UPPSALA
UNIVERSITET

SAMINT-MIA-IEM-24004
Master's Thesis 30 credits
June 2024

Empirical Study of Neural Network and other Machine Learning Models in Time Series Forecasting using Sales Data

Abhinava Changa Ramesh

Muhammad Usman

Master's Programme in Industrial Anatlytics

Masterprogram i industriell Analys

UPPSALA
UNIVERSITET

Faculty of Technology

Visiting address:
Ångströmlaboratoriet
Lägerhyddsvägen 1

Postal Address:
Box 536
751 21 Uppsala

Telephone:
+46 (0)18 – 471 30 03

Telefax:
+46 (0)18 – 471 30 00

Web page:
http://www.teknik.uu.se/education/

Abstract

Empirical Study of Neural Network and other Machine Learning Models in Time Series Forecasting using Sales Data

Abhinava Changa Ramesh, Muhammad Usman

Time series forecasting is a critical component of decision-making in various domains, from finance to retail. With the increasing complexity of real-world datasets, selecting the most appropriate modeling techniques has become crucial for accurate predictions. This thesis evaluates the performance of a traditional gradient boosting ensemble modeling, which constructs a stronger model by sequentially adding weaker models in a gradient descent manner to enhance accuracy and robustness. It also assesses a Neural network modeling that implements a mix of backward and forward residual links and a deep stack of fully-connected layers in a hierarchical structure. These models are evaluated using three distinct datasets, each differing in their structure, features, and origin. Through comprehensive analyses, including hyperparameter optimization, cross-validation, and evaluation of model performance metrics, insights into the strengths and limitations of these approaches are provided.

Findings reveal that traditional gradient boosting ensemble methods like LightGBM demonstrate commendable speed and competitive performance, particularly in datasets with regular patterns. Conversely, Neural Network models like N-BEATS excel in capturing complex temporal dependencies but may face challenges due to longer training times and computational demands. While N-BEATS generally exhibit higher accuracy, LightGBM showcases superior efficiency and speed, which is valuable in scenarios with limited computational resources or real-time forecasting requirements.

The study also delves into the effectiveness of different cross-validation strategies, shedding light on their impact on model performance assessment. The comparison between traditional and deep learning models highlights the nuanced trade-offs between accuracy, computational resources, and model complexity, emphasizing the importance of selecting the most suitable forecasting model based on specific dataset characteristics and forecasting requirements. In addition, this thesis investigates the effectiveness of a public python library Darts, that is effective in handling panel and multivariate hierarchical datasets, expanding the evaluation to cover different time series forecasting challenges.

In conclusion, this thesis contributes to improved understanding of the performance of traditional and deep learning models in time series forecasting, providing practical insights for decision-makers in various industries.

Supervisor: Farrukh Nauman (RISE)
Subject reader: Kaveh Amouzgar
Examiner: Mats Gustafsson
SAMINT-MIA-IEM-24004
Printed by: Uppsala Universitet

Preface

In 2023, after spending two incredible years in Sweden and at Uppsala University as master's students, we decided to undertake our master's thesis with a research organization. We were fortunate to collaborate with the Research Institutes of Sweden (RISE) for this endeavour. RISE is Sweden's national research institute and innovation partner.

We want to extend our sincere thanks to RISE for allowing us to conduct this thesis at their institution. We are particularly grateful to Farrukh Nauman, our thesis supervisor from RISE, whose guidance and support were invaluable throughout our thesis. His instructions and teachings will undoubtedly benefit us for years to come.

We also want to express our sincere gratitude to our subject readers, Assistant Professor Kaveh Amouzgar, Professor Mats Gustafsson, and Study Advisor Lena Eliasson from Uppsala University. Their expertise and assistance were crucial to the success of our research, and we greatly appreciate their continuous support.

Lastly, we would like to thank our families, who have been supportive throughout our studies. Their encouragement has been a constant source of strength for us.

Table of Contents

List of Figures

List of Tables

Introduction

Background

The selection of modeling techniques has changed significantly in the field of time series forecasting. Simple regression models were popular at first due to their ease of use and understandability. These models, due to their ability to capture linear trends of the timeseries, have frequently shown excellent performance in a range of predicting tasks (Brockwell and Davis, 2016). Their nature of being concise made them easier to understand and served as a standard by which the effectiveness of more complex techniques could be evaluated. As the data increasingly reflects the real-world observations, models that can detect subtle patterns and fine details are necessary.

Traditional & Ensemble Models

As timeseries datasets became increasingly intricate and dynamic, the limitations of simple regression models became apparent. In the field of time series forecasting, ensemble methods and conventional machine learning models have become strong competitors (Makridakis et al. 2022a). Ensemble methods combine predictions from multiple base models to improve accuracy, robustness, and adaptability. Among the advantages of machine learning is the ability to work with high-dimensional data, as well as capture nonlinear relationships with much ease through algorithms like Gradient Boosting and Random Forests. Their ensemble nature, aggregating predictions from multiple base learners, increased their robustness and adaptability, enabling them to outperform their simpler counterparts in many scenarios.

Deep learning Models

While traditional and ensemble models gained momentum, deep learning approaches initially lagged behind in time series forecasting tasks (Makridakis et al. 2018). The complex architectures and intensive computational requirements posed challenges, and the performance gains were not always parallel with the investment. But there has been a resurgence of interest due to recent developments in deep learning as well as the accessibility of large datasets and more computational power (Hwang Tim, 2018) (Sagiroglu and Sinanc, 2013). Neural networks can be used to extract significant characteristics from time series data and reveal hidden temporal patterns, as some recent research has shown (Wellens et al. 2022). These developments to a considerable extent revamped the role of deep learning in time series forecasting and reconsidered the effectiveness of these methods in a real-world scenario.

Problem Statement

This thesis is in collaboration with RISE Research Institutes of Sweden, within a project focused on Study of Time Series Forecasting of Sales. The focus is on the Template, Hierarchical, and Multimodal based datasets, each introducing its unique complexities and challenges majorly for sales data. These variations of datasets provide distinct challenges to traditional, novel deep learning and non-deep learning models in time series forecasting. While individual studies and competitions (Makridakis et al. 2018, 2020) have shown how these models perform on each dataset separately, a holistic analysis surrounding a range of models is necessary to understand their overall effectiveness and the specific difficulties they face across distinct types of time series datasets.

