

Sukel, Maarten, and Rudinac (2023) study combines tabular sales data with multimodal product information and the geographical context of each product. Forecasting becomes even more complex when different data sources are integrated, such as variations in delivery times, warehouses, and product types. The authors implement a Temporal Fusion Transformer (TFT) alongside Convolutional Neural Networks (CNNs) to extract image features and transformers to generate text embeddings. These multimodal features are then input into an LSTM model. Their study utilizes data from Picnic, an online grocery delivery service operating in Germany, France, and the Netherlands.

For extended forecasting horizons, Sukel, Maarten, and Rudinac (2023) observe that models such as TiDe, which utilize multilayer perceptrons, may surpass TFTs in performance while also offering faster training and inference times. Geographic information is integrated through a graph-based method. Overall, their proposed model consistently outperforms the baseline TFT across all test set delivery dates and product categories including fruits, beverages, and meat spreads.

Bryan Lim and Sercan Ö. Arık (2021) developed the Temporal Fusion Transformer (TFT), an advanced architecture utilizing attention mechanisms to address forecasting across multiple time periods. This technique enables the model to generate forecasts for several future intervals at once, which is particularly beneficial for retailers looking to manage inventory for entire seasons in advance. In practice, multi-horizon forecasting often draws on a wide range of data, such as upcoming events like holidays, records of customer visits, and fixed attributes like store locations, even if the relationships among these variables are not clearly defined. The wide variety and complexity of this information make forecasting for different time horizons especially demanding. Deep learning models like TFT have outperformed traditional time series methods. Although recurrent neural networks have long been common, attention-based models such as TFT are gaining popularity due to their effectiveness in pinpointing the most important parts of historical data for making predictions.

The findings indicate that TFT significantly outperforms all benchmark models across multiple datasets from various domains such as Electricity, Traffic, Volatility, and Retail. Of particular relevance to this research, the authors utilize the Favorita Grocery Sales Dataset (Favorita, 2018), which integrates data from various products and store locations as well as additional daily-changing external factors. The task involves predicting product sales for future 30 days using 90 days of registered history of sales. For the retail dataset, TFT achieves average improvements of 7.2% in P50 and 3.1% in P90 quantile losses compared to the next best model MQRNN.

Kozodoi, N., Zinovyeva, L., and Valentin (2023) emphasize that demand forecasting should account for relationships between individual items, which can be effectively modeled using GNNs. Their paper introduces GraphDeepAR, a model that combines a GNN encoder with the DeepAR forecasting framework. Specifically, the GNN encoder captures relationships between items, while the DeepAR decoder is responsible for demand prediction. The paper also proposes an item attribute similarity graph construction method without predefined graph structure.

While DeepAR and TFT models are trained on time series data and can learn patterns across similar series, they cannot incorporate inter-series relationships during inference. Although VAR models can capture such relationships during both training and inference, they are not scalable for high-dimensional retail environments. In contrast, GNNs offer a scalable solution for modeling complex relationships in large datasets.

The researchers construct a graph $G_t = (N_t, E)$, with each node corresponding to a product sold at time t, and edges E capturing the pairwise associations between items based on the cosine similarity of their attributes. All edges are undirected and have static features.

The researchers utilize retail sales datasets sourced from Kaggle along with Adidas's historical sales records. The information is structured as a time series, capturing weekly demand, product descriptions, fixed attributes, and dynamic variables that reflect seasonality, such as the week of the year, week number, and day of the month. To maintain the temporal sequence, the datasets are divided in order into training, validation (the 13 weeks following the training period), and test sets (the 26 weeks after the validation period).

The findings reveal that GraphDeepAR surpasses DeepAR by 4.36% on retail datasets and by 31.98% on e-commerce datasets, with the more substantial gains in e-commerce linked to the larger volume of data. When evaluated on the Adidas dataset, GraphDeepAR delivers a 2.05% increase in financial performance over DeepAR. However, this enhanced accuracy comes at the cost of efficiency, as GraphDeepAR takes 159% more time to train and 34% more time for inference compared to DeepAR.

Aktas, M. Y., and Ji, T. (2024) introduced the LSTMGraph model for forecasting demand across multiple products. This method combines LSTM networks, which are adept at capturing changes over time such as price fluctuations, with GCN that model overall relationships between products. Within the LSTMGraph framework, products serve as nodes, and three distinct graphs are built to capture different types of relationships: (1) Weekly Sales, (2) Customer links, and (3) Invoice connections. By merging these graphs, the model leverages diverse temporal and relational information, ultimately improving its forecasting accuracy.

To gain deeper insights into how various graph structures affect model performance, the authors performed an ablation study aimed at identifying which graph type most effectively reduces prediction error. The research assessed the model's accuracy when each graph type was used alone, as well as in different combinations, as summarized in Table 4.1.

Table 4.1: Graph Results by Aktas, M. Y., and Ji, T. (2024)

Graph 1	Graph 2	MAE
Weekly Sales Similarity	-	11.6834
Invoice Similarity	-	11.8377

Graph 1	Graph 2	MAE
Customer Relationship	-	12.3293
Weekly Sales Similarity	Customer Relationship	11.9633
Customer Relationship	Invoice Similarity	11.9417
Weekly Sales Similarity	Invoice Similarity	11.9802

The LSTMGraph model achieved a low MAE of 11.4776, highlighting its effectiveness in learning both time-based trends and relationships captured by the graph. Notably, the standalone Weekly Sales Similarity Graph outperformed all combined graph configurations, suggesting that adding information from the other graphs might have resulted in redundant data or added unneeded complexity.

Bryan Lim and Sercan (2021), the creators of TFT, highlight that one of the main difficulties is making the model interpretable when dealing with varied types of data. The TFT functions much like a ‘black box,’ and conventional methods for interpreting deep neural networks are often ineffective for time-series problems due to the model’s intricate and non-linear internal structure. Techniques like LIME and SHAP do not adequately capture the sequential characteristics of time-series inputs, leading to insufficient interpretability. Although transformer models can provide some level of explanation for applications in text and speech, they face challenges in pinpointing the significance of features at specific time points in complex, horizon-based datasets.

CHAPTER 5: METHODOLOGY

The approach outlined in Figure 5.1 involves evaluating and comparing the effectiveness of three models - GNN, TFT, and a hybrid GNN-TFT - for forecasting time series data across multiple products.

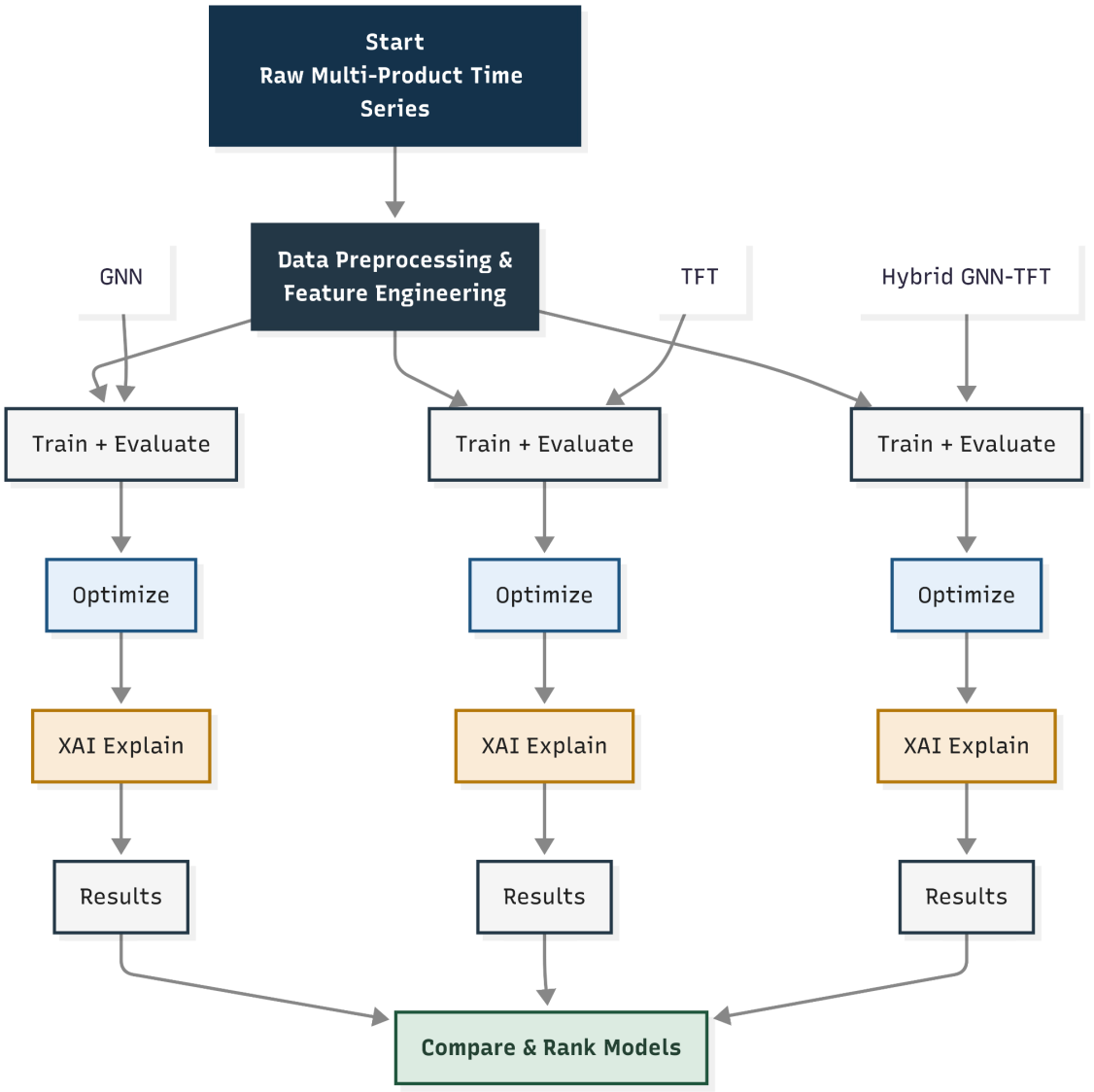


Figure 5.1: Project Methodology Workflow

5.1 Statistical Exploratory Data Analysis

Statistical exploratory data analysis is conducted to reveal patterns within the sales data and support modeling strategies. Time series visualizations are used to study sales trends and pinpoint seasonal effects and unusual fluctuations. Analytical methods are applied to investigate relationships between various features, such as comparing the influence of retailer locations on sales figures, and observing how demand shifts across product categories.

Analysis extends to tracking operating profit for each product throughout the observed period, monitoring how unit sales vary across regions, and evaluating seasonal behaviors to identify both peak and low-profit times for each item.

City performance is measured through unit sales over time, highlighting which cities are most profitable. Regional sales distributions are also studied to uncover patterns in retailer performance. The analysis involves detecting missing values, identifying outliers, and comparing outcomes across various stores and product categories. Findings from this process help improve attribute engineering and guide the selection of key features for the model building, including a closer look at how unit sizes for each product change over time.

5.2 Model Development

This section describes the architecture and key components of models ARIMA, GNN, TFT and hybrid GNN-TFT.

5.2.1 Baseline Model

The ARIMA model serves as a standard reference for unit sales forecasting over time series. The model integrates three core mechanisms. The

autoregressive part captures the influence of prior observations on current values. The integrated part transforms the data by calculating differences between consecutive points to remove underlying trends and stabilize the chains. And the moving average part defines the connection among current values and past forecast errors. Collectively, these elements allow ARIMA to reveal and forecast patterns in time series by leveraging both historical values and residuals from previous predictions.

5.2.2 Graph Neural Network

GNN captures the relationships among products using a graph structure G = (N, E), where every node in N represents a single product, and the edges in E represent pairwise connections between products. These edges are undirected, and the weights are inferred from data, by calculating cosine similarity score between product features and sales histories.

Each product node is characterized by a time series of sales at each time step t. To enrich the node representation, the feature set for each product at time t is described as a tuple A = (X, Z, Y). X is a vector containing M static features, such as retailer, location, which remain constant over time. Z is a matrix of L time-varying features, including known-in-advance factors like month, year, seasonal events, and promotions. Y denotes the demand or sales count for the product at time t.

The main structure of the architecture (see Figure 5.2) features two graph convolutional layers. The initial layer collects information from adjacent nodes and maps the input features into a hidden space, using a ReLU activation to introduce non-linearity. The subsequent layer continues to aggregate and refine

these hidden representations, generating forecast outputs for each product. For each node, the resulting output is a vector of length K, which corresponds to predicting daily demand over the next K days.

By leveraging information from both the sales history of each product and the relationships encoded in the graph, the GNN is able to forecast demand over the horizon. These multi-step predictions are especially valuable for inventory replenishment, enabling retailers to anticipate future demand and optimize stock levels for each product category.

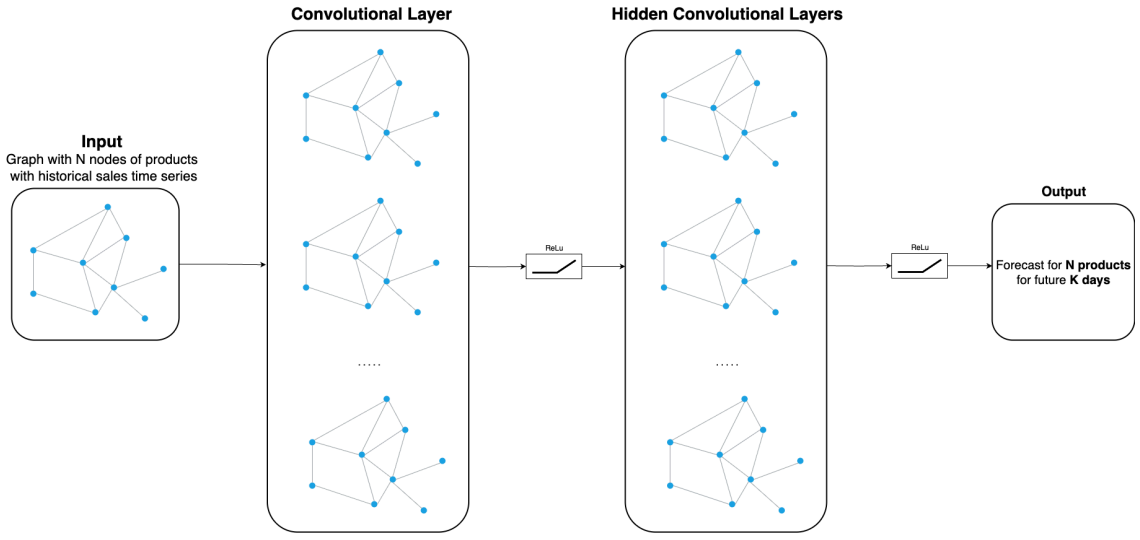


Figure 5.2: Graph NN

5.2.3 Temporal Fusion Transformer

The TFT (Figure 5.3) is designed to handle a variety of input features for time series forecasting. It takes in static features, which remain unchanged throughout the observation period, as well as dynamic features that can vary over time, such as calendar-related factors (month, year), seasonal effects, or special promotions. Additionally, the model utilizes the actual demand data for each product at specific time points.

