MSc Thesis
Enhancing Sales Forecasting: Leveraging Retail Sales Data for Advanced AI Predictive Models

by

Vasileios Karlis

Submitted
in partial fulfilment of the requirements for the degree of

Master of Artificial Intelligence

at the

UNIVERSITY OF PIRAEUS

February 2024

Author .... Vasileios Karlis

II-MSc "Artificial Intelligence"

Month 02, 2024

Enhancing Sales Forecasting: Leveraging Retail Sales Data for Advanced AI Predictive Models

By

Vasileios Karlis

Submitted to the II-MSc "Artificial Intelligence" on Month 02, 2024, in
partial fulfillment of the
requirements for the MSc degree

Abstract

This thesis aims to improve sales forecasting in the retail sector through the application of advanced Artificial Intelligence techniques. It addresses the issue of fluctuations in retail sales, which are influenced by a variety of external factors, including economic changes and shifts in consumer behavior. The study develops and evaluates multiple AI forecasting models, such as FBProphet, NeuralProphet, XGBoost, LSTM, TFT and TimeGPT, to enhance the accuracy and flexibility of predictions. Moreover, the thesis provides an in-depth comparison between conventional time series forecasting methods such as ARIMA and the aforementioned machine and deep learning approaches. The findings underscore the superior performance of state-of the-art AI-based models in handling complex patterns and adapting to new data, thereby providing more accurate and adaptable sales forecasts. Additionally, the thesis emphasizes the importance and impact of thorough data preprocessing.

Thesis Supervisor:
Michael Filippakis
Title:
Professor at University of
Piraeus

Acknowledgments

Any opinions, findings, conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the «funding body» or the view of University of Piraeus and Inst. of Informatics and Telecom. of NCSR "Demokritos".

This thesis was prepared and submitted as part of the requirements for the Inter-institutional Postgraduate Program "Artificial Intelligence," a collaborative effort between the University of Piraeus and NCSR Demokritos. I extend my deepest gratitude to my supervisor, Prof. Michael Filippakis, whose unwavering support, guidance, and advice have been instrumental throughout my journey in completing this master's thesis. His invaluable assistance was essential for the fruition of this work.

I would also like to express my sincere appreciation to all the professors and researchers of the program for their valuable knowledge and insights into the field of artificial intelligence, which have greatly enriched my learning experience.

Lastly, I must extend my heartfelt thanks to my family and friends, who have provided me with endless support and encouragement during this period. Their presence and backing have been a constant source of strength for me. Special thanks go to my dog, Kevin, who patiently waited for my research and studies to conclude, so we could enjoy our walks together. His companionship has been a comforting and cherished part of this journey.

Content

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
ARIMA	Autoregressive Integrated Moving Average
XGBoost	Extreme Gradient Boosting
SVM	Support Vector Machines
LSTM	Long Short Term Memory
TFT	Temporal Fusion Transformers
RNN	Recurrent Neural Network
GRU	Gated Recurrent Units
MAE	Mean Absolut Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Square Error
ADF	Augmented Dickey-Fuller
PACF	Partial Auto - Correlation Function
ACF	Auto - Correlation Function

1 Introduction

Sales forecasting in retail is a multifaceted problem, influenced by a myriad of factors ranging from market trends to unforeseen global events. The retail sector's susceptibility to various external influences, including economic shifts, technological advancements, and consumer behavior changes, makes forecasting a particularly challenging task. This thesis seeks to unravel these complexities by employing advanced artificial intelligence (AI) techniques, aiming to transcend traditional forecasting methods and provide more accurate, adaptive, and robust predictions.

1.1 Problem Description

The primary challenge in retail sales forecasting arises from the unpredictability of external events. Factors such as severe weather conditions, global pandemics like COVID-19, and socio-economic changes significantly impact consumer behavior and sales patterns. These events often lead to anomalies in data, making it difficult for traditional forecasting models to adapt and maintain accuracy. The current reliance on models like moving averages in the retail sector has proven inadequate, as they often fail to capture the nuances and irregularities of the market. Therefore, there is the need to address these shortcomings and provide a more effective forecasting approach for a company where the gap between forecasted and actual sales has been notably wide.

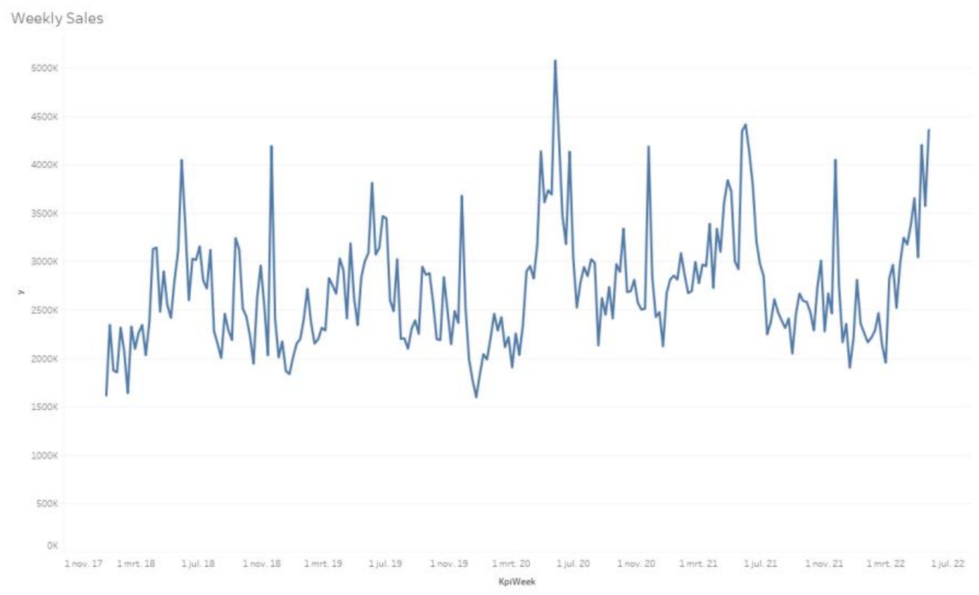


Figure 1: Weekly Sales in Euros (×1000) [1]

To overcome the challenges posed by the unpredictability of external events on retail sales forecasting, it is imperative to adopt more dynamic and resilient forecasting methods. Advanced techniques, such as predictive analytics and machine learning, hold the key to enhancing forecasting accuracy by incorporating a broader spectrum of data inputs, including real-time market trends, consumer behavior analytics, and external environmental factors. These models are adept at identifying complex patterns and adapting to new information, enabling them to adjust forecasts in light of unexpected events more effectively than traditional methods. By integrating these technologies, retailers can achieve a more nuanced understanding of market dynamics, allowing for more accurate and flexible sales predictions. This strategic shift not only helps bridge the gap between forecasted and actual sales but also empowers retailers to make informed decisions in a rapidly changing marketplace.

1.2 Motivation

The primary goal of this thesis is to explore and train artificial intelligence (AI) models with the aim of achieving optimal forecasting accuracy. It is noteworthy that this work endeavors to offer solutions to a real-world business problem faced by a company with an annual turnover of approximately 4.5 billion euros.

This research is motivated by the substantial gap between forecasted and actual sales figures observed in the company's current forecasting practices. The reliance on moving averages and other traditional statistical methods has proven inadequate for capturing the full spectrum of factors influencing sales trends. The adoption of AI-based forecasting models promises a more nuanced understanding of these dynamics, offering the potential for more accurate, timely, and actionable predictions. This thesis aims to bridge the gap between traditional forecasting methodologies and the advanced capabilities offered by AI, providing a comprehensive comparison to demonstrate the value added by AI techniques. Consequently, there is a strong motivation to apply AI methods and compare them with the techniques currently in use.

1.3 Thesis Outline

In the second section, "2. Problem Definition", the nature of the problem is described and defined using mathematical formulas. The third section, "3. Related Work", references the bibliography and presents both traditional and contemporary approaches that have been made in forecasting time series. In the "4. Methods" section, the theoretical framework of the models and methods to be used in this thesis is presented. This chapter will introduce both traditional and state-of-the-art methods. The "5. Experiments" section provides a detailed reference to the experimental process. Specifically, it analyses the dataset and the methods of cleaning and data preprocessing, as well as the results of each model used. Additionally, it examines the parameters in the models that were utilized. It also presents the theoretical background of the metrics used in the evaluation of the forecasting models. Furthermore, this chapter includes an ablation study that concerns experimental processes with suboptimal results. In the semi-final chapter, "6. Conclusion and Future Work", a summary of the results of this thesis is provided, and further work that could be conducted in future research based on the results and conclusions of this thesis is suggested. The "7. References" section contains the bibliographic references in IEEE format.

2 Problem Definition

This thesis endeavors to encapsulate the challenge into forecasting models capable of generating both monthly and daily sales predictions. It is crucial to highlight that daily forecasting presents a greater challenge due to the increased complexity of the problem. The data exhibit higher variation on a daily basis compared to monthly figures, and a large number of factors influence daily sales, making the forecasting task significantly more intricate. The primary challenge in retail sales forecasting stems from the unpredictability of the aforementioned external events as profoundly impact consumer behavior and sales patterns. These events frequently result in data anomalies, posing significant challenges for traditional forecasting models to adapt and maintain accuracy. The prevalent reliance on methods like moving averages has been found wanting, as these approaches often fail to account for the nuances and irregularities present in the market. This inadequacy underscores the pressing need for a more sophisticated forecasting approach, especially given the notable discrepancy between forecasted and actual sales experienced by the company.

Consider a dataset D comprising a sequence of values $(\mathrm{Y}_{\mathrm{t}}, \mathrm{Y}_{\mathrm{t+1}}, \mathrm{Y}_{\mathrm{t+2}}, \dots, \mathrm{Y}_{\mathrm{t+n}})$, where n represents the total sample size. Here, $\mathbf{Y}$ is the dependent variable, indicating the outcome or the response that we aim to explain or predict through our model.

Correspondingly, we have an independent variable X, which consists of features $\{\mathrm{X}_{\mathrm{t}}, \mathrm{X}_{\mathrm{t+1}}, \mathrm{X}_{\mathrm{t+2}}, \dots, \mathrm{X}_{\mathrm{t+n}}\}$. Each Y is a numeric value, whereas each X is a vector that can contain an arbitrary number of numeric values, representing different attributes or characteristics that may influence the outcome Y.

The relationship between the dependent variable Y and the independent variables X can be expressed as follows:

$$Y_t = f(X_t) + \epsilon_t$$

Where f represents the functional form that models the relationship between Y and X, and ϵ_t denotes the error term at time t, capturing the deviation of the observed outcomes from those predicted by the model.

The objective of this thesis is to explore and model the underlying relationship between the dependent and independent variables, employing appropriate machine and deep learning techniques to accurately predict or explain the behavior of Y based on the information contained in X. By addressing the limitations of traditional models and adapting to the volatile nature of retail sales, this research aims to close the gap between predicted and actual sales outcomes.

3 Related Work

In previous sections it introduced the necessary need of time series sales forecasting. These concepts tend to get more and more necessary to create advanced analytics. That is why a lot of research effort has been addressed towards that domain and different approaches, based on statistical methods and machine learning and deep learning have been proposed.

ARIMA and multifactorial linear regression, applied in diverse sectors such as energy and e-commerce. This comparison underscores the criticality of model selection, tailored to specific data traits and forecasting needs, to enhance sales trend predictions and parameter accuracy. This examination aids in substantiating the theoretical underpinnings and practical applications of these models in enhancing business decision-making processes [2].