This detailed comparison will help determine whether models maintain consistent performance across various time series datasets or whether they encounter specific struggles with each variation. Additionally, this research intends to delve deeper from a generalized approach to a specific approach to look into the inherent characteristics of these forecasting models, especially in terms of handling traditional time series specific complexities of each dataset such as missing values, resource utilization, accuracy over time, and multi-modality (Skenderi et al., 2022) (Makridakis et al. 2022b). Understanding how different models perform in these aspects is crucial to assessing their reliability from general to specific performance and vice versa in predicting sales/or traditional forecasts when using the mentioned types of time series data.

Moreover, the issue of model validation in time series forecasting is a critical area, particularly when considering cross-validation techniques. Therefore, a significant focus will be on examining the existing traditional cross-validation techniques in time series forecasting. In time series, forecasting cross-validation is also instrumental in enhancing model accuracy but brings its own set of time series-specific challenges, for example, computation-time requirements and the risk of data leakage due to the datasets temporal structure. Evaluating the effectiveness of existing cross-validation techniques will help identify the extent to which these issues affect sales predictions/or traditional forecasts and the overall strength and reliability of forecasting models when applying these techniques across various datasets. This analysis aims to provide insights that could lead to improvements in forecasting methodologies.

Research Questions

This thesis aims to answers the following questions:

1. Which model among the ones evaluated seems promising for the time series datasets?
2. Which model provides more accurate predictions for sales data when using a short observation window?
3. What are the complexities and challenges associated with implementing selected cross-validation methods on timeseries datasets?
4. What is the comparative performance of Neural Network models versus other Machine Learning models in time series forecasting?

Objective & Scope of the Study

Comparing Ensemble & Deep Learning Methods

The primary objective of this study is to compare the performance of traditional machine learning models, particularly ensemble methods such as LightGBM (Ke et al., 2017), with a deep learning model, specifically the N-BEATS (Oreshkin et al., 2019) architecture, in the context of time series forecasting. We aim to conduct a comprehensive evaluation of these models across a diverse nature of time series datasets, considering various metrics for performance evaluation such as accuracy, computational efficiency, and robustness. By comparing the strengths and limitations of traditional/ensemble models with those of deep learning approaches, we seek to provide valuable insights into the relative efficacy of different modeling techniques in different forecasting scenarios.

Implementing N-BEATS and LightGBM Models

As part of the study's scope, we will focus on implementing and evaluating the N-BEATS and LightGBM models on three distinct datasets, the Visuelle 2.0 dataset—a real-world time series dataset representative of the retail industry and M5 and M4 datasets, known for their complexity and diversity in the field of timeseries forecasting. Our goal is to evaluate how well these models predict important variables that are essential for retail operations decision-making, such as sales, inventory levels, or client demand. We want to identify patterns, trends, and seasonality in retail data by applying these sophisticated forecasting algorithms to these datasets. This will enable us to provide practical insights for improving inventory management, resource allocation, and strategic planning in retail enterprises.

Overview of the Thesis Structure

This chapter gives an introductory idea about the thesis topic, goal (research question), and objective behind it. Chapter 2, the literature review provides an overview of the major concepts and models used in this field of knowledge. In Chapter 3, Methodology is discussed, starting from the dataset preprocessing, followed by the deep learning and non-deep learning models, furthermore, extended to cross validation techniques, error & evaluation metrics, and hyper-parameter optimization. In Chapter 4, the results are critically analyzed. Lastly, Chapter 5 we'll discuss the results obtained to connect with our research objective. In Chapter 6, we will conclude our research with answering the research questions and its future suggestions.

Literature Review

The chapter focuses on existing literature on time series forecasting. It explains time series forecasting and the different methods used for forecasting. The current methods are divided into classical/traditional, machine learning based and hybrid/ensemble methods that are a combination of both. As stated by the research questions, it is clear that the focus is to compare forecasting accuracy of machine learning models on the datasets and to devise the best cross validation strategy for the datasets and models in terms of results. Hence, this chapter covers an overview of models and cross validation strategies implemented in the past for the said datasets. Further, this chapter reviews the pros and cons of all the methods with the research gap and how they are used to deduce our research questions and later make our way towards our research.

Classical Approaches in Time Series Forecasting

Time series is a stretch of values on the same scale indexed by a time-like parameter. It comes in various shapes and forms and is as diverse as the functions of real numbers themselves (Brockwell & Davis, 2016). The analysis of these time series encompasses a range of techniques aimed at extracting valuable insights, enhancing our understanding, and facilitating various objectives such as summarization, decision-making, description, and prediction. From economic indicators to environmental fluctuations, from stock market trends to physiological signals, Time series data is present across various fields, providing a great deal of information that can be analyzed using specific methods.

We concentrate our thesis on Time series forecasting. It is a significant discipline of data modeling where future values are predicted based on historical patterns and trends in sequential data. The key idea behind time series forecasting is to capture and utilize the patterns such as seasonality, and trends present in the data. It assumes that future values will exhibit similar patterns to past observations, allowing for predictions beyond the available data (Box, Jenkins and Reinsel, 2013).The prominence of time series forecasting lies in different use cases across various domains, like economics, weather, stock price, business development etc. Our thesis would be implementing time series forecasting in the fashion domain to forecast the upcoming sales trends of a fast fashion company for various seasons.

Forecasting using models dates as far back as 1927 when autoregressive model was used by Udny Yule, a British statistician. This model was a crucial component in developing ARIMA model in the 1970s by George Box and Gwilym Jenkins (Box, Jenkins and Reinsel, 2013). While this model was limited in its use cases, it paved the way for development of variations of this model that would overcome the limitations, like Seasonal ARIMA, Vector ARIMAX etc. These ARIMA models are in present times, considered classical time series approaches since technological and statistical advancements have paved the way for innovative approaches and algorithms (Cerqueira et al., 2019).

Machine Learning Approaches in Time Series Forecasting

Traditional machine learning models offer a compelling alternative for time series forecasting, often providing a balance of accuracy, speed, and interpretability compared to complex neural networks. These models, including linear regression, decision trees, random forest, and gradient boosting methods such as XGBoost (Chen and Guestrin, 2016) and CatBoost (Prokhorenkova et al., 2019), demonstrate proficiency in managing intricate patterns and non-linear relationships. Decision trees are a fundamental component of these methods, and they are a type of supervised learning algorithm used for classification and regression tasks. They work by splitting the data into subsets based on the value of input features, creating a tree-like model of decisions. However, their performance tends to be suboptimal when confronted with high-dimensional features and large datasets.