The primary output generated by the TFT is a forecast for each product, predicting sales for the next K days using historical data from the previous N days.

What distinguishes TFT is its implementation of attention mechanisms that enable the model to learn complex, evolving relationships and patterns over time. Its architecture centers on Gated Residual Networks, which facilitate the handling of intricate, non-linear data interactions. The designers of TFT incorporated four essential components into the model.

Static covariate encoders transform constant features into context vectors that enrich the entire model.

Gating and variable selection ensures that only the most relevant inputs contribute to each prediction, effectively reducing noise from less important features.

The sequence-to-sequence processing layer manages both observed inputs and those known ahead of time, enabling the model to seamlessly incorporate various forms of time-dependent information.

The temporal self-attention decoder is tailored to capture and represent long-range dependencies, allowing the model to identify patterns and effects that extend throughout the entire dataset.

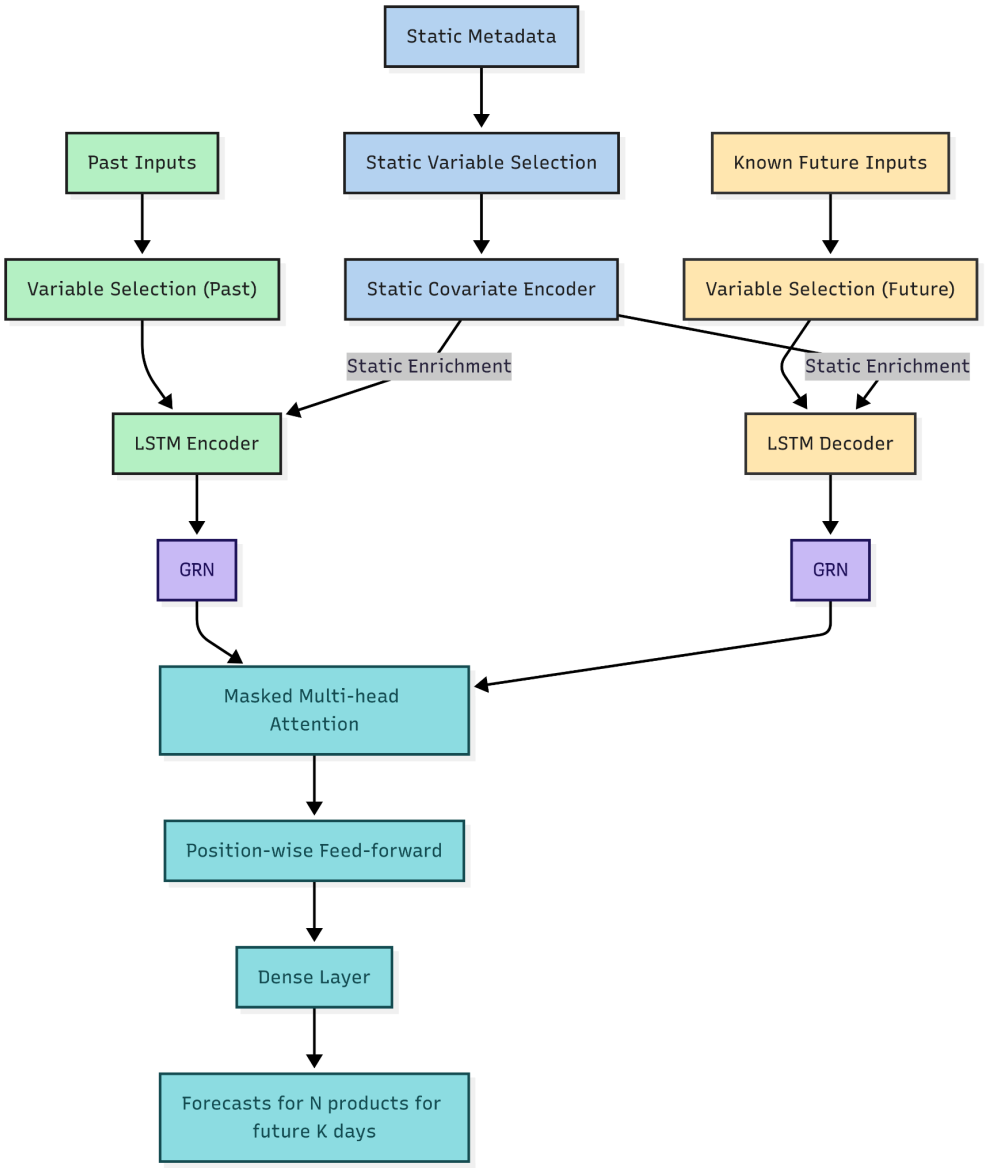


Figure 5.3: Temporal Fusion Transformer Architecture (Sukel, Maarten (2023))

5.2.4 GNN-TFT Hybrid

The main concept behind this approach is to leverage the GNN to derive relational features for each product, and then integrate these with static, historical, and known future features before inputting them into the TFT. This combination produces a comprehensive, context-sensitive representation for each product at every time point.

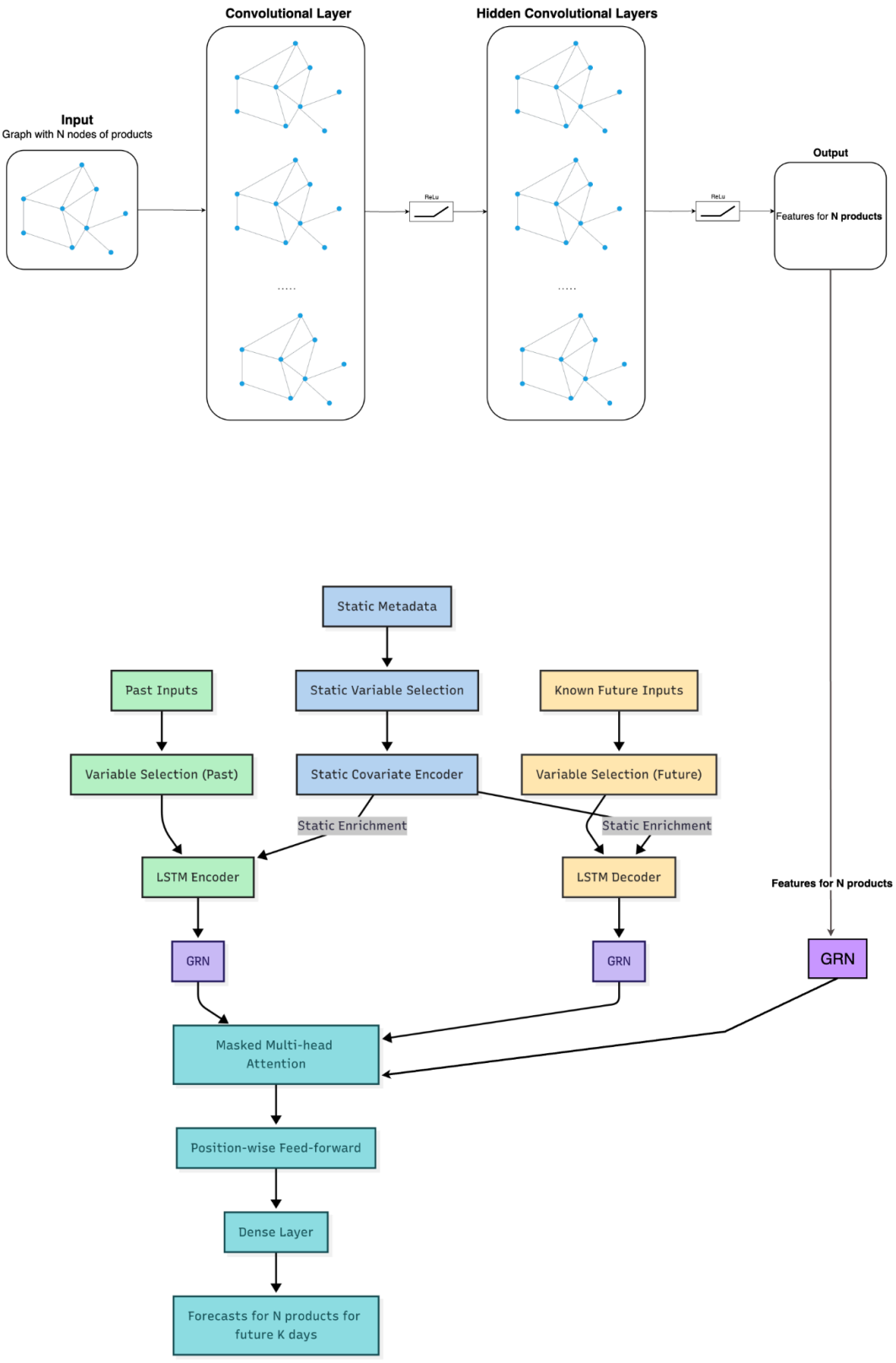


Figure 5.4: Hybrid architecture GNN and TFT (Sukel, Maarten (2023))

The integrated pipeline (Figure 5.4) works as follows:

1. The product graph is processed by the GNN to generate node embeddings that encode both individual and relational information.
2. These embeddings are concatenated with static and temporal covariates.
3. The fused features are input into the TFT, which performs feature selection, sequential modeling, and attention-based forecasting.

5.3 Explainability and Feature Importance

The models are interpreted using both model-specific and model-agnostic techniques. The approaches help pinpoint the features that have the greatest impact on the model's predictions.

The TFT architecture is inherently interpretable, allowing it to reveal (1) the most influential features, (2) consistent temporal patterns, and (3) key events. Additionally, the TFT model results are interpreted using model-agnostic XAI techniques such as LIME and SHAP.

For the GNN, prediction transparency is achieved with the model-specific GNNExplainer, along with the model-agnostic LIME and SHAP methods.

5.4 Optimizations

Enhancements to the model involve updating layers and refining the network architecture. Hyperparameters are carefully tuned to maximize performance, and a systematic assessment is conducted to balance model complexity, predictive accuracy, and practical utility.

5.4.1 Optimizations for GNN

The main GNN utilized in this study is the Convolutional Graph Network. To enhance its capacity to differentiate among neighboring nodes, an attention mechanism will be integrated. This addition allows the network to allocate varying importance to each neighbor, enabling it to focus on the most pertinent nodes when aggregating information.

To thoroughly evaluate the effectiveness of attention and model architecture choices, experiments will be conducted using several GNN variants, including Convolutional GNN, GraphSAGE, and Attention GNN. Each architecture will be tested to analyze how well it models the correlations within the data and to identify the optimal approach for the given task.

5.4.2 Optimizations for TFT

Following the recommendations of Bryan Lim and Sercan Ö. Arık (2021), evaluating feature importance is essential for optimizing the TFT. The model focuses on a group of input features that logically contribute to its predictive performance. Understanding persistent temporal patterns helps to reveal time-dependent relationships within the data.

Building on these insights, this research refines the forecasting model by expanding the receptive field to consider a longer historical context when attention is highest at the start of the lookback window, or by engineering features that more effectively capture seasonal trends. Leveraging attention weights from the temporal fusion decoder's self-attention layer enables the detection of recurring patterns, particularly by analyzing how features at specific time lags impact forecasts across various horizons. This strategy involves

examining variable importance, visualizing enduring temporal trends, and pinpointing distinct regimes and important events.

5.5 Dataset to be used

Daily sales data for multiple product families across Ecuadorian retail stores, designed for multi-series forecasting. Suitable for advanced temporal and relational models (TFT, GNN, hybrids) due to rich exogenous signals, hierarchical (store–family) structure, and event-driven dynamics.

The dataset contains 3,000,888 records. Each record includes the following features in the Table 5.1

Table 5.1. Dataset features

Feature	Description
store	The sold product store location
family	Type of product
sales	The total purchases of a product family at a specific shop and date
on-promotion	The number of promoted product family items for a given store and date

Key supplemental data:

- Store metadata: Includes details such as city, state, store type, and cluster (which groups together similar stores)
- Daily oil price: serves as a macroeconomic indicator relevant to Ecuador's economy
- Holiday calendar with special semantics:

    - Transferred holidays: the original date is treated as a normal day, and the celebration occurs on the “Transfer” date
    - Bridge days: extended holidays with additional days
    - Work days: compensate for extended breaks
    - Additional days: days added a regular calendar holiday
- Daily transaction counts

Domain factors influencing demand include:

- Promotions (on-promotion)
- Holiday and event effects (with careful handling of transferred holidays)
- Bi-monthly public sector paydays (15th and end of the month) can boost sales
- The April 16, 2016 earthquake caused a temporary spike in demand
- Oil price fluctuations provide macroeconomic context

The dataset includes:

- 33 product families
- 54 unique store numbers
- 22 different states
- 5 store types
- 17 store clusters
- No missing values

A high number of stores are located in Quito among 22 cities with significant differences showing the imbalance of sales in the dataset across cities.

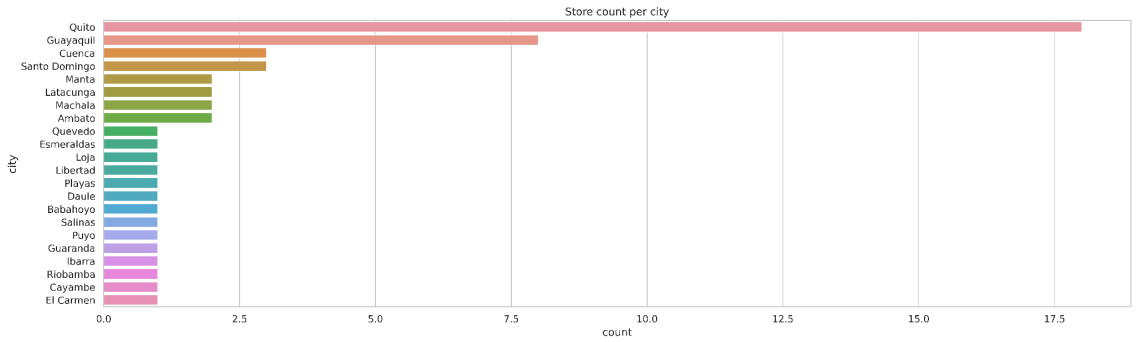


Figure 5.5: Stores per city

The heatmap in Figure 5.6 shows that core grocery and household product families have consistently high daily non-zero sales from 2013 to 2016, with a peak in 2016 and a drop in 2017. Other categories, such as HOME CARE, BABY CARE, and BOOKS, either started later or have much sparser sales.