Early efforts in sales forecasting have predominantly relied on statistical models due to their simplicity and effectiveness in capturing linear trends. The ARIMA model stands out for its ability to model a wide range of time series data with its integrated approach to smoothing out data irregularities. Studies have demonstrated ARIMA's utility in forecasting newspaper sales, showcasing its relevance in sectors undergoing digital transformation [3].

Another approach in time series forecasting is XGBoost. XGBoost has been used in time series analysis, focusing on the sales volume in the retail industry. It was employed due to its superior performance over traditional methods such as ARIMA, SVM, and others, thereby enhancing prediction accuracy. XGBoost has demonstrated efficiency and effectiveness in real-world retail sales forecasting scenarios [4].

In the realm of sales forecasting, the evolution from statistical methods to machine learning and deep learning has marked significant advancements. Traditional statistical approaches, while foundational, often fall short in capturing the dynamic nature of market changes and consumer behavior.

Machine learning techniques, such as random forests and support vector machines, have improved forecast accuracy by utilizing a broader range of sales-related factors. However, their inability to process time-series data directly has led to the exploration of deep learning methods. Among these, Long Short-Term Memory (LSTM) networks have demonstrated superior performance in harnessing the temporal dependencies of sales data. This progression underscores the continuous quest for more precise and adaptive forecasting models in the business sector. [5]

The application of deep learning models such as RNN, LSTM, GRU, and Transformer, In the Time Series Forecasting, has been explored to predict future values based on past data. Among these, Transformer models have shown superior performance due to their ability to handle longer-term dependencies and their use of attention mechanisms. This advancement is particularly evident in experiments with varying look-back window sizes, where Transformers outperform traditional models, especially when predicting further into the future. This shift highlights the evolving landscape of Time Series Forecasting techniques, underscoring the increasing reliance on deep learning methodologies for enhanced prediction accuracy.

4 Methods

In the methods section of this thesis, we explore a comprehensive suite of forecasting techniques applied to two distinct types of problems. Our objective is to evaluate and compare the effectiveness of six advanced forecasting methods: ARIMA (AutoRegressive Integrated Moving Average), FB Prophet (developed by Facebook for forecasting with daily observations), Neural Prophet (an extension of FB Prophet integrating neural network components), XGBoost (eXtreme Gradient Boosting for regression and classification problems), LSTM (Long Short-Term Memory networks for sequential data), TFT (Temporal Fusion Transformers) for interpretable multi-horizon forecasting, and TimeGPT (leveraging generative pre-trained transformers for time series forecasting). This diverse array of methodologies encompasses both traditional statistical models and cutting-edge machine learning algorithms, providing a broad perspective on the current state of forecasting technology. Through rigorous experimentation, this section aims to delineate the strengths and limitations of each approach within the context of our specific forecasting challenges, thereby contributing valuable insights to the field.

At this point, it is important to briefly mention the definition of time series. Time series analysis is a statistical technique that deals with time-ordered sequence of data points. It is widely used in various fields such as economics, finance, environmental science, and engineering to analyze temporal data, forecast future trends, and understand patterns over time. All the models used have applications in time series, and their theoretical background and architecture are described below.

4.1 AutoRegressive Integrated Moving Average – (ARIMA)

The ARIMA model, which stands for AutoRegressive Integrated Moving Average, is a cornerstone in time series forecasting. This model combines three key components: autoregression (AR), differencing (I), and moving average (MA), to capture different aspects of the time series data. The ARIMA model's flexibility in modeling data with various patterns of temporal dependence makes it a widely applied tool in economic forecasting, stock market analysis, and many other fields.

The AR part of the model captures the momentum or the continuation of the time series from one time point to the next. It is predicated on the assumption that the current value of the series can be explained as a linear combination of its previous values. This can be mathematically represented as:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t,$$

where(Y_t) is the value of the series at time (t), $(\phi_1, \phi_2, \dots, \phi_p)$ are the parameters of the model, and (ϵ_t) is white noise.

The I component, standing for "integrated," involves differencing the series to make it stationary. A series is stationary if its statistical properties, such as mean and variance, are constant over time. Non-stationary data are differenced until stationarity is achieved. This process is represented as:

$$\Delta Y_t = Y_t - Y_{t-1},$$

where (ΔY_t) is the difference between the (t^{th}) and $((t-1)^{th})$ observations. For a series that requires (d) differences to become stationary, this process can be extended to $(\Delta^d Y_t)$.

The MA part models the error of the series as a linear combination of error terms from the past. The idea is that the current value of the series can be affected by previous forecast errors. This is represented as:

$$Y_t = \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q}$$

where $\theta_1, \theta_2, \dots, \theta_q$ are the parameters of the model, and ϵ_t is white noise.

Combining these components, the ARIMA model can be expressed as:

$$\phi(B)\Delta^d Y_t = \theta(B)\epsilon_t,$$

where $\phi(B)$and $\theta(B)$ are the polynomials of the lag operator (B) corresponding to the AR and MA parts, respectively. The model is typically denoted as ARIMA(p,d,q) [6], where p is the order of the AR term, d is the degree of differencing, and q is the order of the MA term.

The ARIMA model was first introduced by George Box and Gwilym Jenkins [7] in the early 1970s, revolutionizing the way statistical analysis of time series data was conducted. Their seminal work, "Time Series Analysis: Forecasting and Control," laid the foundation for the ARIMA model's theoretical background and its application to real-world data. The model's comprehensive approach to understanding and forecasting time series data has made it a standard tool in the statistical analysis toolkit.

4.2 FBProphet

FBProphet [8] is a forecasting model developed by Facebook to address the challenges of producing reliable and high-quality forecasts for a variety of time series data, especially in scenarios where there is a scarcity of time series modeling expertise. It is designed as a decomposable model with three main components: trend, seasonality, and holidays, which are combined in a way that allows for easy adjustments by analysts without deep statistical training. The model is formulated as:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

where $g(t)$models non-periodic changes (trend), $s(t)$ captures periodic changes (seasonality), and $h(t)$ accounts for effects of holidays on potentially irregular schedules, with ϵ_t representing any idiosyncratic changes not captured by the model.

FBProphet utilizes a flexible trend model that can accommodate both linear and logistic growth trends, with the capability to automatically detect points of significant change in the trend (changepoints) and adjust the forecast accordingly. This model's trend component can be modified to incorporate domain knowledge through adjustable parameters like carrying capacity in the logistic growth model, providing a practical way for analysts to include their expertise.

For seasonality, FBProphet leverages Fourier series to model complex seasonal patterns, allowing for flexible representation of cyclical behaviors in the data. The model also uniquely handles holidays and special events by including them explicitly in the forecast, enabling it to predict unusual changes in the trend that regular seasonal or trend components might miss.

FBProphet emphasizes the role of the analyst in the modeling process, providing a framework that combines automated forecasting with the ability to manually adjust and improve forecasts based on domain knowledge. This approach is encapsulated in the "analyst-in-the-loop" concept, where the model's intuitive and interpretable parameters enable analysts to refine forecasts without needing extensive statistical training.

Theoretical aspects of FBProphet, including its model components and fitting process, are detailed in the paper "Forecasting at Scale" by Taylor and Letham, which outlines the rationale behind the model's development and its application to real-world forecasting problems.

4.3 NeuralProphet

The theoretical foundation of NeuralProphet [9] is predicated on extending the Prophet forecasting model through the integration of neural networks, thereby enhancing its predictive accuracy and flexibility. At its core, NeuralProphet is designed to model time series data by decomposing it into several interpretable components, such as trend, seasonality, and holidays, similar to its predecessor, Prophet. However, it distinguishes itself by leveraging neural networks to model these components more effectively, especially for capturing complex patterns and dependencies.

NeuralProphet models trends using a piece-wise linear or logistic growth curve, allowing for flexibility in capturing shifts in the trend over time. This is achieved through the specification of changepoints at which the growth rate is allowed to change. The trend component $T(t)$ can be expressed as a function of time t, with adjustments made at changepoints to account for shifts in trend direction or magnitude.

Seasonality is captured using Fourier series to allow for the modeling of complex seasonal patterns. For each seasonality component $S(t)$, a Fourier series is used to model cyclic changes based on the frequency of the seasonality (e.g., daily, weekly, yearly). This approach enables the model to capture both regular and irregular seasonal effects.

NeuralProphet can incorporate the impact of holidays and special events by including additional regressors $H(t)$ in the model. This allows for the adjustment of forecasts based on known events that are expected to impact the time series.

A significant enhancement introduced by NeuralProphet is the AR-Net, an auto-regressive neural network that models the dependency of future values on past values. This component $AR(t)$ enables the model to capture temporal dependencies and autocorrelations within the time series data, improving forecast accuracy for series with strong auto-regressive patterns.

The final forecast $Y(t)$ is obtained by summing the contributions from each component:

$$Y(t) = T(t) + S(t) + H(t) + AR(t) + \epsilon_t,$$

where ϵ_t represents the error term.

NeuralProphet's use of neural networks not only enhances its ability to model complex relationships within the data but also introduces greater flexibility in handling various types of seasonality and trends. The model is trained using stochastic gradient descent (SGD), optimizing a loss function that is typically the Huber loss to mitigate the influence of outliers. This model architecture facilitates the development of highly accurate, scalable, and interpretable forecasts for a wide range of time series forecasting applications.

4.4 XGBoost

XGBoost [10] stands for eXtreme Gradient Boosting, a scalable and efficient implementation of gradient boosting framework. The theoretical background of XGBoost is based on the principle of gradient boosting, which involves constructing new models that predict the residuals or errors of prior models and then combining these models through a weighted sum to make the final prediction. XGBoost improves upon the traditional gradient boosting method by introducing a regularized model formalization to control over-fitting, which makes it effective for a wide range of data science problems.

The objective function in XGBoost is a combination of a loss function that measures the difference between the predicted and actual outcomes, and a regularization term that penalizes complex models to prevent overfitting. Mathematically, the objective function can be written as:

$$L(\phi) = \sum_i l(y_i, \hat{y_i}) + \sum_k \Omega(f_k)$$

Where $L(\phi)$ is the total loss to be minimized, $l(y_i, \hat{y_i})$ is the loss function that measures the difference between the actual value y_i and the predicted value ($\hat{y_i}$), and $\Omega(f_k)$ is the regularization term that penalizes the complexity of the model. The regularization term is composed of both the number of leaves in the tree, to control the model's complexity, and the magnitude of the leaf weights, to encourage simpler models that generalize better. The regularization term $\Omega(f_t)$ is defined as:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda||\omega||^2$$

where T is the number of leaves in the tree, ω are the scores on the leaves, γ is the penalty on the number of leaves, and λ is the L2 regularization term on the weights.

XGBoost also introduces several system-level optimizations for model training and prediction, such as a sparsity-aware algorithm for handling missing data, a weighted quantile sketch for efficient split finding, and parallel and distributed computing to speed up computations. These optimizations enable XGBoost to efficiently handle large-scale and sparse data, making it a powerful tool for a wide range of machine learning tasks.

To tailor the XGBoost algorithm for time series forecasting [4], the dataset is transformed into a supervised learning problem, incorporating lagged observations as features to predict future values. This transformation allows the algorithm to capture temporal dependencies essential for accurate forecasting. The effectiveness of this approach is validated through extensive experiments, demonstrating the algorithm's capability to provide precise forecasts, highlighting its potential for application in various time series forecasting scenarios.

This methodological approach combines theoretical rigor with practical application, ensuring that the forecasting model not only captures the underlying patterns in the data but also remains generalizable and efficient in handling new, unseen data.