In our research, we primarily focus on LightGBM (Ke et al., 2017), an implementation of the Gradient Boosting Decision Tree. Gradient Boosting is an ensemble learning technique that combines the predictions of several weaker base models (in this case, decision trees) to improve overall performance. It operates by training models sequentially, with each one correcting its predecessor's errors in a gradient descent manner. The LightGBM introduces two core optimizations: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS accelerates computation by strategically discarding data instances with small gradients, effectively reducing dataset size without significant accuracy loss. EFB tackles high-dimensional feature spaces by intelligently grouping mutually exclusive features, leading to a reduction in feature dimensionality. The combined effect of GOSS and EFB enables LightGBM to maintain model quality while significantly improving training speed, making it well-suited for large-scale datasets (Ke et al., 2017).

The original LightGBM (Ke et al., 2017) research paper presents a performance comparison of LightGBM with and two XGBoost variations: xgb_exa (pre-sorted algorithm) and xgb_his (histogram-based algorithm). Additionally, the researchers tested a baseline LightGBM implementation with its optimizations (GOSS and EFB) disabled. Evaluation across five datasets focused on training time and accuracy metrics. LightGBM demonstrated the fastest training times while maintaining accuracy comparable to other implementations. However, xgb_his exhibited high memory consumption, preventing its successful execution on the KDD10 and KDD12 datasets (Ke et al., 2017). In conclusion LightGBM outperformed significantly compared to XGBoost in terms of computational speed and memory consumption.

Deep Neural Network Approaches in Time Series Forecasting

While Deep Neural Network (DNN) models have gained considerable attention in various domains, their performance in time series forecasting has been less than stellar when compared to traditional and classical approaches (Makridakis et al., 2020). This was evident in the M4 competition, where the winning approach, proposed by Smyl, was a hybrid model that combined an LSTM stack with the classical Holt-Winters statistical model (Makridakis et al., 2018; Oreshkin et al., 2019). Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of learning long-term dependencies, making them suitable for sequence prediction problems like time series forecasting. The Holt-Winters model is a statistical forecasting method that decomposes time series data into seasonal, trend, and level components to make predictions. Notably, the model's performance was heavily reliant on the Holt-Winters component. In response to this, N-BEATS was developed as an attempt to assess the performance of a pure deep learning model in time series forecasting, providing a benchmark against Smyl's hybrid approach (Oreshkin et al., 2019).

In the paper under discussion, two distinct architectures of N-BEATS are proposed: the Generic and the Interpretable. The Generic architecture (N-BEATS-G) does not rely on any time series-specific knowledge, while the Interpretable architecture (N-BEATS-I) incorporates trend and seasonality stacks. Given the observation that the majority of top-performing models in the M4 competition utilized ensemble methods, this evaluation also adopts an ensemble approach, combining the Generic and Interpretable models (N-BEATS-I+G) (Oreshkin et al., 2019). This comparative analysis aims to provide a comprehensive understanding of the strengths and weaknesses of each architecture and their potential synergies when used in conjunction.

The paper presents a comprehensive evaluation of three N-BEATS models: Generic (N-BEATS-G), Interpretable (N-BEATS-I), and an ensemble model (N-BEATS-I+G). These models are compared against various other models in the M4 dataset, including Smyl's hybrid Deep Learning/Time Series (DL/TS) model, which was the winner of the M4 competition (Makridakis et al., 2020). The results indicate that all N-BEATS models outperform the DL/TS model when evaluated using certain error metrics and averaging methods (Oreshkin et al., 2019). Furthermore, the Generic N-BEATS model, which uses no time series-specific architectural components, performs exceptionally well across a wide array of time series forecasting tasks (Oreshkin et al., 2019). This leads to the conclusion that deep learning does not necessarily require support from statistical approaches or hand-crafted feature engineering to excel in time series forecasting.

Theoretical Background

Under the scope of our degree project with Research Institutes of Sweden (RISE) we use three different datasets to evaluate the models, namely M4 hourly dataset, M5-Walmart dataset and Visuelle 2.0.

The M Competitions initiated by Professor Spyros Makridakis in 1982, provide a wide range of datasets for forecasting tasks. Specifically, M4 and M5 competitions provide datasets that are used to evaluate the time-series forecasting models. The data provides monthly and daily basis sales data and concentrates its accurate forecasting, keeping in mind the seasonality, irregularities and other characteristics.

The NBEATS model presented in the original paper (Oreshkin et al., 2019), outperformed the M4 competition winner ESRNN (Makridakis et al. 2018) by 3% (Oreshkin et al., 2019). This NBEATS model includes generic, interpretable, and ensemble variants and is based on deep learning architecture. In the following competition, i.e. M5 competition, LightGBM (Ke et al., 2017) model was implemented by all top 50 competitors, in which they implemented different variants of it (Makridakis et al. 2022a).

M4 Dataset

The M4 Competition dataset is a comprehensive resource for time-series forecasting, comprising synthetically generated 100,000 time series across various domains like sales, finance, and other domains. These time series vary in length spanning several hours to a few years, depending on the duration of data collection. For example, a time series might represent hourly sales data for a specific product over a year. Particularly relevant to this study is the hourly dataset, which provides high-frequency sales data (Makridakis et al., 2020). This subset offers unique insights into sales patterns, aiding in the development of effective forecasting models. It also provides other information such as starting data, frequency of data collection and other contextual features related to time-series.

M5 Dataset

The dataset, specifically curated for the M5 accuracy Competition, comprises real-world sales data from Walmart stores across the United States. It encapsulates key elements essential for sales analysis and forecasting. The dataset not only captures sales trends across various temporal scales and seasons but also includes detailed product information such as type, price, brand, and size. Each time series represents daily sales for specific products over the years. For example, a time series might capture the daily sales of a particular brand while also including the effects of promotions and holidays. Furthermore, it provides sales data at different hierarchical levels, offering insights into the impact of promotional events, holidays, and marketing campaigns on sales patterns (Makridakis et al., 2022). This comprehensive dataset serves as a robust foundation for developing sophisticated sales forecasting models.

The Multi-modal Universe of Fast-fashion: The Visuelle 2.0 Benchmark (2022)

The Visuelle 2.0 benchmark (Skenderi et al., 2022) addresses prediction challenges typical in the fast-fashion business and particularly focuses on short-observation forecasting (SO-fore), where sales are forecasted given limited historical data. The benchmark utilizes the Visuelle-2.0 Dataset, a distinctive and valuable resource that provides disaggregated item-shop level data and includes multi-modal exogenous data such as Google Trends for products, weather conditions, customer purchase data, and images of each clothing item. The key difference between multivariate and multimodal data lies in the nature and diversity of the data. Typically, multivariate data are of the same type (say numerical), whereas multimodal data considers different types of data like images, Google trends, weather conditions, etc.