5.6 Evaluation Metrics

Forecast performance is assessed using Mean Absolute Error (MAE), a widely used and interpretable accuracy metric. The GNN, TFT, and hybrid GNN–TFT models are benchmarked against a baseline model and then comparatively evaluated to determine relative strengths.

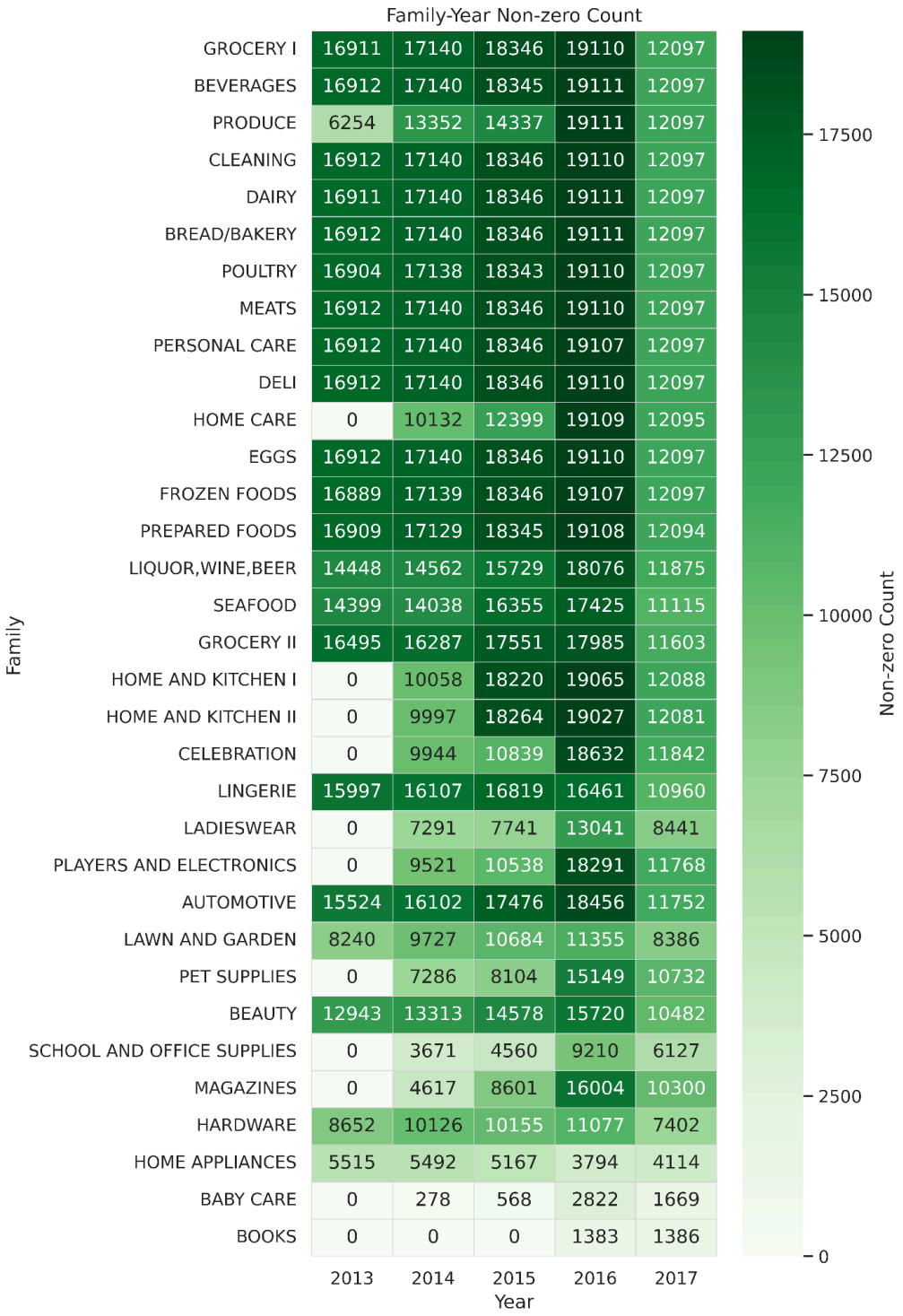


Figure 5.6 Heatmap of yearly sales per product

CHAPTER 6: RESULTS

The project provides stakeholders with sales forecasts for the subsequent 28-day period, corresponding to dates from 2017-07-15 to 2017-08-15, accompanied by detailed explanations of the results.

The time-series characteristics of the Favorita dataset facilitate the modeling and analysis of daily sales activities across multiple product families and diverse retail locations. Effective temporal and spatial analyses require the availability of date-indexed chronological records of purchases, supplemented with metadata such as store location and product identifiers.

Model inputs are constructed by merging multiple raw files, as detailed in Appendix C. A comprehensive sales chronology is created for every calendar day, store number, and product family for the spanning period from 2013-01-01 to 2017-08-15. The data preprocessing and feature engineering pipeline is applied to the sales chronology.

To ensure robust evaluation and maintain temporal-chronology, the dataset is partitioned sequentially into training, validation, and test sets. Specifically, the final 28 days are reserved for the test set, the preceding 56 days constitute the validation set, and all prior data to the last 84 days comprise the training set.

Standardization ensures that variables contribute proportionally to model training and reduces potential biases associated with differing value ranges. The transaction and daily oil price numerical features are normalized using the z-score normalization so that it has a mean of zero and a standard deviation of one.

The transformation of categorical values into binary numerical representations ensures compatibility with the model's requirements. Static categorical features - specifically store number, product family, store state and cluster - are converted into binary representation using a one-hot encoding method.

The holidays feature is further refined into "transferred" and "non-transferred" types, represented as binary labels (0 and 1), while the 'workday' indicator was explicitly defined according to the official calendar as a boolean value.

6.1 Temporal Fusion Transformer

Figure 6.1 presents the TFT model forecasts for the "Automotive" product type in store 1, while Figure 6.2 illustrates the forecasts for this product across all stores. Further detailed forecasts for each individual store are provided in Appendix D.

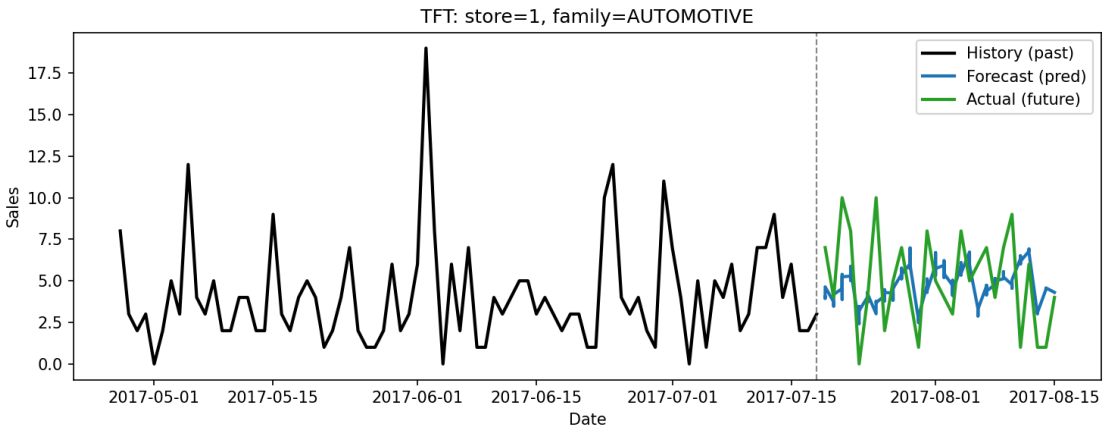


Figure 6.1: TFT Model forecast for Automotive in Store 1

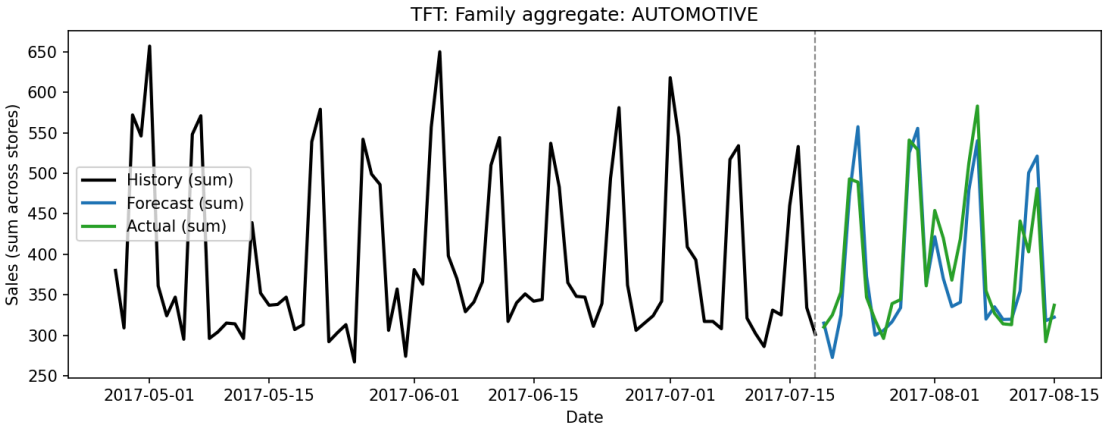


Figure 6.2: TFT Model forecast for Automotive in all stores

6.1.1 TFT Dataset

The TFT architecture requires three categories of input features: observed features from the past, known future inputs, and static features, as summarized in Table 6.1.

A sliding window mechanism provides the model with input-target pairs for each anchor time point t during training, validation and testing.

Table 6.1: TFT Input Data Category Details

Feature Category	Feature Name
Past observed (9 total)	sales transactions daily oil price promotion day of the week month week of the year holiday workday
Known future (6 total)	promotion day of the week month week of the year holiday workday

Static (4 total)	store number product family store state store cluster

6.1.2 TFT Explanation

To interpret the TFT model and evaluate the impact of individual features on forecasting decisions, both model-agnostic and model-intrinsic explainable AI (XAI) approaches are employed.

6.1.2.1 Model-Agnostic Method: Permutation Importance

The relevance of features is evaluated using the permutation importance method on the test set, using the Weighted Absolute Percentage Error (WAPE) metric. In this approach, features from each group (encoder, decoder, static) are randomly shuffled one at a time. The increase in WAPE relative to the baseline quantifies the importance of the permuted feature - the greater the increase in WAPE, the more influential the feature for accurate prediction.

As shown in Table 6.2, among decoder (future) variables, “onpromotion” causes the greatest increase in WAPE, indicating that it is the most critical feature for forecast accuracy. For encoder (historical) features, “sales” is the most important, while “transactions” and “onpromotion” have smaller effects. Static features, such as store information and product metadata, have negligible impact on forecast performance.

Table 6.2: Model-Agnostic Method: Permutation Importance Results

Space	**Variable**	**ΔWAPE**
Decoder (future)	**onpromotion**	**0.1053**
	dow	0.0216

	month	0.0130
	weekofyear	0.0107
	is_holiday	0.0024
	is_workday	0
Encoder (historical)	**sales**	**0.0280**
	transactions	0.0020
	onpromotion	0.0018
	is_holiday	0.0002
	dow	0.0001
	is_workday	0
	weekofyear	-0.0001
	month	-0.0004
	dcoilwtico	-0.0016
Static	state	~0
	cluster	~0
	family	~0
	store_nbr	~0

6.1.2.2 Model-Instrinsic Method: Attention and Variable Selection Weights

The internal mechanisms of the TFT architecture offer interpretability through its attention and variable selection networks. After performing inference on the test set, attention maps can be extracted to reveal which temporal segments the model prioritizes during prediction.

The variable selection network (VSN), implemented with gated residual networks, outputs context-dependent weights for encoder (past), decoder (future), and static features. Each feature group is encoded, followed by

application of softmax function, resulting weighted sum produces a fused representation with interpretable, per-feature importances.

For example, as shown in Figure 6.3, for the "Automotive" product family in store 1, the highest model-derived variable importances show the holiday, oil price, and family type. Factors such as store state, cluster, sales, transactions, workday, and week of the year had minimal impact, while features such as onpromotion, day of the week, month, week of the year, month, and store number have moderate influence. The Figure 6.4 shows the input data and prediction vs actual results.