4.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) [11] networks represent a significant advancement in the field of deep learning, particularly for processing sequences such as time series, language, and other ordered data. Developed by Sepp Hochreiter and Jürgen Schmidhuber in 1997, LSTMs were designed to overcome the limitations of traditional Recurrent Neural Networks (RNNs), especially the difficulty in learning long-range dependencies due to the vanishing gradient problem. This problem arises during the training of standard RNNs, where the gradients, essential for updating network weights through backpropagation, tend to either shrink to insignificance (vanish) or grow excessively large (explode) as they propagate back through time in the network, making it challenging to learn dependencies between distant elements in a sequence.

The architecture of LSTM networks is specifically crafted to address this issue. It consists of a series of recurrently connected blocks, known as LSTM cells or units, each designed to regulate the flow of information. The central component of an LSTM cells is the cell state, which acts like a conveyor belt, running straight down the entire chain of the network. It has the ability to add or remove information, regulated by structures known as gates.

LSTMs possess three gates, each performing a specific function: the forget gate decides which information should be discarded from the cell state; the input gate updates the cell state with new information; and the output gate controls the output to the next hidden state. The forget gate takes the previous hidden state and the current input, applies a sigmoid function, and outputs a number between 0 and 1 for each number in the cell state, indicating whether to retain or discard the information. The input gate filters out certain information from the current state and the previous hidden state through a sigmoid function and simultaneously, a tanh layer creates a vector of new candidate values. These candidates are then scaled by the output of the sigmoid gate and added to the cell state, thus updating it with new information. The output gate applies a sigmoid function to the current input and the previous hidden state, deciding which parts of the cell state will be output. Then, it passes the cell state through a tanh function (to normalize the values) and multiplies it by the output of the sigmoid gate, determining the next hidden state, which is used for predictions and passed to the next time step.

The mathematical formulations that govern these operations in an LSTM cell include equations for each gate's operation. The forget gate's operation is described by:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

the input gate's workings are captured by:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

and the candidate layer by:

$$\widetilde{C_t} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

the cell state is updated as:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$$

and the output gate's function is represented by:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

and the new hidden state as:

$$(h_t = o_t * \tanh(C_t))$$

In these equations, σ denotes the sigmoid activation function, (tanh)is the hyperbolic tangent function, W and b represent the weights and biases associated with each gate, and $[h_{t-1}, x_t]$ symbolizes the concatenation of the previous hidden state and the current input. The element-wise multiplication is denoted by $*$, and C_t , and h_t are the cell state and hidden state at time t, respectively. Figure 2 summarizes the above operation.

The elegance of LSTM lies in its ability to selectively remember patterns for long durations of time and its efficacy in dealing with the vanishing gradient problem. This selective memory is achieved through the intricate interplay of its gates, which learn what to retain and what to discard. As a result, LSTMs are particularly suited for tasks where the understanding of context, spread over long sequences, is crucial.

LSTMs have been widely applied in various domains, showcasing their versatility and effectiveness. In natural language processing, they are used for tasks like language modeling, text generation, and sentiment analysis. They have also been pivotal in the development of machine translation systems and speech recognition technology. Beyond language, LSTMs have found applications in time-series analysis, such as predicting stock market movements or weather patterns.

Training LSTMs, however, is not without its challenges. The complexity of their architecture can lead to a substantial computational overhead. Moreover, the design of the network architecture and the tuning of hyperparameters require careful consideration and experimentation. Despite these challenges, the benefits of LSTMs in learning from and making predictions based on long sequences of data have solidified their position as a key tool in the deep learning toolkit.

In summary, the development of LSTM networks marked a significant advancement in the field of deep learning, offering a robust solution to the problem of learning long-term dependencies. Their unique structure, characterized by the interplay of the cell state and multiple gates, enables them to effectively capture temporal relationships in data, a capability that is crucial in many complex sequence modeling tasks. The theoretical understanding of LSTMs, from their architecture to the mathematical principles governing their operation, is essential for leveraging their full potential in a wide range of applications, particularly those involving intricate sequential data.

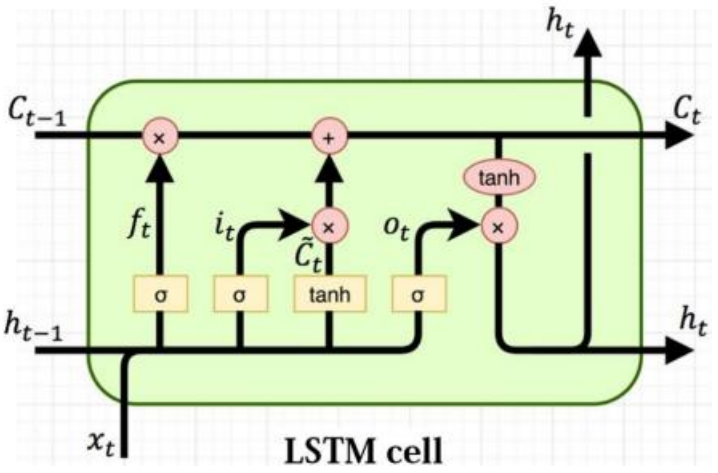


Figure 2: Structure of the LSTM cell.

4.6 Temporal Fusion Transformer (TFT)

The Temporal Fusion Transformer (TFT) [12], represents a sophisticated approach to multi-horizon time series forecasting that integrates multiple data inputs—including static, known future inputs, and historical time series data—to accurately predict future values. By leveraging deep learning techniques, TFT addresses the challenges of capturing complex temporal relationships and providing interpretable insights into how different features and their temporal dynamics influence forecasts. This extended and analytical exploration delves into the architecture, functionality, and mathematical formulations that underpin TFT's forecasting capabilities.

At the heart of TFT's architecture are gating mechanisms that enable the model to manage information flow effectively. These mechanisms are crucial for filtering out irrelevant features and focusing on the most informative aspects of the input data for forecasting tasks. By employing a combination of input, forget, and output gates—similar to those found in LSTM networks—TFT can selectively remember or forget information, thus capturing long-term dependencies in time series data.

Variable selection networks within TFT further refine its ability to concentrate on relevant features. These networks employ attention mechanisms to dynamically adjust the importance of different input features based on their predictive power. This not only enhances model performance but also contributes to the interpretability of the forecasting process, allowing users to understand which features are most influential in predicting future values.

A key feature of TFT is its use of interpretable multi-head attention mechanisms, which enable the model to attend to different segments of the input sequence simultaneously. This is expressed mathematically as:

$$\text{Attention}\ (Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q, K, and V represent queries, keys, and values respectively, and d_k is the dimension of the keys. This mechanism allows TFT to capture complex temporal patterns and dependencies, enhancing its forecasting accuracy.

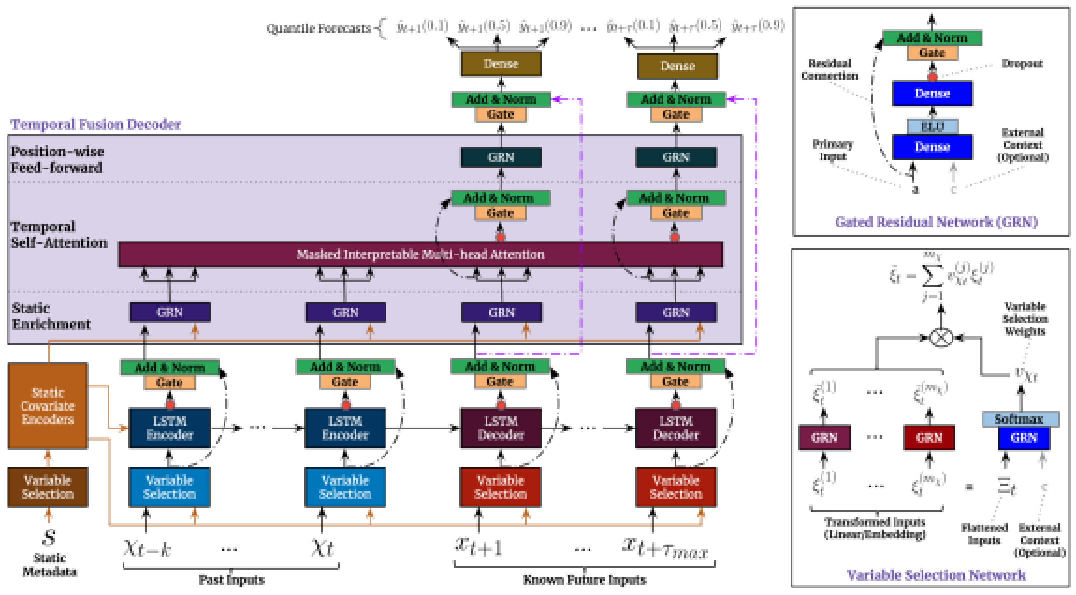
TFT combines convolutional layers for local temporal processing with recurrent layers (or LSTM) to capture global temporal dependencies. This hybrid approach enables the model to understand both short-term fluctuations and long-term trends within the data, making it adept at forecasting over multiple horizons.

For probabilistic forecasting, TFT employs a quantile loss function, allowing it to predict a distribution of possible outcomes rather than just point estimates. The quantile loss is defined as:

$$QL_\tau(y, \hat{y}) = max\ (\tau(y - \hat{y}), (\tau - 1)(y - \hat{y}))$$

Where y is the true value, $\hat{y}$ is the predicted value, and τ is the quantile level. This approach provides valuable insights into the uncertainty of forecasts, which is essential for risk management and decision-making processes.

The Temporal Fusion Transformer offers a powerful and interpretable framework for multi-horizon time series forecasting. By integrating various data inputs and employing advanced techniques such as gating mechanisms, variable selection networks, and multi-head attention, TFT can accurately capture complex temporal dynamics. Its use of quantile loss for probabilistic forecasting further enhances its utility in real-world applications, providing not only precise forecasts but also measures of uncertainty. Through this extended exploration, we gain deeper insights into the theoretical underpinnings and practical applications of TFT, highlighting its significance in the field of time series analysis. The architecture of TFT is depicted by Figure 3.



Quantile Forecasts
Dense
Add & Norm
Gate
Temporal Fusion Decoder
Position-wise Feed-forward
GRN
Temporal Self-Attention
Masked Interpretable Multi-head Attention
Static Enrichment
Static Covariate Encoders
LSTM Encoder
LSTM Decoder
Variable Selection
Static Metadata
Past Inputs
Known Future Inputs
Residual Connection
Dropout
ELU
Primary Input
External Context (Optional)
Gated Residual Network (GRN)
Variable Selection Weights
Softmax
Transformed Inputs (Linear/Embedding)
Flattened Inputs
External Context (Optional)
Variable Selection Network

Figure 3: TFT architecture. TFT inputs static metadata, time-varying past inputs and time varying a priori known future inputs. Variable Selection is used for judicious selection of the most salient features based on the input. Gated Residual Network blocks enable efficient information flow with skip connections and gating layers. Time-dependent processing is based on LSTMs for local processing, and multi-head attention for integrating information from any time step [13].

4.7 TimeGPT

TimeGPT [14] introduces a groundbreaking approach to time series forecasting by leveraging the advanced capabilities of foundation models, specifically designed to address the complexities inherent in predicting future values from historical time series data. At its core, TimeGPT is a model that capitalizes on the Transformer architecture, renowned for its self-attention mechanism, to enhance the forecasting of time series across diverse datasets and domains. This model is distinctive for its ability to perform robust predictions without the necessity for retraining on new datasets, thus embodying the principles of transfer learning and generalization.

