

Prague University of Economics and Business

Faculty of Informatics and Statistics



**Detection of Point-Wise Anomalies in
Large-Scale Hierarchical Time Series:
Methods and Applications**

MASTER THESIS

Study program: Data and Analytics for Business

Author: Bc. Magdalena Marie Šarpatková, MA.

Supervisor: Ing. Pavel Zimmermann, Ph.D.

Prague, May 2025

Acknowledgement

I would like to thank my family for their constant support, patience, and encouragement throughout my academic and professional journey. Their presence has been invaluable. I also wish to thank my supervisor, Ing. Pavel Zimmermann, Ph.D., for his guidance and insight, which were essential in helping me complete this thesis successfully.

Abstrakt

Detekce anomálií ve vícerozměrných časových řadách je klíčovou výzvou v mnoha oblastech typu financí, zdravotnictví, kybernetické bezpečnosti, obchodu a mnoha dalších. Schopnost identifikovat neočekávané odchylky v rozsáhlých datových souborech má významné dopady na rozhodování a efektivitu provozu. Tato práce se zaměřuje na detekci odlehklých pozorování v hierarchických, vícedimenzionálních časových řadách.

Dataset představuje specifickou výzvu kvůli své hierarchické struktuře, která obsahuje více obchodů a oddělení s vzájemně závislými vzory prodeje v čase, což z práce činí komplexní problém časových řad. První část této práce poskytuje rozsáhlý přehled teorie a metod detekce anomálií, přičemž hodnotí jejich vhodnost pro rozsáhlá a hierarchická data časových řad.

Na základě závěrů z teoretické části jsou vybrány a implementovány vybrané metody detekce anomálií. Druhá část práce se věnuje aplikaci těchto metod na data a porovnává jejich efektivitu v identifikaci významných odlehklých hodnot. Výsledky aplikace jsou následně analyzovány s cílem posoudit jejich praktické dopady na reálné aplikace.

Tato práce přispívá v oblasti detekce anomálií identifikací robustních metodologií pro hierarchická časová data a poskytuje využitelné poznatky pro podniky zabývající se rozsáhlým predikováním prodeje. Získané výsledky mohou podpořit informovanější rozhodování, snížit finanční ztráty nebo zlepšit strategické řízení provozu.

Klíčová slova

Bodová detekce anomálií, rozsáhlé hierarchické multivariační časové řady, dekompozice, nesupervizované modely, interpretovatelnost, škálovatelnost, evaluace, přesnost, hierarchická rekonziliace.

Abstract

Anomaly detection in multivariate time series data is a critical challenge in various domains, including finance, healthcare, cyber security, retail, and many other fields. The ability to identify unexpected patterns or deviations in large-scale datasets has significant implications for decision-making and operational efficiency. This thesis focuses on the detection of outliers in hierarchical, high-dimensional time series data.

The dataset presents a unique challenge due to its hierarchical structure of multiple stores and departments with interdependent sales patterns over time, making it a complex time series problem. In the first part of this thesis, a comprehensive review of theory and anomaly detection models is conducted, assessing their suitability for large scale hierarchical time series data.

Based on the findings from the research phase, anomaly detection methods are selected and implemented on the dataset. The second part of the thesis applies these approaches to the data, comparing their effectiveness in identifying significant outliers. The results are then analyzed to assess their practical implications for real-world applications.

This study contributes to the field by identifying robust methodologies for outlier detection in hierarchical time series data and providing actionable insights for businesses dealing with large-scale sales forecasting. The findings can support more informed decision-making, reducing financial losses and enhancing operational strategies.

Keywords

Point-wise anomaly detection, large-scale hierarchical multivariate time series, decomposition, unsupervised models, interpretability, scalability, evaluation, precision, hierarchical reconciliation.

Table of Contents

Introduction.....	10
Motivation.....	10
Research Questions and Objectives.....	10
Methodology.....	11
1 Business Understanding.....	12
2 Machine Learning Understanding.....	14
2.1 Problem Definition.....	14
2.1.1 Point-Wise Anomaly Detection.....	14
2.1.2 Large-Scale Hierarchical Multivariate Time Series.....	15
2.2 Choice of Decomposition Methods.....	16
2.2.1 Prophet Forecasting Mechanism.....	16
2.2.2 Seasonal-Trend Decomposition using LOESS Forecasting Mechanism.....	16
2.2.3 TimeGPT Forecasting Mechanism.....	17
2.3 Choice of Anomaly Detection Methods.....	17
2.3.1 Z-Scores Anomaly Detection Mechanism.....	17
2.3.2 Thresholding Anomaly Detection Mechanism.....	18
2.3.3 Isolation Forest Anomaly Detection Mechanism.....	18
2.3.4 K-Nearest Neighbors Anomaly Detection Mechanism.....	19
2.3.5 Local Outlier Factor Anomaly Detection Mechanism.....	20
2.3.6 HDBSCAN Anomaly Detection Mechanism.....	20
2.3.7 One Class SVM Anomaly Detection Mechanism.....	21
2.3.8 COPOD Anomaly Detection Mechanism.....	21
2.3.9 Bayesian Change Point Detection Anomaly Detection Mechanism.....	22
2.3.10 Mahalanobis Distance Anomaly Detection Mechanism.....	23
2.3.11 Gaussian Mixture Model Anomaly Detection Mechanism.....	23
2.3.12 Autoencoders Anomaly Detection Mechanism.....	24
2.3.13 Deep SVDD Anomaly Detection Mechanism.....	25
2.4 Ensemble Detection.....	26
2.5 Interpretability.....	26
2.6 Chapter Summary.....	27
3 Data Understanding.....	29
3.1 Initial Data Collection.....	29
3.2 Data Description.....	29
3.3 Data Description.....	30
3.3.1 Distribution of Weekly Sales.....	30
3.3.2 Trend and Seasonality.....	31
3.3.3 Feature Relationships.....	32
3.3.4 Data Quality Verification.....	35
3.4 Chapter Summary.....	35
4 Data Preparation.....	36
4.1 Data Selection and Integration.....	36
4.2 Data Cleaning.....	36
4.3 Feature Construction.....	37
4.4 Injecting Anomalies.....	37
4.5 Scaling.....	38
4.6 Chapter Summary.....	39
5 Modelling.....	40
5.1 Model Assessment.....	40
5.2 Decomposition Methods.....	42

5.2.1 Prophet Decomposition.....	42
5.2.2 STL Decomposition.....	44
5.2.3 TimeGPT Decomposition.....	46
5.3 Anomaly Detection Models.....	48
5.3.1 Z-Scores Anomaly Detection Model.....	48
5.3.2 Thresholding Anomaly Detection Model.....	49
5.3.3 Isolation Forest Anomaly Detection Model.....	50
5.3.4 K-Nearest Neighbors Anomaly Detection Model.....	51
5.3.5 Local Outlier Factor Anomaly Detection Model.....	52
5.3.6 HDBSCAN Anomaly Detection Model.....	53
5.3.7 One-Class SVM Anomaly Detection Model.....	54
5.3.8 Bayesian Change Point Detection Anomaly Detection Model.....	56
5.3.9 Mahalanobis Distance Anomaly Detection Model.....	57
5.3.10 Gaussian Mixture Model Anomaly Detection Model.....	58
5.3.11 Plain Autoencoder Anomaly Detection Model.....	59
5.3.12 Variational Autoencoder Anomaly Detection Model.....	60
5.3.13 LSTM Autoencoder Anomaly Detection Model.....	61
5.3.14 Transformer Autoencoder Anomaly Detection Model.....	62
5.3.15 TCN Autoencoder Anomaly Detection Model.....	64
5.3.16 Deep SVDD Anomaly Detection Model.....	65
5.4 Ensemble Anomaly Detection Methods.....	66
5.4.1 Soft Union Ensemble Strategy.....	67
5.4.2 Weighted Ensemble Strategy.....	69
5.4.3 LightGBM Meta-Classifer.....	70
5.4.4 Business Logic Post-Processing.....	72
5.5 Reconciliation Model.....	75
5.6 Interpretability Model.....	77
5.7 Chapter Summary.....	79
6 Business Use.....	81
6.1 Managerial Summaries and Operational Insights.....	81
6.2 Deployment.....	83
7 Evaluation.....	84
7.1 Assessment of Data Mining Results.....	84
7.1.1 Results in Terms of Accuracy.....	84
7.1.2 Results in Terms of Scalability.....	85
7.1.3 Results in Terms of Interpretability.....	86
7.2 Approved Models.....	87
7.3 Review of the Process.....	87
Conclusion.....	89
Answering Research Questions and Objectives.....	89
Limitations.....	91
Future Work.....	92
References.....	93

List of Figures

Figure 3.1 Distribution of Weekly Sales.....	31
Figure 3.2 Weekly National Sales Over Time.....	31
Figure 3.3 Year-over-Year Sales Patterns by Week Number.....	32
Figure 3.4 Trend in Weekly National Sales.....	32
Figure 3.5 Correlation Heatmap of Numeric Features.....	33
Figure 3.6 Weekly Sales Distribution on Holiday vs. Non-Holiday.....	34
Figure 3.7 Weekly Sales Distribution With and Without Promotions.....	34
Figure 4.1 Store 45 – Dept 29: Anomaly Injection Check.....	38
Figure 5.1 Valid Residual Weeks per Method (Filtered).....	40
Figure 5.2 Store 45 – Dept 29: Prophet Decomposition.....	43
Figure 5.3 Mean Regressor Contribution to Prophet Forecast.....	44
Figure 5.4 Store 45 – Dept 29: STL Decomposition.....	45
Figure 5.5 Store 45 – Dept 29: TimeGPT Decomposition.....	47
Figure 5.6 True Positives Overlap Heatmap.....	68
Figure 5.7 Residual Distribution: FP vs FN after Business Post-Processing.....	73
Figure 5.8 Store 45 – Dept 29: Soft Union Business Logic Post-Processing.....	74
Figure 5.9 Store 45 – Dept 29.....	76
Figure 5.10 Store Level: Store 45.....	76
Figure 5.11 National Level.....	77
Figure 6.1 Managerial Summary: Store 45, Week 2011-12-02.....	82
Figure 7.1 Available Machines in Deepnote.....	86

List of Tables

Table 2.1 Specific Autoencoder Variants.....	25
Table 2.2 Anomaly Detection Methods Comparison.....	28
Table 3.1 Descriptive Statistics of Weekly Sales Values.....	30
Table 3.2 Summary of Missing Values.....	35
Table 5.1 Best Runs per Anomaly Detection Method and Decomposition Combination.....	41
Table 5.2 Z-Scores Anomaly Detection Model Best Results.....	49
Table 5.3 Thresholding Anomaly Detection Model Best Results.....	50
Table 5.4 Isolation Forest Anomaly Detection Model Best Results.....	51
Table 5.5 K-Nearest Neighbors Anomaly Detection Model Best Results.....	52
Table 5.6 Local Outlier Factor Anomaly Detection Model Best Results.....	53
Table 5.7 HDBSCAN Anomaly Detection Model Best Results.....	54
Table 5.8 OCSVM Anomaly Detection Model Best Results.....	55
Table 5.9 BCPD Anomaly Detection Model Best Results.....	56
Table 5.10 Mahalanobis Distance Anomaly Detection Model Best Results.....	57
Table 5.11 GMM Anomaly Detection Model Best Results.....	58
Table 5.12 Plain Autoencoder Anomaly Detection Model Best Results.....	60
Table 5.13 Variational Autoencoder Anomaly Detection Model Best Results.....	61
Table 5.14 Transformer Autoencoder Anomaly Detection Model Best Results.....	63
Table 5.15 TCN Autoencoder Anomaly Detection Model Best Results.....	64
Table 5.16 Deep SVDD Anomaly Detection Model Best Results.....	66
Table 5.17 Best Performing Anomaly Detection Models.....	66
Table 5.18 Soft Union Ensemble Runs Performance.....	67
Table 5.19 Unique True Positives per Model.....	69
Table 5.20 Weighted Vote Ensemble Runs Performance.....	70
Table 5.21 Meta-Classifer Runs Performance.....	71
Table 5.22 Business Logic Post-Processing Runs Performance.....	73
Table 5.23 Reconciliation Runs Results.....	75
Table 5.24 Aggregated Department Level SHAP-like Contributions by Feature.....	78
Table 7.1 Pipeline Evaluation Across Varying Anomaly Injection Configurations.....	85

List of Abbreviations

AE	Autoencoder
ARIMA	AutoRegressive Integrated Moving Average
BCPD	Bayesian Change Point Detection
COPOD	Copula-Based Outlier Detection
CPI	Consumer Price Index
CRISP-DM	Cross-Industry Process for Data Mining
CSV	Comma-separated Values File
DBSCAN	Density-based Spatial Clustering of Applications with Noise
ECDF	Empirical Cumulative Distribution Functions
F1	Score, harmonic mean of precision and recall used to evaluate classification
FN	False Negatives
FP	False Negatives
GMM	Gaussian Mixture Model
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
IF	Isolation Forest
IQR	Interquartile Range
KL	Kullback–Leibler divergence
KNN	K-Nearest Neighbors
LOESS	Locally Estimated Scatterplot Smoothing
LOF	Local Outlier Factor
LRD	Local Reachability Density
LSTM	Long Short-Term Memory
MCD	Minimum Covariance Determinant
MSE	Mean Squared Error
OCSVM	One-Class Support Vector Machine
ReLU	Rectified Linear Unit
SHAP	Shapley Additive Explanations
STL	Seasonal-Trend Decomposition using LOESS
STD	Standard Deviation
SVDD	Support Vector Data Description
TAE	Transformer-based Autoencoder
TCN	Temporal Convolutional Network
TN	True Negatives
TP	True Positives
VAE	Variational Autoencoder
ZIP	Archive file format

Introduction

In the Introduction chapter, motivation for this work, research questions and objectives, and used methodology are described.

Motivation

In today's data-driven world, the ability to detect anomalies in time series data is crucial for numerous applications, including finance, healthcare, industrial monitoring, and retail. Anomalous data points can indicate events such as fraud, system failures, or shifts in consumer behavior. Timely detection of such outliers allows organizations to make informed decisions, mitigate risks, and optimize operational strategies.

A particularly challenging domain of anomaly detection lies in hierarchical and multivariate time series data. These datasets with complex dependencies across levels require specialized methods to identify meaningful deviations from forecast. An example is the retail sector, where sales data is structured hierarchically by regions, stores, and product categories.

This thesis focuses on the anomaly detection using the Walmart Store Sales Forecasting dataset from Kaggle as a case study.

Addressing the gap in methods for interpretable point-wise anomaly detection in large-scale hierarchical multivariate time series is essential for advancing anomaly detection models and methodologies and improving real-world decision-making processes.

Research Questions and Objectives

The goal of this research is to explore, implement, and evaluate anomaly detection methods for hierarchical multivariate time series point-wise anomaly detection. The key research questions guiding this study are:

1. What are the most suitable anomaly detection methods for large-scale hierarchical multivariate time series?
2. How do different approaches compare in terms of accuracy, scalability and interpretability?
3. Can an effective anomaly detection framework be developed to support real-world applications in hierarchical time series analysis?

To address these questions, the study is structured around the following objectives:

- Review methods for point-wise anomaly detection in multivariate and hierarchical time series, assessing their theoretical foundations and practical applicability.

- Implement and benchmark selected methods on the Walmart Store Sales Forecasting dataset.
- Compare the results of different approaches to determine their strengths, weaknesses, and suitability for real-world scenarios.
- Propose recommendations and insights based on the findings to contribute to the development of scalable and interpretable anomaly detection techniques for hierarchical time series data.

Methodology

In this thesis, the robust and well-proven methodology Cross-Industry Process for Data Mining (CRISP-DM) is followed (Smart Vision Europe, 2025).

Work begins with Business Understanding. This chapter describes the business context and the real-world problems addressed by this thesis.

Next is Machine Learning Understanding, where suitable models are introduced and their applicability in interpretable point-wise anomaly detection in large-scale hierarchical multivariate time series assessed.

In Data Understanding, it is demonstrated that the available data is appropriate for solving the problem as defined in the Business and Machine Learning Understanding chapters.

Data Preparation summarizes the initial technical steps taken to obtain and preprocess the dataset for modeling.

The Modelling chapter is one of the most important sections of this thesis. It details the process of training, validation, model configuration, and method selection. All decisions are supported by results, and alternatives are compared directly. The best-performing method is identified.

Business Use chapter outlines how the best-performing pipeline could be deployed within the company and used by decision-makers.

In Evaluation, the results of the modeling phase are interpreted based on the criteria established in the Introduction chapter.

Finally, in the Conclusion, the research questions from the Introduction are answered, the main insights are summarized, limitations discussed, and directions for future work are proposed.

1 Business Understanding

Walmart is currently the world's largest retailer, with 2023 retail revenue of \$635 billion and 10,569 stores across 19 countries. In the United States alone, Walmart generated \$534 billion in domestic retail sales across 5,321 locations, maintaining its position as the country's leading retailer. It is followed by Amazon, Costco, Home Depot, Kroger, Walgreens, CVS, Target, and others (Capital One, 2025; The Produce News, 2024).

In 2012, the time period relevant to this work, Walmart recorded approximately \$444 billion in sales. If Walmart had been a country at that time, it would have ranked as the 26th largest economy in the world (Spector, 2012). Since 1988, Walmart (then Wal-Mart) has been the most profitable retailer in the United States (Hayes & Times, 1990).

Remaining at the top for such a long period, while continuing to grow, requires strong strategy, operational excellence, and outstanding data analytics, which is essential for grounding strategic and operational decision-making in real insights and facts. Forecasting future performance based on historical trends is one critical part of that process. The other, and the focus of this thesis, is identifying and understanding why outcomes differ from forecasts. In the data science world, this is known as anomaly detection.

In the retail context, many types of anomalies can occur in the sales data. For the purposes of this work, two primary categories are defined and analyzed:

- Unexpected sales drops — These events occur when actual sales fall below forecasted values during periods of markdowns and/or holidays, which would typically boost sales. Such drops are particularly critical because they may indicate direct revenue loss. The risk is even greater when these anomalies go undetected across the organizational hierarchy, as no one may be connecting the dots. Possible causes include stockouts, operational failures, or ineffective promotions.
- Unexpected sales spikes — These events happen when actual sales exceed forecasts despite the absence of markdowns or holidays. While not as urgent as drops, they are important to investigate. Understanding the root cause can lead to valuable business insights and opportunities for growth.

Other, more specific anomaly types — such as flatlining sales, inconsistent promotion effects, or shifts in price sensitivity — may be business-relevant but are outside the scope of this thesis.

In practice, forecast values and detected anomalies are typically reported to store managers. For these reports to be useful, context is essential. Managers reviewing weekly reports need to understand their store's performance, baseline, and also comparison to other stores and national level — not just detecting individual outliers, but identifying hierarchically consistent patterns and providing clear explanations of what happened and why.

To achieve that, the hierarchical structure of the data must be leveraged. Detecting anomalies only at the lowest level (e.g., a department in a store) risks missing larger patterns that emerge across higher levels of aggregation. In cases of severe, undetected drops, this could lead to significant financial losses. Additionally, it is important that anomalies are communicated in understandable, business-relevant terms — not overly technical or statistical language.

Finally, it is essential not to overwhelm store managers with many flags that turn out to be a false alarm. Even statistically valid anomalies must be filtered through a business lens: if they do not imply a meaningful financial impact, they should not be surfaced. Toward the end of this thesis, the focus shifts to delivering concise, reliable, and actionable outputs, enriched with sufficient business context to support informed decision-making.

2 Machine Learning Understanding

This chapter defines the anomaly detection problem addressed in this thesis and presents the theoretical foundations of the chosen approach. It outlines the challenges of point-wise detection in hierarchical, multivariate, large-scale time series. The chapter also explains the rationale behind the selected decomposition methods, detection algorithms, meta-classifier, and interpretability tools.

2.1 Problem Definition

Problem Definition chapter translates the business goal, detecting unexpected sales events across departments and stores, into a machine learning problem: point anomaly detection of spikes and drops relative to forecasted values in large-scale hierarchical, multivariate time series.

2.1.1 Point-Wise Anomaly Detection

Anomaly detection refers to identifying data points that deviate significantly from expected behavior or patterns (Bajaj, 2021). It distinguishes between inliers, which follow normal patterns, and outliers, which differ so substantially from the rest of the data that they may have been generated by a different process (Bajaj, 2021; Blázquez-García et al., 2021; Tuychiyev & DataCamp, 2021). Statistically, an outlier is a data point with significantly abnormal features. However, the final decision, whether a data point is or is not an outlier, often depends on human judgment (Tuychiyev & DataCamp, 2021). In time series contexts, outliers are typically referred to as anomalies (Bajaj, 2021).

There are several types of anomalies:

- Point-wise anomalies (or point outliers) are individual values that behave unusually either compared to the entire series (global) or to their immediate context (local) (Bajaj, 2021; Blázquez-García et al., 2021; Lai et al., 2021). These are often the focus of anomaly detection research due to their potential impact.
- Contextual anomalies are values that are only anomalous within a specific context (e.g., time of day, season), but not when viewed globally (Lai et al., 2021).
- Collective anomalies involve groups of values that are anomalous as a sequence, even if individual values are not. These are common in time series due to temporal dependencies (Lai et al., 2021).
- Subsequence anomalies (or pattern-wise outliers), a subclass of collective anomalies, are consecutive data points whose joint behavior is unusual. They can result from unusual shapes, seasonal disruptions, or trend changes (Bajaj, 2021; Blázquez-García et al., 2021; Lai et al., 2021).

- Time series–level anomalies refer to entire sequences that behave abnormally in comparison to other sequences (Blázquez-García et al., 2021).

2.1.2 Large-Scale Hierarchical Multivariate Time Series

Time series are a fundamental data structure across domains. Their defining characteristic is the temporal dependency between observations, known as autocorrelation, which makes traditional modeling techniques insufficient (Bajaj, 2021; Blázquez-García et al., 2021; Tychiyev & DataCamp, 2021). This sequential nature requires specialized approaches for effective anomaly detection (Rajan, 2021).

In many real-world scenarios, time series are multivariate, with each timestamped observation accompanied by additional features such as calendar effects or external variables. This structure introduces complex temporal and cross-variable dependencies, especially in domains where contextual factors heavily influence behavior (Sun et al., 2024). As a result, multivariate time series anomaly detection has emerged as a fast-growing research field, with modern deep learning models achieving state-of-the-art results on benchmark datasets (Challu et al., 2022).

In addition to temporal and multivariate complexity, many systems produce time series with an inherent hierarchical structure. In retail, for example, data may be nested by department, store, region, and national level, with higher-level series representing aggregated values from lower levels (Hyndman & Athanasopoulos, 2021). In such settings, maintaining coherence — ensuring that forecasts or anomalies at aggregate levels align with the sum of their parts — is essential for interpretability and decision-making consistency (Mancuso et al., 2021).

In hierarchical time series, coherence means ensuring that forecasts or anomaly signals remain consistent across all levels of aggregation. Several reconciliation strategies address this. Top-down approaches begin at the highest level and allocate forecasts downward using historical ratios or model-based weights. While efficient, they risk masking important low-level variations (Hyndman & Athanasopoulos, 2021). Middle-out methods forecast from an intermediate level and reconcile both upward and downward, but they rely on the existence of a meaningful middle tier, which is not always available (Hyndman & Athanasopoulos, 2021). In contrast, the bottom-up approach models the most granular time series and aggregates upward to produce coherent higher-level signals. This preserves maximum detail and is especially suited to point-wise anomaly detection, where subtle, localized deviations are most relevant. Anomalies are identified directly in the raw, disaggregated data, then propagated upward to inform store-wide or national insights. While modeling bottom-level series can be noisier and more complex, the added precision and interpretability make this tradeoff worthwhile in large-scale hierarchical systems (Hyndman & Athanasopoulos, 2021).

A major challenge in anomaly detection is the lack of labeled ground truth. Historical anomalies are rarely annotated, and new anomalies are inherently unpredictable, making supervised learning difficult to apply. As a result, most models operate in unsupervised or semi-supervised settings, learning normal patterns and flagging deviations without labeled anomalies (Chen et al., 2023; Darban et al., 2024).

2.2 Choice of Decomposition Methods

This chapter discusses suitable decomposition methods to separate signal, the forecast (trend and seasonal components), from noise (residuals) and briefly describes their mechanisms, advantages, and limitations in the context of this use case.

2.2.1 Prophet Forecasting Mechanism

Prophet by Meta is a modular, additive forecasting model designed for business time series with strong seasonality and historical depth (Facebook, 2019; Taylor & Letham, 2017). It decomposes the signal into trend, seasonality, and holiday effects, plus an error term. Trends can follow piecewise linear or logistic growth, seasonality is modeled via Fourier series, and holidays are handled using indicator variables with Gaussian priors. The model is robust to missing data, outliers, and structural changes, and is fit using maximum a posteriori estimation (Taylor & Letham, 2017).

Prophet is well-suited for detecting anomalies in sales time series due to its ability to explicitly model key retail patterns through separate trend, seasonality, and holiday components. It handles multiple seasonalities, allows inclusion of domain-specific events via custom regressors, and is robust to missing or irregular data. This makes it both flexible, easy to use, and practical for large-scale retail forecasting tasks (Melanie, 2024).

Despite its flexibility, Prophet has limitations when applied to hierarchical, multivariate retail data. It models each time series independently, requiring post-hoc reconciliation to maintain consistency across aggregation levels. It does not support multivariate modeling beyond external regressors and lacks mechanisms to capture temporal autocorrelation or residual structure. Additionally, it may overfit when changepoint flexibility is high or underperform on series with abrupt trend shifts (Melanie, 2024).

2.2.2 Seasonal-Trend Decomposition using LOESS Forecasting Mechanism

STL, Seasonal-Trend Decomposition using LOESS (Locally Estimated Scatterplot Smoothing), is a classical, non-parametric method for decomposing time series into trend, seasonal, and residual components, making it a strong baseline for anomaly detection. It uses LOESS smoothing to estimate the trend and seasonal patterns without assuming a global model form. The residual component captures irregularities and potential anomalies. While STL does not perform forecasting itself, it is often paired with models like ARIMA (AutoRegressive Integrated Moving Average) for forward prediction using the cleaned components (Cleveland et al., 1990; Statsmodels, 2025).

The non-parametric decomposition of trend and seasonality could be an advantage, which adapts to evolving patterns across departments and stores. Its LOESS smoothing makes it robust to outliers (Cleveland et al., 1990; Statsmodels, 2025).

Despite its flexibility, STL has several limitations for anomaly detection in hierarchical, multivariate sales data. It is strictly univariate, requiring separate decomposition per series with no cross-series information sharing or built-in reconciliation. It lacks support for external regressors, so key drivers like holidays or markdowns are not modeled explicitly. It does not handle missing data out of the box and expects evenly spaced data — requiring precise preprocessing (Cleveland et al., 1990; Statsmodels, 2025).

2.2.3 TimeGPT Forecasting Mechanism

TimeGPT by Nixtla is a transformer-based forecasting model trained on millions of global time series to learn generalizable temporal patterns. It generates forecasts using pretrained weights, with no local training required, and supports anomaly detection via residuals between actual and predicted values. Anomalies are flagged using the `detect_anomalies()` method, which returns forecast. TimeGPT-based detection is non-parametric and does not involve local training or changepoint modeling. Instead, it relies on the generalization capabilities of the foundation model, which has been trained to implicitly capture seasonal, trend, and event-based patterns through its transformer architecture (Nixtla, 2025).

TimeGPT is a pretrained, transformer-based forecasting model that enables fast, scalable prediction across thousands of time series without local training or manual configuration. Its standardized input format and lack of seasonality or holiday tuning make it ideal for large, heterogeneous datasets. By capturing both short- and long-term temporal patterns, it produces residuals suitable for anomaly detection across complex hierarchies (Nixtla, 2025).