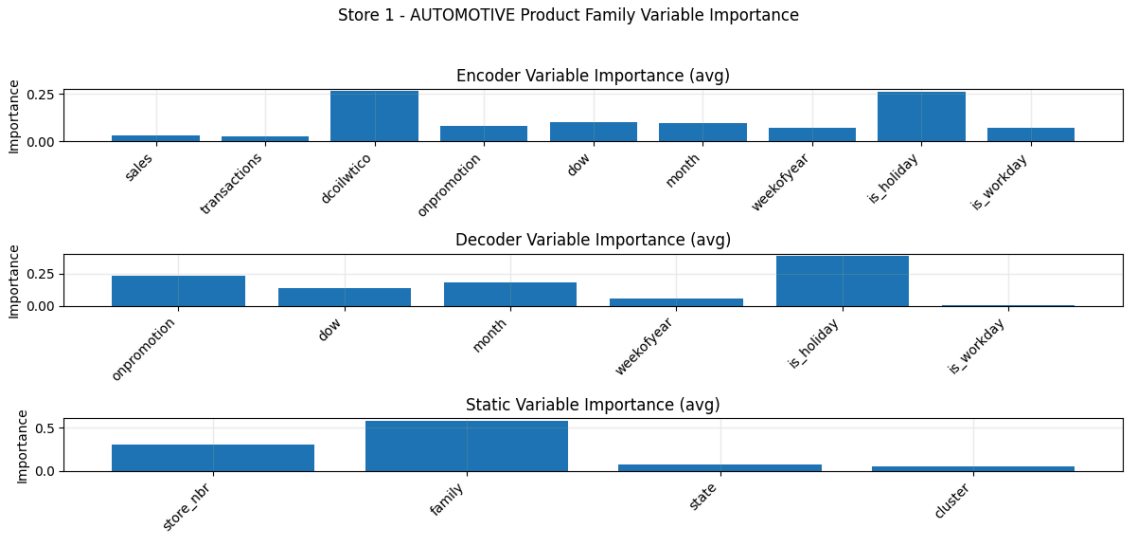


Figure 6.3: TFT-Instrinsic XAI Variable Importances for Automotive in Store 1

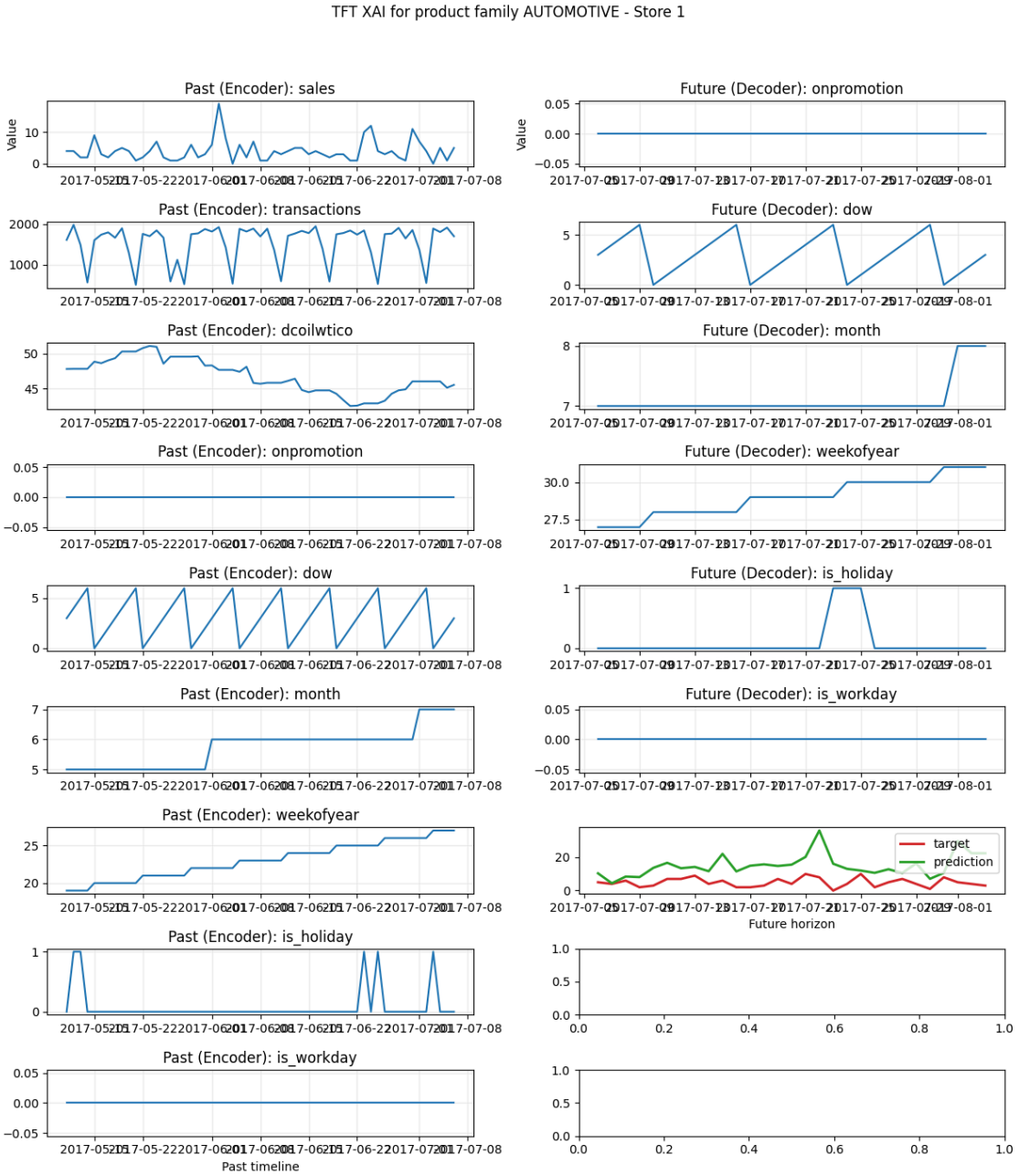


Figure 6.4: TFT Model-Instrinsic XAI Input for Automotive in Store 1

6.1.3 TFT Model Performances

To compare various combinations of TFT model architectures, an experiment was conducted evaluating four distinct sets of key architectural parameters: hidden layer dimension, embedding dimension, number of attention heads, LSTM hidden state size, and the number of stacked LSTM layers, as detailed in

Table 6.3. The findings indicate that the configuration consisting of a hidden layer dimension of 128, an embedding dimension of 64, four attention heads, an LSTM hidden state size of 64, and a single stacked LSTM layer outperformed the other configurations, as evidenced by a MAE of 21.6821 (see Figure 6.5).

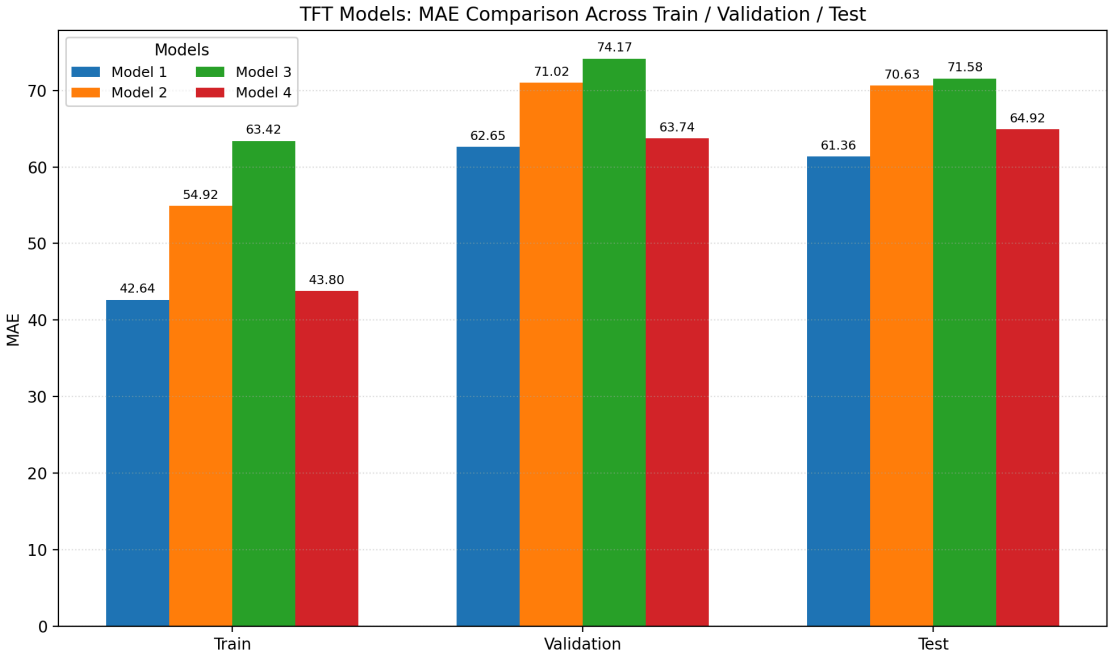


Figure 6.5: TFT Model MAE comparison

Table 6.3: Multiple TFT Architecture Performance Results

Model	**Parameters**	**Results in Metrics**
1	--hidden-dim: 128 --d-model: 64 --heads: 4 --lstm-hidden: 64 --lstm-layers: 1 --dropout: 0.1	Early stopping at Epoch 81/200 Training loss: 15.4443 - MAE: 42.6364 - WAPE: 0.1188 - SMAPE: 0.8810 Validation - MAE 62.6490 - WAPE 0.1298 - SMAPE 0.5188 Test loss: 21.6821

		- MAE 61.3641 - WAPE 0.1287 - SMAPE 0.5249
2	--hidden-dim: 64 --d-model: 32 --heads: 2 --lstm-hidden: 32 --lstm-layers: 1 --dropout: 0.1	Early stopping at Epoch 14/100 Train Loss: 18.7323 - MAE 54.9210 - WAPE: 0.1530 - SMAPE: 0.8948 Validation Loss: 24.6209 - MAE: 71.0156 - WAPE: 0.1472 - SMAPE: 0.6222 Test Loss: 24.6628 - MAE: 70.6296 - WAPE: 0.1481 - SMAPE 0.6176
3	--hidden-dim: 32 --d-model: 16 --heads: 2 --lstm-hidden: 16 --lstm-layers: 1 --dropout: 0.1	Early stopping at Epoch 27/100 Train Loss: 21.5347 - MAE: 63.4196 - WAPE: 0.1767 - SMAPE: 0.9006 Validation Loss: 25.5215 - MAE: 74.1687 - WAPE: 0.1537 - SMAPE: 0.6290 Test Loss: 25.1365 - MAE: 71.5808 - WAPE: 0.1501 - SMAPE: 0.6283
4	--hidden-dim: 256 --d-model: 128 --heads: 8 --lstm-hidden: 128 --lstm-layers: 1 --dropout: 0.3	Early stopping at Epoch 19/100 Train Loss: 15.2155 - MAE': 43.8038 - WAPE: 0.1220 - SMAPE: 0.8935 Validation Loss: 22.3384 - MAE: 63.7361 - WAPE: 0.1321

		- SMAPE: 0.5174 Test Loss: 23.0166 - MAE: 64.9195 - WAPE: 0.1362 - SMAPE: 0.7567

6.2 Spatio-Temporal Graph Neural Network (STGNN)

The STGNN is an architecture developed for multi-horizon demand forecasting that effectively captures both spatial and temporal dependencies. It offers an integrated approach to demand forecasting by modeling the interconnected entities - such as product and store features - over multiple time steps. The model processes the feature inputs alongside node information to output demand predictions for the next 28 days. Figure 6.6 illustrates the forecast for the Cleaning product category in Store 1, while Figure 6.7 presents the forecast for Cleaning product across all stores Favorita stores in Ecuador region. Detailed demand forecasts for each individual store is provided in Appendix E.

The core of the architecture consists of spatio-temporal blocks. Each spatio-temporal block is composed of temporal and graph convolutional blocks, linear layers, and the ReLU activation function.

The temporal blocks are responsible for capturing the sequential dependencies using dilated, causal, one-dimensional convolutions with causal padding. This ensures that predictions for a given time step are influenced by current and historical data, maintaining temporal causality.

Temporal convolutions operate per node, with shared weights across nodes. The graph convolution blocks capture the spatial relationships by processing

the underlying graph structure. At each time-step, a first-order graph convolution is performed according to the equation $H = \sigma(\hat{A} X W + b)$, where Â denotes the normalized adjacency matrix, X represents input model features, W and b are learnable parameters. σ is an activation function ReLU.

Linear layers are used to map hidden representations to the final output predictions, while the ReLU activation function introduces necessary non-linearity into the model.

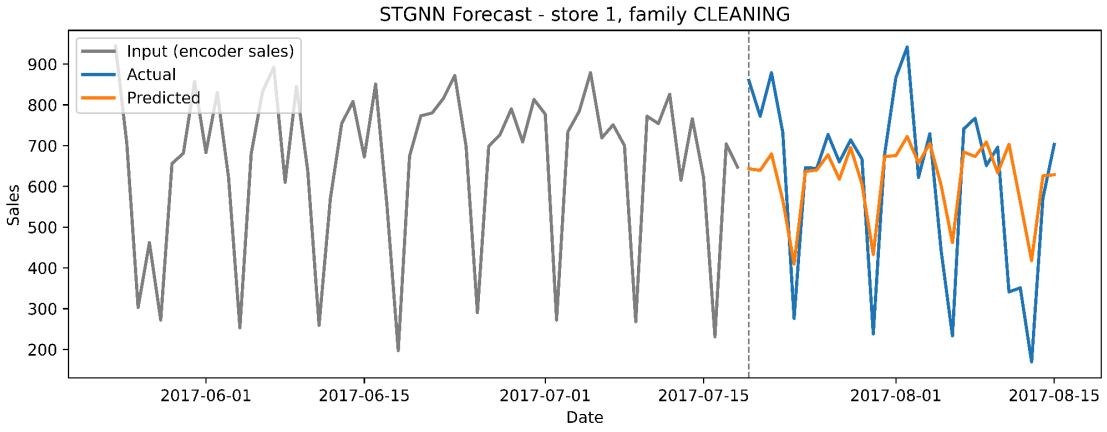


Figure 6.6: STGNN Model forecast for Cleaning in store 1

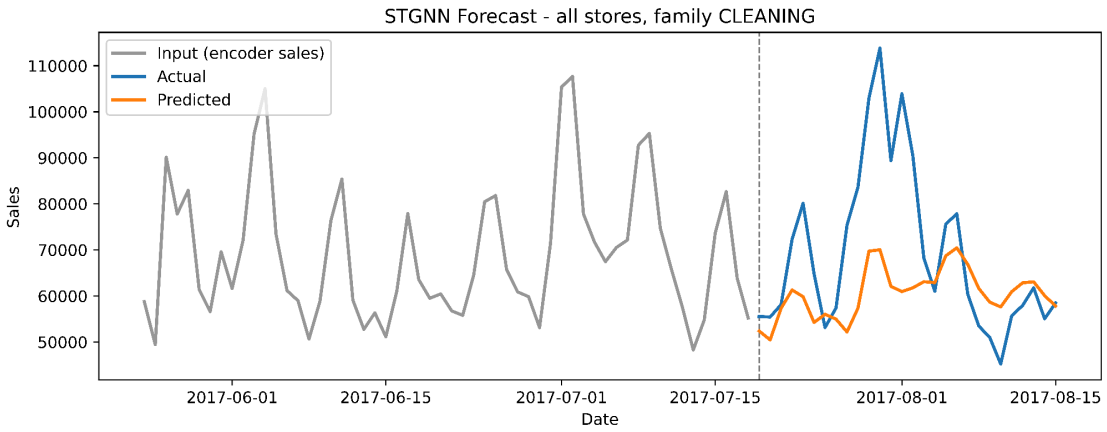


Figure 6.7: STGNN Model forecast for Cleaning in all stores

6.2.1 Graph Dataset

The spatio-temporal sliding window mechanism is employed to construct training, validation, and test sets for the STGNN model. In this framework, each node in the graph corresponds to a unique (store, family) pair, defined by assigning a unique node identifier to every combination of product family and store within the chronological data.

Feature values for each date and node are organized into matrices, which are subsequently stacked into a three-dimensional tensor with shape [T, N, F], where:

- **T**: Number of dates
- **N**: Number of nodes (distinct pairs of store and family)
- **F**: Number of feature columns

For each (store, family) pair, the feature set includes: *store_nbr, family, date, dow, month, weekofyear, id, sales, onpromotion, state, store_type, cluster, transactions, dcoilwtico, is_holiday, is_workday.*

6.2.1.1 Graph Construction

The STGNN operates on a unified graph structure that captures both relational dependencies between stores and product families. To construct this unified representation, two separate graphs are initially built - one for modeling relationships among stores (see Section 6.2.3) and other for modeling relationships among product families (see Section 6.2.4). These individual graphs are then integrated using the Kronecker-sum operation, resulting in a combined graph whose node set consists of all possible (store, family) pairs.

6.2.1.2 Store Graph

The store graph encodes relationships among stores based on both metadata and transaction correlation. It is constructed as an undirected, weighted graph. Nodes are individual stores, while edges are established between stores that exhibit meaningful similarity, when the computed edge weight exceeds zero.

The edge weight between a pair of stores (a, b) is computed according to predefined rules as follows:

- **+0.5**: if both stores belong to the same cluster
- **+0.3**: if both stores are in the same state
- **+corr(transactions_a, transactions_b)**: the Pearson correlation of the transaction time series

The Figure 6.8 shows the closer look into nodes and edges, while Appendix F provides a comprehensive overview of the entire constructed graph.