In their study, (Skenderi et al., 2022) evaluated four models, emphasizing two that incorporate exogenous data alongside sales records. The K-Nearest Neighbours (KNN) model demonstrated superior performance without utilizing image data, while the Cross Attention Recurrent Neural Network (CrossAttnRNN) excelled when image data was included. These findings highlighted the impact of incorporating diverse data sources like images and Google Trends alongside sales data to enhance neural network model performance. This combination of diverse data sources enhances the potential for accurate forecasting, thereby making the Visuelle-2.0 dataset a significant resource in the realm of fast-fashion analytics.

Darts Library

Darts is an open-source Python library, launched in 2022 (Herzen et al., 2022). This library is specifically designed to make time series analysis and forecasting easier to implement as it incorporates built-in time series techniques, e.g. handling univariate and multivariate data. Darts provide built-in implementations of renowned models like N-BEATS and LightGBM, along with a variety of other models ranging from classical to deep neural networks. The library offers a user-friendly interface with fit and prediction functions tailored for time series forecasting. Also, it includes relevant pre-processing techniques for time-series based data, such as handling missing values and multimodality. This makes Darts an appealing choice for our research in time series forecasting.

Cross Validation Strategies

Under the scope of our degree project with Research Institutes of Sweden (RISE) we use these Cross Validation strategies to evaluate the models. Cross-validation (Bergmeir et al. 2018) is an essential technique in machine learning for evaluating model performance, particularly useful in assessing the generalizability of time series forecasting models.

Walk-Forward Cross Validation strategy is designed to mimic real-world forecasting tasks. It begins with a small initial dataset and sequentially updates the model to capture various patterns over time as it becomes available. In this technique, the model is trained on a small set of historical data and then forecasts the next values, it then evaluates its predictions against actual values and the training size increases in the subsequent iterations. This linear iterative process continues until all available data has been used for training and evaluation. A variation of this method, known as Rolling Cross Validation, limits the training size to ensure that the model is continually updated with only the most recent data.

Grouped Time-Series Cross Validation (Smit J.J.H., 2020) approach involves dynamic group expansion to make the model robust to changes and to incorporate advanced data characteristics. Dynamic group expansion refers to a process of adding new, related time series to the existing groups as more data becomes available. It is particularly useful when dealing with large collections of related time series. Purged Group Time Series Cross Validation (MARKETNEUTRAL, 2020) incorporates a purging step during the validation process. This step ensures that when training data is added to the validation set, any data points that could present bias are removed. It is especially applicable for dealing with issues of closely spaced data.

Combinatorial Purged Group Cross Validation (Gort and Yang, 2022) technique introduces a gap to prevent future information from leaking into the training process, which is crucial for handling auto-correlation in data. Auto-correlation occurs when observations in a time series are correlated with previous observations and can bring bias to the training process if not properly managed. CPCV introduces a gap period between the training and testing sets to prevent future information from leaking into the training process. The gap refers to an intentionally introduced period where data points are excluded from both the training and testing sets. This ensures that the model does not use information from future data points when making predictions. By doing such a process this technique effectively handles auto-correlation and minimizes the risk of information leakage. CPCV provides the exact number of combinations of training/testing sets required to construct a set of backtesting paths. This method systematically purges (removes) leaked information from training data and helps ensure that each backtesting path remains unaffected by future data.

This literature review discussed various models, datasets, cross validation, and other evaluation techniques that are for time series forecasting in sales. By leveraging these models, datasets, and evaluation techniques, businesses can make accurate sales forecasts and improve their decision-making processes (Bergmeir et al., 2018). Keeping in view the similar aim, we are further deducing the comparative analysis for deep learning and non-deep learning-based models for sales forecast using the benchmark datasets while evaluating through cross validation and other evaluation techniques answering the research questions devised in introduction chapter.

Methodology

This chapter outlines the research methodology, preparing for the analysis in the following sections. It gives a detailed overview of the scientific processes and techniques used, setting the groundwork for further discussion and analysis.

Research Approach and Design:

The research approach is based on 'positivist epistemology', as the results are derived from scientific methods to confirm or derive the claims mentioned in the research questions. The design of our study is Quantitative in nature, as it involves the analysis of numerical and discrete sales data from time series forecasting.

This study embarks on the exploration of time series forecasting, a scientific approach that leverages historical time-stamped data to make informed predictions. The procedure involves the construction and utilization of models to make observation and determine future decisions. The research aims to construct an N-BEATS ensemble model and compare its performance with the LightGBM model in the context of time series forecasting for sales. This study employs three distinct time series datasets for comparison: M4-Hourly, M5-Walmart and Visuelle 2.0. Further details and specifications of these datasets will be discussed in the following sections of this chapter.

In addition to the model comparison, this research also aims to evaluate the efficacy of different cross-validation techniques across various models and datasets. It also incorporates optimization techniques to enhance the performance of the models. Specifically, Optuna (Akiba et al., 2019), a hyperparameter optimization framework, is utilized to fine-tune the models' parameters.

Error and Evaluation Metrics

There are several error evaluation methods and metrics that can help to understand how well the applied model's predictions align with the actual observed values. Following are the evaluation metrics that are used:

Mean Absolute Error (MAE)

MAE calculates the average absolute difference between the predicted values and the actual observed values. The MAE provides a clear representation of the magnitude of errors without considering their direction. The error value close to zero represents the least amount of error (a perfect forecast). The MAE is defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

The predicted value at time is represented by $\hat{y}_i$ and the actual observed value at time is represented by y_i, where n is the number of observations.

Symmetric Mean Absolute Percentage Error (SMAPE)

The Symmetric Mean Absolute Percentage Error (SMAPE) can be employed as a percentage-based error metric, particularly when dealing with small or zero actual values. It quantifies the percentage difference between the predicted and actual values. It is called symmetric because it treats under-forecasts and over-forecasts equally. SMAPE is defined as follows:

$$sMAPE = \frac{200\%}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{\frac{(|y_i| + |\hat{y}_i|)}{2}}$$

where n is the number of observations, y_i represents the actual values, and $\hat{y}_i$ represents the predicted value at time i.

There are two variations of SMAPE, depending on the range. The first variation forces the metric to range between 0% and 100%. The second variation, although identical, widens the range to 200% . The built-in SMAPE metric in the darts library is the second variation, and in our research, we would be using the variation available through Darts.