Despite its scalability, TimeGPT has several limitations for anomaly detection in structured retail settings. It functions as a black box, offering no interpretable components like changepoints or seasonal effects, which limits explainability. The model's anomaly scoring relies on fixed internal thresholds that cannot be tuned, potentially leading to over- or under-detection. It does not support hierarchical reconciliation, requiring external aggregation logic for consistency across store and national levels. TimeGPT also lacks true multivariate modeling. Additionally, it is sensitive to missing or irregular timestamps, requiring careful preprocessing. In some structured domains, traditional statistical models have outperformed TimeGPT in forecasting accuracy, highlighting that pretrained deep models are not universally superior (Lee, 2024, Netsch, 2024; Nixtla, 2025).

2.3 Choice of Anomaly Detection Methods

This chapter presents the selected anomaly detection methods, briefly describing their mechanisms, advantages, and limitations in the context of this use case.

2.3.1 Z-Scores Anomaly Detection Mechanism

The Z-Score method is a statistical technique that detects anomalies by measuring how far a data point deviates from the average, relative to the overall variability in the dataset. In time series, this involves computing the distance of each point from the mean in terms of standard

deviations (STD) and flagging those that exceed a predefined threshold. It assumes the data follows a normal distribution and is particularly effective for univariate time series where anomalies appear as large, isolated deviations from the typical pattern (Kumar, 2023; Peixeiro, 2023; RisingWave, 2024). Data points with Z-Scores beyond a certain threshold (commonly ± 3 or 4) are considered anomalies (Peixeiro, 2023; Srivastava, 2023).

The Z-Score method is simple to implement and interpret, making it accessible even to non-experts and easily understood by business stakeholders. Its high computational efficiency allows it to scale to large datasets and enables real-time anomaly detection in high-volume environments. As a lightweight baseline, it offers a fast way to assess data before applying more complex models (MindBridge, 2025; Moffitt, 2024; RisingWave, 2024; Romeu, 2021).

The Z-Score method assumes a normal distribution, but this often fails in real-world business data, reducing detection accuracy. Outliers can distort the mean and standard deviation, further undermining reliability. The standard approach is univariate and does not naturally extend to multivariate series without additional methods. Moreover, it does not account for temporal dependencies or seasonality, making it prone to false positives or negatives unless additional safeguards are applied (Kumar, 2023; MindBridge, 2025; Peixeiro, 2023; RisingWave, 2024).

2.3.2 Thresholding Anomaly Detection Mechanism

Thresholding methods detect anomalies by comparing values against predefined or adaptive cutoffs, using simple and interpretable rules. Approaches include absolute thresholds (fixed constants), interquartile range (IQR) filtering (e.g., $1.5 \times \text{IQR}$), percentile-based cutoffs (e.g., 10th/90th percentiles), and adaptive thresholds that adjust to evolving data distributions. These techniques are computationally efficient. (Eslava, 2023; Zwingmann, 2022;)

Thresholding methods, especially adaptive ones, scale efficiently across thousands of time series by avoiding model training and requiring minimal computation (Ebenezer & Sharma, 2023). Techniques like percentile cutoffs and IQR rules are intuitive and easily explainable to stakeholders (Eslava, 2023). Thresholds can be applied globally or per group (e.g., by department or store) without complex tuning, and they require no retraining, fitting, or specialized infrastructure.

Thresholding treats each time series independently. It typically operates on single variables (e.g., residuals) and does not capture inter-feature dependencies. Static thresholds can perform poorly in non-stationary or seasonal data unless dynamically adapted (Ebenezer & Sharma, 2023) or supported by a business reasoning. Choosing an inappropriate cutoff (e.g., 90th vs. 99.9th percentile) can result in excessive false positives or missed anomalies (Zwingmann, 2022).

2.3.3 Isolation Forest Anomaly Detection Mechanism

Isolation Forest is a tree-based anomaly detection algorithm that isolates anomalies rather than modeling normal behavior. It assumes anomalies are “few and different,” making them

easier to separate through recursive random partitioning. The method builds multiple isolation trees by randomly selecting features and split values, with anomaly scores based on how quickly a point is isolated—shorter paths indicating greater anomaly likelihood (Liu et al., 2008; Yoon, 2022).

Isolation Forest is well-suited for detecting anomalies in high-dimensional, large-scale retail data due to its linear time complexity, low memory footprint, and strong performance without assuming any specific data distribution (Lu et al., 2023; Yoon, 2022). It scales efficiently across thousands of series and performs robustly even when features are noisy or sparse (Agyemang, 2024; Carletti et al., 2020). Its ability to isolate outliers without needing feature selection or prior distributional knowledge makes it ideal for retail pipelines with heterogeneous multivariate inputs (Lu et al., 2023).

Despite its efficiency, Isolation Forest suffers from poor interpretability due to its use of random, axis-aligned splits (Carletti et al., 2020; Konefal, 2023). The model requires manual threshold tuning, and performance can degrade when anomalies are not sharply distinct from normal data (Agyemang, 2024; Konefal, 2023). It also lacks native mechanisms for hierarchical aggregation or temporal awareness, limiting its ability to reconcile anomaly patterns across store or national levels (Liu et al., 2024). Isolation Forest can overfit when too many trees are used, leading to reduced accuracy and poor generalization (Yadav, 2023).

2.3.4 K-Nearest Neighbors Anomaly Detection Mechanism

K-Nearest Neighbors (KNN) is a non-parametric, distance-based method that identifies anomalies by measuring how far each data point is from its k nearest neighbors in the feature space. Points that lie far from others are flagged as outliers. The algorithm memorizes the entire dataset and computes distances during prediction, making it unsupervised and highly interpretable (Bajaj, 2023; Schmidl et al., 2022; Tuychiyeve & DataCamp, 2021). Anomaly scores are based on the average or minimum distance to neighboring points. The choice of distance metric (e.g., Euclidean, Manhattan, or Minkowski) and the `n_neighbors` parameter significantly affects sensitivity, especially in high-dimensional or scaled data (Schmidl et al., 2022; Tuychiyeve & DataCamp, 2021).

The method is intuitive, easy to interpret, and highly accessible for anomaly detection (Bajaj, 2023; Schmidl et al., 2022). It requires only one core parameter — the number of neighbors — and makes no statistical assumptions about the data distribution, which increases its applicability across varied datasets (Tuychiyeve & DataCamp, 2021). Because KNN does not require model training, it is computationally efficient during fitting and scales well in batch detection scenarios (Tuychiyeve & DataCamp, 2021). Its simplicity makes results easier to explain to non-technical stakeholders. KNN can also be adapted to various distance metrics to better accommodate high-dimensional or categorical feature spaces (Tuychiyeve & DataCamp, 2021).

Despite its simplicity, KNN suffers from several limitations in anomaly detection. Its computational complexity makes it inefficient for large multivariate datasets (Bajaj, 2023). The method is memory-inefficient, as it must retain the entire dataset and compute distances at prediction time, which can be slow in practice (Tuychiyeve & DataCamp, 2021). It is also

sensitive to feature scaling and may perform poorly if features have disproportionate magnitudes or relevance — a problem partially mitigated through scaling techniques (Tuychiyev & DataCamp, 2021). Additionally, in high-dimensional spaces, KNN suffers from the curse of dimensionality, where distances become less meaningful and anomalies harder to distinguish (Bajaj, 2023). Finally, tuning the number of neighbors is non-trivial when the contamination ratio is unknown, and Euclidean distance performs poorly in more than two or three dimensions, requiring careful metric selection (Tuychiyev & DataCamp, 2021).

2.3.5 Local Outlier Factor Anomaly Detection Mechanism

Local Outlier Factor (LOF) is a density-based anomaly detection algorithm that identifies outliers by comparing the local density of a point to the densities of its nearest neighbors (Eslava, 2023; Tuychiyev & DataCamp, 2021). Each point's local density is measured using the local reachability density (LRD), and its LOF score is computed as the ratio of the average LRD of its neighbors to its own (Shabou, n.d.). Points that lie in areas of significantly lower density than their neighbors are flagged as outliers. This local focus allows LOF to detect both global and subtle local anomalies, especially in datasets with varying density (Eslava, 2023; Eyer, 2024).

LOF is particularly effective in detecting anomalies in datasets with varying local densities, as it compares each point's density only to that of its neighbors rather than the entire dataset (Eslava, 2023; Shabou, n.d.). This local sensitivity allows it to identify both global and nuanced local outliers, making it well-suited for complex, heterogeneous data (Tuychiyev & DataCamp, 2021). It performs well in moderately high-dimensional settings and does not assume any specific data distribution (Eyer, 2024). LOF is also advantageous in finding clusters of anomalous points and capturing subtle patterns that global methods might miss (Eyer, 2024).

Because LOF relies on comparing a point's density to its neighbors, LOF may miss outliers that do not strongly deviate from their immediate context, even if they differ from the global pattern (Eslava, 2023; Shabou, n.d., Tuychiyev & DataCamp, 2021). It is also sensitive to the choice of the `n_neighbors` parameter, which can impact its ability to detect anomalies in regions with varying density (Tuychiyev & DataCamp, 2021). Additionally, LOF can be computationally intensive and prone to overfitting small local variations in complex datasets (Eyer, 2024).

2.3.6 HDBSCAN Anomaly Detection Mechanism

Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) is a hierarchical, density-based clustering algorithm that identifies clusters of varying densities and classifies low-density points as noise (Blachowicz et al., 2025). Unlike DBSCAN (Density-Based Spatial Clustering of Applications with Noise), it does not require a fixed number of clusters and automatically adapts to data structure by pruning unstable clusters from a hierarchy (Iuhasz et al., 2025). It is especially useful in time series anomaly detection for spotting those missed by traditional methods (Ruberts, 2020).

HDBSCAN offers key advantages for anomaly detection in complex time series. It automatically determines the number of clusters, making it more adaptive than k-means or DBSCAN (Blachowicz et al., 2025, Iuhasz et al., 2025). Its ability to model clusters of varying density and exclude noise points improves detection of true anomalies while avoiding overfitting (Blachowicz et al., 2025). HDBSCAN has been shown to detect both spikes and drops, including cases missed by classical models (Ruberts, 2020), and its outputs, such as real-time noise volume and cluster counts, support operational monitoring and visual interpretation (Blachowicz et al., 2025).

Despite its strengths, HDBSCAN has several limitations. It is a transductive method, meaning new data cannot be evaluated without retraining, which leads to long inference times and scalability issues (Iuhasz et al., 2025). It is primarily designed for clustering, not anomaly detection, and may struggle to identify subtle anomalies—especially when trained on short time spans (Iuhasz et al., 2025). Its effectiveness is also sensitive to parameter tuning, such as the minimum cluster size or quantile threshold, which requires careful adjustment to avoid mislabeling (Ruberts, 2020).

2.3.7 One Class SVM Anomaly Detection Mechanism

The One-Class Support Vector Machine (OCSVM) is a one-class unsupervised anomaly detection method that learns a decision boundary around the normal data in a transformed feature space. Using kernel functions OCSVM maps inputs into a higher-dimensional space where it separates the majority (normal) class from potential anomalies (Dey, 2024). Unlike traditional binary classifiers, it trains solely on normal data and flags deviations from the learned boundary as outliers (Kumar, 2023). This makes it especially useful in domains where anomalies are rare or unlabeled.

OCSVM offers strong advantages in settings where only normal data is available, making it well-suited for unsupervised anomaly detection in highly imbalanced datasets (Dey, 2024). The method supports various kernel functions, allowing flexibility across different data structures (Dey, 2024). Its decision boundaries may be interpretable using gradient-based methods, which adds transparency to its otherwise opaque kernel mappings (Nguyen and Vien, 2018).

Despite its strengths, One-Class SVM struggles with scalability on large or high-dimensional data due to its computationally intensive optimization process (Dey, 2024; Nguyen & Vien, 2018). It is also highly sensitive to kernel and hyperparameter choices, requiring careful tuning for good performance (Dey, 2024). Moreover, OCSVM does not natively capture temporal or hierarchical structures, limiting its flexibility in complex multivariate time series.

2.3.8 COPOD Anomaly Detection Mechanism

COPOD (Copula-Based Outlier Detection) is a parameter-free, unsupervised anomaly detection algorithm that estimates how extreme a data point is within the multivariate distribution. It works in three stages: first, it computes empirical cumulative distribution functions (ECDFs) for each feature; second, it constructs an empirical copula to capture joint dependencies across variables; and third, it estimates tail probabilities for each point to

quantify its level of outlyingness. Higher tail scores indicate more likely anomalies. COPOD does not require training, scales efficiently to high-dimensional data, and produces interpretable results through per-feature anomaly contributions (Feasel, 2022; Li et al., 2020).

COPOD offers strong advantages for multivariate anomaly detection due to its parameter-free design, computational efficiency, and interpretability. It estimates tail probabilities using empirical copulas, avoiding the need for training or hyperparameter tuning (Li et al., 2020). By modeling the joint distribution through ECDFs, it scales well to high-dimensional datasets and large sample sizes with minimal overhead (Feasel, 2022). Its deterministic scoring and ability to attribute anomaly contributions to individual dimensions make it especially useful for explainable outlier detection across diverse domains (Li et al., 2020).

Despite its efficiency, COPOD has notable limitations. It assumes feature independence when computing ECDFs, which may not hold in complex real-world data (Feasel, 2022). This can reduce detection accuracy when strong feature interactions are present. COPOD also breaks temporal order, as it evaluates all points jointly rather than respecting sequence, making it unsuitable for time series anomaly detection without major adaptation. Its reliance on static distributions may lead to poor performance in dynamic or evolving data (Feasel, 2022; Li et al., 2020).

Although COPOD's copula-based algorithm initially appeared promising, it was excluded from the modelling because no implementation of sliding-window or train/test-split configurations can enforce the chronological ordering needed to maintain temporal integrity, leading to irrecoverable distortions at sequence boundaries.

2.3.9 Bayesian Change Point Detection Anomaly Detection Mechanism

Bayesian Change Point Detection (BCPD), implemented via the Binseg algorithm with the L2 cost function, is a fast and interpretable method for detecting structural breaks in time series (Perry, 2019; ruptures, n.d.). Binseg recursively partitions the series by identifying change points that mark significant shifts in the mean under the assumption of normally distributed segments (Rajasekaran, 2025).

The method offers fast, interpretable segmentation (Rajasekaran, 2025; ruptures, n.d.). Its computational efficiency makes it suitable for large datasets like retail sales (ruptures, 2017), and each detected change point corresponds to a clear structural shift, aiding business interpretability (Perry, 2019).

Despite its efficiency, BCPD with Binseg has key limitations in this use case. The algorithm detects multiple change points in a time series, rather than point-wise anomalies (ruptures, n.d.). It treats each time series independently, lacking the ability to capture dependencies across hierarchical or multivariate structures (Perry, 2019). It also assumes changes occur as mean shifts under normality (Rajasekaran, 2025), which may not capture more complex retail patterns like variance changes or seasonal distortions.

2.3.10 Mahalanobis Distance Anomaly Detection Mechanism

This method detects anomalies in multivariate data by computing the Mahalanobis distance between each point and the dataset's multivariate centroid, adjusting for correlations between features. Robust estimation is achieved via covariance, which fits a Gaussian model resistant to outliers. Points with large Mahalanobis distances, indicating they are far from the mean in a covariance-adjusted space, are flagged as outliers. This approach is best suited for unimodal, symmetric data and captures anomalies even in correlated or anisotropic feature spaces (Holbert, 2022; Kaya, 2020; Zhao, 2022).

Mahalanobis distance, particularly when paired with the Minimum Covariance Determinant (MCD), offers several advantages for multivariate anomaly detection. Unlike Euclidean distance, it accounts for correlations between variables and scales appropriately for skewed, non-isotropic data (Holbert, 2022; Kaya, 2020). This makes it effective in high-dimensional retail datasets where features often exhibit strong dependencies. Mahalanobis-based methods are mathematically principled, measuring how far a point deviates from the multivariate mean in standard deviation units, and can isolate subtle anomalies that occur in low-variance directions of the feature space (Kamoi & Kobayashi, 2020).

Despite its strengths, the Mahalanobis distance with MCD has notable limitations. It assumes that data follows a unimodal, symmetric Gaussian distribution, and may perform poorly on multimodal or irregular datasets (Zhao, 2022). The method is sensitive to high-dimensional geometry and relies on meaningful covariance estimates, which can degrade in small sample sizes or highly noisy settings (Kamoi & Kobayashi, 2020). Its effectiveness also depends on the stability of low-variance directions in the data, making it less robust when feature variance is unstable or when distributional assumptions are violated (Holbert, 2022).

2.3.11 Gaussian Mixture Model Anomaly Detection Mechanism

Gaussian Mixture Model (GMM) is a probabilistic model that assumes data points are generated from a mixture of multiple Gaussian distributions, each representing a latent subpopulation. GMM allows overlapping, elliptical clusters and estimates the probability of each point belonging to each component (Edge Impulse, 2024; Sayago, 2024). Anomalies are identified as points with low likelihood under the learned mixture distribution (Apgar, 2023; Edge Impulse, 2024). The model uses the Expectation-Maximization algorithm to iteratively refine component means, variances, and weights, guided by Bayes' Theorem. This flexible framework enables unsupervised classification and density-based anomaly scoring across multimodal data. (Apgar, 2023; Edge Impulse, 2024; Sayago, 2024)

GMM is well-suited for detecting both isolated outliers and anomalous subgroups in complex retail sales data, due to its ability to model multimodal and elliptical distributions (Apgar, 2023; Edge Impulse, 2024). Unlike hard clustering methods like K-means, GMM assigns soft probabilistic cluster memberships, allowing flexible detection in noisy and overlapping series (Sayago, 2024). It handles variable cluster shapes and captures hidden structure without requiring labeled data, making it effective in unsupervised, high-variance retail environments (Apgar, 2023).

GMM assumes that data is generated from a mixture of Gaussian distributions, which may not hold in real-world retail datasets with irregular, seasonal, or hierarchical patterns (Edge Impulse, 2024; Sayago, 2024). The model is sensitive to initialization and may converge to suboptimal solutions, especially when clusters are poorly separated or the number of components is misestimated (Sayago, 2024). GMM also treats observations as independent and identically distributed (Apgar, 2023).

2.3.12 Autoencoders Anomaly Detection Mechanism

Autoencoders (AEs) are neural networks trained to reconstruct their input by encoding data into a compressed latent space and decoding it back to its original form. This bottleneck architecture forces the model to learn the essential structure of normal data, and reconstruction error becomes a natural anomaly score—high error signals potential anomalies (Despois, 2017; Chollet, 2016; Singh, 2024; Schmidl et al., 2022). The approach is typically semi-supervised, trained only on normal data using sliding windows to preserve temporal context (Schmidl et al., 2022). Variants such as Variational Autoencoders (VAEs), Long Short-Term Memory (LSTM) Autoencoders, Transformer-based Autoencoders (TAEs), and Temporal Convolutional Autoencoders (TCNs) extend this principle to probabilistic, sequential, and convolutional architectures (Govindaraj, 2024; Neloy & Turgeon, 2024; Thill, 2020; Tuhin et al., 2025).

Autoencoders are highly effective for detecting point anomalies in high-dimensional, multivariate time series, especially when patterns are non-linear or hard to capture with classical models (Bajaj, 2023; Kumar, 2023). Their ability to extract meaningful latent representations allows them to model complex temporal and cross-feature dependencies—such as promotional cycles or economic indicators — especially when extended via LSTM or Transformer layers (Hong, 2024; Govindaraj, 2024; Tuhin et al., 2025). These models scale well across large hierarchical datasets and support flexible architectures suited for different input modalities (Neloy & Turgeon, 2024).

Despite their flexibility, autoencoders require clean and representative normal data, which may be difficult to guarantee in domains with structural noise or concept drift (Hong, 2024; Teuwens, 2021). Deep variants like LSTM or Transformer AEs are also computationally intensive, requiring careful tuning and significant resources (Tuhin et al., 2025; Lawton, 2024). Furthermore, interpretability remains limited due to opaque latent features, complicating real-world deployment and anomaly explanation (Tuhin et al., 2025).

Further in this thesis, autoencoder variants summarized in Table 2.1 *Specific Autoencoder Variants* below will be used.

Table 2.1*Specific Autoencoder Variants*

Type	Architecture	Temporal Modelling	Latent Space Type	Anomaly Detection Mechanism	Strengths	Weaknesses
Plain AE	Fully connected encoder and decoder with bottleneck layer	Not temporal; input often flattened	Deterministic fixed-size latent representation	High reconstruction error on unseen patterns	Simple, effective, non-linear encoding	Cannot capture temporal dependencies
VAE	Probabilistic encoder/decoder; learned latent space distribution	Assumes independent samples	Probabilistic; usually Gaussian	Outliers fall in low-probability regions; latent uncertainty as score	Captures variability, robust to noise	Complex loss function; unstable optimization
LSTM AE	Stacked LSTM layers in encoder and decoder	Explicitly models temporal dependencies	Deterministic temporal latent vector	High reconstruction error on abnormal time sequences	Good memory for time patterns; detects contextual or collective anomalies	High compute and memory cost; hard to interpret
TAE	Self-attention layers process sequences in parallel	Captures long-range dependencies via attention	Deterministic embeddings updated by context	High error signals anomalous deviation	Efficient for long sequences; highly parallel	Requires large datasets and compute power
TCN AE	Dilated convolutions for encoding and decoding sequences	Models temporal dependencies with dilation	Deterministic compressed sequence encoding	High reconstruction error at anomalies + distance	Fast training; large receptive field	Sensitive to dilation/kernel tuning

AE = Autoencoder, VAE = Variational Autoencoder, LSTM = Long Short-Term Memory, TAE = Transformer-based Autoencoder, TCN = Temporal Convolutional Network.
Note: Table created by the author based on multiple sources including: (Al-Marie, 2023; Al-Selwi et al., 2023; Asperti & Trentin, 2020; Bajaj, 2023; Brownlee, 2018; Cai et al., 2023; Chollet, 2016; Despois, 2017; Dhapre, 2024; GeeksForGeeks, 2019; GeeksForGeeks, 2020; Govindaraj, 2024; Helen, 2019; Hong, 2024; IBM, 2024; IBM, 2025; Intel, 2024; Ippolito, 2023; Kar, 2024; Kennedy, 2025; Lachehab et al., 2024; Lawton, 2024; Lozovsky, 2024; Neloy & Turgeon, 2024; Ögretir et al., 2023; Owoh et al., 2024; Pykes, 2024; Schmidl et al., 2022; Singh, 2024; Teuwens, 2021; Thill, 2020; Thill et al., 2021; Tuhin et al., 2025; Xu & Duraisamy, 2020; Yadav, 2024).

2.3.13 Deep SVDD Anomaly Detection Mechanism

Deep Support Vector Data Description (Deep SVDD) is a one-class anomaly detection method that trains a neural network to map normal data into a minimal-volume hypersphere in latent space (Ruff et al., 2018; Sendera et al., 2021; Yi, 2020). The center of this hypersphere is fixed, typically set from an initial forward pass, to prevent collapse (Pérez-Carrasco et al., 2023). At inference, the anomaly score is the distance from an input's representation to the center — larger distances suggest higher anomaly likelihood (Sendera et al., 2021).

Deep SVDD bypasses input reconstruction by learning compact, discriminative latent features from normal data alone (Ruff et al., 2018; Sendera et al., 2021). This makes it well-suited to unsupervised settings with limited anomaly labels (Pérez-Carrasco et al., 2023). The model is end-to-end trainable, architecture-agnostic, and effective even in high-dimensional or structured inputs like multivariate time series (Sendera et al., 2021; Yi, 2020). Its tight latent embeddings support robust anomaly separation across diverse tasks (Ruff et al., 2018).

A key issue is representation collapse, where the network maps all inputs, normal and anomalous, to the same latent point, undermining anomaly detection (Ruff et al., 2018;

Sendera et al., 2021). This is mitigated by removing bias terms and fixing the center, but it doesn't address the deeper problem of multi-modal normal data. When normal behavior spans diverse patterns, as in hierarchical or multivariate time series, the assumption of a single compact latent region breaks down, leading to poor generalization (Pérez-Carrasco et al., 2023; Ruff et al., 2018).

2.4 Ensemble Detection

Ensemble strategies have proven highly effective in anomaly detection by combining the strengths of diverse methods. Instead of relying on a single detector, recent studies emphasize combining outputs from univariate, multivariate, and deep models to boost robustness and recall (Furnari et al., 2021; Xin et al., 2023). These ensemble approaches perform particularly well in complex or noisy datasets, where individual models often miss complementary anomaly signals.

To systematically learn from such heterogeneity, a meta-learning approach is recognized. Following the stacked generalization paradigm, a second-level classifier is trained on the binary outputs of base detectors to synthesize their decisions into a single, refined prediction (Brownlee, 2020). This layer learns which models to trust under which conditions, increasing precision and reducing false positives without compromising sensitivity (Jeffrey et al., 2024; Milvus, 2025).

Weighted voting or simple unions can be used to create the ensemble, but also gradient boosting methods, like LightGBM can be used as the meta-classifier due to its scalability and strong empirical performance in prior stacking studies. Its leaf-wise tree growth and support for mixed inputs make it ideal for aggregating detector outputs in large-scale, structured time series. This design enables flexible, interpretable, and high-performance anomaly detection across diverse retail scenarios (Muruganandham et al., 2024; Muslim et al., 2023).

2.5 Interpretability

As machine learning systems increasingly influence real-world decisions, interpretability becomes essential for trust, transparency, and accountability. This is especially important in high-stakes applications like fraud detection or anomaly surveillance, where understanding a model's behavior is critical (Awan, 2023; Mesameki, 2025).

To interpret complex or black-box models, surrogate modeling was adopted — a well-established approach where a simple, interpretable model is trained to approximate the decisions of a more complex one (Chen et al., 2022). For each anomaly detector, a supervised surrogate model was trained to mimic binary outputs, then applied SHAP (SHapley Additive exPlanations) to extract feature attributions (Mesameki, 2025).

SHAP is based on Shapley values from cooperative game theory and offers consistent, locally accurate, and additive feature importance values (Awan, 2023; Huang & Marques-Silva, 2024). Although originally developed for tree models like LightGBM, it also generalizes to

black-box models (Mesameki, 2025). Since unsupervised models lack native interpretability, the surrogate approach allows SHAP-like models to be used reliably across all detectors (Chen et al., 2022; Mesameki, 2025), preserving a unified explanation layer for heterogeneous methods.

2.6 Chapter Summary

This chapter defined the theoretical foundations and design choices behind the proposed anomaly detection framework. The problem was framed as point-wise anomaly detection in large-scale, hierarchical, and multivariate time series, with thousands of department–store sales series influenced by holidays, markdowns, and macroeconomic factors. Given the absence of labeled anomalies, this is an unsupervised learning problem.

Decomposition methods, Prophet, STL, and TimeGPT, representing additive, non-parametric, and deep learning-based modeling approaches, were described to distinguish the signal space from residual space. Seventeen anomaly detection methods were selected across four categories: statistical methods, classical unsupervised models, probabilistic approaches, and deep learning techniques. Each was evaluated against six criteria: point-wise detection, scalability, multivariate fit, hierarchical compatibility, temporal awareness, and interpretability, as shown in Table 2.2: *Anomaly Detection Methods Comparison*.

To enhance robustness, ensemble learning was introduced to combine individual model outputs for higher accuracy, robustness, and prediction ability, since no single model proved sufficient across all criteria. Importance of interpretability of the results was highlighted and a method via SHAP, using supervised surrogate models to explain both base detectors and the ensemble, described.