The theoretical foundation of TimeGPT is anchored in its unique ability to process and learn from a vast array of time series data, thereby encapsulating the rich temporal dynamics and patterns prevalent across various fields such as finance, healthcare, and energy, among others. The model is pre-trained on a substantial corpus of time series data, encompassing over 100 billion data points from multiple domains, enabling it to capture a broad spectrum of temporal dependencies and nuances.

Mathematically, TimeGPT's forecasting capability can be formulated as estimating the conditional probability distribution:

$$\mathrm{P}(\mathrm{y}[\mathrm{t}+1{:}\,\mathrm{t}+\mathrm{h}] \mid \mathrm{y}[0{:}\,\mathrm{t}],\ \mathrm{x}[0{:}\,\mathrm{t}+\mathrm{h}])$$

where $y[t+1{:}\,t+h]$ represents the future values to be forecasted, $y[0{:}\,t]$ denotes the historical observations, and $x[0{:}\,t+h]$ encompasses any exogenous variables that may influence the forecast. The model, denoted by (f_θ) parameterized by θ, learns to approximate this distribution through a training process that optimizes for prediction accuracy across a wide range of time series characteristics, including trend, seasonality, and volatility.

The process of transfer learning in TimeGPT is twofold: zero-shot learning and fine-tuning. Zero-shot learning allows the model to be directly applied to new forecasting tasks without any additional training, leveraging the knowledge it has acquired during the pre-training phase. In contrast, fine-tuning involves adjusting the model's parameters slightly on a target dataset to tailor its predictions more closely to the specific characteristics of the new task. This dual approach ensures that TimeGPT is not only versatile across different domains but also capable of delivering high precision forecasts.

Furthermore, TimeGPT introduces a novel method for uncertainty quantification in forecasts. By employing probabilistic forecasting techniques, the model provides not just point forecasts but also prediction intervals, offering insights into the confidence levels of the forecasts and the potential range of future values. This aspect is particularly crucial for decision-making processes where understanding the uncertainty of predictions can significantly impact the outcomes.

In conclusion, TimeGPT represents a significant leap forward in the realm of time series forecasting. By harnessing the power of large-scale pre-trained models and the Transformer architecture, TimeGPT sets a new standard for accuracy, efficiency, and generalization in forecasting tasks. Its ability to perform zero-shot learning and fine-tuning, along with its proficiency in uncertainty quantification, positions TimeGPT as a versatile and powerful tool for a wide array of forecasting applications.

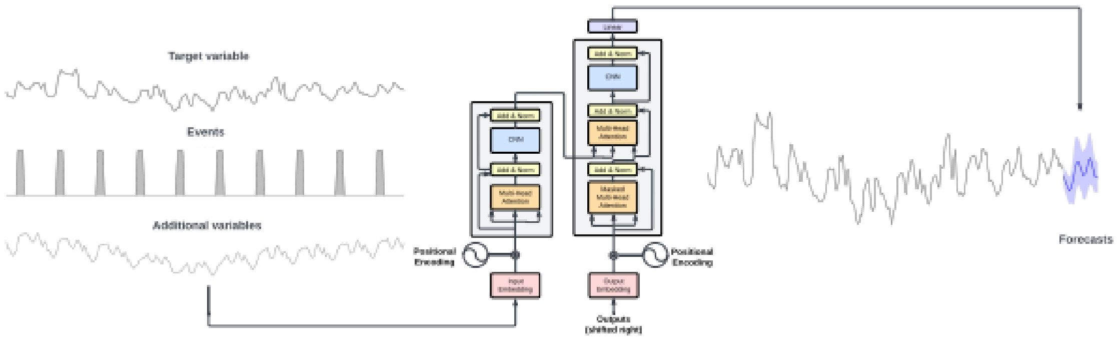


Figure 4: Inference of new time series. TimeGPT takes the historical values of the target values and additional exogenous variables as inputs to produce the forecasts. We rely on conformal predictions based on historic errors to estimate prediction intervals [14].

5 Experiments

5.1 Experimental Setting

In this section, we delve into an in-depth examination of the dataset employed in this study, offering comprehensive insights into its structure, origin, and the preprocessing methods applied to ensure its suitability for the analysis conducted. Furthermore, we provide a thorough explanation of the metrics used to evaluate the performance and effectiveness of the methodologies implemented. This includes both the rationale behind the selection of these metrics and an explanation of how they contribute to a nuanced understanding of the results obtained.

In our approach towards evaluating the effectiveness of stochastic models, we conduct a series of 10 independent experiments. This methodology is deliberately chosen to rigorously assess the variability and reliability of these methods, thereby capturing the inherent randomness and providing a robust statistical foundation for our conclusions. In contrast, for deterministic models, where outcomes are the at every iteration, we perform a single experiment. This differentiation in experimental design between stochastic and deterministic models is critical, as it allows for an appropriate evaluation framework that aligns with the nature of each model type, ensuring that our analysis is both accurate and reflective of the underlying model characteristics.

5.1.1 Dataset

In this section, we will provide a detailed description of the dataset provided to us. The initial dataset consists of 788,620 rows, encompassing daily total sales in euros for the entirety of the company's physical stores as well as its online store. Chronologically, the dataset spans five calendar years, starting from January 1, 2018, and ending on December 31, 2023. The dataset originally contained 9 columns as described in the Table 1 below:

Table 1: Initial columns of the dataset

Column Name	Description
year	The year of sales (e.g. 2019)
month	The year of sales (e.g. 11)
week	The week of the year (e.g. 44)
week_day	The day of the week (e.g. Tue from Tuesday)
date	The date (e.g. 2022-08-20)
store_id	The distinct alphanumeric (id) of the store
store_descr	The description of the according to the address and area
sales_dt	The net sales (before tax)
sales	The gross sales (after tax)

However, from the aforementioned columns, the [sales_dt] column, which pertain to pre-tax sales, and [store_descr] and [store_id], which provide information (ID and Description for each store), were dropped. These columns were dropped because the problem at hand focuses on predicting the company's total gross sales rather than sales for each store individually. Therefore, after the initial data cleaning stage, the dataset comprises the following columns, as detailed in the Table 2 below. It is noted that the reduction in rows results from aggregating store sales by date.

Table 2: Dataset columns after first cleaning process.

year	month	week	week_day	date	sales

At this point, it is important to reference the lags that result from the differences in sales, which we utilize as features in the XGBoost and LSTM models.

Let's assume the current day is t, the day before the forecast begins, then the sales will be Y_t . Thus, the lags can describe as:

$$Y_{(t-1)} - Y_{(t-2)},\ Y_{(t-1)} - Y_{(t-3)},\ \dots\ ,\ Y_{(t-1)} - Y_{(t-n)}$$

Subsequently, the dataset is divided into two cases:

- monthly
- daily

Monthly dataset

For the dataset used for monthly predictions, total sales per month are added, resulting in a dataset with 60 rows, i.e., 12 months for the five years.

Next, the dataset is split into train and test sets. Thus, we keep four years for training, from 2018 to 2022, and predictions and model evaluations are conducted for 2023 into test set. In the Figure 5 is presented the train set with blue color and the test set with orange color. Furthermore, from the Figure 5, we can observe an increasing trend as well as seasonality. Additionally, we can identify certain unexpected events, primarily in the period between 2020 and 2022, which are likely attributable to the variations in demand during the COVID-19 period.

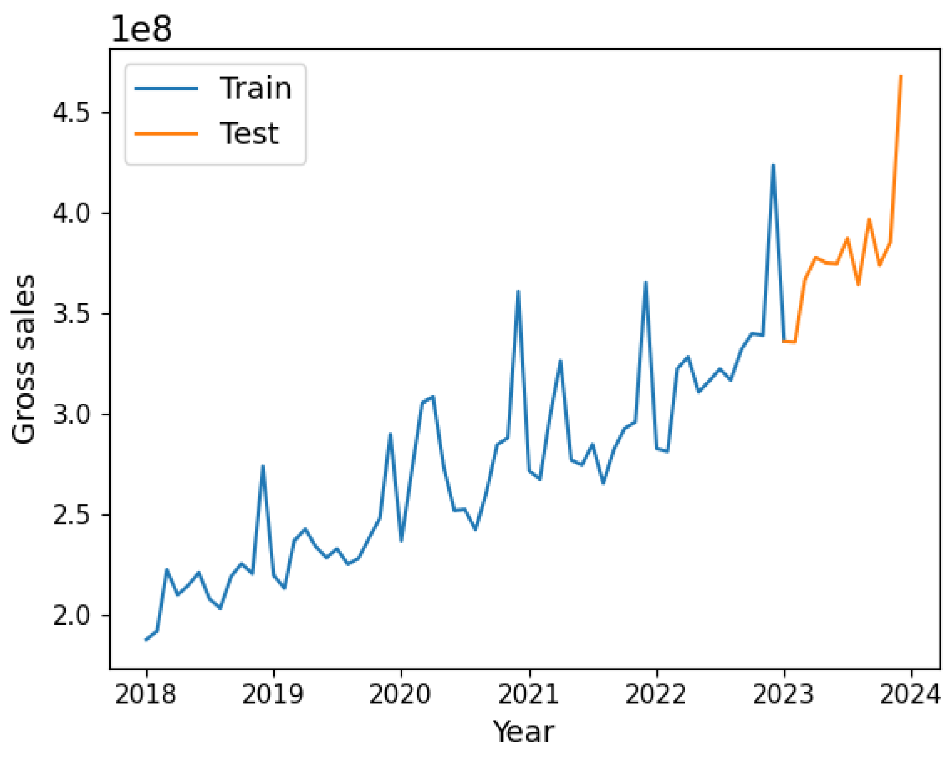


Figure 5: Monthly dataset splits into train and test sets.

In Figure 6 and Figure 7 present two boxplots. Figure 6 relates to the boxplot for the training dataset, while Figure 7 pertains to the boxplot for the test set. These diagrams assist in understanding the differences between the populations. The distances between various segments of the boxplot reveal the extent of dispersion and the asymmetry of the data.

However, outliers, in the case of monthly data, are observed in only a few instances in both the test and train datasets. Therefore, it is not deemed necessary to apply methods for their removal.

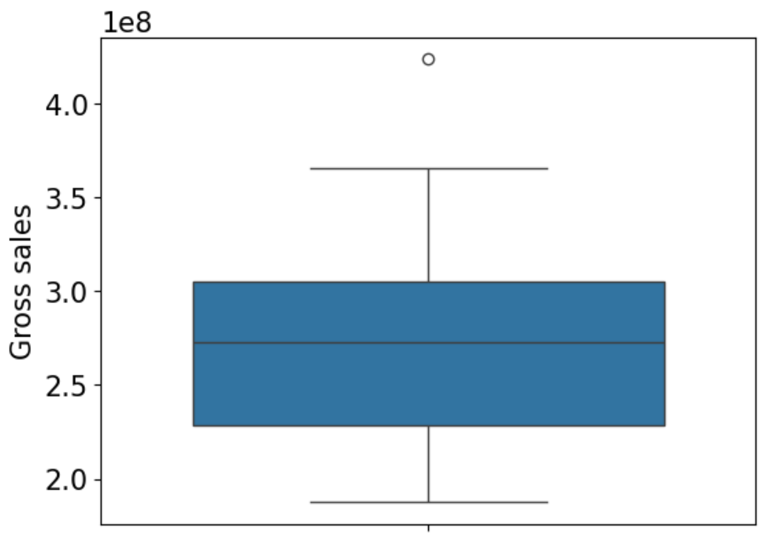


Figure 6: Boxplot for monthly train dataset.