Weighted Absolute Percentage Error (WAPE)

When assessing the accuracy of forecasting models, a statistic known as the Weighted Absolute Percentage Error (WAPE) is used, especially in cases where multiple forecasts or observations have varying weights or degrees of relevance. This expansion of Mean Absolute Percentage Error (MAPE) estimates the average absolute percentage difference between actual and expected data. WAPE measures the relevance of each observation by applying a specified weight to the absolute percentage error. The importance or priority of various data pieces in the study may be reflected in these weights. The WAPE formula is:

$$WAPE = \frac{\sum_{i=1}^{n} w_i \left|\frac{A_i - F_i}{A_i}\right|}{\sum_{i=1}^{n} w_i}$$

The weight associated with observation I is denoted by w_i, the projected value is represented by F_i, the actual values are represented by A_i, and the total number of observations is marked by n.

While the Darts Library does not include a built-in Weighted Absolute Percentage Error (WAPE) metric, implementing it manually is relatively straightforward. This can be accomplished with or without leveraging the related functions provided in the Darts library. For M4 Dataset, it was implemented using the Darts library and for Visuelle 2.0, It was custom implemented using the Numpy library (Harris et al., 2020) along with scikit-learn's MAE function.

Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is another error metric used in our study. It is particularly useful for assessing the performance of models where large errors are particularly undesirable as it penalizes larger errors more than smaller ones. It is calculated by taking the square root of the average of the squared differences between the predicted and actual values. This metric provides an overall measure of the accuracy of a model's predictions by reflecting both the magnitude and frequency of errors. The formula of RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (A_i - F_i)^2}$$

In the above equation, A_i represents the actual value and F_i represents the forecasted value, and n is the total number of observations.

The error value close to zero represents the least amount of error while greater values represent the higher error. In this study, we used RMSE to evaluate the forecasting accuracy. The Darts library provides built-in support for calculating RMSE, which was utilized for the M5 dataset.

Data Analysis and Preprocessing:

M4 Dataset:

The M4 dataset, developed for the fourth edition of the Makridakis Competition, was created through a random selection from the ForeDeCk database (Spiliotis et al., 2020). It is an extensive compilation of 100,000 time series data points, consisting of yearly (Y), quarterly (Q), monthly (M), weekly (W), daily(D), and hourly (H) frequencies. The dataset is partitioned into training and test sets, It should be noted that the lengths of the time series in this dataset are not always equal (Makridakis, Spiliotis and Assimakopoulos, 2020).

The M4 Dataset was chosen for this research due to its nature as univariate time series, meaning it solely contains time series data without any additional covariate features that could potentially influence the time series. The intent is to examine how machine learning and deep learning models would interact with the data and assess which one would offer the most consistent results.

We primarily considered only the hourly timeseries in M4 dataset. This choice was due to its manageable size, which allowed for efficient training, forecasting, hyperparameter tuning, and cross-validation within our available resources.

The M4 hourly time series data comprises two distinct observation lengths: either 700 or 960 observations. Notably, unlike other datasets, the M4 hourly data is not categorized into diverse groups, as confirmed by the `m4-info.csv` file. This streamlined structure further facilitated our research process.

The initial step involves reading the M4-hourly_train.csv file and the metadata file m4-info.csv using a pandas dataframe. We then filter the M4 info file to include only IDs containing 'H', thereby selectively filtering the 'Hourly' frequency from the other frequencies previously mentioned. Subsequently, we merge the hourly dataset and the filtered info file based on the common IDs present in these two data frames and then drop the columns which are not relevant for our research.

Following this, we reshape the merged dataframe using the 'melt' function (Jose, 2021), setting 'id' and 'starting date' as variables, and the time series observations as values. In this reshaped dataframe, the TimeSeries column contains the IDs of various time series, the StartingDate column represents the starting date and time of these time series, the Time column denotes the hours for a time series with an hourly frequency, the Observation column contains the values of the time series for each hour, and the Timeseries_id and StartingDate_id columns are encoded values of the time series and starting dates, respectively.

	TimeSeries	StartingDate	Time	Observation	TimeSeries_id	StartingDate_id
0	H1	01-07-15 12:00	1	605	0	0
1	H2	01-07-15 12:00	1	3124	1	0
2	H3	01-07-15 12:00	1	1828	2	0

Figure 1: First few rows of the final preprocessed M4-Hourly dataset.

The resulting dataframe, as illustrated in figure 1, presents the first three rows of the reshaped data. This conversion of the dataframe from wide format to long format not only enhances performance speed, but also simplifies the conversion to time series format used by Darts.

Darts (Herzen et al., 2022) is a library, specifically designed for time series forecasting. Darts contains various pre-built models, which simplify the process of training and forecasting, eliminating the need for extensive manual model construction. The models are custom-designed to perform optimally on time series datasets. Darts also employs its own data type, known as the Darts TimeSeries data type, for both training the models and outputting the results. To convert our pandas dataframe into this TimeSeries format, we use the "from_group_dataframe()" function provided by Darts. This function groups each observation into a separate time series based on its time series ID.

To further enhance the training process, Darts provides a feature that utilizes static covariates. These are characteristics of a time series that remain constant over time. Utilizing this feature has been proven to improve forecasting accuracy when dealing with multiple time series (Unit8 SA, n.d.). Therefore, we utilize this feature to assign the Starting Date and TimeSeries ID as

static covariates, to capture the temporal and categorical uniqueness of each series respectively. After this we scale the dataset between 0 to 1 and then split the scaled dataset into training and validation sets.

In our methodology, we instantiate various models using the Darts library, setting and adjusting certain hyper-parameters. We develop a custom function that trains the model, generates forecasts, and then calculates metric values by comparing the forecasted values with the validation set. The streamlined approach provided by Darts for training and forecasting time series data enables us to use the same dataset to generate forecasts using a variety of models, not limited to just N-BEATS and LightGBM.

Darts simplifies the calculation of various error metrics by easily matching predicted and validation values with their respective indexes, either by integer range index or time index. It supports the calculation of error metrics for multiple time series and also offers multivariate support (Herzen et al., 2022). Additionally, Darts allows for the creation of custom metrics that can work with its TimeSeries data type. For our research on the M4 dataset, we used the Weighted Absolute Percentage Error (WAPE) metric. While the Symmetric Mean Absolute Percentage Error (SMAPE) was used in the M4 competition, we opted for WAPE as it was the metric used in the Visuelle 2.0 paper (Skenderi et al., 2022). Our aim was to maintain consistency in our metric across our research. Although Darts did not have built-in support for the WAPE metric, we developed a custom WAPE function that integrates seamlessly with the Darts TimeSeries data type.