Together, these components define the complete machine learning strategy that is implemented and evaluated in the following chapters.

Table 2.2*Anomaly Detection Methods Comparison*

Anomaly Detection Method	Point-Wise Detection	Large-Scale Suitability	Multivariate Capability	Hierarchical Compatibility	Temporal Awareness	Interpretability
Z-Scores	Yes	Yes	No	No*	No*	Yes
Threshold	Yes	Yes	No	No*	No*	Yes
IF	Yes	Yes	Yes	No*	No*	Partial*
KNN	Yes	Yes	Yes	No*	No*	Yes
LOF	Yes	Partial	Yes	No*	No*	Yes
HDBSCAN	Yes	Yes	Yes	Yes	No*	Partial*
OCSVM	Yes	No	Yes	No*	No*	Partial*
COPOD	Yes	Yes	Yes	No*	No	Yes
BCPD	Partial	Yes	No	No*	Yes	Yes
GMM	Yes	Partial	Yes	No*	No*	Yes
Mahalanobis	Yes	Yes	Yes	No*	No*	Partial*
Plain AE	Yes	Partial	Yes	No*	No*	No*
VAE	Yes	Partial	Yes	No*	No*	No*
LSTM AE	Yes	No	Yes	No*	No*	No*
TAE	Yes	Partial	Yes	No*	No*	No*
TCN AE	Yes	Partial	Yes	No*	No*	No*
Deep SVDD	Yes	Partial	Yes	No*	No*	No*

IF = Isolation Forest, KNN = K-Nearest Neighbors, LOF = Local Outlier Factor, HDBSCAN = Hierarchical Density-Based Spatial Clustering of Applications with Noise, OCSVM = One-Class Support Vector Machine, COPOD = Copula-Based Outlier Detection, GMM = Gaussian Mixture Model, AE = Autoencoder, VAE = Variational Autoencoder, LSTM = Long Short-Term Memory, TAE = Transformer-based Autoencoder, TCN = Temporal Convolutional Network, SVDD = Support Vector Data Description.

Note: (i) "Point-Wise Detection" indicates whether the method is capable of identifying individual anomalous time points.

(ii) "Large-Scale Suitability" denotes whether the method can scale to thousands of time series in realistic runtimes.

(iii) "Multivariate Capability" refers to a method's ability to natively process multiple input features.

(iv) "Hierarchical Compatibility" assesses whether the method can be applied across multiple organizational levels with coherent aggregation.

(v) "Temporal Awareness" assesses whether the method inherently captures temporal dependencies without preprocessing.

(vi) "Interpretability" evaluates whether the method inherently offers explainable outputs.

(vii) * This limitation is partially addressable through a practical workaround.

(viii) Table created by the author based on multiple sources including: (Al-Marie, 2023; Apgar, 2023; Asperti & Trentin, 2020; Bajaj, 2023; Blachowicz et al., 2025; Brownlee, 2018; Cai et al., 2023; Carletti et al., 2020; Dey, 2024; Despois, 2017; Dhapre, 2024; Edge Impuls, 2024; Ebenezer & Sharma, 2023; Eslava, 2023; Eyer, 2024; Feasel, 2022; Govindaraj, 2024; Helen, 2019; Holbert, 2022; Hong, 2024; Iuhasz et al., 2025; Kar, 2024; Kaya, 2020; Kennedy, 2025; Kumar, 2023; Lachehab et al., 2024; Lawton, 2024; Li et al., 2020; Liu et al., 2008; Lozovsky, 2024; Lu et al., 2023; MindBridge, 2025; Moffitt, 2024; Nelay & Turgeon, 2024; Nguyen & Vien, 2018; Ögretir et al., 2023; Owoh et al., 2024; Perry, 2019; Pérez-Carrasco et al., 2023; Peixeiro, 2023; Pykes, 2024; Rajasekaran, 2025; RisingWave, 2024; Romeu, 2021; Ruberts, 2020; Ruff et al., 2018; Kamoï & Kobayashi, 2020; Sayago, 2024; Schmidl et al., 2022; Sendera et al., 2021; Srivastava, 2023; Teuwens, 2021; Thill, 2020; Thill et al., 2021; Tuhin et al., 2025; Tuychiyyev & DataCamp, 2021; Xu & Duraisamy, 2020; Xu et al., 2023; Yadav, 2023; Yadav, 2024; Yi, 2020; Yoon, 2022; Zhang et al., 2021; Zhao, 2022).

3 Data Understanding

This chapter outlines the dataset’s structure, content, and relevance to the anomaly detection task. It summarizes the source, key features, high-level patterns, and data quality to confirm suitability for modeling and prepare for the preprocessing steps that follow.

3.1 Initial Data Collection

The dataset used in this thesis was obtained from the Walmart Recruiting – Store Sales Forecasting competition, hosted on the Kaggle platform (Kaggle, 2014). It was provided as a ZIP archive containing several structured CSV files. Three of these, `train.csv`, `stores.csv`, and `features.csv`, were used for this project. Each file served a distinct purpose: `train.csv` contains weekly sales data per store and department, `stores.csv` provides additional store-level attributes such as type and size, and `features.csv` includes external variables such as temperature, fuel price, and economic indicators, alongside markdown and holiday-related information. These files formed the basis for constructing a multivariate time series dataset suitable for forecasting and anomaly detection tasks. No issues were encountered during the acquisition or loading of the raw data.

3.2 Data Description

The dataset used in this project was constructed by merging three sources: `train.csv` (weekly sales), `features.csv` (economic indicators, markdowns, holiday flags), and `stores.csv` (store metadata). The merged data preserves the original weekly granularity for each Store–Department combination, resulting in 421,570 rows across 17 columns.

The summary statistics suggest the influence of outliers: the median sales value is \$7,612.03, while the mean is substantially higher at \$15,981.26, maximum at \$693,099.36, and minimum at \$−4,988.94, as depicted in Table 3.1 *Descriptive Statistics of Weekly Sales Values* below.

Table 3.1*Descriptive Statistics of Weekly Sales Values*

Statistic	Weekly Sales
Count	421,570
Mean	15,981.26
Standard deviation	22,711.18
Minimum	-4,988.94
25th Percentile	2,079.65
Median	7,612.03
75th Percentile	20,205.85
Maximum	693,099.36

Note: Table created by the author based on the original dataset.

Each row represents a weekly observation for a specific department in a specific store. In total, the dataset covers 45 stores, 81 departments, and 143 weeks, forming a hierarchical structure of 3,331 unique time series. The time series spans from the week of 2010-02-05 to 2012-10-26.

The feature set includes external variables such as temperature, fuel prices, consumer price index (CPI), unemployment rates, and five markdown columns (interpreted as promotional campaigns). One binary feature is `_holiday` represents national holidays.

3.3 Data Description

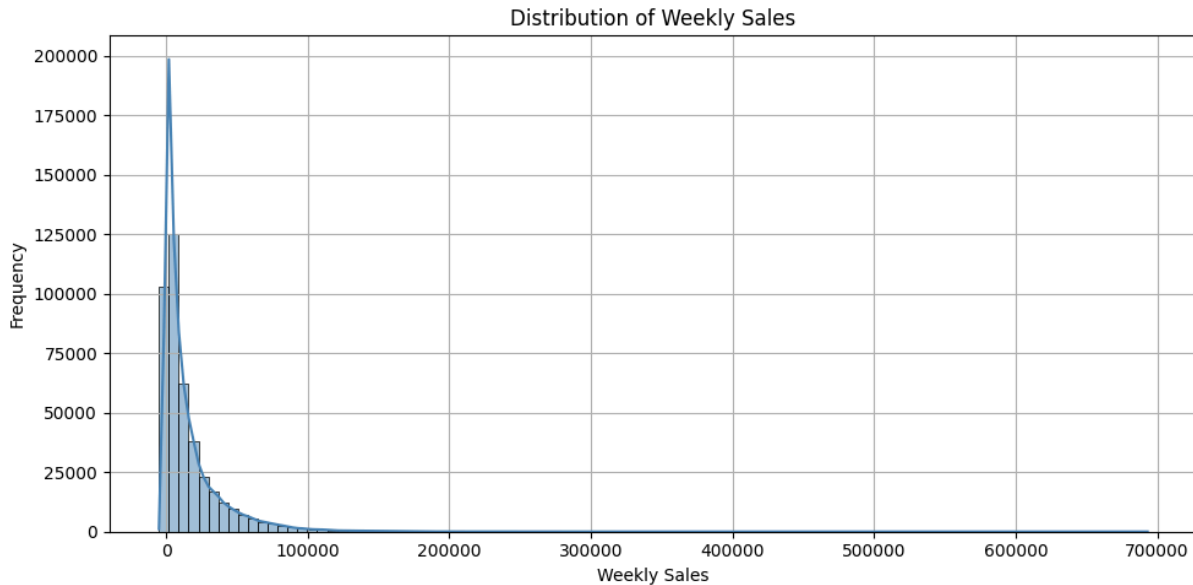
This section presents an initial exploration of the dataset, with the aim of validating its suitability for time series anomaly detection. The focus is on uncovering temporal patterns, detecting seasonal effects, and understanding how external factors such as holidays and markdowns influence sales behavior.

3.3.1 Distribution of Weekly Sales

To better understand the underlying sales behavior, the distribution of the target variable weekly sales was analyzed across all Store–Department combinations. As visualized in Figure 3.1 *Distribution of Weekly Sales*, the distribution is highly right-skewed, with the majority of weekly sales clustered below \$50,000 and a long tail of extreme values reaching up to \$693,099.36.

Figure 3.1

Distribution of Weekly Sales



Note: Figure created by the author based on the original dataset.

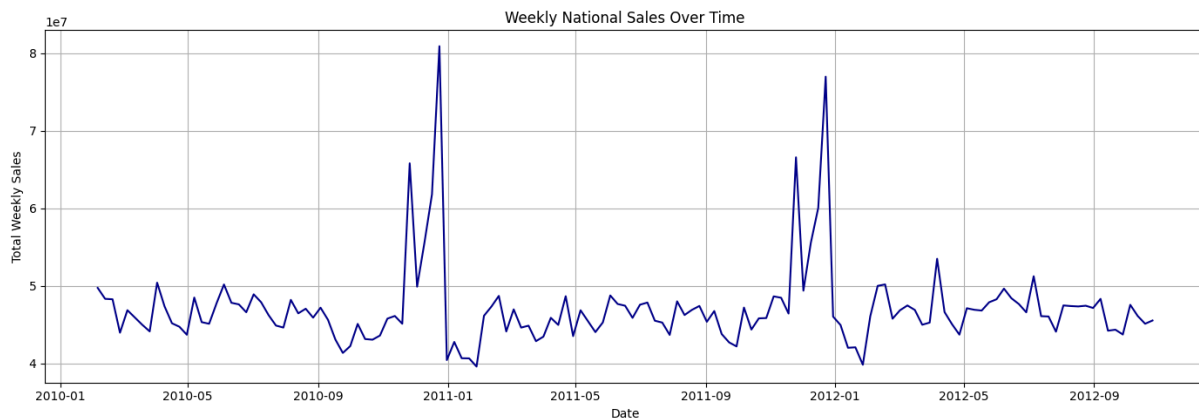
The calculated skewness of 3.26 quantitatively reinforces this heavy-tailed distribution. These patterns highlight the importance of robust anomaly detection methods that can account for high variance and asymmetry in the data.

3.3.2 Trend and Seasonality

A clear seasonal structure is immediately apparent when visualizing the national-level weekly sales over time (see Figure 3.2 *Weekly National Sales Over Time*).

Figure 3.2

Weekly National Sales Over Time



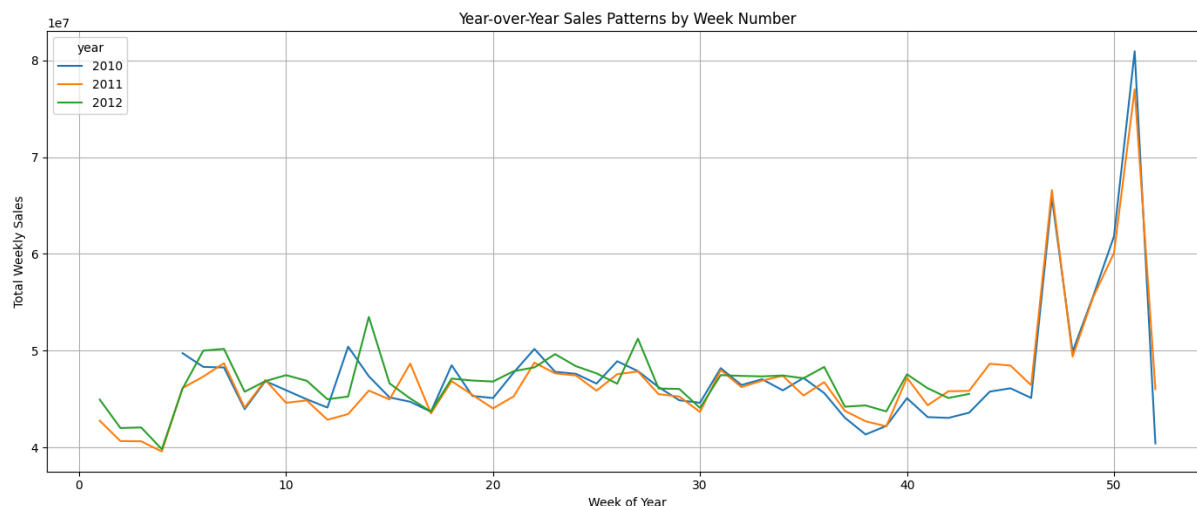
Note: Figure created by the author based on the original dataset.

Regular spikes, most prominently during late November and December, highlight recurring holiday-driven peaks in consumer activity. This visual pattern strongly suggests seasonality and temporal dependencies in the data. To further validate this, a year-over-year comparison

was plotted by aggregating sales across calendar weeks for 2010, 2011, and 2012 (see Figure 3.3 *Year-over-Year Sales Patterns by Week Number*). The near-identical shape of the curves across years confirms consistent seasonal effects on the national level.

Figure 3.3

Year-over-Year Sales Patterns by Week Number

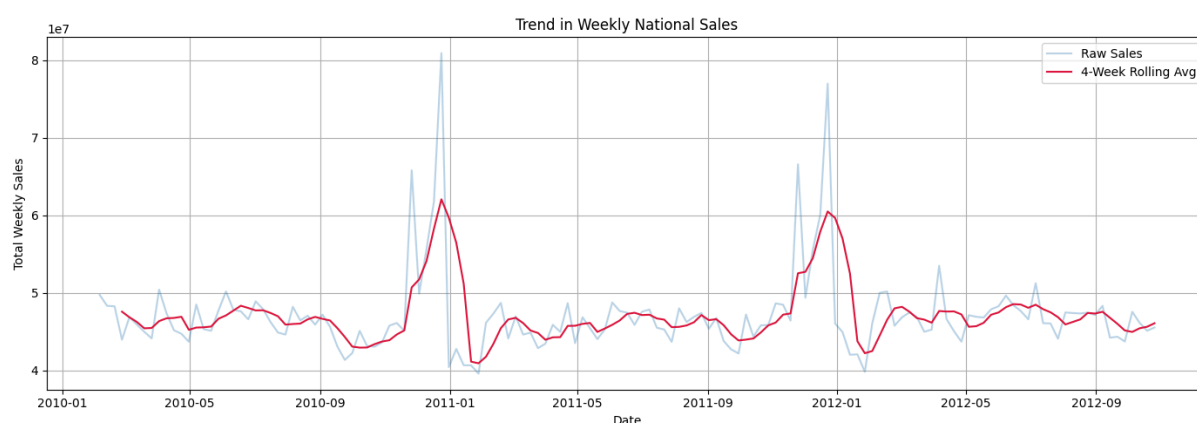


Note: Figure created by the author based on the original dataset.

Finally, a 4-week rolling average was applied to the national time series to emphasize the underlying trend (see Figure 3.4 *Trend in Weekly National Sales*), further supporting the presence of long-range temporal structure. These findings justify the application of time series decomposition methods and temporally-aware anomaly detection models throughout this thesis.

Figure 3.4

Trend in Weekly National Sales



Note: Figure created by the author based on the original dataset.

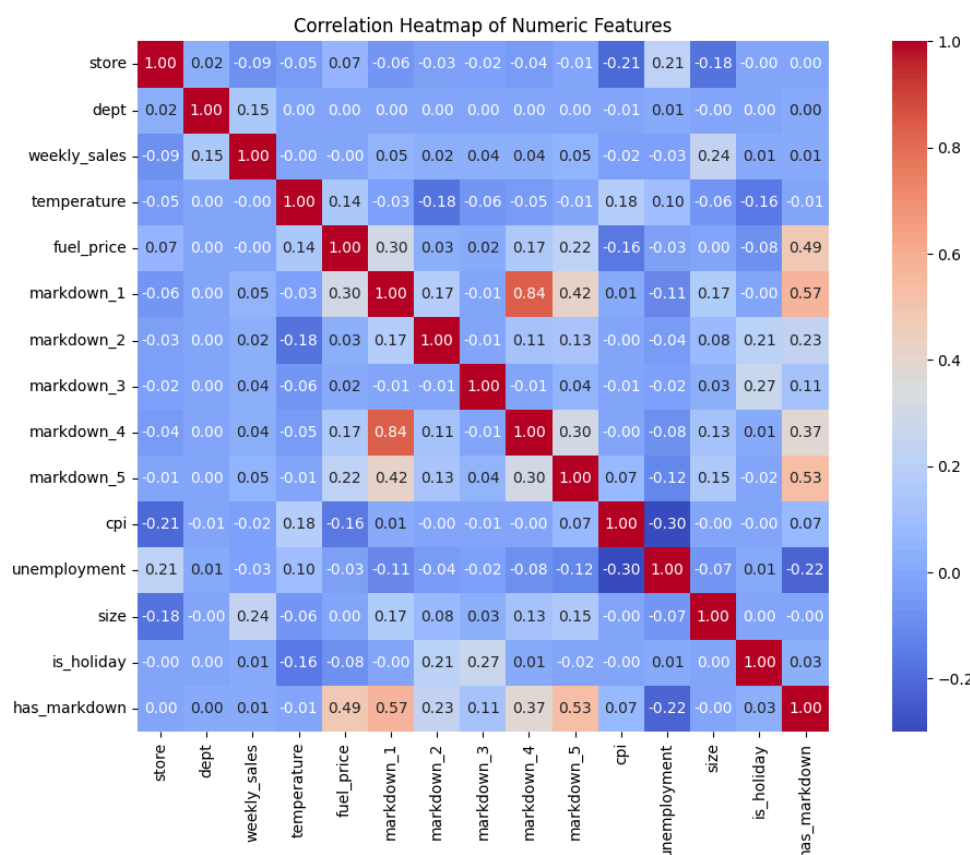
3.3.3 Feature Relationships

To better understand how the numerical features relate to each other and to the target variable, a correlation heatmap was generated (see Figure 3.5 *Correlation Heatmap of*

Numeric Features). As expected, the strongest correlation with weekly sales was observed for the size of the store ($r = 0.24$), which aligns with the assumption that larger stores tend to generate higher sales. Moderate positive correlations were also noted with `markdown_1` ($r = 0.17$) and `markdown_5` ($r = 0.22$), suggesting that certain promotional activities may be associated with elevated sales volumes. Interestingly, no strong correlations were found between economic indicators (e.g., CPI, unemployment) and weekly sales, indicating that these features may have more localized or indirect effects. Overall, the low to moderate correlations support the inclusion of these variables in multivariate modeling, where non-linear and interaction effects can be more effectively captured.

Figure 3.5

Correlation Heatmap of Numeric Features

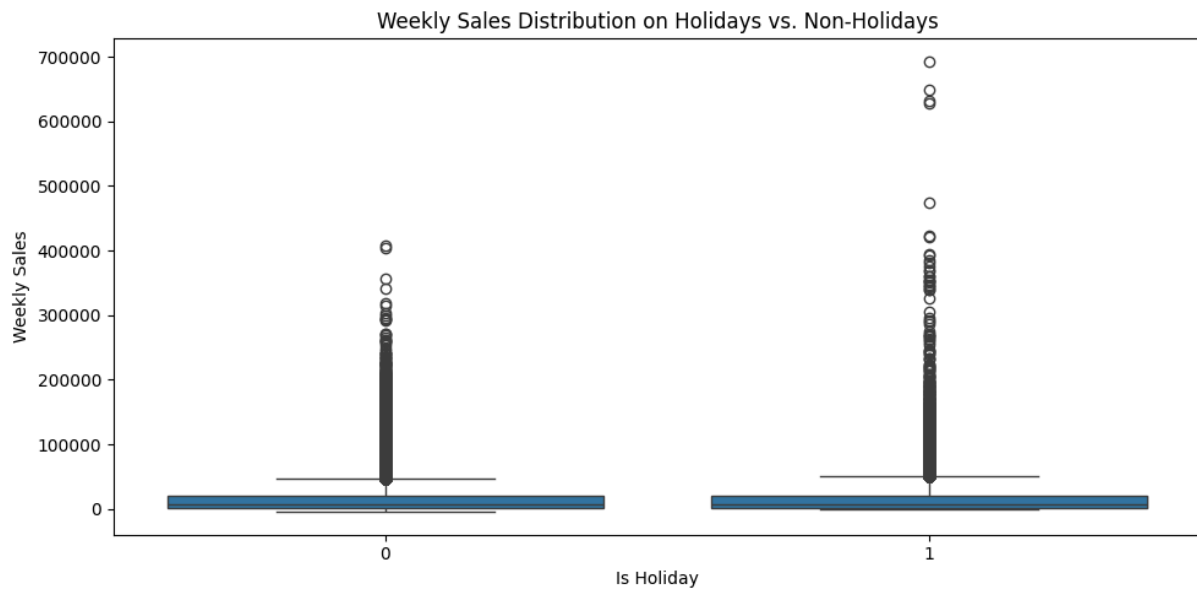


Note: Figure created by the author based on the original dataset.

To assess how binary features influence sales behavior, distribution of `weekly_sales` across two categorical indicators: `is_holiday` and `has_markdown` is visualized. As shown in Figure 3.6 *Weekly Sales Distribution on Holiday vs. Non-Holiday*, holiday weeks tend to show a wider distribution of weekly sales, with multiple high-value outliers, suggesting the presence of holiday-driven spikes.

Figure 3.6

Weekly Sales Distribution on Holiday vs. Non-Holiday

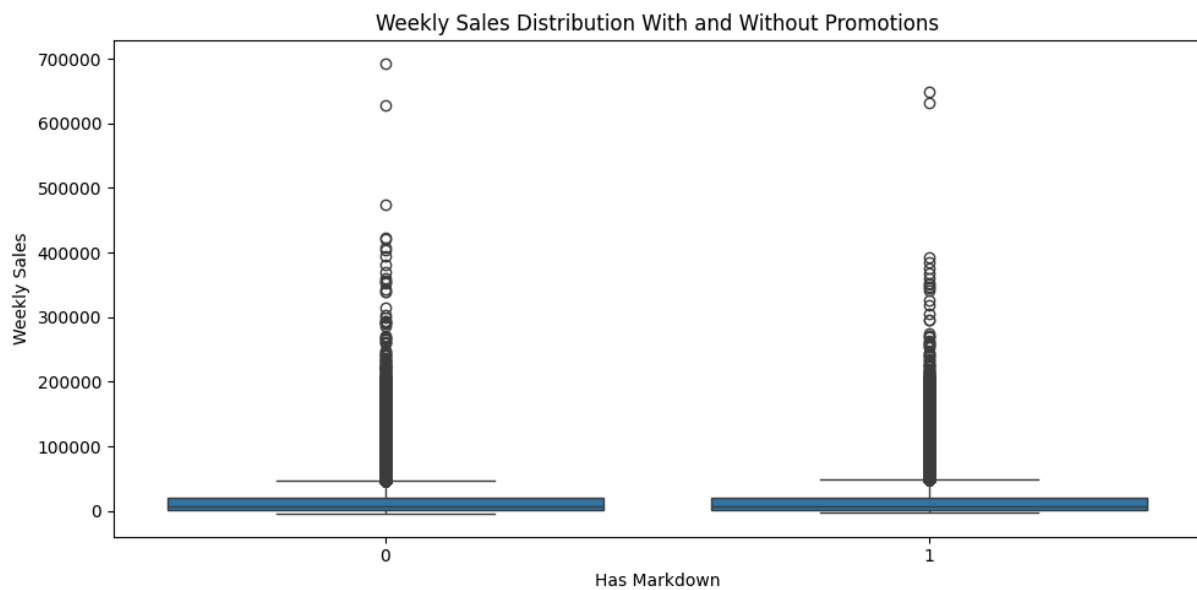


Note: Figure created by the author based on the original dataset.

As seen in Figure 3.7 *Weekly Sales Distribution With and Without Promotions*, weeks with and without markdown promotions have a comparable median and overall spread in weekly sales. However, non-promotion weeks display more extreme outliers, suggesting that unusually high sales spikes can occur even in the absence of promotions.

Figure 3.7

Weekly Sales Distribution With and Without Promotions



Note: Figure created by the author based on the original dataset.

3.3.4 Data Quality Verification

A numerical analysis of missing values (see Table 3.2 *Summary of Missing Values*) revealed substantial gaps in the five markdown features.

Table 3.2

Summary of Missing Values

Column	Count	Percentage
markdown_2	310,322	73.61
markdown_4	286,603	67.98
markdown_3	284,479	67.48
markdown_1	270,889	64.26
markdown_5	270,138	64.08

Note: Table created by the author based on the original dataset.

A detailed quality check revealed that 671 individual Store–Department time series (20.14% of the total 3,331) contained irregularities in their weekly continuity. These gaps reflected implicitly missing data — i.e., weeks for which no entry was present in the dataset at all. The number of missing weeks per affected series ranged from just 1 to as many as 142, with a median of 94. These inconsistencies would have made direct modeling or decomposition unreliable and required targeted preprocessing steps to reconstruct a consistent weekly structure. The specific correction strategy is described in Section 4.2.

Next, the `weekly_sales` column was examined for invalid or unexpected values. A total of 1,285 records contained negative sales, with most values close to zero and a few larger outliers, such as the global minimum of \$−4,988.94. These negative values likely reflect product returns or internal sales corrections and were retained in the dataset, as they may carry important signals for anomaly detection.

Overall, the dataset is of high quality and well-suited for the modeling and evaluation tasks that follow.

3.4 Chapter Summary

This chapter outlined the structure and content of the dataset, which spans 143 weeks across 45 stores and 81 departments. Skewed sales distribution, seasonal trends, and contextual influences were found. Missing weeks were identified in 20% Store–Department series. Establishing the key data challenges and patterns, this chapter set the stage for the cleaning and transformation steps detailed in the next chapter.

4 Data Preparation

This chapter explains how the raw sales data was transformed into a modeling-ready format for hierarchical, multivariate, point-wise anomaly detection. The preparation process included cleaning and merging inputs, handling missing values, creating derived features, injecting synthetic anomalies, and scaling.

4.1 Data Selection and Integration

In this particular case, the data were obtained in a ZIP file from the Kaggle Website as part of the Walmart Recruiting — Store Sales Forecasting Prediction Competition (Kaggle, 2014).

It contains four CSV files:

- stores.csv containing additional information about each store — its size and type (anonymized as either A, B, or C),
- train.csv containing weekly sales for each Store–Department–Date combination, including a column to indicate holidays in a particular week,
- test.csv containing further Store–Department–Date combinations and the holiday column, but without weekly sales (intended only for the original prediction task),
- features.csv containing additional data related to the store, department, and regional activity for the given dates — including temperature, fuel price, CPI, unemployment, and five markdown columns related to promotional markdowns.