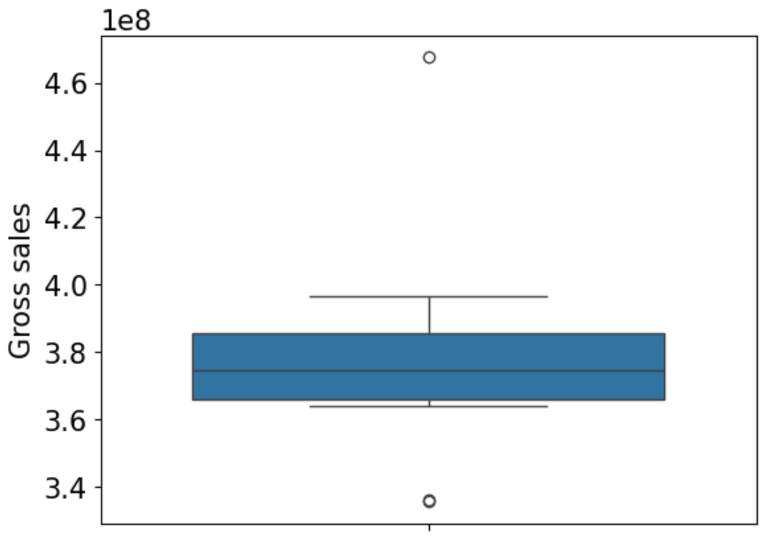


Figure 7: Boxplot for monthly test dataset.

Finally, for the monthly prediction problem, five different models were used: ARIMA, fbProphet, NeuralProphet, XG-Boost, TimeGPT. The Table 3 presents the colums, i.e., the columns that were utilized for each model.

Table 4: Columns are used in each model of the monthly dataset

ARIMA	FBProphet	NeuralProphet	XG - Boost	Time GPT
date, sales	date, sales	date, sales	date, month, week, week_day, sales	date, sales

Daily Dataset

The daily data, as well as the monthly dataset, is split into train and test sets. We keep 5 years and 10 months, 2134 days(01/01/2018 -05/11/2023) for training and 6 days(06/11/2023 – 11/11/2023) as test set for predictions and model evaluations. Additionally, after the split, we kept 20% of the train set for validation. More specifically, we kept 20% from the beginning of the train for validation and not random values, so as not to break the continuity of the time series. The reason we chose this specific period for our test set is because it represents a typical period. We aim to obtain results for our models that reflect a standard time frame where national holidays, movable feasts, or the beginning of the month, which often involves salary and pension payments, are not included. These events do not follow a specific frequency but are determined by various decisions from companies, governments, and trade associations. All these factors can influence demand according to domain experts.

In the Figure 8 the train set is presented with blue color and the test set with orange color. Furthermore, from the Figure 8, we can observe an increasing trend as well as seasonality. We can also discern many zero or very low values. These arise from days with no sales, such as holidays and Sundays. However, since the original data for days with zero sales had no entries, resulting in a disruption of the time series continuity, we decided to perform a left join of a calendar data-frame onto our data, thereby creating zero entries.

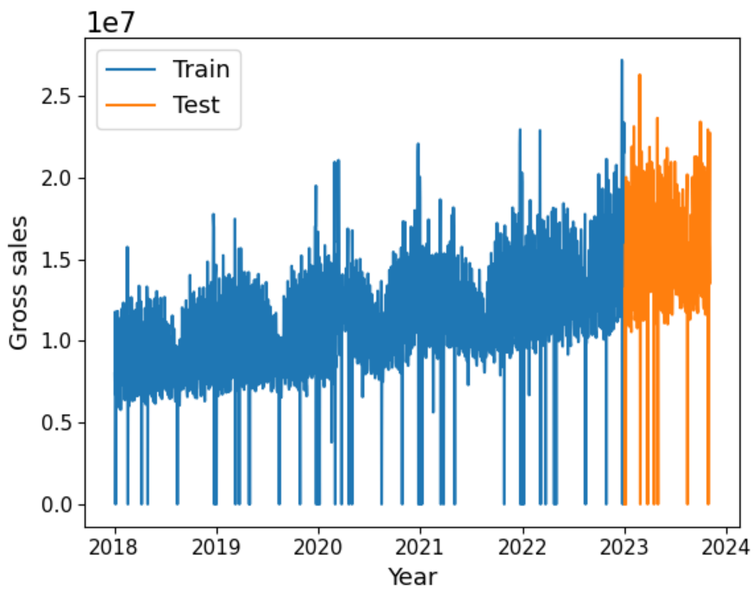


Figure 8: Daily dataset splits into train and test sets.

Regarding the challenges in daily dataset are different. Sundays are removed from the entire dataset because only a small subset of stores, which are not open all year but only for a specific period decided by the management based on undefined factors, operate on Sundays. Additionally, some Sundays throughout the year the stores remain open, but again, their frequency is not fixed and certain. Therefore, it was decided to exclude Sundays from the dataset to avoid confusion in the time series.

Another challenge is that on days with no sales, such as holidays, these do not appear as zero values in the time series, thereby breaking the continuity of the series. For this reason, we created a dataset as a complete calendar and merged it with our dataset to generate zero entries on these dates.

The next challenge was dealing with outliers. These outliers, apart from zero values, were also due to unexpected events. For example, the quarantine period , during COVID-19, and certain periods with extreme weather conditions negatively affected sales. Additionally, system errors or system tests on holidays would create data-entries in the database (e.g., 0.7 euros when the average weekly sale is 12,000,000 euros). We can discern the accumulated low outliers in the boxplots in Figure 9 and Figure 10.

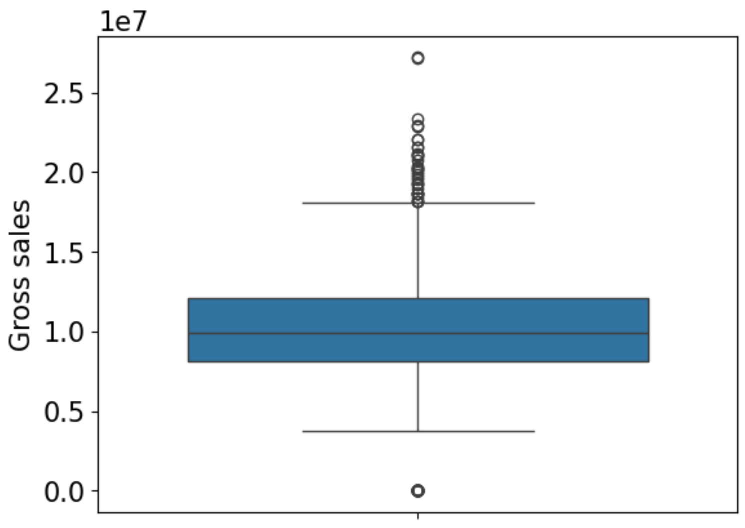


Figure 9: Boxplot for daily training dataset before the removal of the outliers.

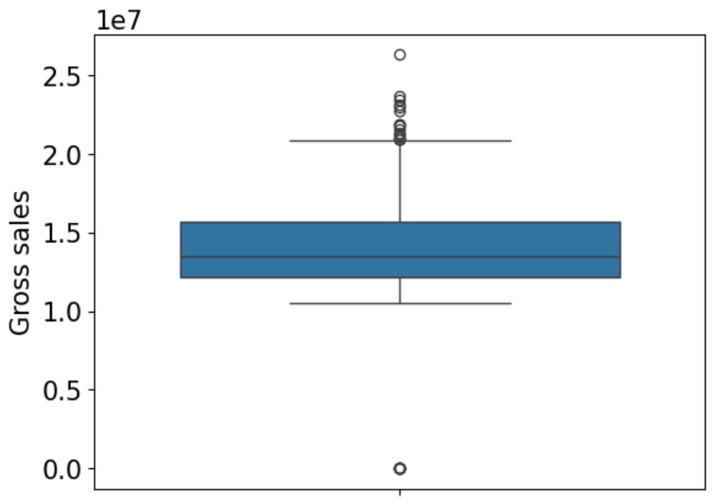


Figure 10: Boxplot for daily testing dataset before the removal of the outliers.

Therefore, it was considered appropriate to remove all these extreme values since they did not reflect real demand. As their simple removal would cause continuity issues in the time series, it was decided to replace very low outliers based on the following conditions which cover all cases of outlier and are be detailed with pseudocode in *Algorithm 1*.

Algorithm 1 Outlier Replacement in Sales Dataset

```
1: Initialize an empty list outliers
2: for $i$ = 1 to 2134 do
3:     if $sales[i] < 100000$ then
4:         Add $i$ to outliers
5:     end if
6: end for
7: for $i$ = 1 to 2134 do
8:     if $i > 6$ and $i < 2129$ then
9:         if ($i - 6$ not in outliers) and ($i + 6$ not in outliers) then
10:            $sales[i] \leftarrow$ AVERAGE($sales[i - 6], sales[i + 6]$)
11:        else if ($i - 6$ in outliers)
           and ($i + 6$ not in outliers)
           and ($i + 12$ not in outliers) then
12:            $sales[i] \leftarrow$ AVERAGE($sales[i + 6], sales[i + 12]$)
13:        else if ($i + 6$ in outliers)
           and ($i - 6$ not in outliers)
           and ($i - 12$ not in outliers) then
14:            $sales[i] \leftarrow$ AVERAGE($sales[i - 6], sales[i - 12]$)
15:        end if
16:    else if $i \leq 6$ then
17:        $sales[i] \leftarrow$ AVERAGE($sales[i + 6], sales[i + 12]$)
18:    else if $i \geq 2129$ then
19:        $sales[i] \leftarrow$ AVERAGE($sales[i - 6], sales[i - 12]$)
20:    end if
21: end for
```

After the process of removing outliers, we can see in Algorithm 1 that we have a clean dataset without extreme fluctuations that resulted from unexpected events or system errors in data entry.

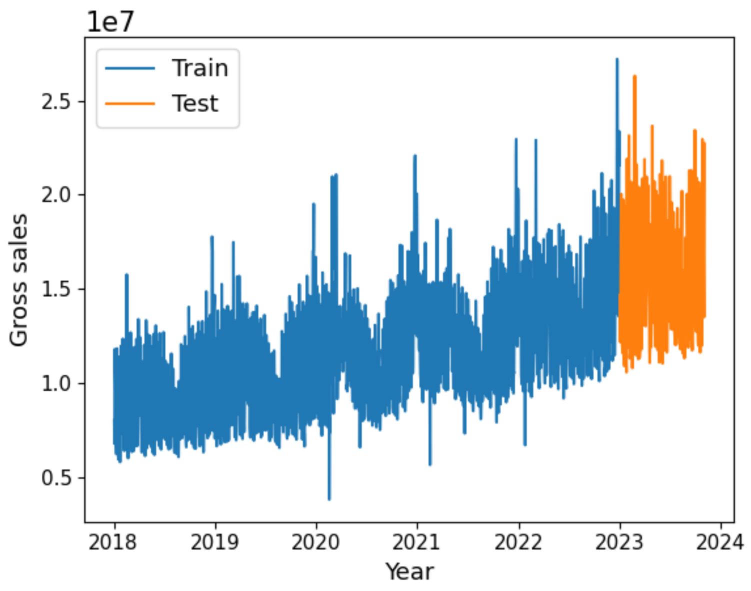


Figure 11: Daily dataset splits into train and test sets and after removal of the outliers.

Additionally, it is necessary to examine the variance of the dataset after the removal of the low outliers. As seen in the Figure 11 and Figure 12 for both the train set and the test set, low outliers are no longer observed. Regarding the high outliers, we decided not to take any action because in most cases they represent an increase in sales during periods of high demand, such as Christmas, which is something we want the prediction model to learn.