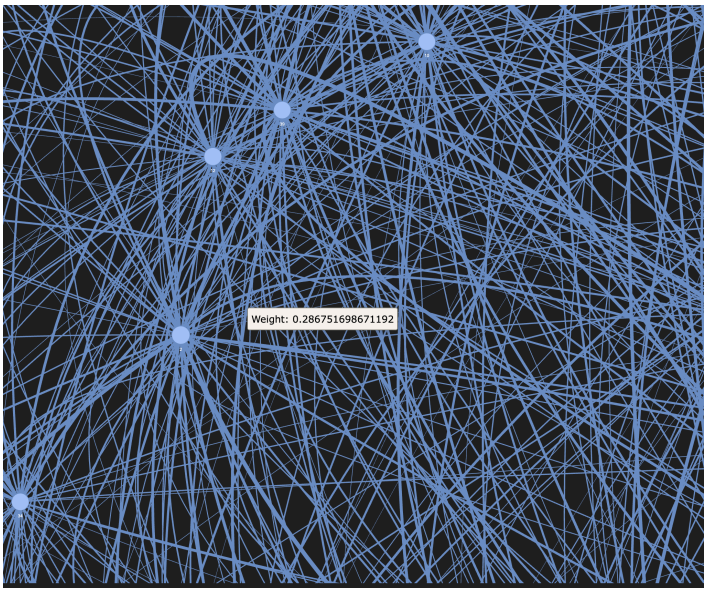


Figure 6.8: Store Graph close look

6.2.1.3 Family Graph

The family graph captures similarities between product families by analyzing their sales patterns over time. In this graph, each node represents a distinct product family. The relationship between pair of families (i, j) are quantified through weighted, undirected edges, with edge weights determined by the following procedure:

1. The Pearson correlation coefficient is computed over a 60-day sales window for products (i) and (j)
2. If the correlation coefficient exceeds 0.3, an undirected edge is established between product (i) and (j), with the correlation value assigned as the weight

The higher weights indicate stronger alignment in sales dynamics between families. Figure 6.9 illustrates the Product Family Graph at closer look, and the further big picture is allocated in Appendix F.

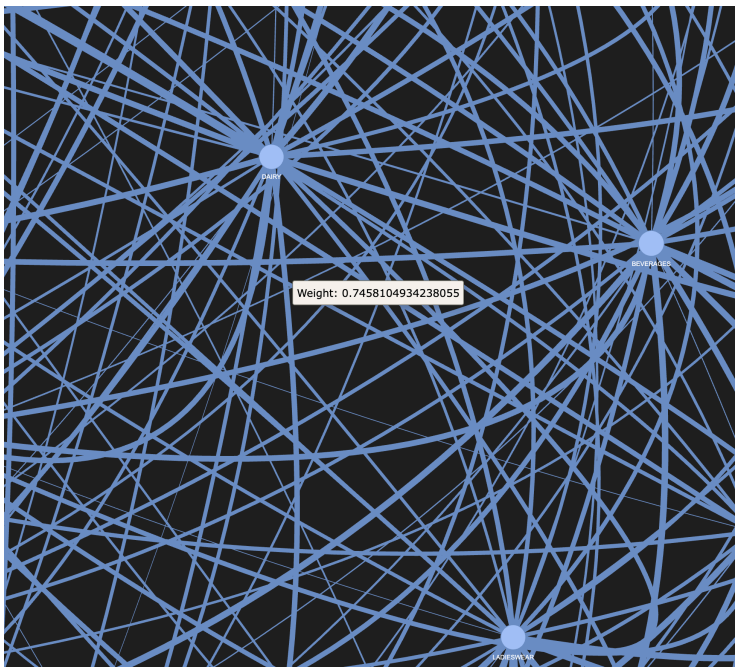


Figure 6.9: Product-Family Graph closer look

6.2.1.4 Kronecker-Sum Adjacency Matrix

The final graph structure for spatio-temporal GNN is constructed by normalizing and combining the adjacency matrices derived from previously built store and family graphs. This combination is achieved through the Kronecker-sum operation:

$$A \oplus B = A \oplus I_b + I_a \oplus B$$

where A and B are squared matrices of order a and b, respectively. I_n is the identity matrix of order n, and $\oplus$ denotes the Kronecker product.

6.2.2 XAI: GNN Explanation

The results of STGNN model predictions are interpreted at both the feature and neighbor levels, utilizing SHAP for feature-level explanations and an edge mask optimizer for neighbour-level explanations. Figure 6.10 depicts the input sequences for Node 2, which corresponds to Store 1 and product family Beauty, serving as the subject for interpretability analysis for the period from 2017-07-19 to 2017-08-15.

6.2.2.1 Feature-Level XAI: SHAP DeepExplainer

To quantify the contribution of individual input features for a certain node (corresponding to a store-product combination), the SHAP Deep Explainer algorithm is employed. In this experiment, 28 test samples were served to the Deep Explainer algorithm to establish a representative background distribution. The DeepExplainer calculates Shapley values, which estimate the contribution of each feature to the model's prediction for each individual sample. To assess overall feature importance, the absolute SHAP values are averaged across all

background test samples, yielding a global ranking of features based on relative influence on the model's prediction.

Figure 6.11 depicts the mean absolute Shapley values for each input feature, quantifying their relative contributions to the STGNN model's predictions.

Transactions importance 0.6492 is by far the most influential feature, indicating that historical transaction attribute is the primary driver in the model's demand predictions.

The next most important features—dow (day of week) 0.0033, sales 0.0031, weekofyear 0.0021, dcoilwtico (oil price indicator) 0.0020—have small but non-negligible impact.

Seasonality features such as promotional and calendar indicators - onpromotion 0.0002, is_holiday 0.0002, month 0.0002 - contribute minimally to the predictions.

Notably, is_workday has a mean absolute SHAP value of 0, indicating it has no contribution to the current model's decision-making.

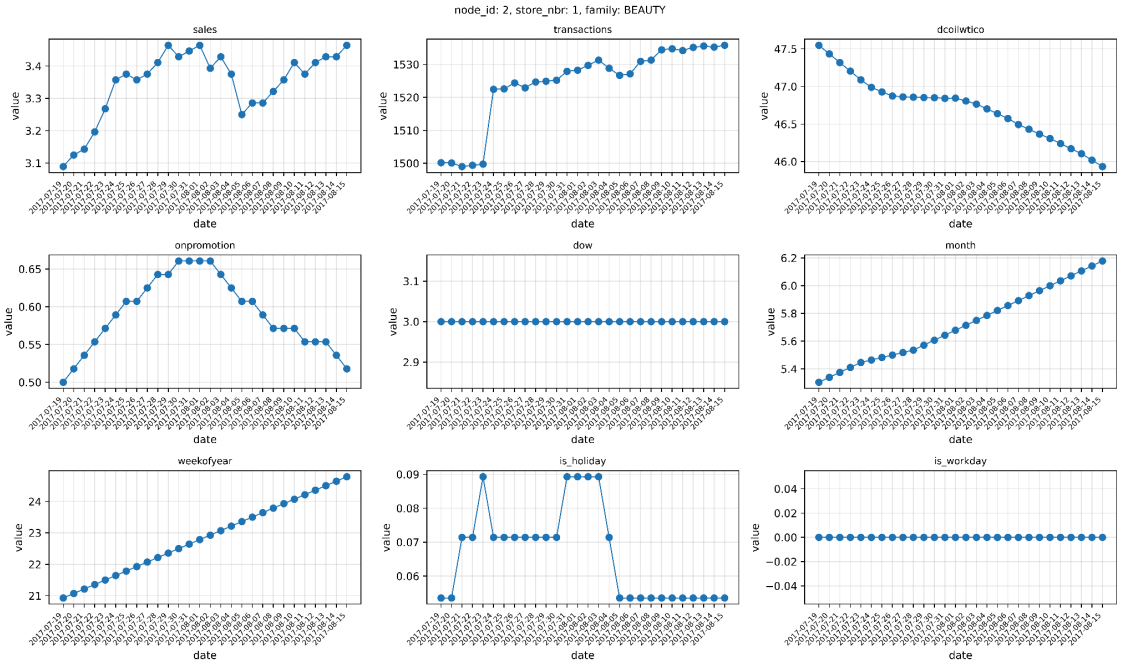


Figure 6.10: GNN explanation input - Node 2 (store 1, Beauty product)

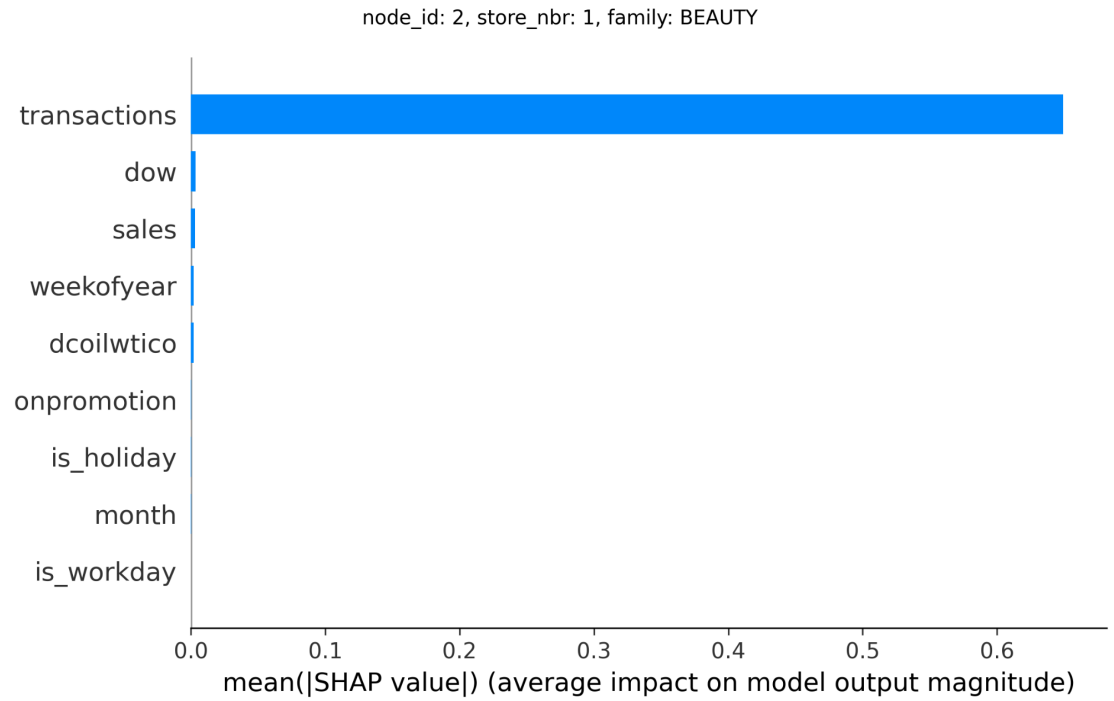


Figure 6.11: GNN explanation with SHAP

Table 6.4: Shapley Feature Mean Absolute Values

Feature	Mean absolute Shapley value
transactions	0.6492

dow	0.0033
sales	0.0031
weekofyear	0.0021
dcoilwtico	0.0020
onpromotion	0.0002
is_holiday	0.0002
month	0.0002
is_workday	0.0

6.2.2.2 Structure-Level XAI: Custom Edge-Mask Explainer

The underlying relationships of neighboring nodes is explained using the custom edge mask optimization methodology. This mask assigns a weight to each neighboring connection, representing its learned importance. The defined edge mask variable is optimized using gradient descent, incorporating L1 and entropy regularization. The optimization encourages the selection of the minimal necessary subset of connections, resulting in a sparse and interpretable mask. Thus the optimized edge mask quantifies the relative importance of each neighbour in forecasting the target node. Neighbors with larger mask values are interpreted as stores and products whose temporal patterns exert the greatest influence on the target node's prediction. Figure 6.12 shows the node 10 (eggs product at store 1) neighbours that influence demand prediction of the product at store 10, see Appendix 10 for details.

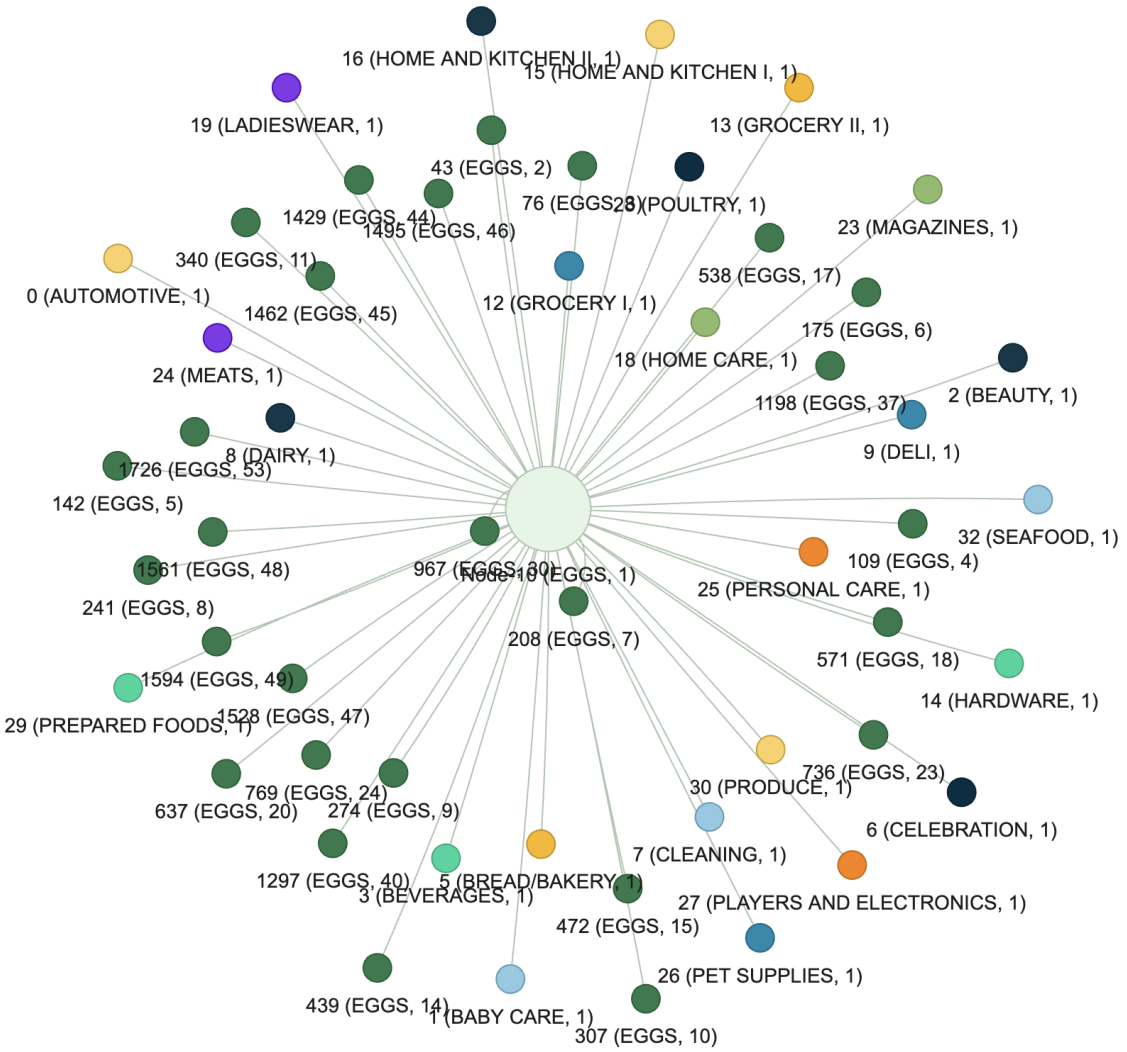


Figure 6.12: GNN explanation with Custom Edge Mask: Top neighbors

6.2.3 GNN Performance

The model training is conducted with a mini-batch size of 8, learning rate of 1×10^{-3}, and a dropout rate of 0.1 to mitigate overfitting. The model is designed to estimate the conditional quantiles at levels 0.1, 0.5, and 0.9, thereby providing a probabilistic range of forecast outputs. To ensure the reproducibility of results, a random seed of 42 is set prior to training. Overfitting is further prevented by implementing early stopping, with a patience threshold of 5 epochs. Table 6.5 presents the experiment results of three model configurations.