To prepare the dataset for anomaly detection, three sources were merged — train.csv, stores.csv, and features.csv. The merged dataset preserved the original Store–Dept–Date granularity, resulting in a total of 421,570 rows — one for each department in each store for each week.

4.2 Data Cleaning

Several cleaning steps were necessary to prepare the dataset for modeling. Column names were standardized to snake_case, and redundant fields introduced by merging (e.g., IsHoliday_x, IsHoliday_y) were removed to maintain a clean schema.

The five markdown-related features (markdown_1 to markdown_5) showed between 64% and 74% of values absent. These were filled with 0, based on the assumption that missing values indicate the absence of a promotion.

Although the column weekly_sales did not contain nulls at the dataset level, a detailed audit revealed that 671 Store–Department time series (20.14%) were incomplete. To address this, missing values were filled using mean imputation per Store–Dept group.

After these steps, the dataset was fully cleaned and confirmed to be complete, consistent, and suitable for hierarchical, multivariate time-series modeling.

4.3 Feature Construction

To support robust anomaly detection, several new features were derived from the raw dataset. The five promotional columns (markdown_1 through markdown_5) were retained. In addition, a new binary column, has_markdown, was created to flag whether any markdown was active in a given week. The categorical feature type, representing store format as A, B, or C, was one-hot encoded into binary indicators to ensure compatibility with machine learning models.

Finally, a special label column injected_anomaly was added, marking known artificial anomalies for model evaluation. This label was never used in training and was preserved throughout all downstream transformations to enable fair and reproducible benchmarking. Values in weekly_sales were modified with injected anomalies, but original weekly sales values were retained in the weekly_sales_original column.

4.4 Injecting Anomalies

To enable robust and repeatable evaluation of anomaly detection models, a controlled set of artificial anomalies was injected into the multivariate time series. These synthetic anomalies were added after data cleaning and before forecasting to simulate realistic sales deviations while preserving the underlying time series structure.

Two types of point anomalies were injected according to defined business logic:

- Spike anomalies (n = 2,107) were introduced into weeks where there was no holiday or promotion. These simulate unexpected surges in sales that would typically trigger business curiosity or investigation.
- Drop anomalies (n = 2,107) were injected into weeks that contained a holiday or promotion. These represent concerning underperformance during periods that would normally be expected to yield higher sales.

Each anomaly was introduced by modifying the original weekly sales value based on the group-level rolling mean and standard deviation (window = 8 weeks). Specifically, the anomaly was created by adding or subtracting 4 times the rolling standard deviation from the rolling mean. Anomaly positions were randomized across the timeline. This approach was designed to produce statistically significant deviations that are detectable yet remain within a plausible range, simulating realistic irregularities rather than extreme outliers.

To validate the injection mechanism, Z-scores were recalculated post-injection using a separate rolling baseline. The resulting anomalies had an average Z-score magnitude of 3.73. Only 0.05% of anomalies had an absolute Z-score ≥ 5 , meaning that the injected points were

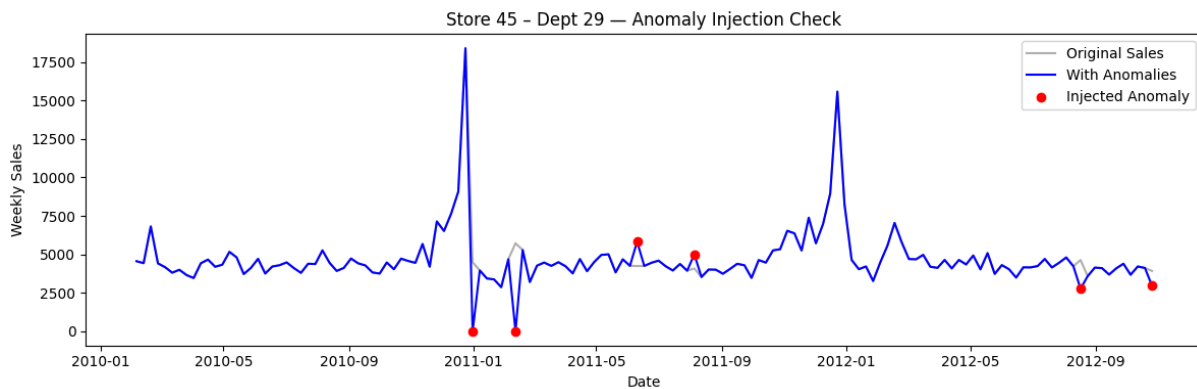
subtle enough to challenge detection models, while still being anomalous by statistical standards.

The injected anomaly labels were stored in the `injected_anomaly` column and strictly reserved for evaluation only, never used for training.

To validate the correctness and realism of the injected anomalies, they were also inspected visually. Plots comparing the original and modified sales clearly confirmed that anomalies appeared in plausible locations and magnitude (see example on Figure 4.1 *Store 45 – Dept 29 – Anomaly Injection Check*). This visual inspection served as a sanity check to ensure that injected anomalies would not trivially dominate the time series but instead pose a meaningful challenge for detection models.

Figure 4.1

Store 45 – Dept 29 – Anomaly Injection Check



Note: Figure created by the author based on the created dataset.

4.5 Scaling

To prepare the dataset for machine learning, all numeric features were scaled to stabilize input distributions while preserving anomaly-relevant signals. Given the presence of outliers, skewed distributions, and heavy tails in the sales data, the `RobustScaler` from `scikit-learn` was selected. Unlike standard techniques such as `StandardScaler` or `MinMaxScaler`, `RobustScaler` uses the median and IQR to mitigate the influence of extreme values (Scikit Learn, 2018). This choice was particularly important for ensuring that downstream models, especially those sensitive to feature magnitude, would not misinterpret outliers as distributional noise.

Crucially, this scaling was applied only after decomposition, targeting the residual component of the signal, which reflects deviations from trend and seasonality. By isolating and scaling only the residuals, the models received a normalized view of the "unexpected" behavior.

For real-time anomaly detection, it was essential to avoid using future data when scaling past observations. Global scaling approaches, which compute scaling parameters across the full series, inadvertently leak information from the future into the past. To address this, the

pipeline applies rolling-window robust scaling, ensuring that each point is transformed based only on values available up to that point in time.

Among the tested configurations, an 8-week rolling window emerged as the most suitable choice. It provides a responsive scaling horizon—short enough to highlight recent distributional shifts and sharp deviations, yet long enough to remain robust to volatility and avoid overfitting to local noise.

4.6 Chapter Summary

The dataset was assembled by merging three source files into a unified Store–Department–Date format at weekly resolution. After column standardization and removal of redundancies, the final dataset contained 421,570 rows. All column names were converted to snake_case, and categorical features such as type and is_holiday were cleaned for consistency. Missing values in the markdown columns were filled with zeroes. A new binary feature has_markdown was introduced to indicate promotional activity. The type column was one-hot encoded to convert store categories A, B, and C into binary features.

20% Store–Department time series with missing weeks were aligned to a complete timeline (February 2010–October 2012) and missing entries were filled using groupwise mean imputation.

Artificial anomalies were then injected into the cleaned weekly_sales column: 2,107 spikes and 2,107 drops. Injection logic was based on contextual business rules and calibrated using local rolling statistics. Anomalies were labeled using injected_anomaly and anomaly_type, and used strictly for evaluation.

Scaling was deferred until after decomposition. Residuals from each decomposition method were scaled independently to ensure comparability across models, while preserving the raw signal structure for injection and forecasting.

5 Modelling

This chapter applies the selected models from Chapter 2 Machine Learning Understanding to the prepared dataset from Chapter 4 Data Preparation, evaluates their performance across all decomposition sources and anomaly detection methods, and identifies the best-performing approach for reliable anomaly detection.

5.1 Model Assessment

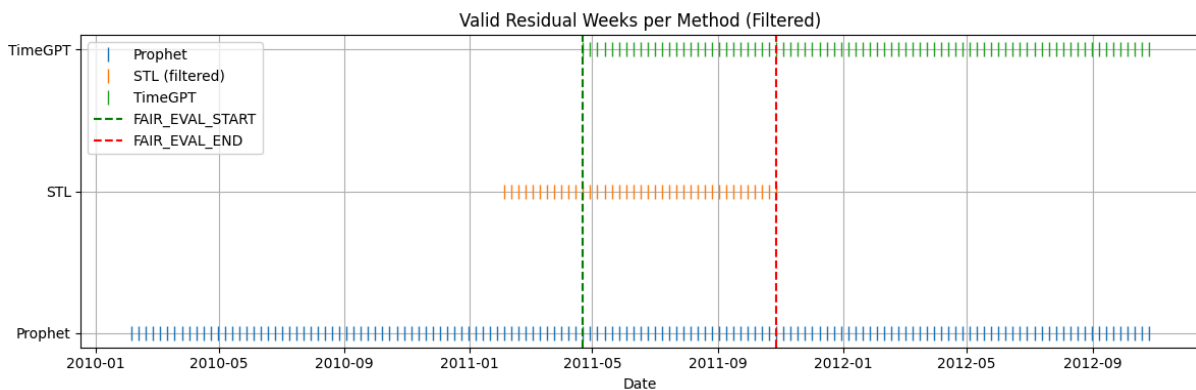
To ensure an unbiased comparison across models and decomposition sources, a fair evaluation window was defined as the intersection of all weeks where valid residuals were available from all three decomposition methods.

Each anomaly detection method was then evaluated under two distinct evaluation windows:

- Fair window: The overlapping weeks across all decomposition sources, used for fair cross-method comparison. In the comparative setting, the Fair window spanned 2011-04-22 to 2011-10-28, covering 27 weeks, as shown in Figure 5.1 *Valid Residual Weeks per Method (Filtered)*.
- Full window: The full original date range available, capturing the method's overall detection behavior (143 weeks, from 2010-02-05 to 2012-10-26).

Figure 5.1

Valid Residual Weeks per Method (Filtered)



Note: Figure created by the author based on the created dataset.

The evaluation process computed the following for each method and source:

- confusion matrix components — true positives (TP) and its breakdown into spikes and drops, false positives (FP), false negatives (FN), true negatives (TN),
- standard metrics: precision, recall, and F1 score (harmonic mean of precision and recall).

To ensure consistency, the entire evaluation process was fully automated. A custom pipeline dynamically identified the correct prediction column from each detection result dataframe, inferred method and decomposition source names from variable names, created the two evaluation windows, computed and logged all metrics into a global results log.

This setup enabled comprehensive comparison combining 16 detection methods and 3 decomposition strategies, as depicted in detail in Table 5.1 *Best Runs per Anomaly Detection Method and Decomposition Combination*. Evaluation results were used for model tuning, ranking, and the training of the final ensemble.

Table 5.1

Best Runs per Anomaly Detection Method and Decomposition Combination

Anomaly Detection Model	Decomposition Method	Time	Fair F1	Full F1	Anomaly Detection Model	Decomposition Method	Time	Fair F1	Full F1
Z-Scores	Prophet	36.3s	0.2943	0.1949	Mahalanobis	Prophet	77.5s	0.0547	0.0451
Z-Scores	STL	35.3s	0.0802	0.0205	Mahalanobis	STL	78.8s	0.0129	0.0081
Z-Scores	TimeGPT	33.42s	0.2159	0.1096	Mahalanobis	TimeGPT	131.9s	0.0471	0.0375
Threshold	Prophet	36.03s	0.0747	0.0652	GMM	Prophet	59.9s	0.0255	0.0216
Threshold	STL	35.59s	0.0201	0.0138	GMM	STL	58.5s	0.0092	0.0135
Threshold	TimeGPT	34.29s	0.0485	0.0567	GMM	TimeGPT	116.2s	0.0248	0.0211
IF	Prophet	46.55s	0.0482	0.0357	Plain AE	Prophet	725.1ss	0.0131	0.0128
IF	STL	44.23s	0.0143	0.0089	Plain AE	STL	702.3s	0.0048	0.0023
IF	TimeGPT	44.50s	0.0402	0.0305	Plain AE	TimeGPT	690.5ss	0.0067	0.0028
KNN	Prophet	52.1s	0.1318	0.0852	VAE	Prophet	138.2s	0.0144	0.0121
KNN	STL	50.0s	0.0499	0.0327	VAE	STL	137.6s	0	0
KNN	TimeGPT	45.7s	0.1116	0.0774	VAE	TimeGPT	136.4s	0	0
LOF	Prophet	8.9s	0.0044	0.0099	LSTM AE	Prophet	2400.s	-	-
LOF	STL	9.0s	0.0052	0.0059	LSTM AE	STL	2400.s	-	-
LOF	TimeGPT	8.4s	0.0033	0.0054	LSTM AE	TimeGPT	2400.s	-	-
HDBSCAN	Prophet	73.3s	0.0238	0.0280	TAE	Prophet	785.3s	0.0104	0.0101
HDBSCAN	STL	70.9s	0.0094	0.0082	TAE	STL	804.6s	0.0042	0.0037
HDBSCAN	TimeGPT	71.7s	0.0120	0.0157	TAE	TimeGPT	778.9s	0.0137	0.0110
OCSVM	Prophet	1130.6s	0.0098	0.0091	TCN AE	Prophet	113.3s	0	0.0074
OCSVM	STL	750.5s	0.0035	0.0032	TCN AE	STL	112.4s	0	0.0069
OCSVM	TimeGPT	1432.3s	0.0085	0.0079	TCN AE	TimeGPT	112.0s	0	0.0082
BCPD	Prophet	31.5s	0.0090	0.0101	Deep SVDD	Prophet	631.7s	0.0084	0.0071
BCPD	STL	26.2s	0.0057	0.0061	Deep SVDD	STL	627.8s	0.0069	0.0069
BCPD	TimeGPT	26.5s	0.0061	0.0072	Deep SVDD	TimeGPT	632.5s	0	0

AE = Autoencoder, COPOD = Copula-Based Outlier Detection, GMM = Gaussian Mixture Model, HDBSCAN = Hierarchical Density-Based Spatial Clustering of Applications with Noise, IF = Isolation Forest, KNN = K-Nearest Neighbors, LOF = Local Outlier Factor, LSTM = Long Short-Term Memory, OCSVM = One-Class Support Vector Machine, SVDD = Support Vector Data Description, VAE = Variational Autoencoder, TAE = Transformer-based Autoencoder, TCN = Temporal Convolutional Network.

Note: (i) "Time" indicates only runtime of the Anomaly Detection Model. (ii) Table created by the author based on the created dataset.

5.2 Decomposition Methods

Due to the clear seasonality and trend patterns in the sales data, decomposition was applied to separate predictable components from unpredictable residuals used for anomaly detection. Three methods, Prophet, STL, and TimeGPT, were selected for their diverse modeling approaches. Comparing results of anomaly detection models across these residuals allows for a robust evaluation of model performance.

5.2.1 Prophet Decomposition

This model was chosen to decompose weekly sales to trend, seasonality, and residuals. Multivariate form of Prophet with `.add_regressor()` with external features was tested. A separate Prophet instance was trained per Store–Department combination.

Modelling Assumptions

Prophet models trend and seasonality additively and works best with regularly spaced data. It assumes residuals are independent and doesn't capture interactions between observations. It naturally handles missing values without special treatment. External regressors are assumed to have linear, independent effects, meaning no interactions or nonlinearities are modeled.

Test Design

Prophet decomposition results were not evaluated directly; instead, their impact was assessed via anomaly detection model outputs. Precision, recall and F1 were computed in both the Fair and Full windows against known, injected spikes and drops. A visual inspection of decomposition quality and of residuals was conducted for representative Store–Department combinations.

Parameter Setting

The highest F1 score was achieved using only the `weekly_sales` column with `weekly_seasonality=True`, `yearly_seasonality=True`, `daily_seasonality=False`, without additional regressors or tweaks.

Model Assessment

Residuals produced by Prophet consistently outperformed those from STL and TimeGPT across nearly all detection methods. In 12 anomaly-detection models, higher Full F1 scores were achieved by Prophet-based residuals, while for the Fair window, Prophet achieved highest F1 in 11 models (see Table 5.1 *Best Runs per Anomaly Detection Method and Decomposition Combination*).

Average runtime in the univariate setting was 7.5 minutes, while the full multivariate configuration required just over 2 hours.

In Figure 5.2 *Store 45 – Dept 29: Prophet Decomposition*, decomposition into trend and seasonality and residuals with highlighted injected anomalies are shown, confirming consistency across the full timeline.

Figure 5.2

Store 45 – Dept 29: Prophet Decomposition



Note: Figure created by the author based on the created dataset.

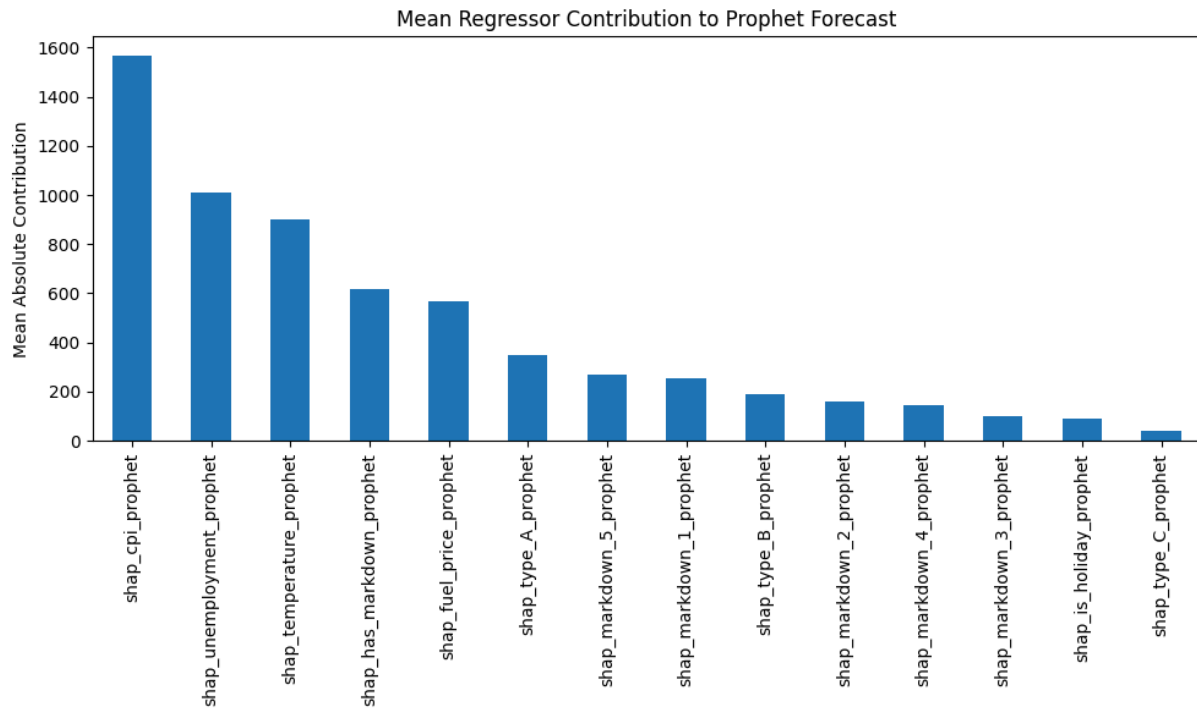
Given its superior F1, precision and recall across most models and its reasonable computational time, Prophet was chosen for the final model.

Model Refinement and Observations

Multiple Prophet configurations were evaluated; residuals from each were processed through the anomaly detection pipeline and underperformed relative to the basic univariate weekly_sales model. SHAP-style analysis was applied to all candidate regressors, revealing CPI, unemployment and temperature as the only meaningful contributors (Figure 5.3 *Mean Regressor Contribution to the Prophet Forecast* below). Runs using all business variables or the top seven SHAP-ranked features yielded $F1 = 0.0000$, while the second-best configuration, incorporating just two best regressors, achieved only half the F1 of the univariate setup in the same anomaly detection models.

Figure 5.3

Mean Regressor Contribution to Prophet Forecast



Note: Figure created by the author based on the created dataset.

5.2.2 STL Decomposition

The STL decomposition was run across each Store–Department series. The outputs included three new columns per observation: `stl_trend`, `stl_seasonal`, and `residual` (calculated manually as `weekly_sales - stl_trend - stl_seasonal`). The implementation was based on `statsmodels.tsa.seasonal.STL`.

Modelling Assumptions

STL assumes a univariate, regularly spaced time series with stable, approximately additive seasonality. While it can accommodate some missing values via interpolation, it is not intended for irregular gaps or heavily censored sequences. External regressors, autocorrelation, and structural breaks are not explicitly modeled.

Test Design

STL decomposition results were not evaluated directly; instead, their impact was assessed via anomaly detection model outputs. Precision, recall and F1 were computed in both the Fair and Full windows against known, injected spikes and drops. A visual inspection of decomposition quality and of residuals was conducted for representative Store–Department combinations.

Parameter Setting

The highest F1 score was achieved applying STL separately to each group using a fixed period=52 to match weekly seasonality.

Model Assessment

STL was the weakest of the three decomposition methods tested. Across all evaluated anomaly-detection models, STL never produced the best-performing residuals in either the Full evaluation window and only once in the Fair window (see Table 5.1 *Best Runs per Anomaly Detection Method and Decomposition Combination*). These issues are likely due to LOESS smoothing's reduced support near the edges, which prevents reliable trend estimation without sufficient surrounding data. This severely impacted anomaly detection, as noise at the series boundaries led to both false positives and missed injected anomalies.

STL decompositions were extremely fast to compute — finished in 2 minutes and 10 seconds.

In Figure 5.4 *Store 45 – Dept 29: STL Decomposition* below, the trend and seasonality components and the residuals with highlighted injected anomalies are presented, showing major residual inconsistency between the incomplete years 2010 and 2012 and the full middle year 2011.

Figure 5.4

Store 45 – Dept 29: STL Decomposition



Note: Figure created by the author based on the created dataset.

The low runtime did not compensate for the instability observed in the residuals and therefore, STL was not chosen as the final decomposition method.

Model Refinement and Observations

The period = 52 value was retained, as consistent weekly periodicity was assumed across all series. Significant modifications were applied during preprocessing in an attempt to make STL fit adequately in 2010 and 2012 (including various backfill and forward-fill techniques and filling missing weeks at the beginning and end of each series with the mean). Padding was applied before STL to enforce a complete weekly timestamp range across all units, ensuring equal decomposition windows and later comparability. However, all downstream evaluations were restricted to the original rows (`is_filled == False`) to avoid contamination from artificially added timestamps.

Because major residual issues in the incomplete years were observed and remained unresolved, no further extensive tuning was performed. The decomposition method was retained in the comparison pipeline only in case STL outperformed the other two methods in the Fair window; however, that did not occur.

5.2.3 TimeGPT Decomposition

The TimeGPT decomposition was run across each Store–Department series. For this study, the `detect_anomalies()` function was used to extract forecasts across each Store–Dept series. Residuals were then computed as the difference between weekly sales and predicted values (`yhat`).

Modelling Assumptions

As a pretrained black-box model, TimeGPT makes several implicit assumptions. Input series must be continuous, regularly spaced. Series should have minimal missing timestamps; dense, clean inputs yield better forecasts. Model internally assumes that sales patterns follow trends learned from general time series behavior. Forecasts are generated only after an initial context window. No custom holiday, event, or regressor data is supported. Exogenous feature support is limited or disabled in the free version.

Test Design

TimeGPT decomposition results were not evaluated directly; instead, their impact was assessed via anomaly detection model outputs. Precision, recall and F1 were computed in both the Fair and Full windows against known, injected spikes and drops. A visual inspection of decomposition quality and of residuals was conducted for representative Store–Department combinations.

Parameter Setting

The highest F1 score was achieved using `detect_anomalies()` in its basic configuration: `time_col="date", target_col="y", freq="W-FRI"`.

Model Assessment

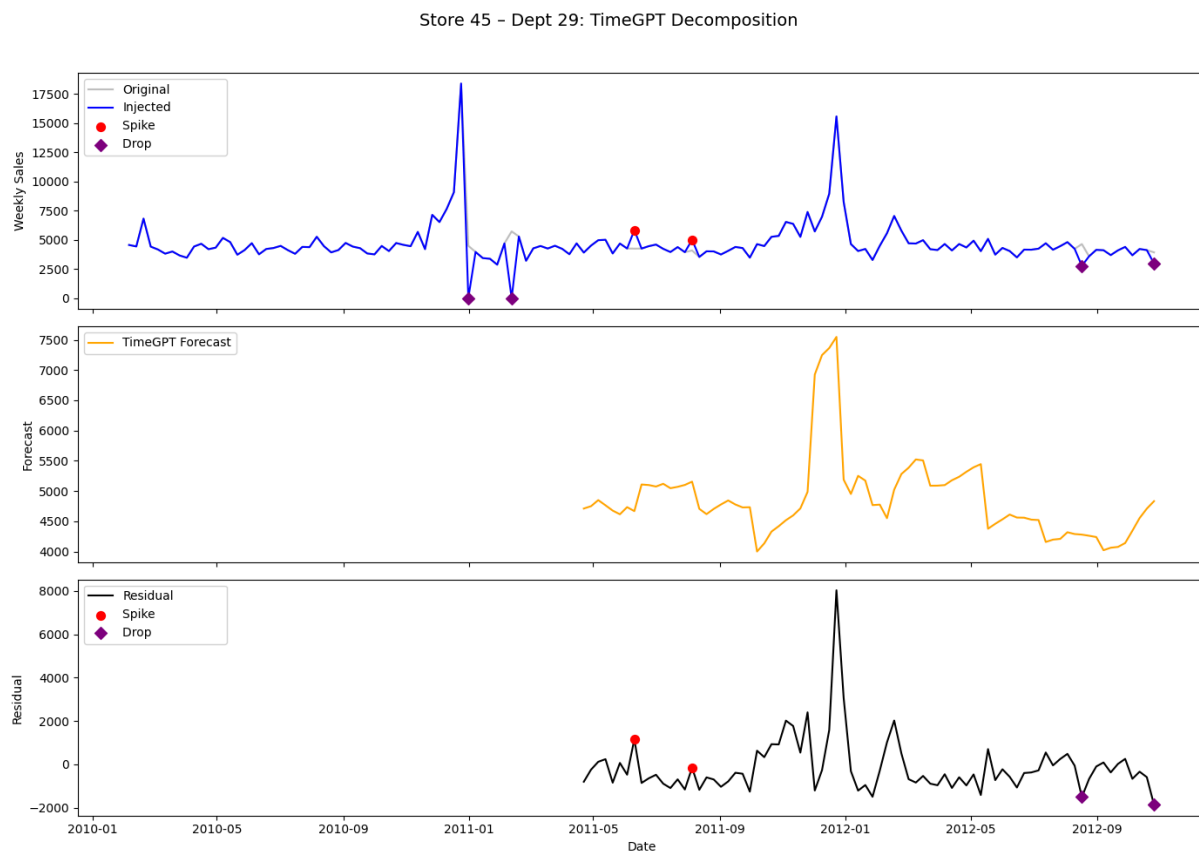
Across the all evaluated anomaly detection models, TimeGPT produced better results than Prophet only twice in the Full window and twice in the Fair window. In the Full window, TimeGPT provided better results in certain deep learning anomaly detection settings — Transformer-based Autoencoder and TCN Autoencoder (see Table 5.1 *Best Runs per Anomaly Detection Method and Decomposition Combination*).

The decomposition across the series completed in approximately 32 minutes.

In Figure 5.5 *Store 45 – Dept 29: TimeGPT Decomposition* below, the trend and seasonality components and the residuals with highlighted injected anomalies are presented, beginning forecast only in 2011-04-22, but then showing well formed forecast and residuals.

Figure 5.5

Store 45 – Dept 29: TimeGPT Decomposition



Note: Figure created by the author based on the created dataset.