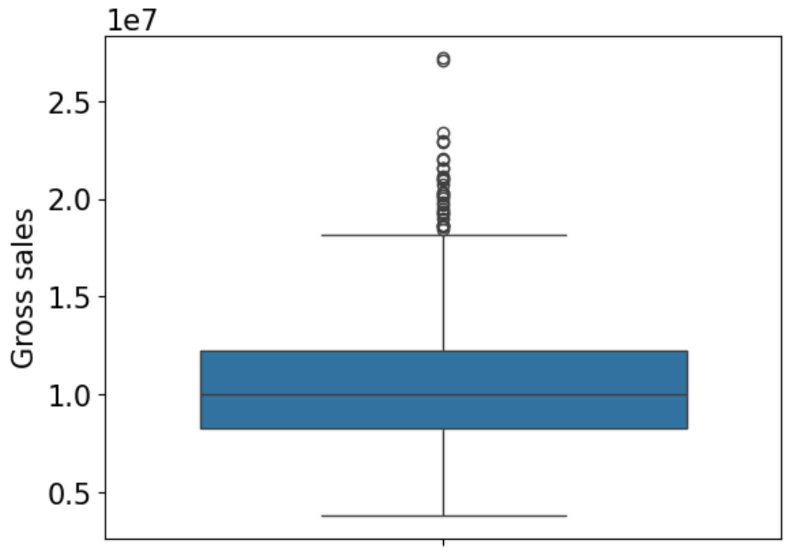


Figure 12: Boxplot for daily training dataset after the removal of the outliers.

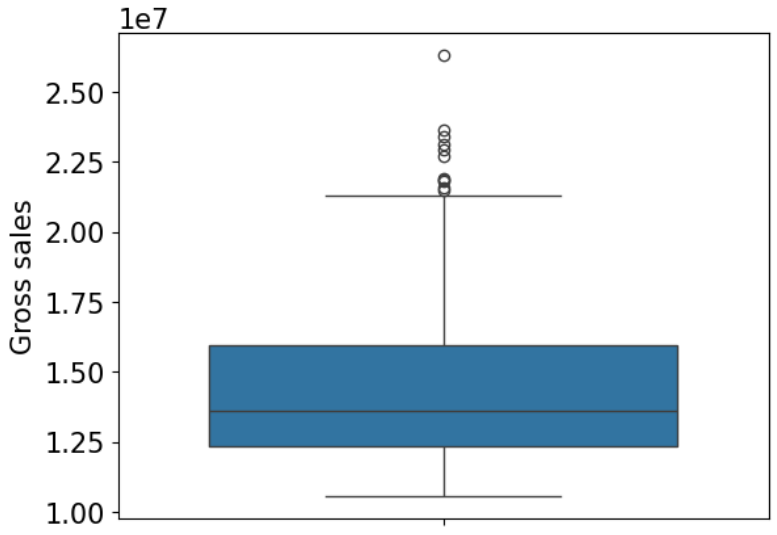


Figure 13: Boxplot for daily testing dataset after the removal of the outliers.

Finally, for the daily prediction problem, five different models were used: fbProphet, NeuralProphet, LSTM, TFT, TimeGPT. The Table 5 presents the colums, i.e., the columns that were utilized for each model.

Table 5: Columns are used in each model of the daily dataset

FBProphet	NeuralProphet	LSTM	TFT	TimeGPT
date, sales	date, sales	date, month, week, week_day, sales	date, month, week, week_day, sales	date, sales

5.1.2 Metrics

In this section of the thesis, we present the metrics utilized for the evaluation of the models. Specifically, we employ Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Mean Squared Error (MSE) as detailed below. The differences between Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) primarily lie in how they quantify the accuracy of models in regression analysis, forecasting, and other predictive analytics applications. Each of these metrics evaluates the difference between the actual values and the predicted values generated by a model, but they do so in different ways, leading to distinct sensitivities and interpretations.

The differences between Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) primarily lie in how they quantify the accuracy of models in regression analysis, forecasting, and other predictive analytics applications. Each of these metrics evaluates the difference between the actual values and the predicted values generated by a model, but they do so in different ways, leading to distinct sensitivities and interpretations.

Mean Square Error (MSE)

The Mean Square Error (MSE) is a widely used measure in statistics, signal processing, and machine learning for estimating the performance of predictive models. It quantifies the difference between the values predicted by a model and the actual values observed. The MSE is particularly useful because it provides a single value that encapsulates the quality of a predictor in predicting outcomes across a dataset, making it an essential tool for model evaluation and comparison.

Mathematically, the MSE is defined as the average of the squares of the errors. The error is the amount by which the prediction deviates from the actual value. For a set of n predictions, where y_i represents the actual value and $\widehat{y_i}$ represents the predicted value for the ith observation, the MSE is given by the formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2$$

This formula encapsulates the essence of MSE by squaring the difference between the predicted and actual values before averaging these squared differences over all observations in the dataset. The squaring is crucial because it ensures that errors are positive and emphasizes larger errors more than smaller ones, making MSE sensitive to outliers in the dataset.

One of the key properties of MSE is that it is always non-negative, as the square of a real number cannot be negative. The best possible score is 0, which indicates that the model predictions perfectly match the actual values. However, achieving a zero MSE is often impossible in practical scenarios, especially in the presence of noise in the data or when the model does not perfectly capture the underlying data generation process.

Despite its widespread use, MSE has limitations. It is highly sensitive to outliers due to the squaring of errors. This means that a small number of very large errors can dominate the MSE, potentially giving a misleading impression of the overall model performance. Furthermore, MSE does not necessarily provide a clear indication of how the model performs on different segments of the data unless it is decomposed or analyzed in conjunction with other metrics.

In conclusion, MSE is a fundamental metric for assessing the accuracy of predictive models. Its simplicity and the rich theoretical background make it an indispensable tool in the repertoire of methods for model evaluation. Nonetheless, it is essential to consider its characteristics and limitations when interpreting its value and to complement it with other metrics for a comprehensive assessment of model performance.

Mean Absolute Error

Mean Absolute Error (MAE) is a widely used statistical measure to evaluate the performance of predictive models in various fields such as finance, weather forecasting, and machine learning. It quantifies the accuracy of a model by calculating the average magnitude of errors between predicted values and observed (actual) values, without considering the direction (positive or negative) of those errors. The simplicity and interpretability of MAE make it a popular choice for assessing model performance, especially when it is important to understand the average error magnitude in the same units as the data being predicted.

Mathematically, the MAE is defined as the average of the absolute differences between the predicted values $\widehat{y_i}$ and the observed values y_i over a dataset of N observations. The formula for MAE is given by:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\widehat{y_i} - y_i|,$$

where N is the total number of observations in the dataset, $\widehat{y_i}$is the predicted value for the ith observation, $y_i$is the actual value for the ith observation, $|\widehat{y_i} - y_i|$

represents the absolute difference between the predicted and actual values for the *ith* observation.

The absolute value in the MAE formula ensures that all errors are treated equally, regardless of their direction. This contrasts with other metrics such as the Mean Squared Error (MSE), where the squaring of errors amplifies the influence of larger errors on the overall metric.

One of the main advantages of MAE is its interpretability. Because it is expressed in the same units as the data, the MAE can be easily understood and communicated to non-technical stakeholders. It tells us how wrong the predictions are on average in the dataset. For example, in the context of predicting house prices, an MAE of 50,000 means that, on average, the predicted prices are \$50,000 away from the actual prices.

However, while MAE provides a straightforward measure of average model error, it has its limitations. It treats all errors equally, which may not be desirable in applications where larger errors are more significant than smaller errors. Moreover, MAE does not provide any indication of the error distribution, meaning it could be the same for models that make consistently small errors and" for models that make a few large errors along with many accurate predictions.

In conclusion, Mean Absolute Error is a valuable metric for assessing the average error magnitude of predictive models, offering clear interpretability in the same units as the predicted variable. Despite its simplicity, researchers and practitioners should consider its characteristics and limitations in the context of their specific application to ensure it aligns with their model evaluation objectives.

Mean Absolute Percentage Error (MAPE)

The Mean Absolute Percentage Error (MAPE) is a statistical measure used to assess the accuracy of a forecasting model. It calculates the average of the absolute percentages by which forecasts differ from actual values, providing a clear, intuitive measure of forecast accuracy that can be easily interpreted. The MAPE is particularly useful in scenarios where the magnitude of the prediction error relative to the actual value is more important than the error's absolute size. The formula for MAPE is defined as follows:

$$\mathrm{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{A_i - F_i}{A_i} \right|$$

where n is the number of observations, A_i represents the actual values, and F_i denotes the forecasted values. The absolute value of the difference between the actual and the forecasted values is divided by the actual value to calculate the percentage error for each observation. This process ensures that all errors are treated equally, regardless of their direction (positive or negative). The errors are then averaged across all observations and multiplied by 100 to convert the result into a percentage.

One of the main advantages of MAPE is its interpretability. A MAPE of 5%, for example, tells us that the forecasted values are, on average, within 5% of the actual values, which provides a straightforward way to understand the model's accuracy. However, the MAPE also has its limitations. It can be heavily influenced by zero or near-zero actual values, as these can lead to undefined or infinitely large percentage errors, skewing the overall MAPE. Furthermore, because it is based on percentage errors, the MAPE can be more sensitive to relative errors in lower actual values than to the same absolute error in higher actual values.

Despite these limitations, the MAPE remains a popular choice for evaluating forecast accuracy in various fields, including finance, supply chain management, and economics. It provides a useful benchmark for comparing the performance of different forecasting models or methods under consistent conditions. When using MAPE in research or practical applications, it's important to be aware of its assumptions and limitations, considering complementary metrics if necessary to obtain a comprehensive evaluation of a model's performance.

We can summarize the evaluation metrics as follows:

- Sensitivity to outliers : MSE > MAE > MAPE
- Interpretability: MAE (units of the original data) > MSE (squared units) > MAPE (percentage, but can be skewed by values close to zero)
- MSE is often preferred in machine learning because it penalizes larger errors more harshly, which can be desirable in many contexts.
- MAE is used when you want a measure that is robust to outliers.
- MAPE is useful when you want to express the error as a percentage of actual values, making it easier to communicate the magnitude of errors in a more understandable way, particularly in business settings or forecasting.

5.1.3 Models Hyperparameters

In the current section, we will introduce the hyperparameters of the models. To the extent that it was feasible, given computational resources, time constraints, and literature review, fine-tuning of the model hyperparameters was conducted. Fine-tuning is a critical process that involves adjusting the hyperparameters of a model to optimize its performance. This process is heavily dependent on the availability of computational resources, as extensive experiments can be computationally expensive. Time is also a significant factor, as finding the optimal set of hyperparameters can be time-consuming, requiring multiple iterations of training and evaluation. Furthermore, the literature plays a crucial role in guiding the initial selection of hyperparameters and fine-tuning strategies, offering insights from previous research on effective ranges and values for different model architectures and problem domains. Through this meticulous process, we aimed to enhance the accuracy, efficiency, and generalizability of our models, ensuring that they are well-suited to address the specific challenges and complexities of our research tasks.

ARIMA. In determining the optimal parameters for our ARIMA model, we began by examining the stationarity of the time series. The Augmented Dickey-Fuller (ADF) test yielded a p-value of 0.991603 and a test statistic of 0.797977, which was higher than the critical values at the 1%, 5%, and 10% levels. This indicated the presence of a unit root and non-stationarity in the original series, necessitating differencing. Despite the lack of explicit visual evidence of stationarity in the 1st and 2nd order differenced plots Figure 14, we proceeded under the assumption of a potentially required second order of differencing, informed by the conservative interpretation of the ADF test results, setting $d = 2$.