M5-Walmart Dataset:

The M5 Dataset reflects a hierarchical and multivariate structure. It is a collection of Walmart Sales Data that has different product-based data. Its hierarchical structure contains a variety of product collections at different levels. The M5 Dataset is a multivariate dataset and is considered an advancement over our previous dataset i.e. M4 which is univariate. The purpose of working with the M5 Dataset is to apply the same models that were used in the M4 competition and observe the differences in these models in terms of compatibility, performance, and accuracy under these new conditions.

The dataset consists of five .csv files. In particular, we import the ‘sales_train_evaluation.csv’, ‘sell_prices.csv’, and ‘calendar.csv’ files into pandas DataFrames using the ‘load_dataset’ function. These DataFrames are then named after their respective files, with ‘_df ’ appended to the end of each title.

The dataset includes categorical events as explanatory variables that influence the time series. To make these events interpretable by Darts models, one-hot encoding (DelSole, 2018) is applied to the event_name_1 and event_name_2 columns in calendar_df, where event 1 is the primary event and event 2 is second event taking place on the same date. This process transforms categorical events such as sports, holidays, global and local festivities into binary columns, with the occurrence of an event represented as 1 and its absence as 0. The original event columns are then dropped to conserve space. The sales_train_evaluation_df and sell_price_df are then merged separately with a subset of calendar_df, to align their indices. This ensures that these two DataFrames have the same index for further processing.

Given the complexity and size of the dataset, and considering our resource constraints, the dataset is subdivided for manageability. The sales_train_evaluation_df and sell_price_df DataFrames are filtered for a specific store and department, rendering the dataset more focused and manageable. This approach also allows for the flexible inclusion of various stores and departments as needed and offers the ability to adjust the filters to analyze different subsets of the dataset based on specific analytical needs or model training objectives.

The filtered train DataFrame is transformed from a wide to a long format through a melting process (Jose, 2021). Each row now corresponds to an individual day's sales for every product. This transformation also involves replacing the day numbers (d_1 to d_1941) into their respective dates, we name the column 'date' and a value column 'sales', containing the sales figures. The sell_prices DataFrame is merged with the calendar_df on the wm_yr_wk column, integrating selling price information with calendar details. A new id column is constructed in this merged price_calendar DataFrame by concatenating item and store ids. This newly melted sales dataframe 'train_melt_df', is then merged with the melted price and calendar dataframe, based on the 'id' and 'd' columns present in the two DataFrames, to create the final dataframe.

	id	date	sell_price	sales	wm_yr_wk	wday	snap_CA	snap_TX	snap_WI	event_Chanukah End	...	event_StPatricksDay
	HOBBIES_1_001_CA_2_evaluation	2013-07-27	8.26	1	11327	1	0	0	0	0	...	0
	HOBBIES_1_001_CA_2_evaluation	2013-07-28	8.26	0	11327	2	0	0	0	0	...	0
	HOBBIES_1_001_CA_2_evaluation	2013-07-29	8.26	1	11327	3	0	0	0	0	...	0

Figure 2: First few rows of final preprocessed M5 sample dataset.

The final DataFrame undergoes a cleanup process to remove unnecessary columns. Products with missing data, indicated by NaN values in the date column, are dropped. The date column is converted to a datetime object, and the columns are restructured to prioritize id, value, sell_price, date, and any event or SNAP columns. This meticulous data preparation process ensures the M5 Walmart dataset is ready for the subsequent stages of our research and compatible with darts library during the conversion of data into timeseries format.

Given the substantial size of the dataset, memory optimization is crucial. Unnecessary DataFrames are discarded to release memory. The reduce_memory function is utilized to decrease the memory footprint of the final DataFrame. Total memory usage is assessed, and garbage collection is performed to eliminate residual memory waste.

Optionally recognizing that models such as LightGBM would not work with string-encoded identifiers, a label encoder is employed. The encoder transforms the textual identifiers into numeric labels, which are then appended as a new id_encoded column to the sorted_data DataFrame.

As this dataset is rich in multivariate data, we need to identify data as past and future covariates (Herzen, 2021) during model training to enhance prediction accuracy. While the process of converting the dataset into a time series format mirrors the approach used for the M4 forecasting, this dataset introduces additional considerations. selling price data and all the events are treated as exogenous variables. These variables, while not the focus of prediction,

play a crucial role in influencing the product sales time series and are thus considered as covariates. Contrary to the target time series, these values are not scaled.

We perform similar operations as before using built-in darts models, setting up parameters and providing train and validation sets, along with this we also include exogenous data during both training and validations, there are built-in methods in darts models to take in multiple lagged values for both the target series as well as the covariates. In terms of metrics, our initial approach was to use the Weighted Absolute Percentage Error (WAPE). However, we encountered issues with division by zero errors. Consequently, we considered using the Weighted Root Mean Squared Scaled Error (WRMSSE), as suggested in the competition guidelines (Makridakis, Spiliotis and Assimakopoulos, 2022b). However, due to its large value and the complexity involved in programming it to work with time series data, we ultimately decided to use the Symmetric Mean Absolute Percentage Error (SMAPE) for our evaluations.

Visuelle 2.0 Dataset

The Visuelle-2.0 dataset is a multi-modal dataset for fast-fashion product sales forecasting. It contains real data for 5355 clothing products of the retail fast-fashion Italian company, Nuna Lie, from 2017-2019. The dataset contains 106879 time series. The data further contains HD images of the products, textual tags (such as category, color, and fabric), time series data (such as sales, inventory stock, prices, and discounts) disaggregated at the shop level. Furthermore, the dataset includes exogenous time series data like Google Trends based on item-based data and weather conditions of store locations and purchase data for 667K anonymized customers to capture personal preferences.

Visuelle 2.0 integrates textual tags, Google Trends, store locations, and images with time series data. Despite the time series' brief span of 12 weeks, the inclusion of diverse features compensates for this limitation during forecasting. This is in contrast to the M4 and M5 datasets, which have longer time series but fewer supporting features. The purpose of working with Visuelle 2.0 is to evaluate how these additional features paired with short length timeseries can impact the performance, speed, accuracy, and compatibility of the models when applied to this dataset.