Overall, while TimeGPT delivered a few competitive results, it was ultimately outperformed by Prophet in accuracy, transparency, control, and speed. TimeGPT was not chosen as the final decomposition method.

Model Refinement and Observations

TimeGPT was used solely for decomposition and residual extraction, not for forward forecasting. Early attempts to call `.forecast()` or specify `h=length` repeatedly failed (returning NaNs, missing outputs, or errors about overly long horizons) due to undocumented free-tier API constraints. As a workaround, `detect_anomalies()` was tried to obtain residuals, but only the `yhat` forecast column was returned. Expected fields for anomalies, confidence levels, or pointwise flags were absent. Consistent weekly padding and interpolation were applied to all series before decomposition, yet the first few weeks of each series were omitted from the API response, presumably because the model requires a minimum warm-up context, so those timestamps were excluded from the Fair window evaluation. The black-box nature of the service precluded any internal inspection, customization of horizons or confidence intervals, and advanced anomaly-detection tuning beyond basic decomposition.

5.3 Anomaly Detection Models

This section details the anomaly detection models applied to the residuals generated by each decomposition method. The models are presented individually to highlight their strengths, limitations, and suitability for large-scale, hierarchical, multivariate anomaly detection.

5.3.1 Z-Scores Anomaly Detection Model

The following section documents the application of Z-score-based anomaly detection.

Modelling Assumptions

Z-scores assume a normal distribution of input features and that anomalies are extreme deviations from the mean. It assumes stationarity within the window used to compute the mean and standard deviation, and requires complete data without missing values.

Test Design

No explicit train/test split was performed. Instead, an 8-week centered sliding window was applied directly over each Store–Dept series to compute local mean and STD to preserve temporality. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach preserves temporality without look-ahead while validating detection on held-out anomaly injections.

Parameter Setting

The best F1 results in Full window were achieved with `z_threshold = 2` on `residual_scaled` and sliding window set at 8 weeks.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.2 *Z-Scores Anomaly Detection Model Best Results* achieved Full F1 of 0.1949. This model finished 1st across all anomaly detection models. The runtime was 36.3 seconds. This model was excluded from the final ensemble because its detection logic directly mirrors the anomaly injection process, which could lead to biased results. It is retained solely for baseline comparison, as Z-Scores represent a standard benchmark in anomaly detection.

Table 5.2

Z-Scores Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.2943	0.2153	0.4643
Full	0.1949	0.1203	0.5105

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

Several alternative parameter settings were tested, including thresholds of 3, 4, 5, and 6, as well as window sizes of 4 and 16. However, none of these configurations outperformed the combination of threshold 2 with an 8-week window. This particular setting achieved the highest F1 score, making it the most effective configuration overall.

5.3.2 Thresholding Anomaly Detection Model

The following section documents the application of quantile-based Thresholding anomaly detection.

Modelling Assumptions

The approach assumes that anomalies lie in the tails of the residual distribution and that normal behavior is centered. It makes no distributional assumption beyond rankability, but requires no missing values in the thresholded variable.

Test Design

No explicit train/test split was performed. Instead, an 8-week centered sliding window was applied directly over each Store–Dept series to compute rolling lower and upper quantile thresholds, enabling local anomaly detection while preserving temporal context. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach preserves temporality without look-ahead while validating detection on held-out anomaly injections.

Parameter Setting

The best F1 results in Full window were achieved with upper threshold at 99,5% and lower threshold at 0.5% on residual_scaled with a sliding window set at 8 weeks.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.3 *Thresholding Anomaly Detection Model Best Results* achieved Full F1 of 0.0652. This model finished 3rd across all anomaly detection models. The runtime was 36.0 seconds. This model was selected for the final ensemble.

Table 5.3

Thresholding Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0747	0.0395	0.6821
Full	0.0652	0.0339	0.8497

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

Several percentile thresholds were tested (1st/99th, 0.25th/99.75th), but the 0.5th/99.5th percentile setting yielded the best F1 score and was retained as the most effective configuration.

5.3.3 Isolation Forest Anomaly Detection Model

The following section documents the application of Isolation Forest anomaly detection.

Modelling Assumptions

Isolation Forest assumes that anomalies are rare and different from normal observations, and that the feature space provides useful separation. It does not rely on any distributional assumptions but assumes all input features are continuous, scaled, and complete.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The Isolation Forest model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in Full window were achieved with `n_estimators = 200`, `max_samples = auto`, `contamination = 0.01` and `jobs = -1`. Final input features were `residual_scaled`, `has_markdown` and `is_holiday`.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.4 *Isolation Forest Anomaly Detection Model Best Results* achieved Full F1 of 0.0357. This model finished 5th across all anomaly detection models. The runtime was 46.55 seconds. This model was selected for the final ensemble.

Table 5.4

Isolation Forest Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0482	0.1034	0.0315
Full	0.0357	0.0325	0.0397

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

Extensive tuning was performed on the Isolation Forest model by varying contamination levels (0.008 to 0.01), `max_samples` settings (auto, 256), `jobs` not specified or `-1`, and input feature sets ranging from minimal to full multivariate representations including all markdowns, economic indicators, and store type encodings. Despite these variations, none of the alternative configurations delivered F1 scores comparable to the final selected setup.

5.3.4 K-Nearest Neighbors Anomaly Detection Model

The following section documents the application of K-Nearest Neighbors anomaly detection.

Modelling Assumptions

KNN assumes that normal observations are located in dense regions of the feature space, while anomalies are isolated and lie in sparse regions. It requires that all features be numerical and properly scaled, assumes no missing values, and does not make assumptions about underlying distributions.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold `TimeSeriesSplit` was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The KNN model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected

anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in Full window were achieved with $k = 20$, metric = Chebyshev, and threshold = 98. Final input features were residual_scaled, cpi_scaled, and unemployment_scaled.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.5 *Local Outlier Factor Anomaly Detection Model Best Results* achieved Full F1 of 0.0852. This model finished 2nd across all anomaly detection models. The runtime was 52.12 seconds. This model was not selected for the final ensemble.

Table 5.5

K-Nearest Neighbors Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.1318	0.1378	0.1257
Full	0.0852	0.0716	0.1049

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

KNN was tuned by varying k (20, 50), percentile thresholds (98, 99), and distance metrics (minkowski, manhattan, chebyshev, euclidean). Baseline runs used residual_scaled, has_markdown, and is_holiday; later runs added cpi_scaled, unemployment_scaled, and temperature_scaled. Chebyshev with $k=20$ and extended features was tested extensively. Threshold lowering from 99th to 98th percentile aimed to improve sensitivity to subtle anomalies.

5.3.5 Local Outlier Factor Anomaly Detection Model

The following section documents the application of Local Outlier Factor anomaly detection.

Modelling Assumptions

LOF assumes that normal data points reside in high-density clusters, while anomalies appear in sparser regions. It requires scaled, numerical features, no missing values, and is sensitive to the choice of distance metric and neighborhood size.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The LOF model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in Full window were achieved with `n_neighbors = 20`, `contamination = 0.02`, `metric = minkowski`, and `novelty = false`. Final input features were `residual_scaled`, `unemployment_scaled`, and `cpi_scaled`.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.6 *Local Outlier Factor Anomaly Detection Model Best Results* achieved Full F1 of 0.0099. This model finished 12th across all anomaly detection models. The runtime was 8.90 seconds. This model was not selected for the final ensemble.

Table 5.6

Local Outlier Factor Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0044	0.0034	0.0063
Full	0.0099	0.0051	0.1594

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

LOF was tuned by varying `n_neighbors` (5, 10, 20, 30), contamination levels (0.01, 0.02), and distance metrics (minkowski, manhattan). Initial runs used core features (`residual_scaled`, `has_markdown`, `is_holiday`), followed by broader multivariate inputs including `cpi_scaled`, `unemployment_scaled`, `fuel_price_scaled`, and `temperature_scaled`.

5.3.6 HDBSCAN Anomaly Detection Model

The following section documents the application of HDBSCAN anomaly detection.

Modelling Assumptions

HDBSCAN assumes that normal data forms dense, hierarchically clusterable regions, and anomalies occur as sparse or noisy points that do not belong to any stable cluster. It requires

continuous, scaled input data and does not support missing values or categorical features. It requires no assumptions about data stationarity or distribution.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The HDBSCAN model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in Full window were achieved with `min_cluster_size = 10`, `min_samples = 1`, `metric = euclidean`, and `cluster_selection = eom`. Final input features were `residual_scaled`, `has_markdown`, `is_holiday`, `unemployment_scaled`, `cpi_scaled`, and `temperature_scaled`.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.7 *HDBSCAN Anomaly Detection Model Best Results* achieved Full F1 of 0.0280. This model finished 6th across all anomaly detection models. The runtime was 73.33 seconds. This model was selected for the final ensemble.

Table 5.7

HDBSCAN Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0238	0.0121	0.6907
Full	0.0280	0.0143	0.6900

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

Initial attempts with non-hierarchical DBSCAN on `residual_scaled` failed due to extreme runtimes and scalability issues. HDBSCAN was then tuned by varying `min_cluster_size` (10, 15, 20, 25, 30, 40), `min_samples` (1, 5, 7, 10), and cluster selection methods (`eom`, `leaf`). Early runs used core features (`residual_scaled`, `has_markdown`, `is_holiday`); later ones added economic signals (`cpi_scaled`, `unemployment_scaled`, `temperature_scaled`). Euclidean and Manhattan metrics were tested.

5.3.7 One-Class SVM Anomaly Detection Model

The following section documents the application of OCSVM anomaly detection.

Modelling Assumptions

One-Class SVM assumes that anomalies lie outside a compact, high-density region of normal data and can be separated from it by a hyperplane in a transformed feature space. It assumes the distribution of normal data is relatively stable. OCSVM assumes no missing values.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The OCSVM model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in Full window were achieved with kernel = 20, nu = 0.03, and gamma = scale. Final input features were residual_scaled, unemployment_scaled, cpi_scaled, and temperature_scaled.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.8 *OCSVM Anomaly Detection Model Best Results* achieved Full F1 of 0.0091. This model finished 13th across all anomaly detection models. The runtime was 1130.6 seconds. This model was not selected for the final ensemble.

Table 5.8

OCSVM Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0098	0.0056	0.0381
Full	0.0091	0.0052	0.0361

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

OCSVM was refined through a series of tuning experiments. Both linear and rbf kernels were tested, with nu values of 0.01 and 0.03, and gamma set to either scale or auto. Due to high computational time, initial runs used a 0.2 subsample, but were excluded from comparison. Various feature sets were tested from residual_scaled only to a multivariate set of residual_scaled, cpi_scaled, unemployment_scaled, and temperature_scaled.

5.3.8 Bayesian Change Point Detection Anomaly Detection Model

The following section documents the application of BCPD anomaly detection.

Modelling Assumptions

BCPD assumes that time series exhibit regime shifts, sustained changes in mean or trend, rather than isolated outliers. The method also assumes enough continuity and variance in the input series. It requires complete data without missing values and no categorical inputs.

Test Design

BCPD was applied independently to each Store–Dept series using the full residual sequence. The method processes data sequentially, updating change point probabilities based only on past and current observations, thereby preserving temporal causality. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in Full window were achieved with model = l2 and penalty = 7. Final input feature was only residual_scaled.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.9 *BCPD Anomaly Detection Model Best Results* achieved Full F1 of 0.0101. This model finished 11th across all anomaly detection models. The runtime was 31.5 seconds. This model was not selected for the final ensemble.

Table 5.9

BCPD Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0090	0.0060	0.0179
Full	0.0101	0.0071	0.0176

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

BCPD was implemented using the Binary Segmentation (Binseg) algorithm from the ruptures library, with the L2 cost function as the primary model. Tuning focused on the penalty parameter, which controls the sensitivity of change point detection: values of 3, 5, 7, and 10 were tested. Most runs used only residual_scaled as input, while one multivariate variant incorporated economic features (cpi_scaled, unemployment_scaled, temperature_scaled). Additional experiments explored the RBF model, but L2 consistently yielded better results.

5.3.9 Mahalanobis Distance Anomaly Detection Model

The following section documents the application of Mahalanobis distance anomaly detection.

Modelling Assumptions

This method assumes that normal points follow a multivariate distribution with measurable covariance, and that anomalies deviate significantly from the distribution center. It is sensitive to feature scaling and correlation structure.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The Mahalanobis model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in Full window were achieved with threshold = 0.99 and estimator = MinCovDet. Final input features were residual_scaled, has_markdown, and is_holiday

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.10 *Mahalanobis Distance Anomaly Detection Model Best Results* achieved Full F1 of 0.0451. This model finished 4th across all anomaly detection models. The runtime was 77.46 seconds. This model was selected for the final ensemble.

Table 5.10

Mahalanobis Distance Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0547	0.0771	0.0424
Full	0.0451	0.0495	0.0413

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

Tuning involved systematically comparing multiple covariance estimators (MinCovDet, EmpiricalCovariance, LedoitWolf, and OAS) to assess their robustness under multivariate noise. Various feature sets were tested from residual_scaled, has_markdown, is_holiday, cpi_scaled, and unemployment_scaled. Thresholds were adjusted from 0.99 to 0.995.

5.3.10 Gaussian Mixture Model Anomaly Detection Model

The following section documents the application of Gaussian Mixture Model anomaly detection.

Modelling Assumptions

GMM assumes the data is generated from a mixture of multiple Gaussian distributions, and that anomalies correspond to points with low likelihood under the fitted mixture. It requires complete and continuous input features.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The GMM model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in Full window were achieved with `n_components = 4`, `covariance = full`, and `threshold = 0.01`. Final input features were `residual_scaled`, `has_markdown`, `is_holiday`, and `cpi_scaled`.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.11 *GMM Anomaly Detection Model Best Results* achieved Full F1 of 0.0216. This model finished 7th across all anomaly detection models. The runtime was 59.92 seconds. This model was selected for the final ensemble.

Table 5.11

GMM Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0255	0.0314	0.0213
Full	0.0216	0.0238	0.0198

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

During GMM tuning, several key parameters were varied to evaluate their impact on anomaly detection performance. The number of components (`n_components`) was tested at values 3, 4, and 5 to balance model complexity and overfitting risk. Both full and diag covariance

structures were explored. A regularization term ($\text{reg_covar} = 1e-3$) was introduced to stabilize estimates in full-covariance settings. Threshold strategies included thresholds (e.g., <0.01 , <0.05) and test runs without thresholding that stored raw log-likelihood scores. Initial experiments used only `residual_scaled`, `has_markdown`, and `is_holiday` as features, while later runs expanded inputs to include `cpi_scaled`.

5.3.11 Plain Autoencoder Anomaly Detection Model

The following section documents the application of Plain Autoencoder anomaly detection.

Modelling Assumptions

The model assumes that normal patterns in the input features can be compactly reconstructed through a lower-dimensional latent space, while anomalies cause larger reconstruction errors. It requires complete data without missing values and assumes a consistent feature scale and distribution.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The Autoencoder model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in the Full window were achieved using a plain feedforward autoencoder with a symmetric architecture: two dense ReLU-activated layers in both the encoder and decoder, and a linear activation in the output layer. The model used a latent dimension = 4 and sequence length = 7. Training was performed for 10 epochs with a batch size of 64 using the Adam optimizer and mean squared error (MSE) loss. Anomalies were identified using a 90th percentile threshold on the reconstruction error. Final input features included `residual_scaled`, `has_markdown`, `is_holiday`, and `cpi_scaled`.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.12 *Plain Autoencoder Anomaly Detection Model Best Results* achieved Full F1 of 0.0128. This model finished 8th across all anomaly detection models. The runtime was 725.1 seconds. This model was not selected for the final ensemble.

Table 5.12*Plain Autoencoder Anomaly Detection Model Best Results*

Window	F1	Precision	Recall
Fair	0.0131	0.0073	0.0886
Full	0.0128	0.0069	0.0884

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

Multiple configurations were tested by varying the latent dimension (2, 4, 8), sequence length (5, 7, 14), batch size (32, 64), and number of training epochs (10, 25, 50), while consistently using MSE loss and the Adam optimizer. Input features were incrementally expanded starting from `residual_scaled` and progressively including `has_markdown`, `is_holiday`, `cpi_scaled`, `fuel_price_scaled`, and `unemployment_scaled` to evaluate multivariate sensitivity. Thresholds on reconstruction error were set using quantile-based methods, with the 90th percentile yielding the best results.

5.3.12 Variational Autoencoder Anomaly Detection Model

The following section documents the application of Variational Autoencoder anomaly detection.

Modelling Assumptions

The model assumes that normal data lies near a continuous latent distribution learned during training, and that anomalies deviate from this space and yield higher reconstruction errors. It assumes a roughly smooth underlying data distribution and complete input features without missing values.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The Variational Autoencoder model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in the Full window were achieved using a variational autoencoder with a symmetric architecture: two dense layers with ReLU (rectified linear unit) activation in both the encoder and decoder, followed by a latent sampling layer. The decoder ended with a linear output layer. The model used a latent dimension of 4. It was trained for 10 epochs with

a batch size of 64 using the Adam optimizer. The loss function combined MSE with KL (Kullback–Leibler) divergence to regularize the latent space. Anomalies were identified using a 0.95 quantile threshold on reconstruction error. Final input features included `residual_scaled`, `has_markdown`, `is_holiday`, and `cpi_scaled`.

Model Assessment

Under the best parameter settings, on Prophet decomposed residuals, results depicted in Table 5.13 *Variational Autoencoder Anomaly Detection Model Best Results* achieved Full F1 of 0.0121. This model finished 9th across all anomaly detection models. The runtime was 138.24 seconds. This model was not selected for the final ensemble.

Table 5.13

Variational Autoencoder Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0144	0.0088	0.0433
Full	0.0121	0.0073	0.0363

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

Tuning for the variational autoencoder involved systematic variation of key hyperparameters. Latent dimensionality was tested at values of 32, 16, and 4. Thresholds on reconstruction error were evaluated at both 0.99 and 0.95. The batch size was fixed at 64 and the Adam optimizer was used consistently across all runs. Input features were expanded incrementally, starting from a minimal set (`residual_scaled`, `has_markdown`, `is_holiday`) and later including `cpi_scaled`, which contributed to better anomaly detection performance.

5.3.13 LSTM Autoencoder Anomaly Detection Model

The following section documents the application of Long Short-Term Memory Autoencoder anomaly detection.

Modelling Assumptions

The model assumes that normal temporal sequences of multivariate features can be accurately reconstructed, and that anomalies produce higher reconstruction error due to temporal deviation. It assumes meaningful sequence structure and no missing values in the input data.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The LSTM Autoencoder model was refit in each fold

using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The LSTM autoencoder followed a sequence-to-sequence design, where input sequences were passed through an encoder LSTM layer with ReLU activation and transformed into a fixed-length representation. A RepeatVector layer ensured that the decoder received the appropriate sequence length. The decoder consisted of an LSTM layer followed by a TimeDistributed dense layer with a linear activation function to reconstruct the input. The model was trained using the Adam optimizer and MSE loss. Input data was processed as rolling sequences over time within each Store–Dept group. Anomalies were identified using a 0.95 quantile threshold on reconstruction error.

Model Assessment

Although extensive tuning and optimization were attempted, no LSTM Autoencoder run was able to complete on the full dataset in a comparative setting alongside other models. This was true for all decomposition sources, Prophet, STL, and TimeGPT, as each run was eventually interrupted by the kernel after tens of minutes of execution.

Model Refinement and Observations

In earlier non-comparative runs, a series of tuning experiments were conducted to identify optimal parameters for the LSTM Autoencoder. Key hyperparameters tested included sequence length (5, 10, and 20), hidden units (64 and 128), number of epochs (5, 10, and 20), and batch size (64 and 128). All runs used the Adam optimizer with MSE loss. Input features varied across configurations, including both minimal sets (residual_scaled, has_markdown, is_holiday) and extended markdown inputs (economical factors and markdown_1 to markdown_5).

5.3.14 Transformer Autoencoder Anomaly Detection Model

The following section documents the application of Transformer Autoencoder anomaly detection.

Modelling Assumptions

The model assumes that attention-based mechanisms can capture long-range temporal dependencies in multivariate sequences and reconstruct normal patterns effectively. Anomalies are expected to cause reconstruction errors due to disruption of attention weights. It assumes sequential data with consistent structure and no missing values.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The Transformer Autoencoder model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in the Full window were achieved using a transformer-based autoencoder with a single transformer encoder block. The architecture included multi-head self-attention, residual skip connections, feedforward dense layers with ReLU activation, and layer normalization. The model received static input vectors (sequence length = 1) and reconstructed them through a final dense output layer. It was trained for 10 epochs with a batch size of 64 using the Adam optimizer and MSE loss. Anomalies were identified using a 95th percentile threshold on the reconstruction error. Final input features included residual_scaled, has_markdown, is_holiday, and cpi_scaled.

Model Assessment

Under the best parameter settings, on TimeGPT decomposed residuals, results depicted in Table 5.14 *Transformer Autoencoder Anomaly Detection Model Best Results* achieved Full F1 of 0.0110. This model finished 10th across all anomaly detection models. The runtime was 778.90 seconds. This model was not selected for the final ensemble.

Table 5.14

Transformer Autoencoder Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0137	0.0075	0.0937
Full	0.0110	0.0059	0.0878

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

The Transformer Autoencoder was tuned across several dimensions. Sequence length was tested at 5, and latent dimension at 64 in earlier exploratory runs. Later configurations included 10 to 30 epochs and batch sizes of 64 and 128. Both MSE and Huber (SmoothL1) loss functions were tested with the Adam optimizer. Thresholds for anomaly detection were varied between the 90th and 95th percentiles of reconstruction error. The input features remained consistent: residual_scaled, has_markdown, is_holiday, and cpi_scaled.

5.3.15 TCN Autoencoder Anomaly Detection Model

The following section documents the application of TCN Autoencoder anomaly detection.

Modelling Assumptions

The model assumes that normal temporal patterns can be reconstructed using dilated causal convolutions, and that anomalies disrupt local or multi-scale temporal filters, leading to higher reconstruction errors. It assumes fixed-length sequences, consistent time intervals, and no missing values.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The TCN Autoencoder model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in the Full window were achieved using a TCN autoencoder with two stacked 1D convolutional layers with ReLU activation in the encoder, followed by a symmetric decoder composed of two transposed convolutional layers. The architecture operated on sequences of length 5, constructed from residual-scaled features over time. The latent representation was flattened to a 32-dimensional vector before reconstruction. The model was trained for 5 epochs with a batch size of 128 using the Adam optimizer and MSE loss. Anomalies were identified using a 99th percentile threshold on the reconstruction error. Final input features included residual_scaled, markdown_1 to markdown_5, and is_holiday.

Model Assessment

Under the best parameter settings, on TimeGPT decomposed residuals, results depicted in Table 5.15 *TCN Autoencoder Anomaly Detection Model Best Results* achieved Full F1 of 0.0082. This model finished 14th across all anomaly detection models. The runtime was 112.01 seconds. This model was not selected for the final ensemble.

Table 5.15

TCN Autoencoder Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0	0	0
Full	0.0082	0.0083	0.0081

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

The TCN Autoencoder was tuned across sequence lengths (5, 10, 14), latent dimensions (16, 32), and loss functions (Huber and MSE). Early runs tested long training configurations (30 epochs) with Huber loss, followed by aggressive reductions in epochs (down to 1) to manage runtime. Latent dimension was varied to test model capacity, and the batch size was consistently held at 128. Input features ranged from residual-only to extended multivariate inputs including markdowns and holidays. Features ranged from `residual_scaled` to the full feature set including `markdown_1–5`.

5.3.16 Deep SVDD Anomaly Detection Model

The following section documents the application of Deep SVDD Autoencoder anomaly detection.

Modelling Assumptions

The method assumes that normal data points cluster tightly in a learned feature space, and that anomalies lie further from this compact hypersphere. It assumes access to mostly normal training data, consistent input scale, and no missing values. It makes no explicit distributional assumptions but relies on structural compactness.

Test Design

Model was applied independently to each Store–Dept series. A 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The Deep SVDD model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies in both the Fair and Full windows using precision, recall, and F1 score. This approach ensures robust validation without look-ahead, aligning with the temporal structure of the data.

Parameter Setting

The best F1 results in the Full window were achieved using a Deep SVDD model with a feedforward encoder architecture composed of three dense layers: 64, 32, and 8 neurons respectively, all with ReLU activation except for the final latent layer. The model operated on flattened sequences of 7 time steps across 6 input features, resulting in input vectors of length 42. After initializing a hypersphere center from the latent embeddings, the model was trained for 10 epochs with a batch size of 64 using the Adam optimizer. The custom loss minimized the distance between predictions and the hypersphere center, encouraging compact representations. Anomalies were identified using a 95th percentile threshold on the squared distance from the center. Final input features included `residual_scaled`, `has_markdown`, `is_holiday`, `cpi_scaled`, `fuel_price_scaled`, and `unemployment_scaled`.

Model Assessment

Under the best parameter settings, on TimeGPT decomposed residuals, results depicted in Table 5.16 *Deep SVDD Anomaly Detection Model Best Results* achieved Full F1 of 0.0071. This model finished 15th across all anomaly detection models. The runtime was 631.71 seconds. This model was not selected for the final ensemble.

Table 5.16

Deep SVDD Anomaly Detection Model Best Results

Window	F1	Precision	Recall
Fair	0.0084	0.0044	0.091
Full	0.0071	0.0043	0.0216

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

Tuning explored multiple aspects: the anomaly threshold was varied between the 90th and 95th percentile; training duration was extended from 10 to 30 epochs; dropout was added to reduce overfitting; and the model center was recompiled after training to improve scoring stability. All runs used a fixed encoder structure ($64 \rightarrow 32 \rightarrow 8$) over flattened temporal windows and a consistent batch size of 64.

5.4 Ensemble Anomaly Detection Methods

To overcome the limitations of individual models this section evaluates ensemble strategies, ranging from simple logic-based unions to a supervised meta-classifier, designed to combine the strengths of multiple methods and deliver more robust anomaly detection in a complex retail setting. Anomaly detection models summarized in table 5.17 *Best Performing Anomaly Detection Models* were chosen based on Full window F1 and are used in following ensembles.

Table 5.17

Best Performing Anomaly Detection Models

Model	F1	Precision	Recall
KNN	0.0852	0.0716	0.1049
Threshold	0.0652	0.0339	0.8497
Mahalanobis	0.0451	0.0495	0.0413
Isolation Forest	0.0357	0.0325	0.0397
HDBSCAN	0.0280	0.0143	0.6900
GMM	0.0216	0.0238	0.0198

GMM = Gaussian Mixture Model, HDBSCAN = Hierarchical Density-Based Spatial Clustering of Applications with Noise, KNN = K-Nearest Neighbors.

Note: Table created by the author based on the created dataset.

The best results in ensemble anomaly detection were achieved by first combining high-recall models to maximize detection coverage, followed by a tailored “business shave” step that filters results using domain-informed criteria to boost precision — yielding a final detection set that captures both injected anomalies (true positives) and plausible real-world anomalies, aligned with business relevance (false positives).

5.4.1 Soft Union Ensemble Strategy

The following section presents the Soft Union strategy, which combines the outputs of multiple anomaly detection models using a logical OR operation to maximize recall.

Modelling Assumptions

It was assumed that different anomaly detection models would capture different aspects of the data structure, and that their combination would minimize false negatives. All models were applied independently before aggregation.

Test Design

Model results are evaluated against injected anomalies only in Full window using precision, recall, and F1 score.

Parameter Setting

The best recall was achieved with all six models unified using simple OR logical parameter.

Model Assessment

As depicted in Table 5.18 *Soft Union Ensemble Runs Performance*, the best recall of 0.9355 was achieved by unioning all six models. The runtime was only 0.2 seconds. This method is further combined with the “business shave” strategy for precision and business relevance.