From a computational standpoint, it is important to note that configuring the model with hidden size of 128 or greater leads to GPU A100 memory allocation errors. Consequently, the model's hidden size is restricted to the maximum feasible model capacity within the available computational resources.

As demonstrated in Figure 6.13, the configuration designated as Model 2 achieves the highest performance on the test set, attaining an MAE value of 61.31 with the specified parameters.

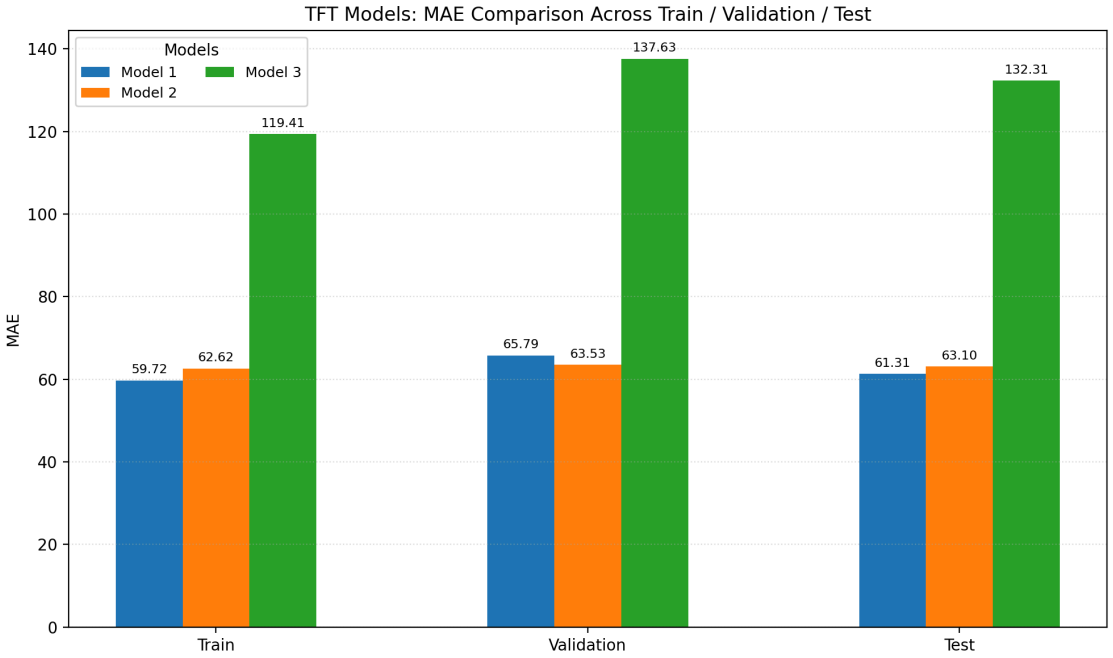


Figure 6.13: GNN Multiple Architecture MAE Comparison

Table 6.5: GNN Comparative Performance Results

Parameters	Result in Metrics
--hidden: 64 --blocks: 3 --kernel: 3	Early stopping triggered at epoch 58 Train Loss: 20.7103 - MAE 59.7155 - WAPE: 0.1664 - SMAPE: 0.9001 Validation Loss 21.6929 - MAE: 65.7865

	- WAPE: 0.1363 - SMAPE: 0.5792 Test Loss: 20.4918 - MAE: 61.3146 - WAPE: 0.1286 - SMAPE: 0.5831
--hidden: 32 --blocks: 3 --kernel: 3	Early stopping triggered at epoch 99 Train Loss: 21.8531 - MAE: 62.6185 - WAPE: 0.1745 - SMAPE: 0.9088 Validation Loss 21.1624 - MAE: 63.5278 - WAPE: 0.1316 - SMAPE: 0.5993 Test Loss: 21.1078 - MAE: 63.1049 - WAPE: 0.1324 - SMAPE: 0.5971
--hidden: 16 --blocks: 1 --kernel: 1	Early stopping triggered at epoch 73 Train Loss: 40.9545 - MAE: 119.4102 - WAPE: 0.3328 - SMAPE: 0.9986 Validation Loss 45.2701 - MAE: 137.6304 - WAPE: 0.2851 - SMAPE: 0.7129 Test Loss: 43.8707 - MAE: 132.3087 - WAPE: 0.2775 - SMAPE: 0.7064

6.3 Hybrid Model

6.3.1 Hybrid Model Architecture

The proposed hybrid model integrates a GNN with a TFT to exploit both relational and temporal patterns in the input data. The architecture is composed of two primary components: the Relational Encoder and the TFT blocks.

The RE employs a GNN to create node embeddings, thereby capturing the relational structure among product family and store number entities. The node embeddings are concatenated with static input features, and then passed to the TFT block static covariate.

Architecturally, the RE block consists of two sequential Linear layers. Each linear layer is followed by a dropout layer and a ReLU activation function to promote regularization and non-linearity. Key hyperparameters for this block include the GNN input dimension, hidden dimension, output dimension, and dropout ratio.

The TFT component is responsible for multi-horizon time series forecasting and requires a range of configurable parameters such as data input dimensions for each input data feature that are encoder, decoder and static attributes. The model specific parameters are adjusted for hidden size, number of attention heads, LSTM size, dropout rate.

Figure 6.14 shows the seafood prediction at store 1, and Figure 6.15 shows the seafood product demand prediction across all Favorita stores.

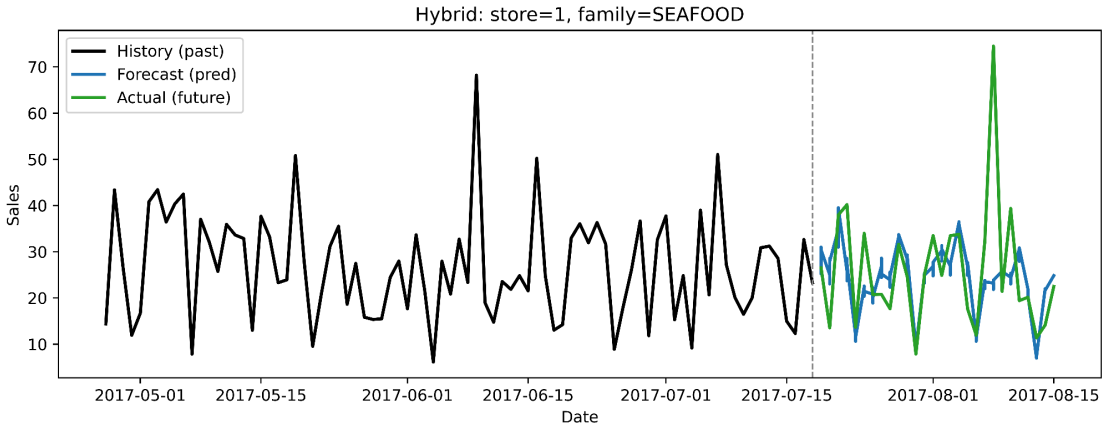


Figure 6.14: Hybrid Model result for store 1, family Seafood

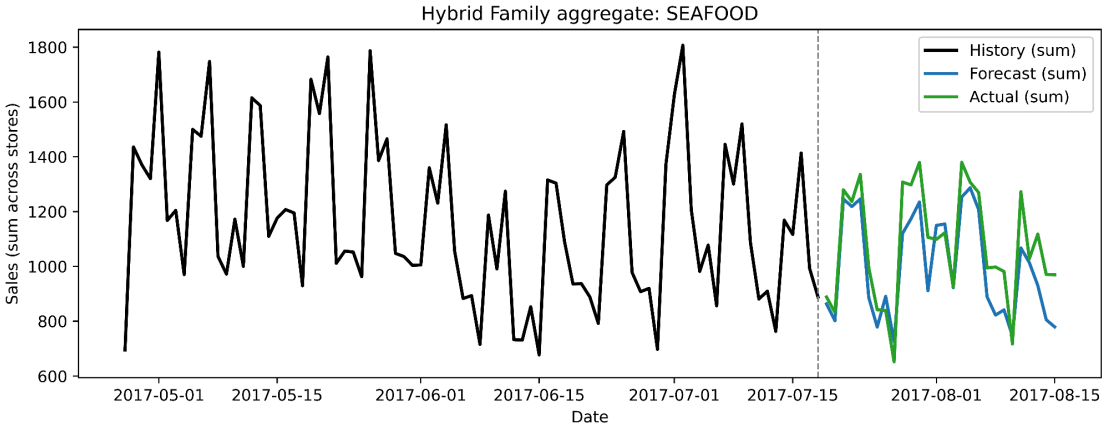


Figure 6.15: Hybrid Model result across all stores, family Seafood

6.3.2 Hybrid Model Data

Node embeddings creation comprises several consequent steps: node indexing, static feature matrix organization, and adjacency matrix construction.

First, product nodes are indexed by mapping each unique combination of store number and product family to contiguous node identifiers. This bi-directional mapping is constructed from the chronological data, enabling efficient and unambiguous identification of nodes throughout the modeling workflow.

Next, for each node - identified by store number and product family- the relevant records are retrieved from the chronological dataset. The static features obtained from these records are concatenated to form a comprehensive static feature matrix, encapsulating static characteristics of each node.

Finally, an adjacency matrix for the training set is created using the Pearson correlation coefficient computation formula. These correlations serve to quantify the similarity in sales patterns between pairs of nodes. For each node, only the top 10 most similar neighbors (those with the highest correlation coefficients) are retained in the adjacency matrix to define the local relational structure.

$$r = \frac{\sum(x - m_x)(y - m_y)}{\sqrt{\sum(x - m_x)^2 \sum(y - m_y)^2}}$$

6.3.3 Hybrid XAI: LIME

To interpret the predictions of the hybrid model, the LIME (Local Interpretable Model-agnostic Explanations) method is employed. LIME is a model-agnostic explanation technique that assesses the impact of individual input features on a given prediction outcome by approximating the model locally with an interpretable surrogate. The explainer utilizes both sample data and feature names to mimic the behavior of the original model in the vicinity of the instance being analyzed.

In this experiment, the LIME explainer employs 200 background samples to fit a surrogate model under a tabular regression setting. This approach yields feature-wise importance scores for the selected instance, quantifying the contribution of each feature to the prediction result. The feature importance

weights reflect both the magnitude and direction of contribution: positive weights indicate features that increase the predicted value, while negative weights suggest a suppressing effect.

The results presented in Figure 6.16 summarize the LIME analysis for store number 1 and product family AUTOMOTIVE, highlighting the most influential features for this specific prediction instance.

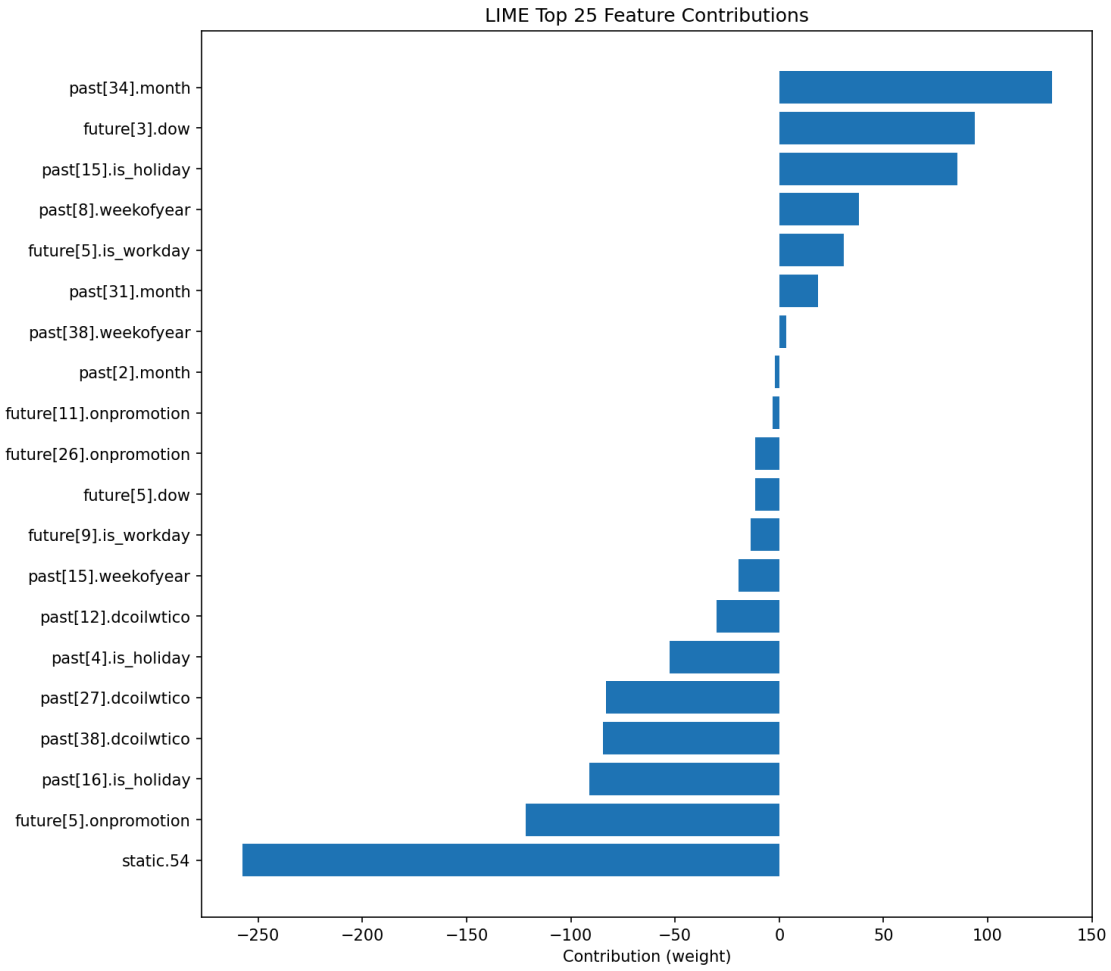


Figure 6.16: Hybrid LIME explanation for store-product (1, Automative)

Several temporal features exhibit substantial positive influence on the model's prediction. Notably, past[34].month (weight = 130.81), future[3].dow (93.73), and past[15].is_holiday (85.42) emerge as the most significant contributors, underscoring pronounced dependencies on seasonality,

day-of-week, and historical holiday effects in the forecast. Additional temporal and calendar-related features, such as past[8].weekofyear (38.06), future[5].is_workday (31.02), past[31].month (18.62), and past[38].weekofyear (3.41), also positively influence predictions, indicating the hybrid model's sensitivity to recurring temporal cycles and workday patterns.