Data preparation involves encoding categorical variables such as "release dates", "category", "color", and "fabric" using scikit-learn's LabelEncoder, integrating Google Trends data, and processing image data. Alternative encodings like one-hot and hash (Pahwa, 2024) were explored for categorical variables, but they added complexity without significant improvement. Image features from the "visuelle2.0" dataset was extracted using the VGG16 model (Simonyan et al. 2015), the input size sent to the VGG16 model was '1*224*224*3' and the output size obtained was '1*7*7*512'. In other words, if flattened, we obtained 25088 features for each image. To manage high dimensionality, we applied Principal Component Analysis (PCA) (Maćkiewicz and Ratajczak, 1993) to the image features, reducing them to 10 parameters that explain 95% variance. Alternatively, we tried taking the mean of the image features of each image to a single value. Both methods yielded similar results, so we opted for the latter method for its simplicity.

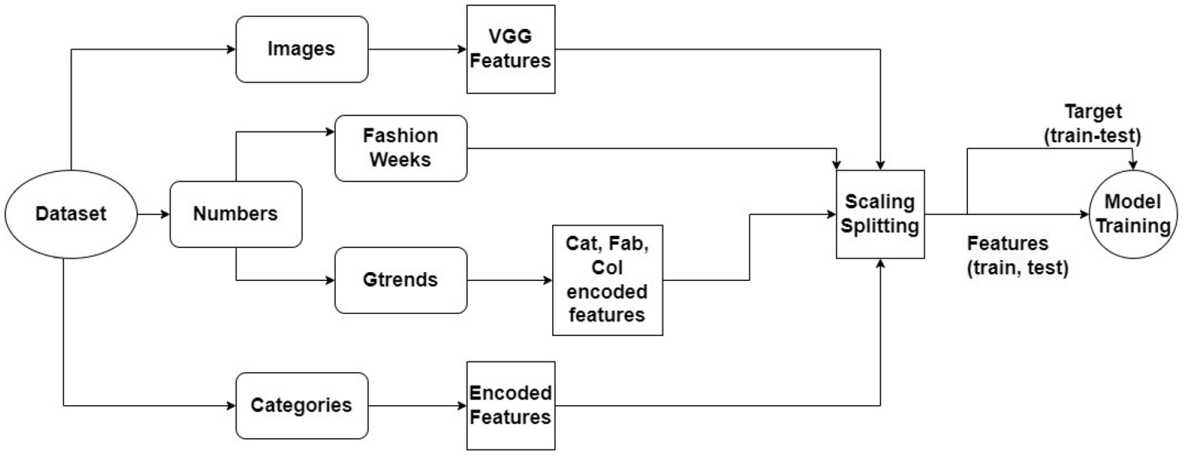

Figure 3: Flowchart of techniques utilized only in Visuelle2.0 dataset processing.

Additionally, Google Trends data is loaded from a separate CSV file, offering insights into search trends based on various textual tags. This Trends data is filtered based on the colour, fabric and category of every individual product. Only the trends data with the span of 52 weeks, before the product's release date is considered for the research. This trends data is then associated with the main dataset by creating new columns (cat_gtrend, col_gtrend, fab_gtrend) and merging trends data for the corresponding products. The final preprocessed dataset is then stored as 'dataFinal.pkl', in pickle format (Radečić, 2021), due to its speed and efficiency.

Data Preparation for the Training of forecasting Model includes some other basic steps. First, the preprocessed dataset is loaded from the 'dataFinal.pkl' file as a pandas dataframe. We then continue by reducing the dimensionality of the three distinct 'g_trend' columns by taking the mean of each of its rows, which simplifies it from 52 dimensions down to 1. The final shape of preprocessed dataframe is as illustrated in Figure 4. We then partition the dataset into training and testing sets. To further enhance the robustness and efficacy of the training process, these datasets are scaled using the StandardScaler from scikit-learn.

color_encoded	0	1	2	3	4	5	6	7	8	9	10	11	image_features	cat_gtrend	col_gtrend	fab_gtrend
4	1.0	3.0	1.0	1.0	2.0	1.0	0.0	0.0	2.0	0.0	0.0	0.0	2.499182	57.584906	17.886792	71.924528
7	1.0	1.0	1.0	0.0	0.0	2.0	0.0	0.0	0.0	1.0	1.0	0.0	2.578126	57.584906	59.207547	71.924528
4	1.0	3.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	2.499182	57.584906	17.886792	71.924528
9	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.695715	59.981132	84.773585	13.283019

Figure 4: First few rows of the final pre-processed Visuelle2.0 dataset before splitting into train and test sets.

These sets are then converted from dataframe into numpy arrays to meet the input requirements of the model. For the time series, we use the initial 10 weeks (0 to 9) along with image features and 'g_trend' columns as our feature set. The sales of the 11th week (10) serve as our target for model training, denoted as 'X_train' and 'y_train' respectively. For forecasting, we shift the time series to weeks 2 to 11 (1 to 10), again paired with image features and 'g_trend'

columns, to form ‘X_test’. We then predict the sales of the 12th week (11) to form ‘y_test’). Finally, we evaluate the model’s performance using the Weighted Absolute Percentage Error (WAPE) metric.

For LightGBM, the training and testing datasets transformed into LightGBM datasets (designated as train_data and test_data, respectively), and as a result, efficient training and evaluation of the LightGBM forecasting model. We don’t have such compatibility considerations with NBEATS.

Deep Learning Model

Deep learning, a subset of machine learning within artificial intelligence, has the remarkable ability to autonomously learn from data without the need for explicit supervision (Goodfellow, Bengio and Courville, 2016). This capability enables deep learning models to analyze vast amounts of unstructured and unlabeled data efficiently. Artificial Neural Networks (ANNs), which are fundamental units of deep learning architectures, employ multiple layers to extract intricate features from the input data. N-BEATS is one such example of a deep learning model that utilizes these principles to achieve superior performance in various tasks.

N-BEATS

N-BEATS is designed to analyze and predict complex, multi-dimensional data. Unlike traditional models, which can struggle with complex patterns, N-BEATS can learn directly from the data, bypassing the need for manual feature engineering. The architecture of N-BEATS consists of a stack of fully connected layers with residual connections placed into building blocks. Each block can be designed to focus on different components of the time series such as trend and seasonality. However, the model is flexible enough to handle other patterns as well. The input to the N-BEATS model is typically a sequence of past time series values and the output is a prediction of future values. Each block in the N-BEATS architecture contains fully connected layers followed by basis expansion layers, and the model iteratively refines its predictions through multiple backcasts and forecasts. What sets N-BEATS apart is its focus on "interpretability". The interpretable N-BEATS model breaks down time series into understandable components providing clear insights into its predictions. This is a clear difference from some traditional models, which can be opaque in their predictions. The following Figure 5 of the N-BEATS architecture illustrates its block structure and flow, highlighting how the model processes timeseries as input to generate forecasts.