Table 5.18

Soft Union Ensemble Runs Performance

Ensemble	Setting	F1	Precision	Recall	Time
Soft Union 1	KNN	0.0650	0.0338	0.8550	0.1s
	Threshold				
	Mahalanobis				
Soft Union 2	KNN	0.0307	0.0156	0.9355	0.2s
	Threshold				
	Mahalanobis				
	Isolation Forest				
	HDBSCAN				
	GMM				

GMM = Gaussian Mixture Model, HDBSCAN = Hierarchical Density-Based Spatial Clustering of Applications with Noise, KNN = K-Nearest Neighbors.
Note: Table created by the author based on the created dataset.

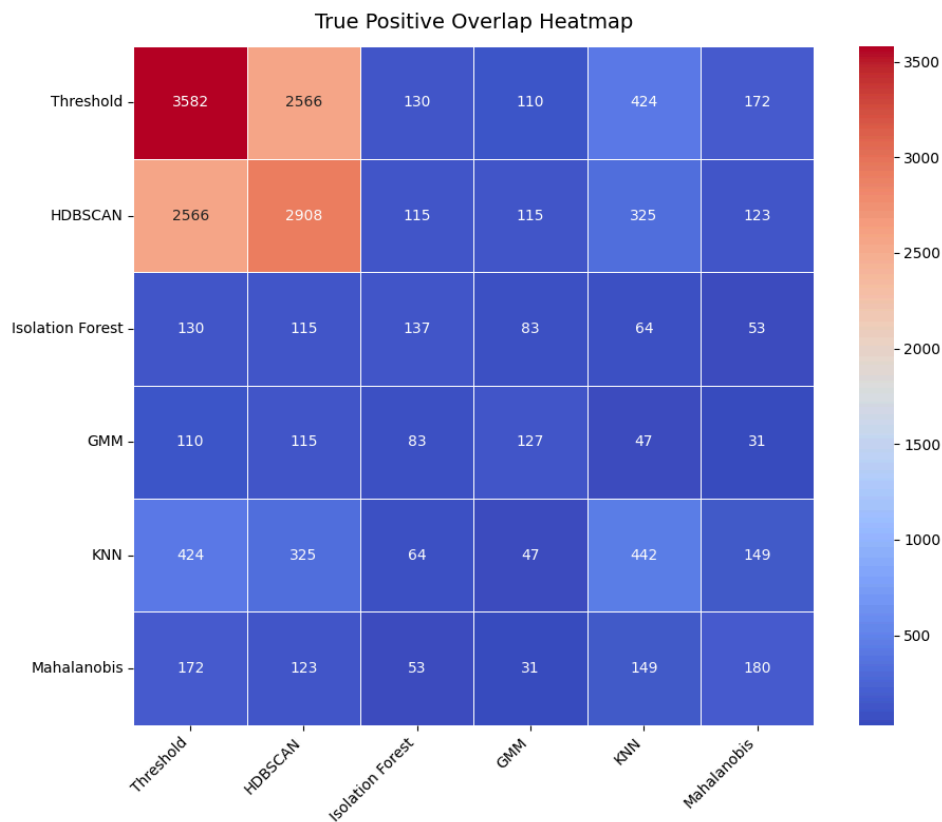
Model Refinement and Observations

Two settings were tested — three best performing and all six best performing anomaly detection models. Naturally, combining all models yielded the highest recall.

To analyze how various models contribute to detecting injected anomalies, Figure 5.6 *True Positives Overlap Heatmap* was constructed. It clearly shows that Thresholding and HDBSCAN, the two models with the highest recall, identify the largest share of true positives.

Figure 5.6

True Positives Overlap Heatmap



Note: Figure created by the author based on the created dataset.

As shown in Table 5.19 *Unique True Positives per Model*, it may be tempting to discard models like GMM, KNN, Mahalanobis, and especially Isolation Forest due to their lower standalone contribution. However, their inclusion is justified by their minimal runtimes and their complementary detection mechanisms. While they may not have captured many anomalies in this dataset, their unique strengths could prove valuable under different anomaly types or data distributions, preserving ensemble robustness.

Table 5.19*Unique True Positives per Model*

Model	Unique TP
Threshold	903
HDBSCAN	321
KNN	6
GMM	5
Mahalanobis	2
Isolation Forest	0

GMM = Gaussian Mixture Model, HDBSCAN = Hierarchical Density-Based Spatial Clustering of Applications with Noise, KNN = K-Nearest Neighbors, TP = True Positives.

Note: Table created by the author based on the created dataset.

5.4.2 Weighted Ensemble Strategy

The following section presents the Weighted Voting ensemble, which aggregates the outputs of multiple anomaly detection models using weighted contributions and a voting threshold to balance detection coverage and precision.

Modelling Assumptions

It was assumed that different anomaly detection models would capture different aspects of the data structure, and that their combination would minimize false negatives. All models were applied independently before aggregation.

Test Design

Model results are evaluated against injected anomalies only in Full window using precision, recall, and F1 score.

Parameter Setting

The highest recall was achieved by combining all six candidate anomaly detection models in a voting ensemble, where each model contributed one vote and anomalies were flagged when at least two models agreed.

Model Assessment

As shown in Table 5.20 *Weighted Vote Ensemble Runs Performance*, the best recall of 0.6417 was achieved with a runtime of just 0.1 seconds. While this recall does not outperform the best soft union run, it offers a strong alternative in scenarios where business constraints require at least two models to agree on an anomaly candidate before applying the business shave.

Table 5.20*Weighted Vote Ensemble Runs Performance*

Ensemble	Setting	Weights	Threshold	F1	Precision	Recall	Time
Weighted Ensemble 1	KNN	3	4	0.0901	0.0809	0.1018	0.1s
	Threshold	2					
	Mahalanobis	1					
Weighted Ensemble 2	KNN	3	3	0.0932	0.0741	0.1255	0.1s
	Threshold	2					
	Mahalanobis	1					
	Isolation Forest	1					
Weighted Ensemble 3	HDBSCAN	1	3	0.0894	0.0746	0.1115	0.1s
	Thresholding	1					
	Isolation Forest	1					
	GMM	1					
	KNN	1					
	Mahalanobis	1					
Weighted Ensemble 4	HDBSCAN	1	2	0.0810	0.0432	0.6417	0.1s
	Thresholding	1					
	Isolation Forest	1					
	GMM	1					
	KNN	1					
	Mahalanobis	1					

GMM = Gaussian Mixture Model, HDBSCAN = Hierarchical Density-Based Spatial Clustering of Applications with Noise, KNN = K-Nearest Neighbors.

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

The weighted vote ensemble was tuned by adjusting both the weights assigned to each individual detector and the voting threshold required to classify an observation as anomalous. Initial runs tested unweighted combinations of 3 to 6 models, including KNN, Thresholding, Mahalanobis, Isolation Forest, GMM, and HDBSCAN. Subsequent experiments explored strategic weighting (e.g., emphasizing KNN or Thresholding) and reduced the consensus threshold from 4 to 2. This systematic tuning revealed that recall was maximized when all six models were weighted equally and a low threshold of 2 was used, allowing more potential anomalies to pass through.

5.4.3 LightGBM Meta-Classifier

The following section presents the LightGBM meta-classifier, a supervised stacked ensemble designed to integrate outputs from multiple unsupervised anomaly detectors alongside key contextual features.

Modelling Assumptions

It was assumed that different anomaly detection models would capture different aspects of the data structure, and that their combination would minimize false negatives. All models were applied independently before aggregation. Anomaly class is well labeled. No missing values are preferred.

Test Design

Applied globally, a 5-fold TimeSeriesSplit was applied across the full dataset, where each fold trains on past observations and tests on future data, preserving temporal causality. The meta-classifier model was refit in each fold using only past data to predict on unseen future windows. Model results are evaluated against injected anomalies only in Full window using precision, recall, and F1 score.

Parameter Setting

The highest recall was achieved by configuration which used all six base model outputs alongside contextual features and residual_scaled, and applied a probability threshold of 0.2 with balanced class weighting to maximize sensitivity.

Model Assessment

As shown in Table 5.21 *Meta-Classifier Runs Performance*, the best recall of 0.7119 was achieved with a runtime of 38.0 seconds. While this recall does not outperform the top soft union run, it offers a strong alternative in business scenarios where labeled anomalies are available and a supervised learning approach is preferred.

Table 5.21

Meta-Classifier Runs Performance

Ensemble	Setting	Predict Proba	Class Weighting	F1	Precision	Recall	Time
Meta-Classifier 1	residual_scaled is_holiday has_markdown HDBSCAN Thresholding Isolation Forest GMM KNN Mahalanobis	≥ 0.3	balanced	0.1450	0.0811	0.6844	38.0s
Meta-Classifier 2	residual_scaled is_holiday has_markdown HDBSCAN Thresholding Isolation Forest GMM KNN Mahalanobis	≥ 0.4	balanced	0.1596	0.0909	0.6545	38.0s
Meta-Classifier 3	residual_scaled is_holiday has_markdown HDBSCAN Thresholding Isolation Forest GMM KNN Mahalanobis	≥ 0.2	balanced	0.1264	0.0693	0.7119	38.0s

GMM = Gaussian Mixture Model, HDBSCAN = Hierarchical Density-Based Spatial Clustering of Applications with Noise, KNN = K-Nearest Neighbors.

Note: Table created by the author based on the created dataset.

Model Refinement and Observations

Tuning focused on varying the prediction threshold (0.2–0.4) and applying `class_weight='balanced'` to handle label imbalance. Earlier runs tested different class weightings, thresholds, and additional feature combinations, and insights from those were used to select the final settings shown here, balancing recall and precision through threshold adjustment.

5.4.4 Business Logic Post-Processing

In this final modeling step, additional business-based post-processing is applied to refine anomaly detection outputs using realistic rules, ensuring greater precision without sacrificing recall of real anomalies.

Modelling Assumptions

The business logic post-processing step assumes that true business-relevant anomalies fall into two interpretable categories: unexpected spikes (high residuals without holidays or markdowns) and unexpected drops (low residuals during weeks with holidays or markdowns). It further assumes that a minimum sales impact is required for an anomaly to be considered meaningful, introducing a domain-specific relevance threshold. This approach preserves temporal causality by avoiding any lookahead operations — all decisions are made using information available at the current or prior time points only.

Test Design

Model results are evaluated against injected anomalies only in Full window using precision, recall, and F1 score. Furthermore, detailed visual and statistical analysis was performed to assess the results.

Parameter Setting

The best F1 was achieved with a financial threshold set at \$5,000 and consistent rules to determine spikes (occurring at no holiday and no markdown week) and drops (occurring when either holiday or markdown event is present).

Model Assessment

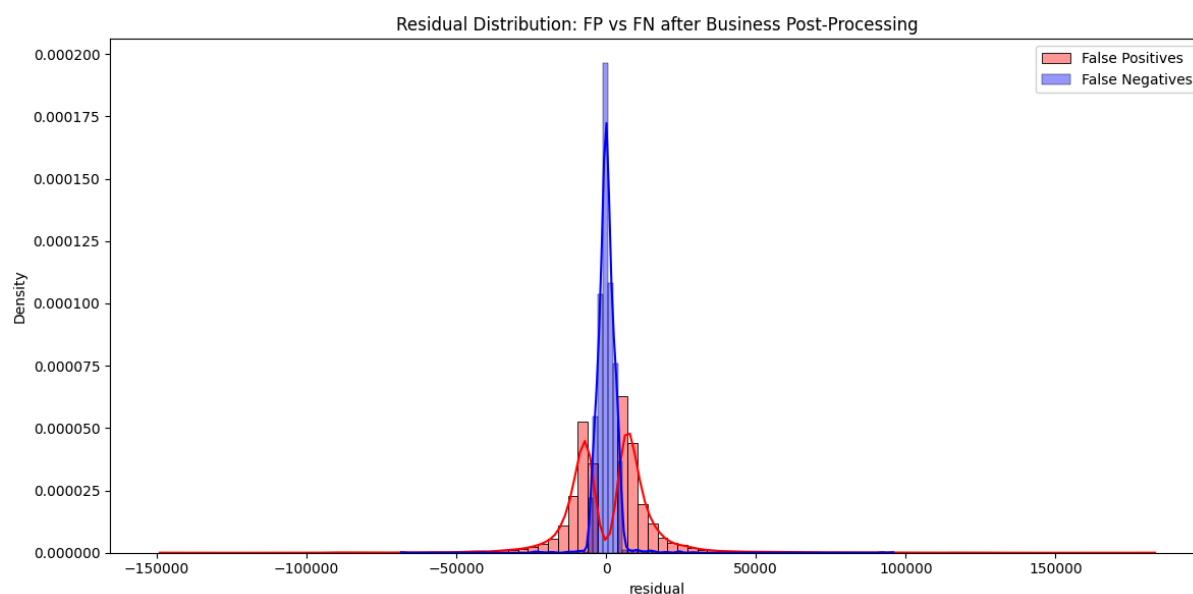
As shown in Table 5.22 *Business Logic Post-Processing Runs Performance*, the best F1 of 0.2011 was achieved with a runtime of 0.1s seconds. The threshold of \$5,000 was selected for the final model.

Table 5.22*Business Logic Post-Processing Runs Performance*

Ensemble	Threshold	F1	Precision	Recall	TP	FP	FN	TN	Time
Business Logic 1	\$5,000	0.2011	0.1402	0.3557	1,499	9,195	2,715	408,161	0.1s
Business Logic 2	\$10,000	0.1929	0.1970	0.1889	3,086	43,797	1,128	373,559	0.1s
Business Logic 3	\$1,000	0.1208	0.0658	0.7323	796	3,245	3,418	414,111	0.1s

TP = True Positives, FP = False Positives, FN = False Negatives, TN = True Negatives.
 Note: Table created by the author based on the created dataset.

Figure 5.7 *Residual Distribution: FP vs FN after Business Post-Processing* visualizes the distributions of residual values for false positives and false negatives using kernel density estimation. The y-axis labeled “Density” shows a probability density function normalized such that the area under each curve equals one. This allows for fair comparison of the shape and spread of the two distributions. The broader, heavy-tailed shape of false positives indicates that these model-flagged anomalies, while not part of the injected set, often involved large deviations from the forecast and may represent real, high-impact anomalies within the dataset. In contrast, false negatives are sharply concentrated around zero, suggesting they were minor deviations unlikely to trigger concern in practice. This supports the rationale behind applying a \$5,000 residual threshold in the business logic step, which was designed to prioritize large, actionable anomalies over small injected ones that may not hold practical significance.

Figure 5.7*Residual Distribution: FP vs FN after Business Post-Processing*

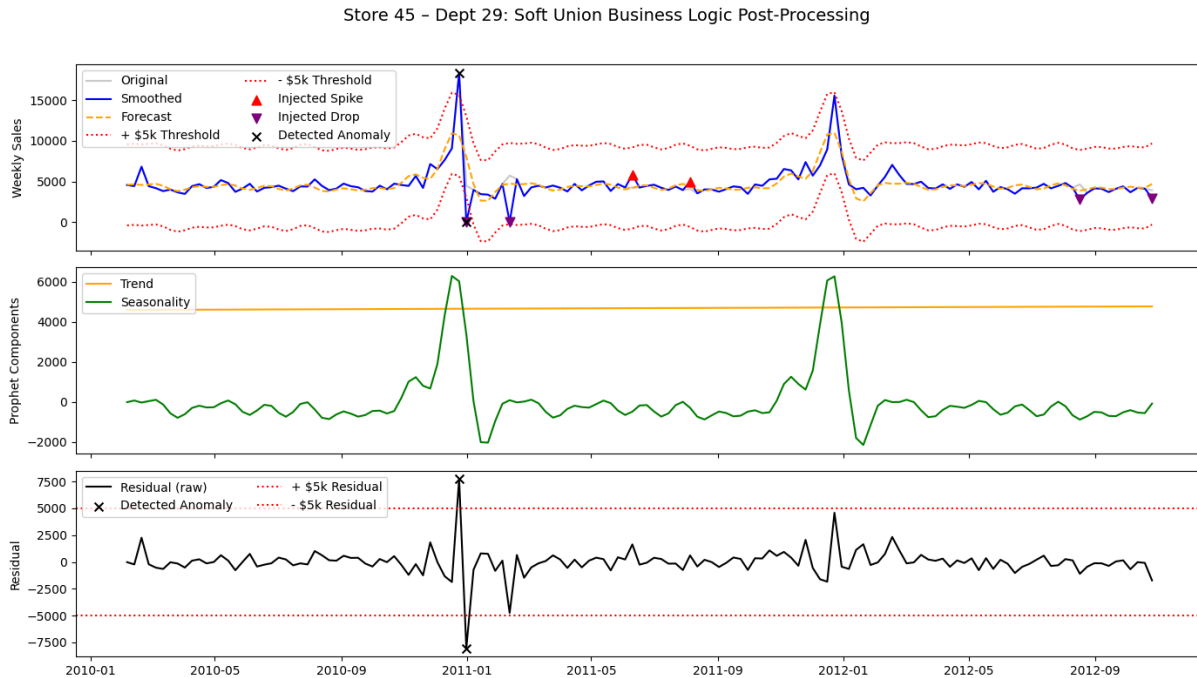
FP = False Positives, FN = False Negatives,
 Note: Figure created by the author based on the created dataset.

Visual inspection in Figure 5.8 *Store 45, Dept 29: Soft Union Business Logic Post-Processing* confirms that detected meaningful anomalies occur outside the $\pm\$5,000$

residual threshold, indicating that the method effectively captures meaningful deviations beyond expected variability.

Figure 5.8

Store 45 – Dept 29: Soft Union Business Logic Post-Processing



Note: Figure created by the author based on the created dataset.

Model Refinement and Observations

In earlier iterations of the pipeline, both STD and ECDF thresholds were explored as candidates for the precision-enhancing “business shave” step. While these methods can be effective in retrospective anomaly scoring, they traditionally rely on global statistics computed over the full time series, rather than on data available up to the current timestep. Since both STD and ECDF inherently depend on the entire dataset to establish thresholds, they were deemed unsuitable for deployment-oriented detection. Instead, a domain-driven thresholding strategy was adopted, using fixed dollar-value cutoffs for weekly sales anomalies. Although the thesis performs retrospective evaluation, the detection logic was designed to be compatible with online deployment constraints. Three thresholds were tested: \$10,000, \$5,000, and \$1,000.

Moreover, the “business shave” step explicitly uses the thesis’s original definitions of unexpected spikes and unexpected drops, as these represent the only types of anomalies relevant to the targeted business use case. Spikes are defined as unusually high sales occurring in the absence of holidays or markdowns, while drops are defined as unusually low sales occurring during weeks that feature either a holiday or a markdown. These definitions align the anomaly detection process with real-world retail dynamics and ensure the filtered results are business-meaningful.

5.5 Reconciliation Model

In order to escalate anomalies from department level to store and national levels, a hierarchical reconciliation strategy was designed and tuned. This step ensures that local department anomalies can be meaningfully interpreted at broader business levels while maintaining focus on severe, business-relevant deviations.

Anomalies confirmed at the department level were aggregated upward using a bottom-up, soft-voting reconciliation strategy. At each higher level, the proportion of underlying anomalies of each type (spikes and drops) was computed for every week. If at least a fixed percentage of underlying entities (departments within a store, or stores within the nation) exhibited spike-like or drop-like anomalies in the same week, the parent level was flagged accordingly. At the national level, if both spike and drop criteria were met in a given week, the signal was marked as a conflict, indicating ambiguity in the aggregate anomaly direction.

Modelling Assumptions

It is assumed that department-level anomalies detected after soft union and business logic post-processing represent a high-quality filtered signal. The model also assumes that sales are additive across levels.

Test Design

The reconciliation model is evaluated through both the number of anomalies propagated to higher hierarchical levels and visual inspection of results.

Parameter Setting

The best results were achieved at reconciliation threshold of 0.05.

Model Assessment

The final reconciliation threshold of 5% was chosen as it maintained strong national-level signal retention (45 spikes, 38 drops) while substantially reducing the volume of store-level anomalies compared to more permissive thresholds (as shown in Figure 5.23 *Reconciliation Runs Results*), achieving balance between coverage and business interpretability.

Table 5.23

Reconciliation Runs Results

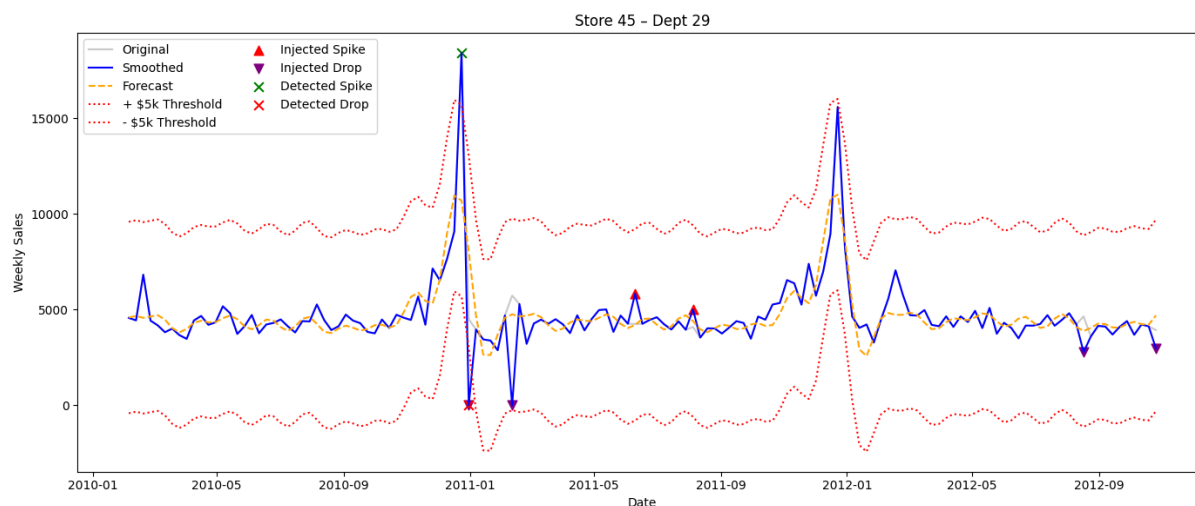
Threshold	Dept Spikes	Dept Drops	Store Spikes	Store Drops	Store Conflicts	National Spikes	National Drops	National Conflicts
0.01	5,637	5,057	2,065	1,574	0	86	57	0
0.05	5,637	5,057	435	407	0	45	38	0
0.1	5,637	5,057	123	125	0	6	5	0

Note: Table created by the author based on the created dataset.

Figure 5.9 *Dept 29 – Store 45* confirms that the post-processed soft union method detects anomalies with business-relevant sales beyond the set threshold.

Figure 5.9

Store 45 – Dept 29

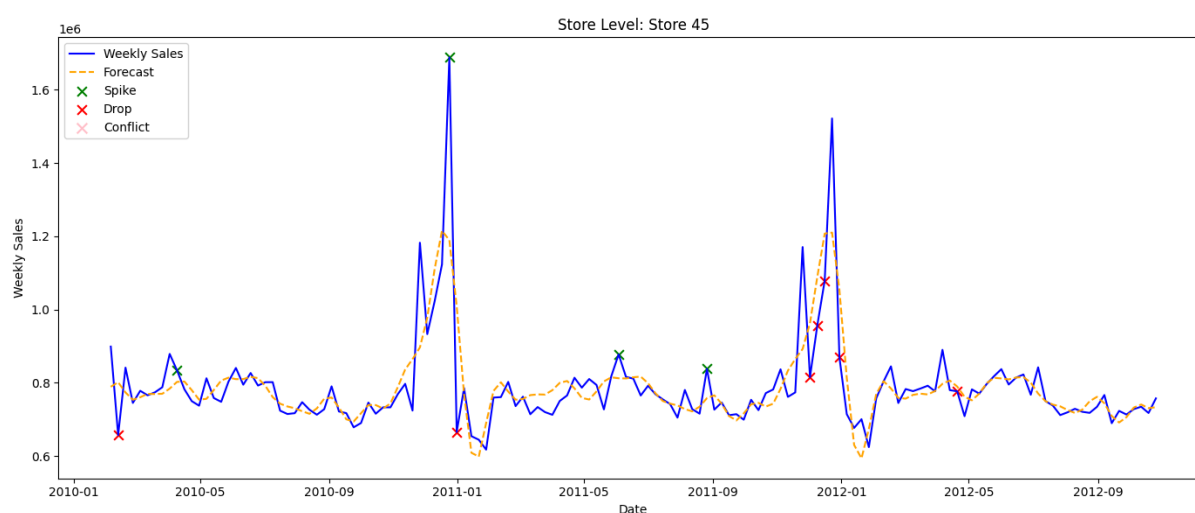


Note: Figure created by the author based on the created dataset.

In Figure 5.10 *Store Level: Store 45*, aggregated detections clearly surface identified anomalies from the lower department level.

Figure 5.10

Store Level: Store 45

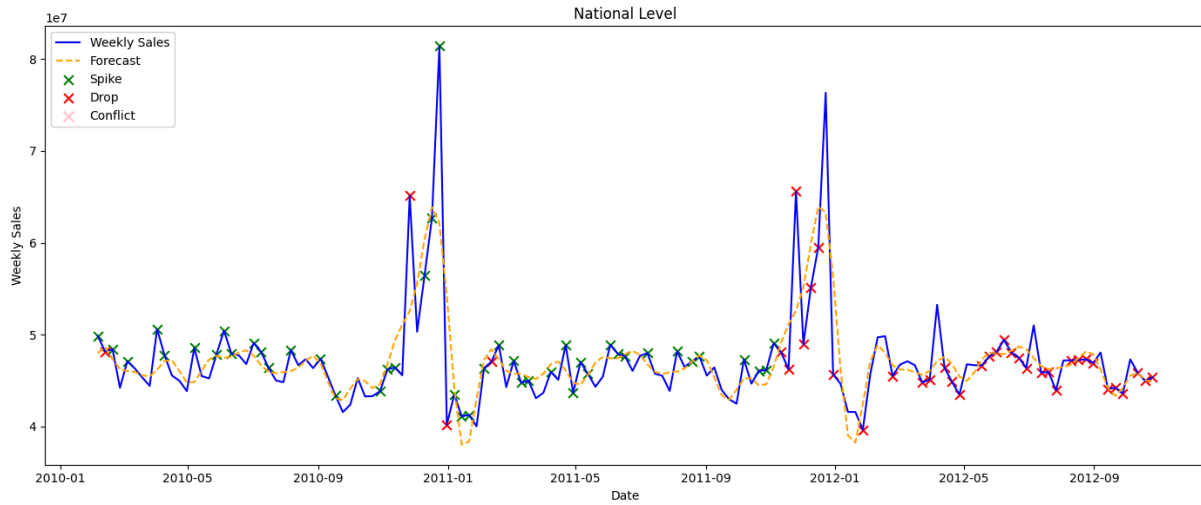


Note: Figure created by the author based on the created dataset.

Finally, Figure 5.11 *National Level* demonstrates that the bottom-up reconciliation strategy surfaces only the most consistent and impactful patterns at the national level — yielding clear, interpretable anomalies aligned with macro-level deviations in weekly sales.

Figure 5.11

National Level



Note: Figure created by the author based on the created dataset.

Together, these plots validate the robustness, business alignment, and hierarchical consistency of the entire detection pipeline.

Model Refinement and Observations

The reconciliation threshold was empirically tuned to balance anomaly coverage with signal quality across hierarchical levels. As shown in Table 5.23 *Reconciliation Runs Results*, lowering the threshold to 0.01 substantially increased the number of spikes and drops detected at the store and national levels but risked introducing noise. Conversely, higher threshold 0.1 significantly reduced detection rates, potentially overlooking meaningful patterns. A threshold of 0.05 was selected as the optimal compromise, offering sufficient national-level signals without overwhelming volume. For simplification and easier business understanding, the same threshold was eventually kept at both department and store level.