Conversely, several features display notable negative contributions, acting to suppress the predicted value. Among these, the static feature static.54 (weight = -257.73) has the most pronounced impact, suggesting that this specific static attribute is strongly associated with reduced forecast outputs. Other features exhibiting negative weights include future[5].onpromotion (-121.68), past[16].is_holiday (-91.28), past[38].dcoilwtico (-84.70), past[27].dcoilwtico (-83.20), past[4].is_holiday (-52.78), past[12].dcoilwtico (-30.07), and past[15].weekofyear (-19.53). These results suggest that certain promotional periods, holidays, and economic indicators (oil prices) have a dampening effect on the model's demand prediction for the analyzed instance.

Taken together, the LIME analysis reveals that the hybrid model's prediction for store number 1 and product family AUTOMOTIVE is primarily driven by its temporal features, particularly those reflecting seasonality and calendar events. In contrast, several static and external variables exert a substantial negative influence. The full spectrum of feature weights thus provides a nuanced understanding of how different data attributes contribute to the forecast for the specific instance analyzed.

6.3.4 Hybrid Model Performance

Among the four evaluated configurations of hybrid model parameters, Model 3 demonstrated the best performance, achieving the lowest MAE of 59.76 on the test set.

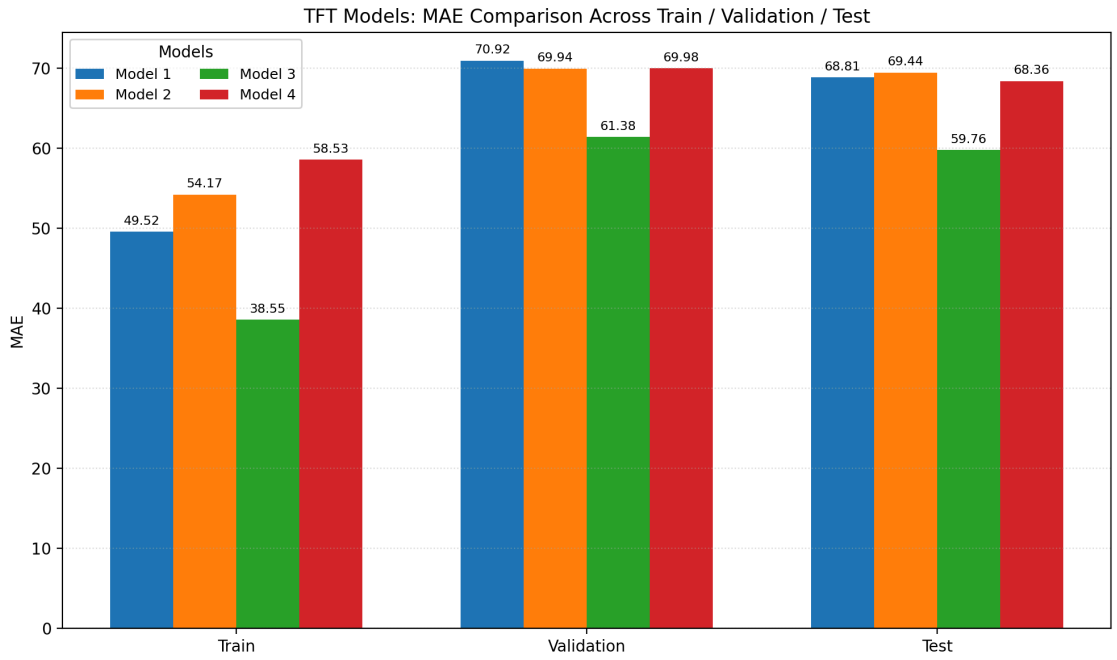


Figure 6.16: Hybrid Model MAE comparison

Table 6.5: Hybrid Comparative Performance Results

Parameters	Performance in Metrics
--hidden-dim: 128 --d-mode: 64 --heads: 4 --lstm-hidden: 64 --lstm-layers: 1 --gnn-hidden: 64 --gnn-embed: 32	Early stopping at epoch 33 Train Loss: 17.0956 - MAE: 49.5245 - WAPE: 0.1380 - SMAPE: 0.9065 Validation Loss 24.7882 - MAE: 70.9200 - WAPE: 0.1470 - SMAPE: 0.5876 Test loss: 24.46641 - MAE: 68.8060 - WAPE: 0.1443

	- SMAPE: 0.5411
--hidden-dim: 64 --d-mode: 32 --heads: 4 --lstm-hidden: 32 --lstm-layers: 1 --gnn-hidden: 32 --gnn-embed: 16	Early stopping at epoch 43 Train Loss: 18.5534 - MAE: 54.1713 - WAPE: 0.1509 - SMAPE: 0.8951 Validation Loss 24.4104 - MAE: 69.9379 - WAPE: 0.1449 - SMAPE: 0.5434 Test Loss: 24.3820 - MAE: 69.4415 - WAPE: 0.1456 - SMAPE: 0.6860
--hidden-dim: 256 --d-model: 128 --heads: 8 --lstm-hidden: 64 --lstm-layers: 3 --gnn-hidden: 128 --gnn-embed: 64	Early stopping at epoch 60 Train Loss: 13.0254 - MAE: 38.5470 - WAPE: 0.1074 - SMAPE: 0.8643 Validation Loss 20.8178 - MAE': 61.3819 - WAPE: 0.1272 - SMAPE: 0.5249 Test Loss: 20.7917 - MAE: 59.7633 - WAPE: 0.1253 - SMAPE: 0.5971
--hidden-dim: 32 --d-model: 16 --heads: 2 --lstm-hidden: 32 --lstm-layers: 1 --gnn-hidden: 16 --gnn-embed: 8	Early stopping at epoch 51 Train Loss: 19.9101 - MAE: 58.5270 - WAPE: 0.1631 - SMAPE: 0.8861 Validation Loss 24.3748 - MAE: 69.9758 - WAPE: 0.1450 - SMAPE: 0.6577 Test Loss: 24.0726 - MAE: 68.3648

	- WAPE: 0.1434 - SMAPE: 0.5494

CHAPTER 7: SUMMARY

References

Lalou, P., Ponis, S.T.. & Efthymiou, O.K.. (2020). Demand Forecasting of Retail Sales Using Data Analytics and Statistical Programming. Management & Marketing, 15(2), 2020. 186-202 https://doi.org/10.2478/mmcks-2020-0012

Sukel, Maarten & Rudinac, Stevan & Worring, Marcel. (2023). Multimodal Temporal Fusion Transformers Are Good Product Demand Forecasters. 10.48550/arXiv.2307.02578.

Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, Tomas Pfister (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting, 37(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Kozodoi, N., Zinovyeva, L., Valentin, S., Pereira, J., & Agundez, R. (2023). Probabilistic demand forecasting with graph neural networks. Retrieved from https://www.amazon.science/publications/probabilistic-demand-forecasting-with-graph-neural-networks

Aktas, M. Y., Ji, T., & Lu, C.-T. (2024). Time series forecasting with GCN-LSTM based unified model for product demand prediction. In 2024 IEEE International Conference on Big Data (BigData) (pp. 5901–5906). https://doi.org/10.1109/BigData62323.2024.10825969

Ngeni, F., Kutela, B., Chengula, T. J., Ruseruka, C., Musau, H., Novat, N., Indah, D. A., & Kasomi, S. (2024). Prediction of bike-sharing station demand using explainable artificial intelligence. Machine Learning with Applications, 17, Article 100582. https://doi.org/10.1016/j.mlwa.2024.100582

Favorita Kaggle Dataset https://www.kaggle.com/competitions/store-sales-time-series-forecasting/overview

Name	Description
Store_nbr	Number of a store in an Ecuador area
family	Product type
date	Date of the purchase
dow	Day of the week
month	Month of the year
weekofyear	Week of the year
id	
sales	Sales of the product family in a specific store
onpromotion	Promotion label of the product type in a store number
state	State in the Ecuador area
store_type	Store type
cluster	A grouping of similar stores
transactions	Number of transactions on that date
dcoilwtico	Oil price on the stated date
is_holiday	1 if the date is a (non-transferred) holiday.
is_workday	1 if the date is explicitly marked a “Work Day”.

Table Appendix 1. Dataset feature descriptions

Table Appendix 2. TFT Model Description

Parameter Name	Type	Explanation
--enc-len	int	Encoder sequence length (number of past time steps used for each forecast).

--dec-len	int	Decoder sequence length (forecast horizon, number of future steps to predict).
--batch-size	int	Number of samples per training batch.
--epochs	int	Number of training epochs.
--lr	float	Learning rate.
--hidden-dim	int	Hidden layer dimension in various network components.
--d-model	int	Model embedding dimension (core internal feature size).
--heads	int	Number of attention heads in multi-head attention layers.
--lstm-hidden	int	LSTM hidden state dimension.
--lstm-layers	int	Number of stacked LSTM layers.
--dropout	float	Dropout rate (fraction of units dropped for regularization during training).
--quantiles	str	Quantiles for prediction output (used for quantile loss/objective, e.g. median, upper/lower).
--stride	int	Step size for sliding windows/indexing in dataset construction.
--seed	int	Random seed for reproducibility.
--early-stopping-patience	int	Number of epochs to wait for validation loss improvement before stopping training.
--early-stopping-min-delta	float	Minimum improvement in validation loss required to reset the patience counter.
--train-flag	bool	Whether to perform training (vs. only evaluation/inference).

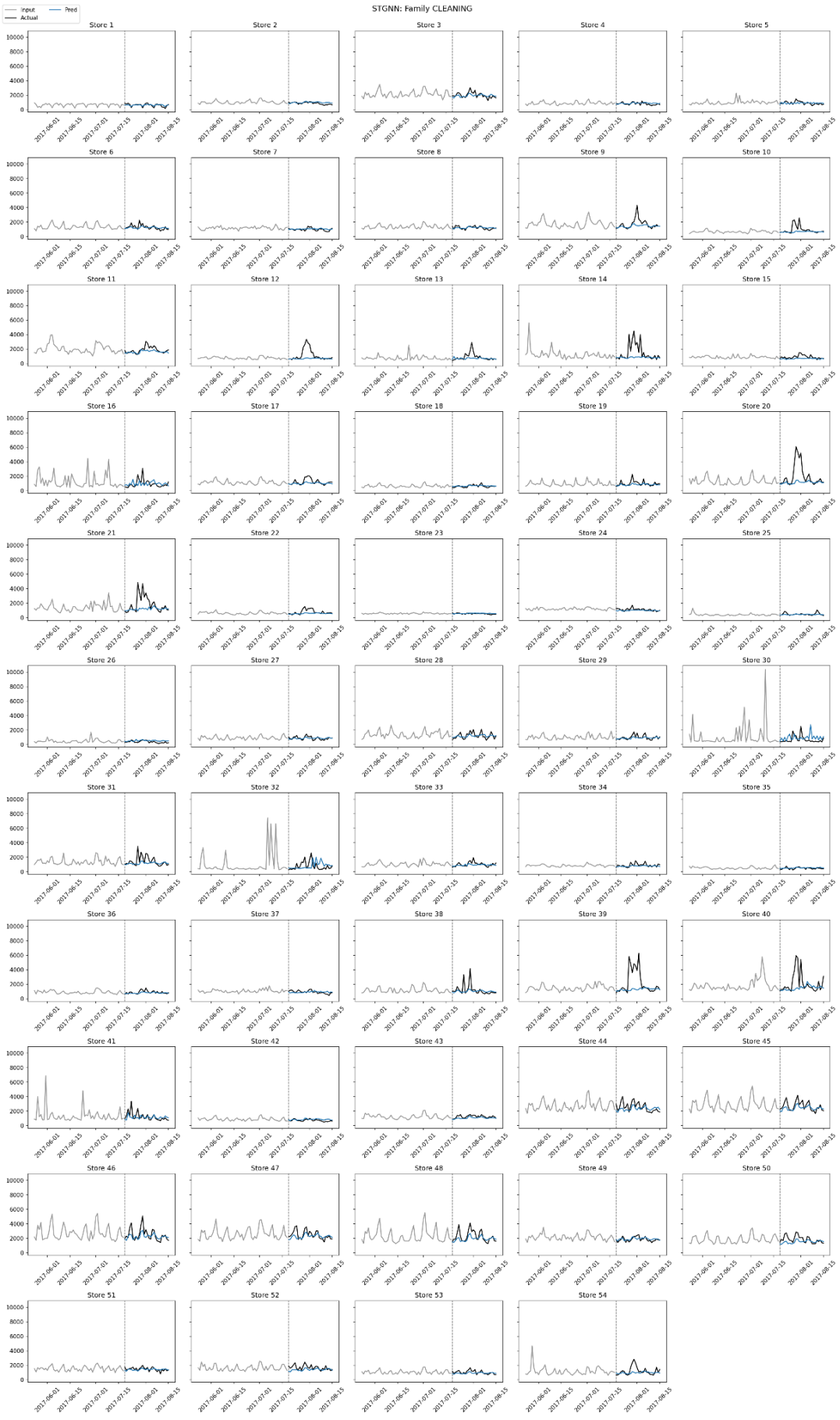
Appendix C: Dataset construction from multiple files

- Sales transactions: *(id, date, store_nbr, family, sales, onpromotion)*
- Store metadata: *(store_nbr, city, state, type, cluster)*

- Oil prices: *(date, dcoilwtico)*
- Transaction volumes: *(date, store_nbr, transactions)*
- Holidays and events: *(date, type, locale, locale_name, description, transferred)*