Next, we identified the autoregressive term p by examining the Partial Autocorrelation Function (PACF) plot Figure 16, where the first lag was significant outside of the limit and the second one is also out and negative so we can select the order of the p=0.

Finally, the Moving Average term q was estimated from the Autocorrelation Function (ACF) plot Figure 15. Given the gradual decline and the fact that all lags quickly fell within the confidence interval, we hypothesized a non-existent MA component, setting q=1. So the hyperparameters ARIMA(0, 2, 1) were test and provide the best results.

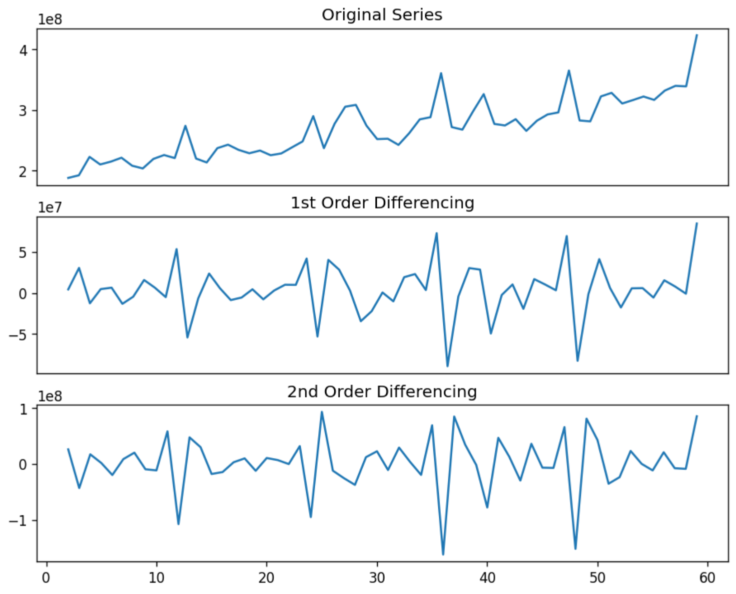


Figure 14: ARIMA Differencing plot

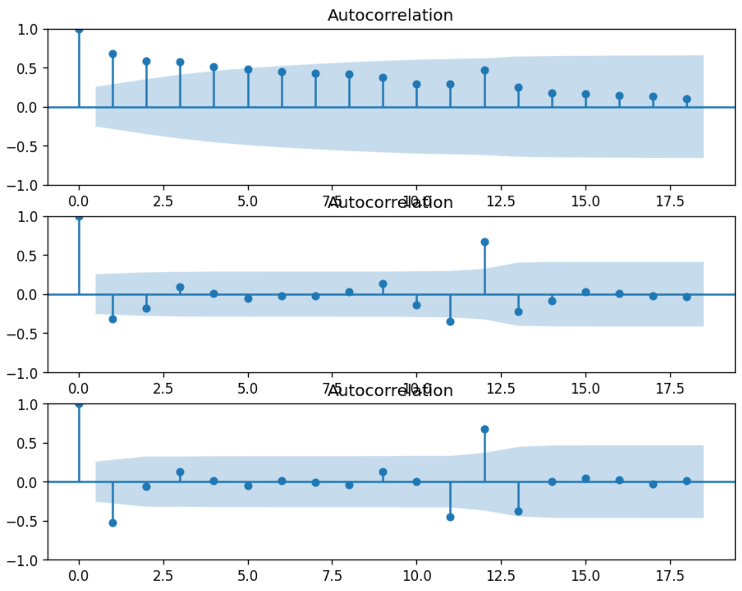


Figure 15: ARIMA ACF plot

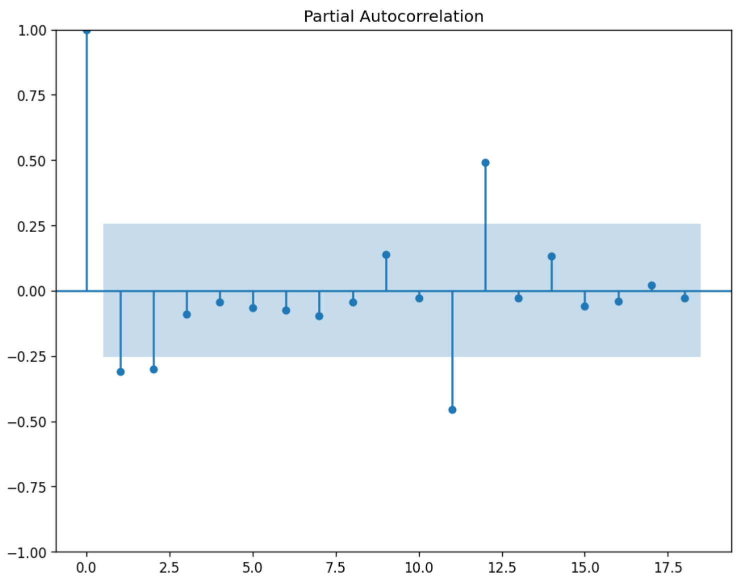


Figure 16: ARIMA PACF plot

FBProphet. If the model does not have hyperparameters except for seasonality which is set to M (Monthly) for the monthly problem and D (Daily) for the daily issue.

Neuralprophet. The model used the following hyperparameters in Table 6. Despite being a neural model, Neural Prophet is designed in such a way that it does not require scaling, except in cases where there are exogenous variables as features, such as weather conditions.

Table 6: NeuralProphet hyperparameters.

Hyperparameters	**Daily**	**Monthly**
epochs	[200] early stop	[100] early stop
yearly_seasonality	[True]	[True]
weekly_seasonality	[False]	[False]
daily_seasonality	[daily]	monthly
period	24	[30.5]
fourier_order	[5]	[5]
n_changepoints	[300]	[300]
optimizer	[AdamW]	[AdamW]
batch_size	[32]	[32]
learning_rate	[0.04]	[0.01]

XGBoost. The model xgb.XGRegressor() used the following hyperparameters as presented in the Table 7. Regarding scaling, it is observed that XGBoost is indifferent to scaling, therefore no scaling was applied to this model.

Table 7: XGBoost hyperparameters.

n_estimators	[50]
max_depth	[2]
seed	[42]
objective	[reg: squared_error]
n_changepoints	[300]
optimizer	[AdamW]
batch_size	[32]
learning_rate	[0.04]

TimeGPT. The GPT model is a pre-trained model that is executed through an API. However, there are hyperparameters that optimize its predictions. The most important of these is the finetune_steps. As indicated in the Table 8, we need a large value for this parameter for the daily dataset, which is large, and a small value for the monthly dataset, which consists of only 60 lines.

Table 8: TimeGPT hyperparameters.

Hyperparameters	**Daily**	**Monthly**
h	[6]	[12]
freq	[D]	[MS]
time_col	[timestamp]	[timestamp]
target_col	[value]	[value]
finetune_steps	[60]	[10]

LSTM. In the table Table 9, the hyperparameters of the model are presented. Also, for this specific model, scaling is essential for its optimal performance and is described below the StandardScaler which is used. The mathematical formula applied by the StandardScaler to each feature is:

$$Z = \frac{(X - \mu)}{\sigma} \quad (1)$$

Here:

- Z is the scaled value.
- X is the original value of the feature.
- μ is the mean of the feature values.
- σ is the standard deviation of the feature values.

Table 9: LSTM Hyperparameters.

epochs	[200] early stop
patience	[20]
batch_size	[32]
lstm_units	[20]
lr	[0.01]
n_changepoints	[300]
optimizer	[Adam]
loss_function	[mse]

TFT. In the table Table 10, the hyperparameters of the model are presented. For this model, scaling is also essential, and the Standard Scaler was used as described in (1).

Table 10: TFT Hyperparametrs.

epochs	[1000] - early stop
patience	[10]
batch_size	[128]
limit_train_batches	[50]
gradient_clip_val	[0.1]
optimizer	[Ranger]
loss_function	[Quantileloss]
hidden_size	[16]
attention_head_size	[4]
dropout	[0.1]
hidden_continuous_size	[8]

5.2 Results

In this section, we delve into the experimental results, from all methods used, derived from our study. This part of the document is methodically divided into two distinct segments to facilitate a clearer understanding and comparison of the findings. The first segment is dedicated to the presentation of the monthly prediction results. Following this, the second segment focuses on the daily prediction results.

5.2.1 Monthly Results

ARIMA. This model is a conventional approach employed for forecasting time series. The primary rationale for its application in our experimental setup was to facilitate a comparative analysis with machine learning (ML) and deep learning (DL) models. As demonstrated in the table, this particular model exhibits the highest error margin among all the models evaluated, deviating from the actual data by 24.28% MAPE as presented in the Table 11: Mean and standard deviation over 10 independent experiments for monthly test set.. Based on the data utilized, it is evident that the ARIMA model does not serve as a dependable option for our forecasting needs.

FBProphet. This model represents a modern tool developed by Facebook's engineers in 2017 to address the problem of business forecasting for time series. Model delivered outstanding performance in our experiment. Exhibiting a mere 2.77% discrepancy from the actual data Table 11, it establishes itself as a credible tool for predicting monthly sales, thus affirming the findings reported in the literature. It is important to mention that this specific model is very user-friendly in terms of implementation, as it can be executed simply by providing it with the correct format of the date and sales data.

Neuralprophet. It represents an advancement of the FBProphet model, integrating neural networks to tackle more intricate issues. Within the context of monthly data, it fails to outperform FBProphet in terms of performance, exhibiting a MAPE of 6.12% and a standard deviation of 1.9%, as indicated in the

Table 11. This seems to arise from the fact that the data for the monthly problem is too limited for a neural model

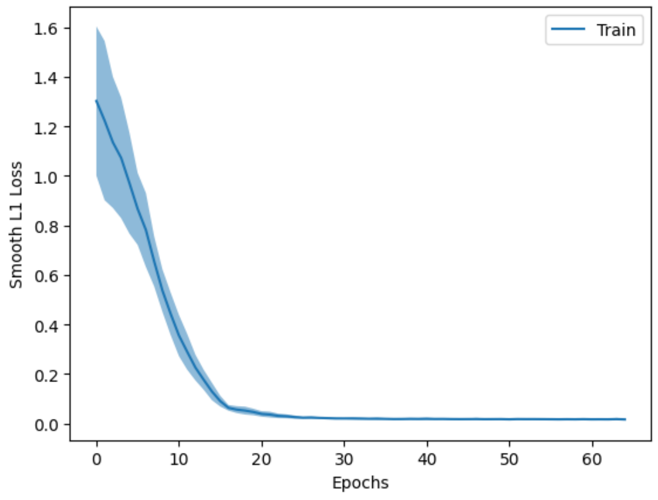


Figure 17: NeuralProphet loss functions monthly.

XGBoost. Represents the decision tree-based approach to the issue at hand. The outcomes achieved are moderate, showcasing a MAPE (Mean Absolute Percentage Error) of 9.13%, as illustrated in the Table 11. However, it is noteworthy that the model exhibits a significantly low standard deviation, signifying its stability as a forecasting model and suggesting that it has been sufficiently trained.

TimeGPT. Is a pre-trained model that utilizes historical data via an API. The performance of this model is outstanding, outperforming other models in our monthly forecasting challenge. According to the table, it achieves a MAPE of 2.23%, closely rivaling the FBProphet, which has a MAPE of 2.77%. Despite its superior performance, it is crucial to note that TimeGPT is not open-source. For our experiment, we secured a trial version from NIXTLA, specifically for academic use, which allotted us 1000 credits to facilitate our experimental procedure.