5.6 Interpretability Model

To improve interpretability of individual anomaly detection models and assess feature influence across the pipeline, SHAP (SHapley Additive exPlanations) values were computed using a surrogate modeling strategy. Since many of the detection algorithms are not inherently explainable, a standard gradient boosting classifier was trained post-hoc for each method to approximate its prediction logic. SHAP-like values were then computed on this surrogate to estimate feature contributions.

To analyze feature importance systematically, a two-step aggregation process was applied:

1. Per-model SHAP-like surrogate aggregation: For each model, the absolute mean SHAP-like values were computed separately for detected anomalies and non-anomalies.
2. Global aggregation across models: To identify which base features contributed most consistently across all models, contributions were grouped by feature. Contributions were summed across models, yielding a global ranking of feature importance.

Table 5.24 *Aggregated Department Level SHAP-like Contributions by Feature* summarizes the relative importance of input features in differentiating anomalies from normal points across the entire pipeline. The column Mean Absolute Contributor to Anomaly captures the average absolute SHAP-like value for each feature on records flagged as anomalies, indicating how strongly a feature contributed to the model's decision in those cases. Conversely, Mean Absolute Contributor to Normal reflects the same metric on normal (non-anomalous) points. Contributor to Anomaly Store shows the signed difference between the two values and reveals the direction and magnitude of the feature's overall shift toward anomaly detection. As expected, `residual_scaled` emerged as the most influential variable. Features like `has_markdown`, `is_holiday`, and `cpi_scaled` had moderate contribution scores to anomalous points, however, when compared with contribution to normal points, their impact was negative.

Table 5.24

Aggregated Department Level SHAP-like Contributions by Feature

Feature	Mean Absolute Contributor to Anomaly	Mean Absolute Contributor to Normal	Contributor to Anomaly Store
<code>residual_scaled</code>	3.633311	3.292823	0.340488
<code>temperature_scaled</code>	0.149606	0.165134	-0.015528
<code>is_holiday</code>	0.649432	0.668787	-0.019355
<code>unemployment_scaled</code>	0.275200	0.339871	-0.064671
<code>has_markdown</code>	0.768487	0.859387	-0.090901
<code>cpi_scaled</code>	0.523950	0.673979	-0.150028

Note: Table created by the author based on the created dataset.

This pipeline-level interpretability approach provided consistent insights across models, helping to demystify otherwise unclear detection logic. The resulting SHAP-like explanations can support business-facing anomaly summaries, improving user trust and facilitating informed decision-making.

5.7 Chapter Summary

First, across the three tested decomposition methods, Prophet consistently delivered the best anomaly detection results, with the highest F1 scores achieved across nearly all models. At the same time, runtime was very reasonable at 8 minutes. TimeGPT, although showing better results in two deep learning models, overall underperformed. This decomposition also ran for the longest time — over 30 minutes. STL decomposition was extremely quick, around 2 minutes, but suffered greatly in quality due to incomplete years and its produced residuals underperformed in all models. As a result, Prophet decomposition was selected as the default base for downstream modeling and evaluation.

Second, out of 16 tested anomaly detection models, only six models were selected for ensemble construction based on their F1 scores exceeding 0.0200 and reasonable computational times: KNN, Thresholding, Mahalanobis, Isolation Forest, HDBSCAN, and GMM. Together, these models cover distance-based, density-based, probabilistic, statistical, and rule-based methods, ensuring diverse perspectives on anomaly formation and minimizing shared blind spots. Notably, none of the deep learning models tested surpassed these classical methods in terms of achieved accuracy or runtime.

Third, three ensemble strategies were developed to combine the strengths of the base detectors: a soft union, a weighted voting ensemble, and a LightGBM meta-classifier. The soft union achieved the highest recall (0.9355) by flagging any data point detected by at least one of the six models, offering a simple and fast method for maximum anomaly coverage. This approach was chosen for the final pipeline and further post-processing step. The weighted voting ensemble enforced a stricter condition, requiring at least two models to agree, yielding lower recall (0.6417) but offering an alternative to business setting with needs for wider consensus before surfacing anomalies. The supervised LightGBM meta-classifier offered a third alternative, achieving a recall of 0.7119 using labeled injected anomalies. Though computationally more intensive, it provides a learnable, tunable solution where labeled data is available. Each ensemble serves a distinct role: soft union maximizes recall, weighted vote enforces cross-model agreement, and the meta-classifier tailors detection to hypothetical supervised settings.

Fourth, to increase business relevance and improve precision, a final post-processing step, referred to as the “business shave”, was introduced. This stage filters anomalies based on a fixed magnitude threshold of \$5,000 and contextual rules aligned with retail expectations. Specifically, spikes are retained only if they occur during non-holiday, non-markdown periods, while drops are kept only if they coincide with holidays or markdowns. The \$5,000 residual threshold ensures only substantial deviations are flagged, preventing minor fluctuations from cluttering the anomaly set, while maximizing interpretability of flagged anomalies.

Fifth, following post-processing, anomalies were reconciled bottom-up to the store and national levels using a simple, interpretable aggregation strategy. At the store level, a spike (or drop) was confirmed if at least 5% of departments within that store exhibited a spike (or drop) in the same week after passing the business logic filters. At the national level, the same rule was applied: a week was flagged nationally if $\geq 5\%$ of stores had a confirmed store-level

anomaly of the same type. This approach ensures that only coherent, distributed patterns are surfaced at higher levels, avoiding false alarms from isolated department-level signals. The 5% threshold was applied consistently across levels to simplify business communication.

Lastly, interpretability was integrated directly into the anomaly detection pipeline using surrogate models to provide SHAP-like explanations for each method. For every anomaly detection model, a gradient boosting classifier was trained post hoc to mimic the model's outputs, enabling SHAP-like contribution value computation over the original input features. These values were saved during detection and later aggregated to produce global interpretability insights.

Throughout the entire pipeline, temporality is strictly preserved, with all models, evaluations, and post-processing steps using only past or current information at each time point — ensuring full compatibility with real-time deployment scenarios.

6 Business Use

This chapter outlines how the developed anomaly detection pipeline can be applied in a real business context, focusing on generating managerial insights and outlining key considerations for deployment.

6.1 Managerial Summaries and Operational Insights

The anomaly detection pipeline developed in this thesis transforms raw weekly retail data into actionable business information. Its primary business value lies in supporting higher-level decision-making and operational oversight. By surfacing statistically and contextually significant anomalies, the system enables retail managers to track emerging issues or unexpected trends and take appropriate action. This data-to-insight transformation aligns directly with the strategic goals of modern retail management: early awareness, operational agility, and data-driven response.

Given the weekly cadence of the data, it is proposed that every Monday morning, each store manager receives an automatically generated summary of the past week's performance and anomalies. These summaries are designed to be concise, human-readable narratives that highlight unexpected spikes or drops in performance. For demonstration, this thesis used OpenAI API to generate such summaries using only data created by the pipeline. An example is provided in Figure 6.1 *Managerial Summary: Store 45, Week 2011-12-02*, illustrating how such communication could look in practice.

Figure 6.1

Managerial Summary: Store 45, Week 2011-12-02



Retail Ops Assistant 8:00 AM

Today ▾

Happy Monday!

Sales Performance Overview:

Store 45 recorded sales of \$815,209 for the week ending 2011-12-02, falling short of the forecasted \$962,597 by \$147,388, a deviation of -15.31%. On a national level, sales reached \$48,927,613 against a forecast of \$55,154,004, marking a deviation of -11.29%. Store 45 contributed 1.67% to the national sales.

Anomaly Report:

During the week, Store 45 experienced no spikes in sales but recorded 1 drop. On a national scale, 1 drop was recorded. The anomaly pattern at Store 45 mirrored the national trend.

Departmental Report:

Department 22 had no sales recorded against a forecast of \$8,349. This was primarily due to unexpected sales behavior and heavy discounting activity. Department 72 recorded sales of \$46,965, falling short of the forecasted \$101,245 by 53.61%. This deviation was mainly due to unexpected sales behavior and heavy discounting activity. Department 18 also recorded no sales against a forecast of \$30,325, primarily due to unexpected sales behavior and heavy discounting activity.

Impact:

- Markdown activity was present in Store 45 during the week.
- There were no holidays during the week.
- The average temperature for the week was a cool 50.2°F. There were no extreme or notable weather conditions.

For details and drill-down follow [this link to your dashboard](#).

Note: Figure created by the author based on the created dataset.

To complement the summaries and build managerial trust, interactive dashboards, although out of scope in this thesis, would be expected to be built. These dashboards allow business users to drill down from national and store-level anomalies, explore affected departments, and inspect detailed metadata for each detected deviation. This combination of proactive reporting and transparent visual exploration ensures the system serves not just as an alerting tool but as a practical assistant to human decision-makers.

Depending on the managerial level and organizational hierarchy, this reporting framework can be tailored to serve not only individual store managers, but also department-level supervisors, regional directors, national operations leaders, and even financial controllers monitoring sales performance across the chain.

6.2 Deployment

While the pipeline developed in this thesis is ready to detect business-meaningful anomalies using only unsupervised models, deploying it in practice requires important adaptations. First, the injection logic used for benchmarking must be removed, preserving only the detection components. Since all final models are unsupervised, this transition is feasible without retraining or requiring labeled data.

If the company has access to daily-level data or lower hierarchies (e.g., product or SKU level), the pipeline can be extended to that granularity. Doing so would yield even more actionable and localized insights, helping managers intervene with higher precision.

To support production use, the pipeline should be redesigned for incremental operation — rather than reprocessing all historical data weekly, it should append new weekly observations to the existing output. This would improve runtime efficiency and align more closely with real-time decision-making.

In addition, a deployment-ready version should also include automated weekly summary generation, and dashboard refresh triggers. Ideally, results should be written to a centralized database accessible to analysts and managers, while summaries and alerts could be distributed via scheduled emails or messaging integrations, as outlined in Figure 6.1 Managerial Summary: Store 45, Week 2011-12-02.

Lastly, monitoring logic should be implemented to detect pipeline failures, data quality issues (e.g., missing values, data delays), or model drift. Although the models are unsupervised, recurring anomalies over time may indicate changing behavior patterns. This could suggest the future need for either retraining or integrating limited supervised learning if labeled anomalies become available.

7 Evaluation

In this Evaluation chapter, first, the results of the data mining process. Then, the approved models are revisited and summarized. Finally, the overall process is reviewed through the lens of the CRISP-DM framework.

7.1 Assessment of Data Mining Results

The data mining results are evaluated against the goals defined in the Introduction chapter. Specifically, the models are assessed in terms of accuracy, scalability, interpretability, and explainability.

7.1.1 Results in Terms of Accuracy

Each model in the pipeline was validated against injected anomalies on the department level using standard metrics: true and false positives and negatives, precision, recall, and F1 score. Each detection method and decomposition strategy was evaluated using both a Fair window (aligned timeframes of available quality residuals across decompositions) and a Full window (entire timeline). Final decisions were based solely on the Full window. In later stages, only Full window evaluation was used. Where relevant, additional statistical and visual analyses supported the results.

A key design choice was to prioritize recall in early stages, accepting false positives as a strength rather than a flaw. This is because many anomalies labeled as false positives could in fact be meaningful deviations in the original data. To first maximize recall, an ensemble was constructed as a soft union of the best-performing individual models based on Full window F1 scores. It combined diverse methods: statistical (Thresholding), classical unsupervised (Isolation Forest, KNN, HDBSCAN), and probabilistic (GMM, Mahalanobis). To improve precision and reduce noise from irrelevant anomalies, a post-processing step called the "business shave" was applied. It retained only anomalies with an absolute residual above \$5,000. In addition, spikes were kept only if no holiday or markdown occurred, while drops were kept only if a holiday or markdown was present. This ensured that final anomalies were not only statistically significant but also aligned with real-world business relevance.

To ensure the approach is not biased toward the single anomaly injection set used throughout the thesis, additional reruns were conducted with varied configurations, as shown in Table 7.1 *Pipeline Evaluation Across Varying Anomaly Injection Configurations*. The first columns describe the injection settings. The Soft Union columns report performance immediately after the ensemble combination step (maximizing recall), while the Business Shave columns reflect results after applying the post-processing rules (maximizing precision). As expected, lowering the standard deviation of injected anomalies made detection harder, with a final Business Shave F1 of only 0.1091. In contrast, increasing the standard deviation improved

detectability, yielding a Business Shave F1 of 0.3003. Varying the percentage of injected anomalies had a similar, though less extreme, effect — injecting only 0.5% resulted in an F1 of 0.1234, while 2% led to an F1 of 0.2817. Changing the random seed (42 vs. 17) produced nearly identical results, confirming consistency. Modifying the rolling window size had minimal impact — shorter windows resulted in an F1 of 0.2012 (just 0.0001 above baseline), and longer windows reached 0.2039. Overall, pipeline runtime remained stable around 14 minutes, with approximately 7.5 minutes spent on Prophet decomposition.

Table 7.1

Pipeline Evaluation Across Varying Anomaly Injection Configurations

Injected Anomalies Rolling Window	Injected Anomalies STD	Injected Anomalies Random State	Injected Anomalies Percentage	Soft Union Precision	Soft Union Recall	Soft Union F1	Business Shave Precision	Business Shave Recall	Business Shave F1	Full Pipeline Runtime
8	4	42	1	0.0156	0.9355	0.0307	0.1402	0.3557	0.2011	14m
8	2	42	1	0.0138	0.8242	0.0271	0.0776	0.1837	0.1091	14m
8	10	42	1	0.0161	0.9606	0.0316	0.2034	0.5733	0.3003	14m
8	4	17	1	0.0157	0.9374	0.0310	0.1364	0.3450	0.1955	14m
16	4	42	1	0.0155	0.9312	0.0306	0.1421	0.3607	0.2039	15m
4	4	42	1	0.0156	0.9343	0.0307	0.1403	0.3552	0.2012	14m
8	4	42	0.5	0.0078	0.9292	0.0155	0.0747	0.3542	0.1234	14m
8	4	42	1	0.0310	0.9293	0.0600	0.2413	0.3384	0.2817	13m

Note: Table created by the author based on the created dataset.

The chosen strategy successfully meets the objective of uncovering actionable anomalies, rather than merely detecting all injected ones. It was shown in Figure 5.7 *Residual Distribution: FP vs FN after Business Post-Processing* and Figure 5.8 *Store 45 – Dept 29: Soft Union Business Logic Post-Processing* that the anomalies identified through this method include both significant injected anomalies and meaningful anomalies naturally present in the original data. This confirms the method's effectiveness for the intended task.

7.1.2 Results in Terms of Scalability

The dataset consists of 421,570 data points across 3,331 department-level time series, representing a large-scale time series setting. Scalability was assessed both in terms of runtime efficiency and infrastructure feasibility.

All final runs in this thesis were executed on Deepnote's GPU L4 instance, which provides 16 vCPUs, 64 GB of memory, and 24 GB of dedicated VRAM, at a cost of \$1.56 per hour (Figure 7.1 *Available Machines in Deepnote*). The final pipeline runs in about 14 minutes on the GPU (L4) instance, compared to 28 minutes on the free CPU (Basic) instance. Even on the paid GPU, the cost remains financially negligible (roughly \$0.39 per run), especially considering that the process is only executed once per week and detects sales anomalies exceeding \$5,000. If this task were to be done manually, identifying anomalies across 3,331 department-level series would be impractical. Moreover, failing to detect these patterns could result in missed revenue opportunities, overlooked operational issues, or poor decisions.

Figure 7.1

Available Machines in Deepnote

Machine selection			
<input type="radio"/>	CPU (Basic)	Free	Internet access
2 vCPU, 5 GB memory			
<input type="radio"/>	CPU (Plus)	\$0.38/h	Internet access
4 vCPU, 16 GB memory			
<input type="radio"/>	CPU (Performance)	\$1.54/h	Internet access
16 vCPU, 64 GB memory			
<input type="radio"/>	CPU (High memory)	\$2.02/h	Internet access
16 vCPU, 128 GB memory			
<input type="radio"/>	GPU (T4)	\$0.14/h	Internet access
8 vCPU, 32 GB memory, 16 GB VRAM			
<input checked="" type="radio"/>	GPU (L4)	\$1.56/h	Internet access
16 vCPU, 64 GB memory, 24 GB VRAM			

Note: From Deepnote (2025).

Each of the anomaly detection models included in the final pipeline finishes in under one minute. Longer runtimes were observed in models not selected for the final ensemble, such as One-Class SVM, autoencoders, and Deep SVDD. Of the total 14-minute runtime, the Prophet decomposition accounts for approximately 7.5 minutes on average. It was retained due to its clear superiority in detection accuracy compared to other decomposition methods. Given that this analysis is performed weekly, the total runtime is entirely acceptable.

In conclusion, the pipeline demonstrates strong scalability, making it both computationally and financially viable for continuous, real-world monitoring of large-scale, hierarchical, multivariate time series data.

7.1.3 Results in Terms of Interpretability

Interpretability was a key objective in this thesis, ensuring that anomalies could be understood both technically and from a business perspective.

The chosen framework prioritizes understandability over complexity — six anomaly detection models run anomaly detection, and if any of them flags a point that exceeds a \$5,000 deviation from forecast (while following specified holiday/markdown rules), it is marked as an anomaly. At higher levels, an anomaly is propagated if at least 5% of the underlying level series flag one. This simple rule-based design avoids edge cases and is easy to explain and remember.

Two mechanisms were employed for interpretability and explainability. SHAP-like feature contributions on department-level anomalies (explaining why a point was flagged) and managerial summaries (explaining what it means and what actions may follow). While these

summaries could be extended into full dashboards, that lies outside the scope of this thesis and is suggested for future work.

A minor limitation is that SHAP-like values were not aggregated beyond the department level, due to the reconciliation strategy. However, this was intentional — anomalies at store or national level are triggered not by individual features, but by the collective signal from multiple flagged departments, providing a clear and actionable drill-down path.

7.2 Approved Models

After extensive evaluation against both technical and business criteria, a final model strategy was selected.

No single anomaly detection model performed well enough across all evaluation dimensions, so an ensemble approach was introduced. The final ensemble combined the best-performing models (KNN, Thresholding, Mahalanobis Distance, Isolation Forest, HDBSCAN, and GMM), capturing strengths from statistical, classical unsupervised, and probabilistic methods.

The most effective ensemble strategy was a soft union, where anomalies detected by any of the selected models were included, maximizing recall. This was followed by a business logic-based post-processing step to improve precision. This step applied stable business rules — only spikes without holidays or markdowns and only drops with either a holiday or markdown were retained, along with a fixed \$5,000 residual threshold to ensure financial significance.

Because all modeling was done at the department level, a hierarchical bottom-up reconciliation step was introduced after post-processing. This logic propagates anomalies to the store or national level only when at least 5% of underlying departments flag an anomaly of the same type. This ensured that higher-level anomalies represent consistent underlying signals, and the approach proved effective in surfacing interpretable, scalable, and actionable results.

7.3 Review of the Process

The modeling process followed the CRISP-DM methodology end to end, covering all core phases without omitting any key steps.

Throughout the project, learnings, evaluation results and insights from later stages were continuously fed back into earlier steps, creating an iterative loop between data preparation, modeling, and evaluation to improve each successive version's reliability and accuracy.

The modular pipeline structure allowed decomposition methods, detection models, and evaluation strategies to be developed and compared independently. All modeling steps were versioned and consistently named, with detailed evaluation logs and serialized outputs

enabling traceability across large-scale experiments. Quality controls were built in to preserve injected anomalies, apply uniform residual scaling, and enforce consistent evaluation logic across decomposition methods. These practices supported reliable comparison and consistent results throughout the research process.

The project ultimately relied solely on historical Walmart data. While two informal interviews with retail professionals informed the early phase of the project, this thesis did not include formal input from end users. The absence of review by operational decision-makers is a known limitation. For any real-world deployment, direct validation and co-design with users such as store managers, merchandisers, or planners would be essential to ensure business relevance, interpretability, and trust.

Conclusion

This conclusion answers the research questions and objectives by summarizing the most suitable anomaly detection methods for large-scale hierarchical multivariate time series, comparing their performance across key criteria, and reflecting on the development of a practical detection framework, along with limitations and directions for future work.

Answering Research Questions and Objectives

What are the most suitable anomaly detection methods for large-scale hierarchical multivariate time series?

Methods for interpretable point-wise anomaly detection in multivariate and hierarchical time series were reviewed, and their theoretical foundations and practical applicability were assessed.

It was determined that for highly seasonal time series, such as those in retail, the most effective approach is to first apply decomposition to separate signal from noise, followed by anomaly detection in the noise space.

From the literature, three suitable decomposition methods were identified: Prophet, STL, and TimeGPT. In addition, seventeen anomaly detection models were selected as candidates for the task, including statistical methods (Z-Scores, Thresholding), classical unsupervised models (Isolation Forest, K-Nearest Neighbors, Local Outlier Factor, HDBSCAN, One-Class SVM), probabilistic approaches (COPOD, BCPD, Mahalanobis Distance, Gaussian Mixture Model), and deep learning models (Plain, Variational, LSTM, Transformer-based, and TCN Autoencoders, as well as Deep SVDD).

Their suitability for interpretable point-wise anomaly detection in large-scale hierarchical multivariate time series was assessed in Table 2.2 *Anomaly Detection Methods Comparison*. However, no single model met all criteria. While some limitations could be addressed, for example, using surrogate models for interpretability or sliding windows and time-aware cross-validation to preserve temporal structure, no method was sufficient on its own.

Due to this, a pipeline and ensemble approach were developed to overcome individual model shortcomings and create a robust framework for interpretable point-wise anomaly detection in large-scale hierarchical multivariate time series.

How do different approaches compare in terms of accuracy, scalability, and interpretability?

Selected methods were implemented and benchmarked on the Walmart Store Sales Forecasting dataset.

To assess accuracy and scalability, all combinations of candidate anomaly detection models were run and tuned on residuals from each decomposition method. Table 5.1 *Best Runs per Anomaly Detection Method and Decomposition Combination* summarizes their accuracy and runtime. The most accurate models were KNN, Thresholding, Mahalanobis Distance, Isolation Forest, HDBSCAN, and Gaussian Mixture Model. These models also showed reasonable runtimes, ranging from 36 to 78 seconds on Prophet residuals. In contrast, lower-performing models, mainly deep learning approaches and One-Class SVM, took over 15 minutes or failed to complete at this data scale.

Interpretability was first assessed theoretically, as summarized in Table 2.2 *Anomaly Detection Methods Comparison*, and then consistently applied throughout the pipeline using SHAP-like surrogate models to capture feature contributions. These contributions were aggregated in an additive manner to provide interpretability across the entire process.

Since simply flagging anomalies without linking them to a meaningful deviation from prediction is of limited value, this principle was emphasized in the demo managerial summaries, such as in Figure 6.1 *Managerial Summary: Store 45, Week 2011-12-02*. These summaries go beyond stating that an anomaly occurred — they provide context and highlight hierarchical relationships essential for recognizing patterns, enabling action, and supporting decision-making in a business setting.

In summary, individual models showed significant variation in accuracy and scalability and natively also in interpretability, but that was addressed and standardized through surrogate models that captured feature contributions.

Can an effective anomaly detection framework be developed to support real-world applications in hierarchical time series analysis?

Yes, this thesis demonstrates that a working framework can be developed. The results of different approaches were compared to identify their strengths, weaknesses, and real-world applicability. Based on these findings, recommendations and insights were proposed to support the development of scalable and interpretable anomaly detection techniques for hierarchical time series data.

The core idea is that the framework first optimizes for high recall — not only to capture injected anomalies as true positives, but also to detect real, naturally occurring anomalies that appear as false positives in the confusion matrix. A business lens is then applied to improve precision by filtering for anomalies that are truly relevant from an operational perspective.

The final framework consists of five sequential steps: (1) Prophet decomposition to separate signal from noise in highly seasonal data, (2) six best-performing anomaly detection models applied to residuals — covering statistical (Thresholding), classical unsupervised (Isolation Forest, KNN, HDBSCAN), and probabilistic (GMM, Mahalanobis) approaches, (3) a soft union of their outputs to maximize recall, (4) business logic post-processing to improve precision using a fixed financial threshold and specific holiday and markdown rules to classify spikes and drops, and (5) hierarchical reconciliation to propagate anomalies to higher levels. SHAP-like surrogate models were used consistently throughout for interpretability.

This framework was developed and proven to be effective in point-wise anomaly detection in large-scale hierarchical multivariate time series even under varying anomaly injection scenarios.

Limitations

A key limitation is the use of artificially injected anomalies for validation, which were generated using a statistical approach without underlying patterns or temporal complexity. This may have favored simpler models and disadvantaged deep learning methods.

Another limitation is the limited amount of data — only one full year and two incomplete years were available. This was not sufficient for decomposition methods like STL and TimeGPT to demonstrate their full potential. With more data, they may outperform Prophet — especially TimeGPT, which is a strong candidate for incremental use. In fact, its residuals showed promising quality toward the end of the timeline, as seen in Figure 5.3 *Store 45 — Dept 29: TimeGPT Decomposition*.

Resource and infrastructure constraints limited the experimental scope. Several deep learning models, such as LSTM Autoencoders and Deep SVDD, could not be fully trained or tuned on the complete dataset due to GPU memory limitations. Some experiments required sampling or feature reduction, which may have introduced bias. Although tests were run on Deepnote’s high-performance L4 GPU tier, full-scale deep learning experiments would require more advanced and costly infrastructure. Additionally, TimeGPT decomposition was evaluated only in trial mode without access to enterprise-grade features, preventing a full assessment of its operational potential.

One of the most unexpected findings was that none of the deep learning models outperformed the simpler methods. This may be due to the nature of the artificial anomaly injection, which lacked underlying patterns, or the limited dataset — only three years, two of which were incomplete. This limitation could potentially be addressed with more extensive tuning, additional data, or by applying the models to real-world anomalies rather than injected ones. Including at least one deep learning model in the ensemble would add a new detection mechanism and strengthen the overall framework. Its absence is a limitation of the current best-performing ensemble and may impact performance on different datasets or injection strategies.

An important finding was that decomposition quality emerged as the dominant success factor in highly seasonal data. During experiments it was proven that the choice of decomposition method had a greater impact on detection performance than the choice of anomaly detection model. Prophet consistently outperformed STL and TimeGPT by producing stable and interpretable residuals that allowed downstream models to perform well. This revealed a key dependency — when residuals fail to clearly separate predictable patterns, even strong anomaly detectors are unable to compensate.

This thesis identified a major gap in hierarchical decomposition, which is critical for anomaly detection in large industries like retail. Although the dataset had a clear hierarchical

structure, no existing decomposition method could natively handle hierarchical multivariate time series. As a result, even hierarchical methods like HDBSCAN were run on each series individually, limiting their effectiveness. Workarounds, such as bottom-up reconciliation using department-level detections, were required.

The pipeline is currently functional and producing meaningful results on the fixed dataset used in this thesis. However, if deployed in a live setting, it would require ongoing monitoring and periodic adjustment of parameters to account for new patterns and changes over time. While the core strategy, prioritizing recall first and refining precision through business logic, should remain unchanged, threshold values and model parameters may need to be tuned again to maintain performance.

A key limitation is the lack of live business feedback. While two informal interviews with retail staff informed early assumptions about current trends in anomaly detection, no business stakeholders reviewed the results of this thesis. As a result, although technical interpretability was achieved, its practical value in real decision-making contexts remains unconfirmed.