Appendix D: TFT all stores

Appendix E: STGNN all stores



Appendix F: Store Graph

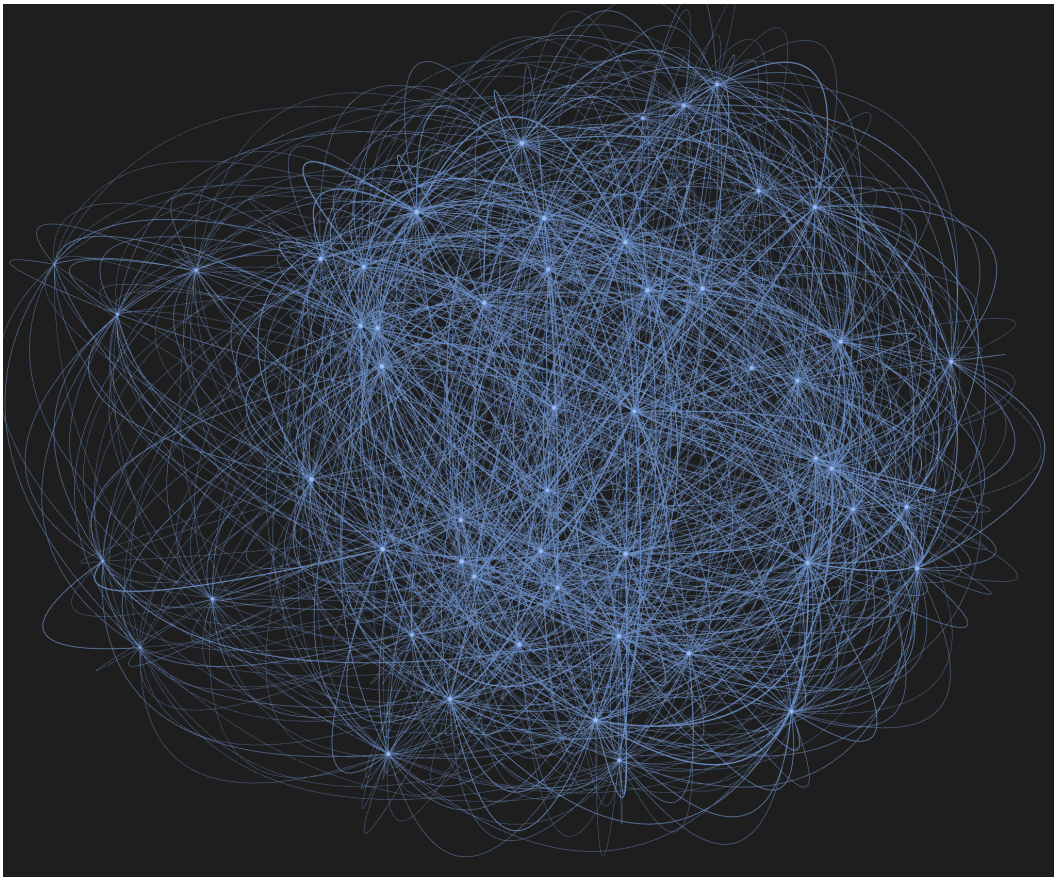


Figure 6.2.3.1 Store Graph illustration

Appendix G: Product Family Graph

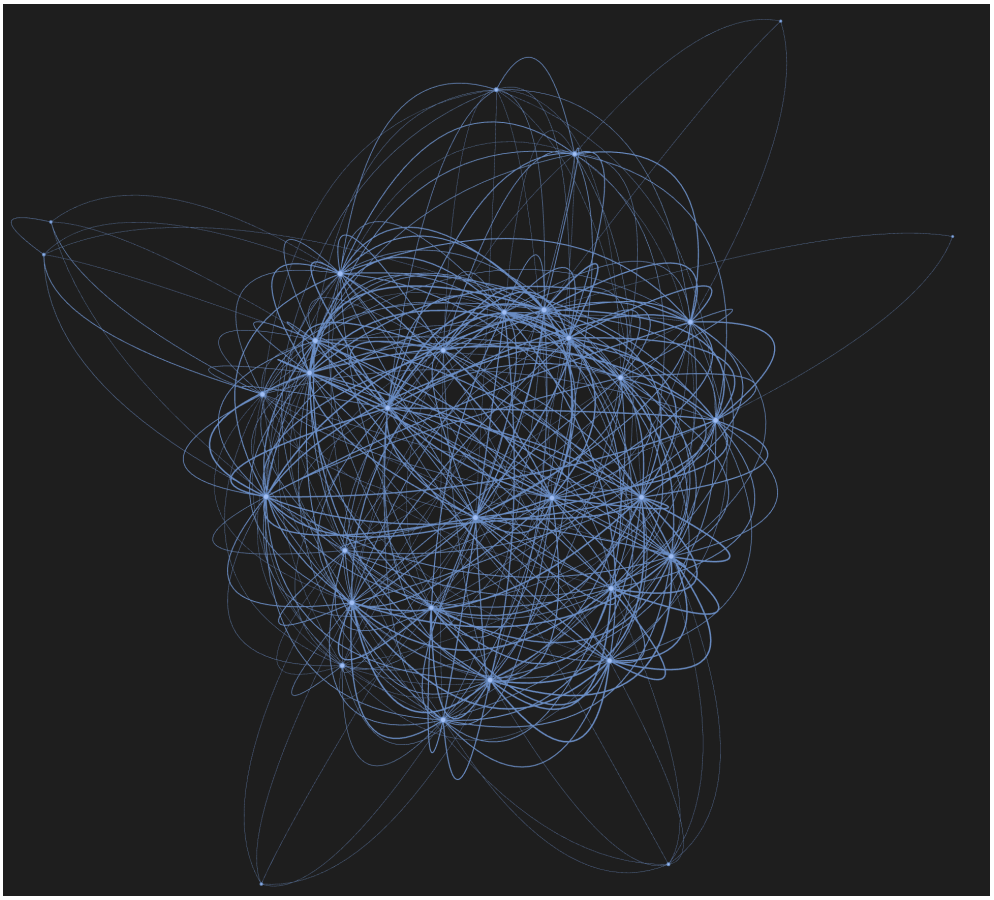


Figure 6.2.4.1 Product-Family Graph

Appendix H: Hybrid all stores

Appendix I: STGNN node 10 top neighbors:

neighbor_node_id,importance,store_nbr,family

12,0.761333167552948,1,GROCERY I

3,0.7608758807182312,1,BEVERAGES

30,0.7601446509361267,1,PRODUCE

8,0.7539601922035217,1,DAIRY

7,0.7528702616691589,1,CLEANING

5,0.742431104183197,1,BREAD/BAKERY

28,0.7401461601257324,1,POULTRY

24,0.738740861415863,1,MEATS

967,0.7372738718986511,30,EGGS

76,0.7368155121803284,3,EGGS

1429,0.736394464969635,44,EGGS

1594,0.734183669090271,49,EGGS

208,0.7336453199386597,7,EGGS

10,0.729555070400238,1,EGGS

241,0.7294285893440247,8,EGGS

769,0.726266622543335,24,EGGS

571,0.7229378819465637,18,EGGS

538,0.7192614674568176,17,EGGS

1462,0.7168235778808594,45,EGGS

1297,0.7151981592178345,40,EGGS

1528,0.7134265899658203,47,EGGS

175,0.7071603536605835,6,EGGS

43,0.7067459225654602,2,EGGS

109,0.7044850587844849,4,EGGS

25,0.7029313445091248,1,PERSONAL CARE

18,0.7027408480644226,1,HOME CARE

1198,0.6967707276344299,37,EGGS

9,0.6851966977119446,1,DELI

274,0.6817722320556641,9,EGGS

1726,0.6541054248809814,53,EGGS

142,0.6370068788528442,5,EGGS

637,0.6151273846626282,20,EGGS

1495,0.603293240070343,46,EGGS

472,0.5985670685768127,15,EGGS

340,0.56599360704422,11,EGGS

1561,0.5513509511947632,48,EGGS

736,0.5073283314704895,23,EGGS

29,0.47849878668785095,1,PREPARED FOODS

15,0.25167056918144226,1,HOME AND KITCHEN I

16,0.23876164853572845,1,HOME AND KITCHEN II

32,0.23790088295936584,1,SEAFOOD

13,0.2366286814212799,1,GROCERY II

307,0.2337900847196579,10,EGGS

439,0.22978700697422028,14,EGGS

6,0.22900919616222382,1,CELEBRATION

19,0.22707080841064453,1,LADIESWEAR

27,0.22509019076824188,1,PLAYERS AND ELECTRONICS

23,0.22469142079353333,1,MAGAZINES

26,0.2244776338338852,1,PET SUPPLIES

14,0.22340546548366547,1,HARDWARE

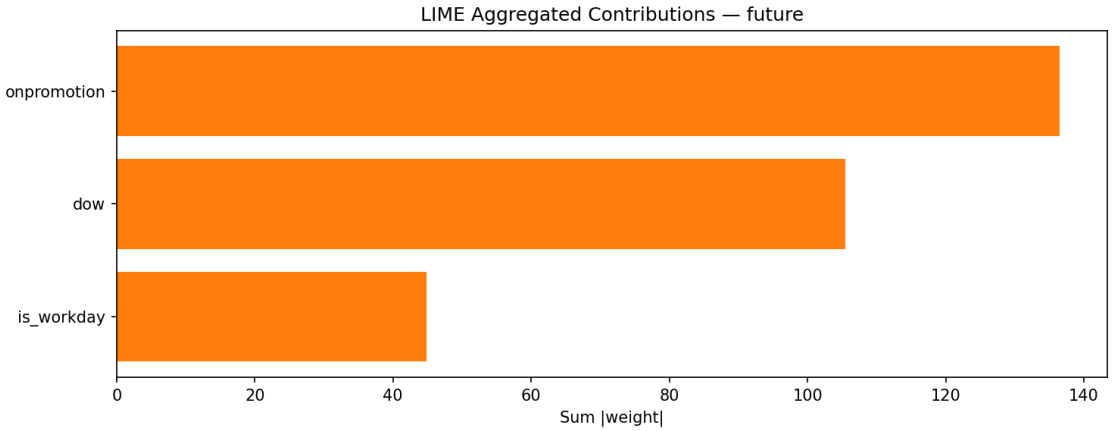
0,0.22337393462657928,1,AUTOMOTIVE

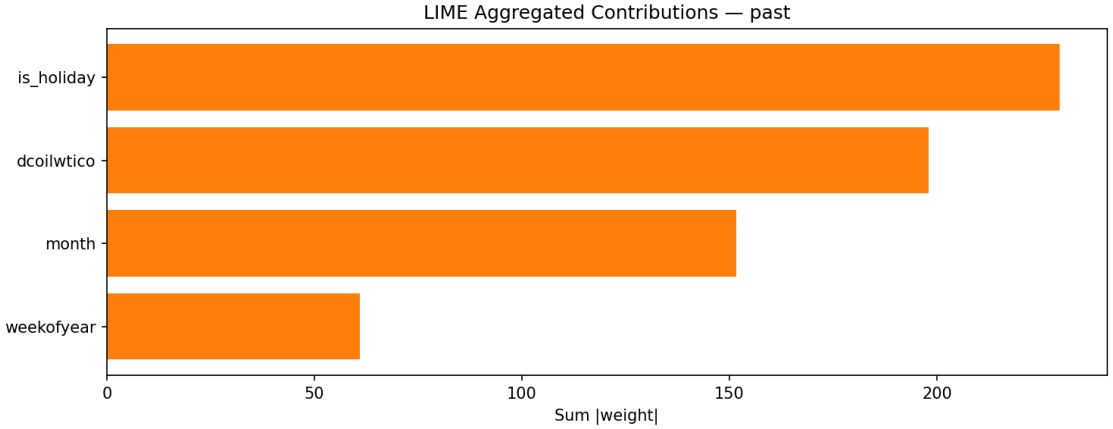
2,0.22313988208770752,1,BEAUTY

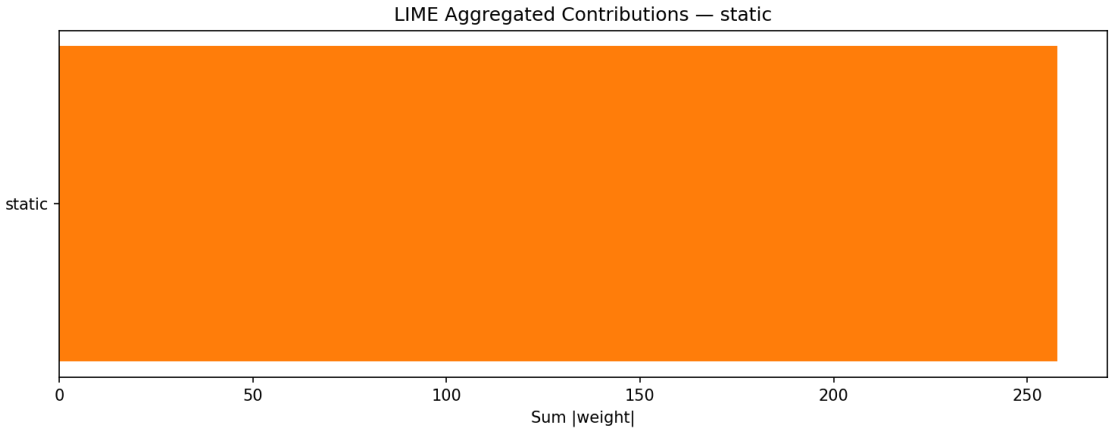
1,0.22244596481323242,1,BABY CARE

Appendix J: Hybrid LIME explanation

feature,weight
past[34].month,130.81417873895026
future[3].dow,93.7328357869461
past[15].is_holiday,85.4225635578236
past[8].weekofyear,38.05082383139763
future[5].is_workday,31.018024249900847
past[31].month,18.615793346165237
past[38].weekofyear,3.4062068889605386
past[2].month,-2.2725689199885992
future[11].onpromotion,-3.161681797265286
future[26].onpromotion,-11.673573360462429
future[5].dow,-11.730119581890312
future[9].is_workday,-13.886950892591363
past[15].weekofyear,-19.533223099411565
past[12].dcoilwtico,-30.072511183298133
past[4].is_holiday,-52.7783944207709
past[27].dcoilwtico,-83.19992389017256
past[38].dcoilwtico,-84.69727768434223
past[16].is_holiday,-91.2766969285098
future[5].onpromotion,-121.6793679484997
static.54,-257.7339500657491


LIME Aggregated Contributions — future
onpromotion
dow
is_workday
0
20
40
60
80
100
120
140
Sum |weight|


LIME Aggregated Contributions — past
is_holiday
dcoilwtico
month
weekofyear
0
50
100
150
200
Sum |weight|


LIME Aggregated Contributions — static
static
0
50
100
150
200
250
Sum |weight|