In summary, our experiment demonstrates that TimeGPT delivered the most superior performance, while FBProphet produced the second-best results overall and stands as the top-performing open-source mode.

Table 11: Mean and standard deviation over 10 independent experiments[1] for monthly test set.

Model	Metric	12 Months Average Error
ARIMA	MAE	9023×10^4
	MAPE	24.28%
	MSE	8587×10^{12}
FBProphet	MAE	1079×10^4
	MAPE	$\mathbf{2.77}\%$
	MSE	200×10^{12}
NeauralProphet	MAE	$2342 \times 10^4 \pm 772\times 10^4$
	MAPE	$6.12\% \pm 1.99\%$
	MSE	$927\times 10^{12} \pm 3150 \times 10^{12}$
XGBoost	MAE	$3513 \times 10^4 \pm 575\times 10^4$
	MAPE	$9.13\% \pm 0.8\%$
	MSE	$1461 \times 10^{12} \pm 200.8 \times 10^{12}$
TimeGPT	MAE	831×10^4
	MAPE	$\mathbf{2.23}\%$
	MSE	110×10^4

5.2.2 Daily Results

FBProphet. In terms of this model's daily forecasting abilities, as highlighted in the table, it ranks as the least effective, showing a MAPE of 14.53%, Table 12. Given this significant deviation from the actual data, FBProphet does not prove to be a dependable model for predictions across a large dataset.

Neuralprophet. Contrary to the FBProphet, the NeuralProphet demonstrates improved outcomes. Nonetheless, the results still do not meet expectations, and it cannot be considered a dependable forecasting model since it has a MAPE of 9.99%, as detailed in theTable 12. However, it is worth noting that it maintains a consistent performance with minimal variations across the 10 independent experiments performed, a fact highlighted by the analysis of the loss function Figure 18.

[1] Mean and standard deviation are calculated only for stochastic models which is NeuralProphet and XGBoost.

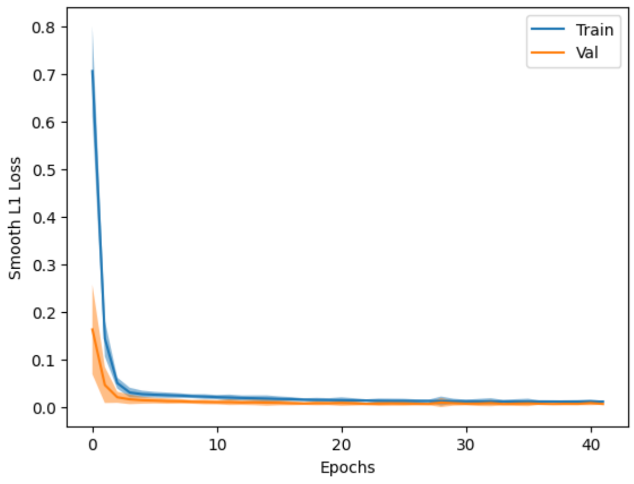


Figure 18: NeuralProphet loss function daily.

LSTM. This model embodies the conventional DL methodology. As indicated by the table, showing a MAPE of 6.68%, Table 12, and from the analysis of the loss functions Figure 19, this particular model demonstrates strong performance and yields favorable outcomes in addressing the complex challenge of daily forecasting.

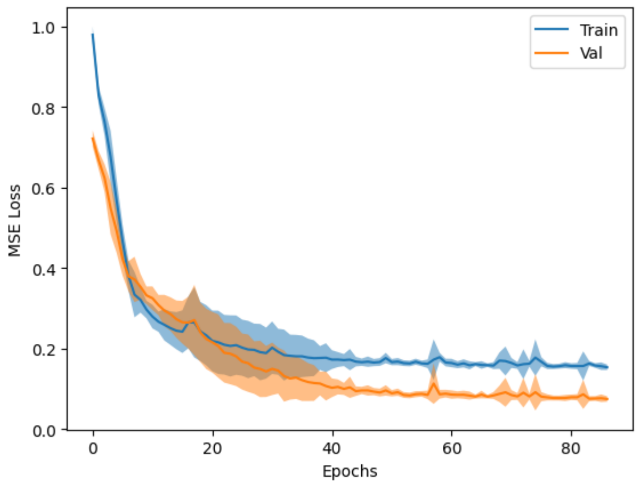


Figure 19: LSTM loss function daily.

TimeGPT. TimeGPT here delivered exceptional results also for daily forecasting, with a MAPE (Mean Absolute Percentage Error) of 5.19%, as shown in the table.

TFT. This model represents a state-of-the-art approach for daily forecasts. With an outstanding performance of 3.90% MAPE and a standard deviation of 1.91%, as indicated in the Table 12: Mean and standard deviation over 10 independent experiments for daily test set., it is the best model for monthly forecasting. This particular model is open source and stands out as the superior option for daily forecasting, outperforming both open source models and proprietary models that are not free and require some cost. Additionally, from the loss functions Figure 20, we can see that it trains quickly and does not exhibit significant discrepancies across the 10 independent experiments.

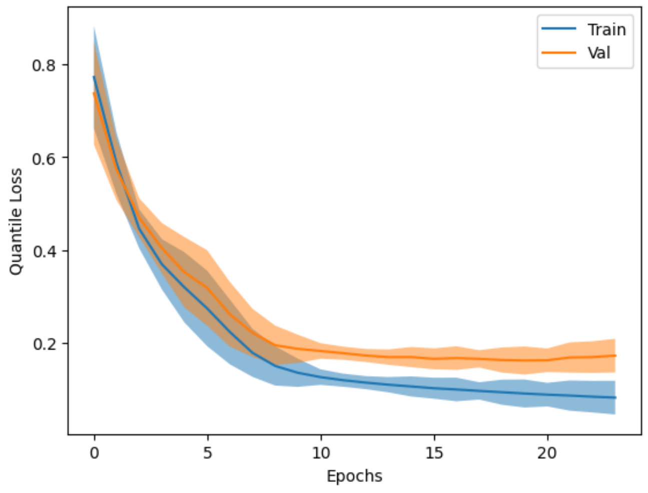


Figure 20: TFT loss function daily.

Table 12: Mean and standard deviation over 10 independent experiments[2] for daily test set.

Model	**Metric**	**6 Days Average Error**
FBProphet	MAE	1923×10^4
	MAPE	14.53%
	MSE	4119×10^{12}
Neauralprophet	MAE	$145.5 \times 10^4 \pm 13.69 \times 10^4$
	MAPE	$9.99\% \pm 2.96\%$
	MSE	$3.548\times 10^{12} \pm 0.218 \times 10^{12}$
LSTM	MAE	$108.1 \times 10^4 \pm 15.27 \times 10^4$
	MAPE	$6.68\% \pm 1.18\%$
	MSE	$2.312 \times 10^{12} \pm 0.528 \times 10^{12}$
TimeGPT	MAE	255×10^4
	MAPE	5.19%
	MSE	0.984×10^{12}
TFT	MAE	$53.94 \times 10^4 \pm 26.78 \times 10^4$
	MAPE	$\mathbf{3.90\% \pm 1.91\%}$
	MSE	$0.509\times 10^{12} \pm 0.574 \times 10^{12}$

[2] Mean and standard deviation are calculated only for stochastic models which are Neuralprophet, LSTM and TFT.

6 Conclusion and Future Work

In summary, our findings indicate that monthly prediction is a more accessible issue compared to daily forecasting. Simpler methodologies, even without the integration of neural networks, have shown to be effective. This contrasts with complex neural network models, which may underperform due to their inability to train effectively with small datasets. Additionally, monthly forecasts can be influenced by unforeseen events and temporary demand fluctuations, such as early salary payments, holidays, or weather conditions that unexpectedly increase or decrease demand. However, consumer demand in this context tends to remain stable monthly, significantly aiding in producing accurate forecasts without overcomplicating the models.

On the other hand, daily forecasting presents a different scenario. This problem is considerably more intricate due to the potential for outliers caused by unexpected events, which can disrupt model performance and time series analysis. Yet, once these outliers are addressed, the results significantly improve. The models that succeed in daily forecasting tend to be more complex and rely heavily on neural networks. This is supported by literature, considering the larger volume of data available for daily forecasting, where deep learning methods tend to perform better. The main challenges in daily forecasting extend beyond the complexity of the models; they also encompass the preparation and processing of data, especially the identification and correction of outliers.

Looking forward, there are several avenues for future work. One area involves feature engineering, specifically the identification of variables that significantly impact sales. Initial factors for consideration should include weather conditions and oil prices. However, the incorporation of holidays as a feature is anticipated to be particularly impactful. We suggest developing a feature that encompasses holidays and other significant events, in collaboration with domain experts, to inform the model about unexpected events that have historically affected demand, as well as predictable fluctuations such as weekends and holidays. Furthermore, it would be beneficial to experiment with a larger dataset to assess how effectively a forecasting model can learn from and adapt to holidays that shift annually, such as Easter. Lastly, a more extensive exploration of model parameters is warranted. Due to previous computational constraints, there was a limitation in the extent of fine-tuning achievable. Addressing this could potentially enhance model performance and forecasting accuracy.

7 References

[1] B. Pavlyshenko, "Machine-Learning Models for Sales Time Series Forecasting," *Data,* vol. 4, p. 15, January 2019.

[2] Z. Zhang, "Sales Prediction Based on ARIMA Time Series and Multifactorial Linear Model," *Highlights in Science, Engineering and Technology,* vol. 38, pp. 1-8, March 2023.

[3] C. Permatasari, W. Sutopo and M. Hisjam, "Sales forecasting newspaper with ARIMA: A case study," 2018.

[4] L. Zhang, W. Bian, W. Qu, L. Tuo and Y. Wang, "Time series forecast of sales volume based on XGBoost," *Journal of Physics: Conference Series,* vol. 1873, p. 012067, April 2021.

[5] Y. Dai and J. Huang, "A Sales Prediction Method Based on LSTM with Hyper-Parameter Search," *Journal of Physics: Conference Series,* vol. 1756, p. 012015, February 2021.

[6] R. Ospina, J. A. M. Gondim, V. Leiva and C. Castro, "An Overview of Forecast Analysis with ARIMA Models during the COVID-19 Pandemic: Methodology and Case Study in Brazil," *Mathematics,* vol. 11, 2023.

[7] J. Fattah, L. Ezzine, Z. Aman, H. Moussami and A. Lachhab, "Forecasting of demand using ARIMA model," *International Journal of Engineering Business Management,* vol. 10, p. 184797901880867, October 2018.

[8] S. Taylor and B. Letham, *Forecasting at scale,* 2017.

[9] O. Triebe, H. Hewamalage, P. Pilyugina, N. Laptev, C. Bergmeir and R. Rajagopal, "NeuralProphet: Explainable Forecasting at Scale," *CoRR,* vol. abs/2111.15397, 2021.

[10] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *CoRR,* vol. abs/1603.02754, 2016.

[11] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation,* vol. 9, pp. 1735-80, December 1997.

[12] B. Lim, S. O. Arik, N. Loeff and T. Pfister, *Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting,* 2020.

[13] B. Lim, S. Ö. Arık, N. Loeff and T. Pfister, "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting,* vol. 37, pp. 1748-1764, 2021.

[14] A. Garza and M. Mergenthaler-Canseco, *TimeGPT-1,* 2023.

[15] Z. Zhang and J. C. Moore, "Chapter 8 - Autoregressive Moving Average Models," in *Mathematical and Physical Fundamentals of Climate Change*, Z. Zhang and J. C. Moore, Eds., Boston, Elsevier, 2015, pp. 239-290.