Future Work

Future work should focus on deploying the framework, including building a dashboard and alerting tool for store managers, based on the generated summaries.

It is also recommended to retrain or fine-tune the models on different data granularities, such as daily data or SKU-level series, if available, to explore performance across different operational layers and even increase business value of this framework. Also, more anomaly types can be considered for detection.

Introducing active learning strategies or feedback loops could help reduce the gap between injected and naturally occurring anomalies by incorporating labels over time.

If resources allow, further exploration of TimeGPT in its paid tier is encouraged, as it may yield better results — especially in an incremental setting. In this thesis, its performance was limited because accuracy was judged using the Full window, while the Fair window (where TimeGPT was available) was too small to showcase its strengths.

Deep models may also be worth revisiting if more training data or compute time becomes available.


Finally, a clear limitation in this work is the lack of hierarchical decomposition. Once reliable hierarchical decomposition methods are available, this innovation should be integrated and the framework re-evaluated.

References

- Agyemang, E. F. (2024). Anomaly detection using unsupervised machine learning algorithms: A simulation study. *Scientific African*, 26, e02386–e02386. <https://doi.org/10.1016/j.sciaf.2024.e02386>
- Al-Marie, M. (2023, April 4). *Exploring Neural Network Architectures: Autoencoders, Encoder-Decoders, and Transformers*. Medium. <https://medium.com/%40mohd.meri/exploring-neural-network-architectures-autoencoders-encoder-decoders-and-transformers-c0d3d6bc31d8>
- Al-Selwi, S. M., Hassan, M. F., Abdulkadir, S. J. & Muneer A.. (2023). LSTM Inefficiency in Long-Term Dependencies Regression Problems. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 30(3), 16–31. <https://doi.org/10.37934/araset.30.3.1631>
- Apgar, V. (2023, July 27). *3 Use-Cases for Gaussian Mixture Model (GMM) | Towards Data Science*. Towards Data Science. <https://towardsdatascience.com/3-use-cases-for-gaussian-mixture-model-gmm-72951fcf8363/>
- Asperti, A., & Trentin, M. (2020, February 18). *Balancing reconstruction error and Kullback-Leibler divergence in Variational Autoencoders*. ArXiv.org. <https://doi.org/10.48550/arXiv.2002.07514>
- Awan, A. A. (2023, June 28). *An Introduction to SHAP Values and Machine Learning Interpretability*. Datacamp.com; DataCamp. <https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability>
- Bajaj, A. (2023, August 22). *Anomaly Detection in Time Series*. Neptune.ai. <https://neptune.ai/blog/anomaly-detection-in-time-series>
- Blachowicz, T., Wylezek, J., Sokol, Z., & Bondel, M. (2025). Real-Time Analysis of Industrial Data Using the Unsupervised Hierarchical Density-Based Spatial Clustering of Applications with Noise Method in Monitoring the Welding Process in a Robotic Cell. *Information*, 16(2), 79–79. <https://doi.org/10.3390/info16020079>
- Blázquez-García, A., Conde, A., Mori, U., & Lozano, J. A. (2021). A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Computing Surveys*, 54(3), 1–33. <https://doi.org/10.1145/3444690>
- Brownlee, J. (2018, November 4). *A Gentle Introduction to LSTM Autoencoders - MachineLearningMastery.com*. MachineLearningMastery.com. <https://machinelearningmastery.com/lstm-autoencoders>
- Brownlee, J. (2020, December 17). *What Is Meta-Learning in Machine Learning? - MachineLearningMastery.com*. MachineLearningMastery.com. <https://machinelearningmastery.com/meta-learning-in-machine-learning>
- Cai, B., Yang, S., Gao, L., & Xiang, Y. (2023). Hybrid variational autoencoder for time series forecasting. *Knowledge-Based Systems*, 281, 111079–111079. <https://doi.org/10.1016/j.knsys.2023.111079>
- Capital one. (2025, January 31). *Largest Retailers in the U.S. and the World (as of 2023): Full List*. Capital One Shopping. <https://capitaloneshopping.com/research/largest-retailers/>
- Carletti, M., Terzi, M., & Antonio, S. G. (2020). *Interpretable Anomaly Detection with DIFFI: Depth-based Isolation Forest Feature Importance*. ArXiv.org. <https://arxiv.org/abs/2007.11117>
- Challu, C., Jiang, P., Wu, Y. N., & Callot, L. (2022). *Deep generative model with hierarchical latent factors for time series anomaly detection*. Amazon Science. <https://www.amazon.science/publications/deep-generative-model-with-hierarchical-latent-factors-for-time-series-anomaly-detection>
- Chen, H., Lundberg, S. M., & Lee, S.-I. (2022). Explaining a series of models by propagating Shapley values. *Nature Communications*, 13(1). <https://doi.org/10.1038/s41467-022-31384-3>
- Chen, Y., Zhang, C., Ma, M., Liu, Y., Ding, R., Li, B., He, S., Rajmohan, S., Lin, Q., & Zhang, D. (2023). *ImDiffusion: Imputed Diffusion Models for Multivariate Time Series Anomaly Detection*. ArXiv.org. <https://arxiv.org/abs/2307.00754>
- Chollet, F. (2016). *Building Autoencoders in Keras*. Keras.io. <https://blog.keras.io/building-autoencoders-in-keras.html>
- Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A Seasonal-Trend Decomposition Based on Loess. *Journal of Official Statistics*, 6(1), 3–33. <https://www.math.unm.edu/~lil/Stat581/STL.pdf>
- Darban, Z. Z., Webb, G., Darban, Z., Pan, S., Aggarwal, C., & Salehi, M. (2024). *Deep Learning for Time Series Anomaly Detection: A Survey*. <https://arxiv.org/pdf/2211.05244>
- Deepnote. (2025). *Deepnote - Data science notebook for teams*. Deepnote. <https://deepnote.com/>
- Despois, J. (2017, February 23). *Latent space visualization — Deep Learning bits #2*. Hackernoon.com. <https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df>

- Dey, R. (2024, March 16). *Anomaly Detection using Support Vectors*. Medium. <https://medium.com/@roshmitadey/anomaly-detection-using-support-vectors-2c1b842213ed>
- Dhapre, M. (2024, September 16). *Using Variational AutoEncoders (VAE) for Time-Series Data Reduction*. Medium. <https://medium.com/%40mrnmayee.dhapre/using-variational-autoencoders-vae-for-time-series-data-reduction-9681338a2e17>
- Ebenezer, I., & Sharma, A. (2023). *Adaptive Thresholding Heuristic for KPI Anomaly Detection*. ArXiv.org. <https://arxiv.org/abs/2308.10504>
- Edge Impuls. (2024, May 22). *Anomaly detection (GMM) | Edge Impulse Documentation*. Edgeimpulse.com. <https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/anomaly-detection-gmm>
- Eslava, A. (2023, July 16). *Outlier Detection Techniques for Time Series - Alex Eslava - Medium*. Medium. <https://medium.com/@alex.eslava96/outlier-detection-techniques-for-time-series-9868db2875c2>
- Eyer. (2024, March 9). *Anomaly Detection in Time Series Data Python: A Starter Guide*. Eyer.ai. <https://www.eyer.ai/blog/anomaly-detection-in-time-series-data-python-a-starter-guide/>
- Facebook. (2019). *Prophet*. Prophet. <https://facebook.github.io/prophet/>
- Feasel, K. (2022). Copula-Based Outlier Detection (COPOD). *Finding Ghosts in Your Data*, 217–228. https://doi.org/10.1007/978-1-4842-8870-2_12
- Furnari, G., Vattiato, F., Allegra, D., Milotta, F. L. M., Orofino, A., Rizzo, R., De Palo, R. A., & Stanco, F. (2021). An Ensembled Anomaly Detector for Wafer Fault Detection. *Sensors*, 21(16), 5465. <https://doi.org/10.3390/s21165465>
- GeeksForGeeks. (2019, June 21). *ML | Auto-Encoders*. GeeksforGeeks. <https://www.geeksforgeeks.org/auto-encoders/>
- GeeksForGeeks. (2020, July 20). *Variational AutoEncoders*. GeeksforGeeks. <https://www.geeksforgeeks.org/variational-autoencoders/>
- Govindaraj, P. (2024, July 19). *Self-Attention Mechanism In Transformers - Priyanthan Govindaraj - Medium*. Medium. <https://medium.com/%40govindarajpriyanthan/self-attention-mechanism-in-transformers-1e46af9e1afb>
- Hayes, T. C., & Times, S. T. the N. Y. (1990, February 28). COMPANY NEWS; Wal-Mart Net Jumps By 31.8% (Published 1990). *The New York Times*. <https://www.nytimes.com/1990/02/28/business/company-news-wal-mart-net-jumps-by-31.8.html>
- Helen. (2019, December 1). *Keras, sequential, and timeseries: should we flatten or not?* Stack Overflow. <https://stackoverflow.com/questions/59125775/keras-sequential-and-timeseries-should-we-flatten-or-not>
- Holbert, C. (2022, March 27). *Outlier Identification Using Mahalanobis Distance*. Charles Holbert. https://www.cfholbert.com/blog/outlier_mahalanobis_distance/
- Hong, Z. (2024, February 2). *Anomaly Detection in Time Series Data using LSTM Autoencoders*. Medium. <https://medium.com/%40zhonghong9998/anomaly-detection-in-time-series-data-using-lstm-autoencoders-51fd14946fa3>
- Huang, X., & Joao Marques-Silva. (2024). On the failings of Shapley values for explainability. *International Journal of Approximate Reasoning*, 171, 109112–109112. <https://doi.org/10.1016/j.ijar.2023.109112>
- Hyndman, R. J., & Athanasopoulos, G. (2021). *Chapter 11 Forecasting hierarchical and grouped time series | Forecasting: Principles and Practice (3rd ed)*. Otexts.com. <https://otexts.com/fpp3/hierarchical.html>
- IBM. (2024, June 12). *Variational autoencoder*. Ibm.com. <https://www.ibm.com/think/topics/variational-autoencoder>
- IBM. (2025, January 28). *Latent Space*. Ibm.com. <https://www.ibm.com/think/topics/latent-space>
- Intel. (2024, March 11). *How to Apply Transformers to Time Series Models*. Intel Tech. <https://medium.com/intel-tech/how-to-apply-transformers-to-time-series-models-spacetimeformer-e452f2825d2e>
- Iuhasz, G., Teodor-Florin Fortiș, & Silviu Panica. (2025). Exploring machine learning methods for the identification of production cycles and anomaly detection. *Internet of Things*, 30, 101508–101508. <https://doi.org/10.1016/j.iot.2025.101508>
- Jeffrey, N., Tan, Q., & Villar, J. R. (2024). Using Ensemble Learning for Anomaly Detection in Cyber–Physical Systems. *Electronics*, 13(7), 1391. <https://doi.org/10.3390/electronics13071391>
- Kaggle. (2014, February 20). *Walmart Recruiting - Store Sales Forecasting*. Kaggle.com. <https://www.kaggle.com/competitions/walmart-recruiting-store-sales-forecasting/data>
- Tuhin, K. H., Nobi, A., Rakib, M. H., & Lee, J. W. (2025). Long short-term memory autoencoder based network of financial indices. *Humanities and Social Sciences Communications*, 12(1). <https://doi.org/10.1057/s41599-025-04412-y>
- Kar, S. (2024, November 2). *What is Transformer Architecture? The Transformer architecture is a deep learning model introduced in 2017 by Vaswani et al. that revolutionized natural language processing (NLP) and paved the way for many modern AI applications, including BERT, GPT, and T5*. LinkedIn.com. <https://www.linkedin.com/pulse/understanding-transformer-architecture-backbone-modern-suman-kar-23u4c/>
- Kaya, ibrahim. (2020, February 12). *Anomaly Detection and Mahalanobis Distance*. Medium. <https://medium.com/@ikaya754/anomaly-detection-and-mahalanobis-distance-25b21b7cfe5b>

- Kennedy, W. B. (2025, January 3). *Deep Learning for Outlier Detection on Tabular and Image Data*. Medium; TDS Archive. <https://medium.com/data-science/deep-learning-for-outlier-detection-on-tabular-and-image-data-90ae518a27b3>
- Konefal, B. (2023, June 16). *LinkedIn*. LinkedIn.com. <https://www.linkedin.com/pulse/anomaly-detection-isolation-forest-bogus%C5%82aw-konefa%C5%82/>
- Kumar, K. (2023, August 17). *Mastering Anomaly Detection in Time Series Data: Techniques and Insights*. Medium; Medium. <https://medium.com/@ketan31kumar/mastering-anomaly-detection-in-time-series-data-techniques-and-insights-98fbc94c4258>
- Lachekhab, F., Benzaoui, M., Tadjer, S. A., ensmaine, A. B., & Hamma, H. (2024). LSTM-Autoencoder Deep Learning Model for Anomaly Detection in Electric Motor. *Energies*, 17(10), 2340–2340. <https://doi.org/10.3390/en17102340>
- Lai, K.-H., Daochen Zha, Xu, J., Zhao, Y., Wang, G., & Hu, X. (2021). *Revisiting Time Series Outlier Detection: Definitions and Benchmarks*. OpenReview. <https://openreview.net/forum?id=r8lvOsnHchr>
- Lawton, G. (2024). *Generative models: VAEs, GANs, diffusion, transformers, NeRFs*. Search Enterprise AI; TechTarget. <https://www.techtarget.com/searchenterpriseai/tip/Generative-models-VAEs-GANs-diffusion-transformers-NeRFs>
- Lee, I. (2024, July 23). *TimeGPT vs Statistical Models for Forecasting SPY | Medium*. Medium. https://medium.com/@_ivylee_/timegpt-vs-statistical-models-for-forecasting-spy-8378ea258a19
- Li, Z., Zhao, Y., Botta, N., Ionescu, C., & Hu, X. (2020). COPOD: Copula-Based Outlier Detection. *ArXiv (Cornell University)*. <https://doi.org/10.1109/icdm50108.2020.00135>
- Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*. <https://doi.org/10.1109/icdm.2008.17>
- Liu, T., Zhou, Z., & Yang, L. (2024). Layered isolation forest: A multi-level subspace algorithm for improving isolation forest. *Neurocomputing*, 581, 127525–127525. <https://doi.org/10.1016/j.neucom.2024.127525>
- Lozovsky, D. (2024, February 15). *The Limitations of Transformers: A Deep Dive into AI's Current Shortcomings and Future Potentials*. Ww.linkedin.com. <https://www.linkedin.com/pulse/limitations-transformers-deep-dive-ais-current-future-lozovsky-mba-vrrdc/>
- Lu, T., Wang, L., & Zhao, X. (2023). Review of Anomaly Detection Algorithms for Data Streams. *Applied Sciences*, 13(10), 6353. <https://doi.org/10.3390/app13106353>
- Mancuso, P., Piccialli, V., & Sudoso, A. M. (2021). A machine learning approach for forecasting hierarchical time series. *Expert Systems with Applications*, 115102. <https://doi.org/10.1016/j.eswa.2021.115102>
- Melanie. (2024, March 20). *Facebook Prophet : All you need to know*. Data Science Courses | DataScientest. <https://datascientest.com/en/facebook-prophet-all-you-need-to-know>
- Mesameki. (2025, March 26). *Model interpretability - Azure Machine Learning*. Microsoft.com. <https://learn.microsoft.com/en-us/azure/machine-learning/how-to-machine-learning-interpretability?view=azureml-api-2>
- Milvus. (2025). *What is ensemble anomaly detection?* Milvus.io. <https://milvus.io/ai-quick-reference/what-is-ensemble-anomaly-detection>
- MindBridge. (2025, February 19). *Anomaly Detection Techniques: How to Uncover Risks, Identify Patterns, and Strengthen Data Integrity*. MindBridge. <https://www.mindbridge.ai/blog/anomaly-detection-techniques-how-to-uncover-risks-identify-patterns-and-strengthen-data-integrity/>
- Moffitt, J. (2024, April 1). *Real-Time Anomaly Detection: Use Cases and Code Examples*. Ww.tinybird.co. <https://www.tinybird.co/blog-posts/real-time-anomaly-detection>
- Muruganandham, P., Jayaraman, S., Tahiliani, K., Tanna, R., Ghosh, J., Pathak, S. K., & Ramaiya, N. (2024). An advanced double-phase stacking ensemble technique with active learning classifier: Toward reliable disruption prediction in Aditya tokamak. *Review of Scientific Instruments*, 95(9). <https://doi.org/10.1063/5.0222189>
- Muslim, M. A., Nikmah, T. L., Pertiwi, D. A. A., Subhan, Jumanto, Dasril, Y., & Iswanto. (2023). New model combination meta-learner to improve accuracy prediction P2P lending with stacking ensemble learning. *Intelligent Systems with Applications*, 18, 200204. <https://doi.org/10.1016/j.iswa.2023.200204>
- Neloy, A. A., & Turgeon, M. (2024). A comprehensive study of auto-encoders for anomaly detection: Efficiency and trade-offs. *Machine Learning with Applications*, 17, 100572–100572. <https://doi.org/10.1016/j.mlwa.2024.100572>
- Netsch, C. (2024, July 2). *3 + 1 Lessons learned from running TimeGPT on Machine Data*. Alpamayo. <https://www.alpamayo-solutions.com/en/blog/3-1-lessons-learned-from-running-timegpt-on-machine-data>
- Nguyen, M.-N., & Vien, N. A. (2018, April 13). *Scalable and Interpretable One-class SVMs with Deep Learning and Random Fourier features*. ArXiv.org. <https://arxiv.org/abs/1804.04888>

- Nixtla. (2025, January). *Anomaly detection*. TimeGPT Foundational Model for Time Series Forecasting and Anomaly Detection. https://docs.nixtla.io/docs/tutorials-anomaly_detection
- Ögretir, M., Ramchandran, S., Papatheodorou, D., & Lahdesm, H. (2023, November 20). *A Variational Autoencoder for Heterogeneous Temporal and Longitudinal Data*. Arxiv. <https://arxiv.org/pdf/2204.09369>
- Owoh, N., Riley, J., Ashawa, M., Hosseinzadeh, S., Philip, A., & Osamor, J. (2024). An Adaptive Temporal Convolutional Network Autoencoder for Malicious Data Detection in Mobile Crowd Sensing. *Sensors*, 24(7), 2353–2353. <https://doi.org/10.3390/s24072353>
- Peixeiro, M. (2023, March 15). *Practical Guide for Anomaly Detection in Time Series with Python*. Datasciencewithmarco.com. <https://www.datasciencewithmarco.com/blog/practical-guide-for-anomaly-detection-in-time-series-with-python>
- Pérez-Carrasco, M., Cabrera-Vives, G., Hernández-García, L., Forster, F., Sánchez-Sáez, P., Arancibia, A. M., Astorga, N., Bauer, F., Bayo, A., Cádiz-Leyton, M., & Catelan, M. (2023, August 10). *Multi-Class Deep SVDD: Anomaly Detection Approach in Astronomy with Distinct Inlier Categories*. ArXiv.org. <https://doi.org/10.48550/arXiv.2308.05011>
- Perry, K. (2019, August 14). *A Brief Introduction to Change Point Detection using Python - Tech Rando*. Tech Rando. <https://techrando.com/2019/08/14/a-brief-introduction-to-change-point-detection-using-python/>
- Ippolito, P. P. (2023, December 14). *Introduction to Autoencoders: From The Basics to Advanced Applications in PyTorch*. DataCamp.com; DataCamp. <https://www.datacamp.com/tutorial/introduction-to-autoencoders>
- Pykes, K. (2024, August 13). *Variational Autoencoders: How They Work and Why They Matter*. DataCamp.com; DataCamp. <https://www.datacamp.com/tutorial/variational-autoencoders>
- Rajan, S. (2021, August 10). *Multivariate Time Series Anomaly Detection using VAR model*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/08/multivariate-time-series-anomaly-detection-using-var-model/>
- Rajasekaran, A. (2025, March 30). *Moments That Matter: Identifying Change Points in Time Series*. Medium. <https://medium.com/%40anithaamalan/moments-that-matter-identifying-change-points-in-time-series-f5ecfe8855c5>
- RisingWave. (2024, July 23). *Effective Anomaly Detection in Time-Series Using Basic Statistics*. Effective Anomaly Detection in Time-Series Using Basic Statistics. <https://risingwave.com/blog/effective-anomaly-detection-in-time-series-using-basic-statistics/>
- Romeu, A. (2021, June 24). *Simple statistics for anomaly detection on time-series data*. Tinybird.co. <https://www.tinybird.co/blog-posts/anomaly-detection>
- Ruberts, A. (2020, July 15). *Antons Ruberts*. Well Enough. <https://antonsruberts.github.io/anomaly-detection-web-2/>
- Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., & Kloft, M. (2018, July 3). *Deep One-Class Classification*. Proceedings.mlr.press; PMLR. <https://proceedings.mlr.press/v80/ruff18a.html>
- ruptures. (n.d.). *Binary segmentation - ruptures*. Github.io. Retrieved April 13, 2025, from <https://centre-borelli.github.io/ruptures-docs/user-guide/detection/binseg/>
- ruptures. (2017). *Binary segmentation — ruptures documentation*. Cnrs.fr. <https://ctruong.perso.math.cnrs.fr/ruptures-docs/build/html/detection/binseg.html>
- Kamoi, R., & Kobayashi, K. (2020). Why is the Mahalanobis Distance Effective for Anomaly Detection? *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2003.00402>
- Sayago, G. S. (2024, May 29). *Decoding Efficiency: GMM (Gaussian Mixture Models) vs KMeans in Anomaly Detection*. Medium. <https://medium.com/@surribasg/decoding-efficiency-gmm-gaussian-mixture-models-vs-kmeans-in-anomaly-detection-f695242d7af1>
- Schmidl, S., Wenig, P., Papenbrock, T., & Anomaly. (2022). Anomaly Detection in Time Series: A Comprehensive Evaluation. *Proceedings of the VLDB Endowment*, 15(9), 2150–8097. <https://doi.org/10.14778/3538598.3538602>
- Scikit Learn. (2018). *Compare the effect of different scalers on data with outliers — scikit-learn 0.20.3 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html
- Sendera, M., Śmieja, M., Maziarka, Ł., Struski, Ł., Spurek, P., & Tabor, J. (2021). *Flow-based SVDD for anomaly detection*. ArXiv.org. <https://arxiv.org/abs/2108.04907>
- Shabou, S. (n.d.). Chapter 5 Outlier detection in Time series | Time Series with R. *S-Ai-F.github.io*. Retrieved April 13, 2025, from <https://s-ai-f.github.io/Time-Series/outlier-detection-in-time-series.html>
- Singh, H. (2024, February 13).  *Anomaly Detection with Autoencoders*. Kaggle.com; Kaggle. <https://www.kaggle.com/code/harshsingh2209/anomaly-detection-with-autoencoders>
- Smart Vision Europe. (2025). *Crisp DM methodology*. Smart Vision Europe. <https://www.sv-europe.com/crisp-dm-methodology/#two>

- Spector, D. (2012, November 15). *18 Facts About Walmart That Will Blow Your Mind*. Business Insider. <https://www.businessinsider.com/crazy-facts-about-walmart-2012-11>
- Srivastava, A. (2023, October). *Detecting Anomalies with Z-Scores: A Practical Approach*. Medium. <https://medium.com/%40akashri306/detecting-anomalies-with-z-scores-a-practical-approach-2f9a0f27458d>
- Statsmodels. (2025, April 2). *Seasonal-Trend decomposition using LOESS (STL)* — statsmodels. [www.statsmodels.org. https://www.statsmodels.org/dev/examples/notebooks/generated/stl_decomposition.html](https://www.statsmodels.org/dev/examples/notebooks/generated/stl_decomposition.html)
- Sun, Q., Li, Y., Hu, Z., Zhou, C., & Liu, L. (2024). Spatial-Temporal Dependency Based Multivariate Time Series Anomaly Detection for Industrial Processes. *Lecture Notes in Computer Science*, 212–223. https://doi.org/10.1007/978-981-97-5618-6_18
- Taylor, S. J., & Letham, B. (2017, September 27). *Forecasting at scale*. Peerj.com. <https://peerj.com/preprints/3190/>
- Teuwens, R. (2021). *Anomaly Detection with Auto-Encoders*. Kaggle.com. <https://www.kaggle.com/code/robinteuwens/anomaly-detection-with-auto-encoders>
- The Produce News. (2024). *Walmart: The world's largest retailer* | Produce News. The Produce News. <https://theproducenews.com/headlines/walmart-worlds-largest-retailer>
- Thill, M. (2020). *Time Series Encodings with Temporal Convolutional Networks*. <https://www.gm.th-koeln.de/ciopwebpub/Thill20a.d/bioma2020-tcn.pdf>
- Thill, M., Konen, W., Wang, H., & Bäck, T. (2021). Temporal convolutional autoencoder for unsupervised anomaly detection in time series. *Applied Soft Computing*, 112, 107751. <https://doi.org/10.1016/j.asoc.2021.107751>
- Tuychiyev, B., & DataCamp. (2021, October 1). *Anomaly Detection in Python*. DataCamp. <https://app.datacamp.com/learn/courses/anomaly-detection-in-python>
- Xin, R., Liu, H., Chen, P., & Zhao, Z. (2023). Robust and accurate performance anomaly detection and prediction for cloud applications: a novel ensemble learning-based framework. *Journal of Cloud Computing*, 12(1). <https://doi.org/10.1186/s13677-022-00383-6>
- Xu, J., & Duraisamy, K. (2020). Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and Engineering*, 372, 113379. <https://doi.org/10.1016/j.cma.2020.113379>
- Xu, W., He, J., Li, W., He, Y., Wan, H., Qin, W., & Chen, Z. (2023). Long-Short-Term-Memory-Based Deep Stacked Sequence-to-Sequence Autoencoder for Health Prediction of Industrial Workers in Closed Environments Based on Wearable Devices. *Sensors*, 23(18), 7874. <https://doi.org/10.3390/s23187874>
- Yadav, A. (2024, November 30). *Transformer Encoder Explained - Amit Yadav - Medium*. Medium. <https://medium.com/%40amit25173/transformer-encoder-explained-0ed866b69083>
- Yadav, S. (2023, May 23). *Harnessing the Power of Isolation Forest for Anomaly Detection*. CloudThat Resources. <https://www.cloudthat.com/resources/blog/harnessing-the-power-of-isolation-forest-for-anomaly-detection>
- Yi, J. (2020). *Patch SVDD: Patch-level SVDD for Anomaly Detection and Segmentation*. https://openaccess.thecvf.com/content/ACCV2020/papers/Yi_Patch_SVDD_Patch-level_SVDD_for_Anomaly_Detection_and_Segmentation_ACCV_2020_paper.pdf
- Yoon, Y. (2022, March 8). *Isolation Forest Anomaly Detection — Identify Outliers*. Medium. <https://medium.com/%40y.s.yoon/isolation-forest-anomaly-detection-identify-outliers-101123a9ff63>
- Zhang, L., Zhang, W., McNeil, M. J., Chengwang, N., Matteson, D. S., & Bogdanov, P. (2021). AURORA: A Unified Framework for Anomaly detection on multivariate time series. *Data Mining and Knowledge Discovery*, 35(5), 1882–1905. <https://doi.org/10.1007/s10618-021-00771-7>
- Zhao, Y. (2022). *pyod.models.mcd - pyod 2.0.3 documentation*. Readthedocs.io. https://pyod.readthedocs.io/en/latest/_modules/pyod/models/mcd.html
- Zwingmann, T. (2022). *Auto-Detect Anomalies In Time Series Data Using ML*. The Augmented Advantage. <https://blog.tobiaszwingmann.com/p/detect-anomalies-automatically-with-ai>

Note: ChatGPT was used to support grammar correction, language clarity, and typographical edits throughout the thesis.