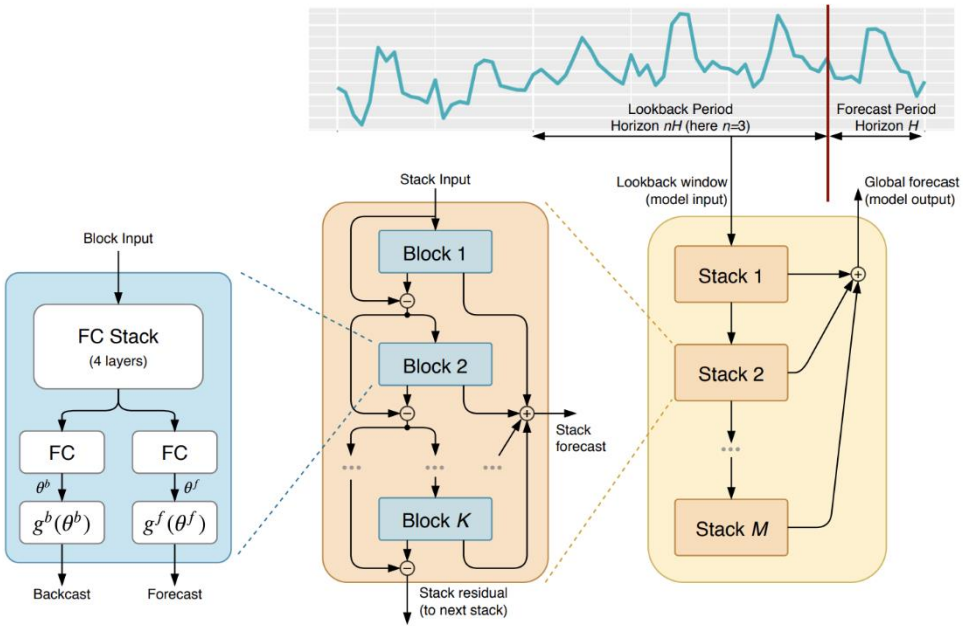

Figure 5: Overview of the N-BEATS architecture showing its stacked blocks of fully connected layers with an example time series data as input (Oreshkin et al., 2019)

In our work, we have utilized and developed two different implementations of the N-BEATS model. Firstly, we employed the N-BEATS implementation provided by the Darts library (Herzen et al., 2022) for forecasting tasks on the M4 and M5 datasets. For the Visuelle 2.0 dataset, we utilize the PyTorch (Paszke et al., 2019) Implementation of the N-BEATS model. PyTorch is known for its flexibility and ease of use, and it let us customize and extend the N-BEATS architecture to better suit the unique characteristics of the Visuelle 2.0 dataset. Our N-BEATS model architecture comprises two generic stacks, supplemented by a seasonal stack and a trend stack. This configuration also enables our model to effectively capture a wider spectrum of patterns in the Visuelle 2.0 dataset. Using different implementations provided us with the opportunity to leverage the strengths of each library and optimize our model for specific datasets.

In addition to this, we also developed a custom N-BEATS model using TensorFlow's Keras Sequential (Chollet et al., 2018).The motivation to develop a custom N-BEATS is for the sake of experimentation and to leverage TensorFlow's extensive support for model customization and scalability. The libraries TensorFlow (Abadi et al., 2016), PyTorch, and Scikit-learn (Pedregosa et al., 2016), provide the necessary modules. Similar to PyTorch model, this custom implementation includes a generic stack. The generic stack is useful because it allows the model to learn a broad set of features that are not specific to trend or seasonality only therefore allowing it to adapt to various types of data inputs. This ability to adapt is crucial for time series data that contain complex patterns that might not be purely trend or seasonal based. The Visuelle 2.0 dataset includes various data such as generic image features, Google trends, and seasonal sales fluctuations, the generic stack can effectively learn and generalize from this diverse data.

Machine Learning Model

Machine learning models use a variety of methods to learn from data and make predictions. They are different from the deep learning model we discussed earlier. These models can learn and predict without being explicitly told how to do so. For example, LightGBM is a type of machine learning model that is good at dealing with large amounts of data.

LightGBM

LightGBM is a machine-learning model known for its efficiency in handling complex datasets. Unlike traditional models that may take a long time while working with complex data patterns, LightGBM was developed to achieve faster training speeds while retaining similar accuracy with other ML models. This enables it to handle large-scale datasets with ease and increases its scalability. Given its ability to learn from both linear (relationships that can be represented as straight lines) and nonlinear patterns (more complex relationships not linearly separable), LightGBM model is well-suited for time-series forecasting tasks as temporal features inherent in time-series datasets are effectively captured.

In our study, we utilized the LightGBM implementation provided by the Darts library for forecasting tasks on the M4 and M5 datasets. In the Darts implementation, we adjusted several key parameters like boosting type, learning rate and a few others to evaluate the model's performance and accuracy. For the Visuelle 2.0 dataset, we inherited a similar approach of model definition/architecture, however, we utilized PyTorch due to its comprehensive data preprocessing capabilities. For training in the model and forecasting of the sales data, we employed the official LightGBM library in conjunction with scikit-learn to provide more flexibility.

Hyper-parameter Optimization

Hyper-parameters are the configuration variables of the model, which are not learned from the data but are set prior to the training process. They can significantly influence the performance of the model, and their optimal values often depend on the specific dataset. In this study, we explored Optuna, to help fine-tune the parameters of all models before applying cross-validation.

Optuna

Optuna (Akiba et al., 2019) is an open-source platform for hyper-parameter tuning. For each model, we created a hyper-parameter search space and utilized Optuna to find the optimum hyper-parameters to reduce validation errors. The search space included various model-specific parameters, such as the number of layers and neurons for neural network models, the number of trees and their depth for tree-based models, etc. The optimization was performed separately for each model to account for their unique characteristics and requirements. The use of Optuna for hyper-parameter optimization helped us ensure that each model was tuned to its best performance before the cross-validation process. This not only improved the accuracy of our

models but also allowed us to make a fair comparison between them because the optimized models were then used in the subsequent cross-validation process.

The most common hyper-parameters that are part of the search space and hyper-parameter in almost all models are:

Epochs : An epoch refers to a single iteration through the entire dataset during training. The number of epochs used has an impact on the model's capacity to learn patterns from data because the model's internal parameters are adjusted with each epoch. As a result, utilizing too few epochs may result in underfitting, which occurs when the model fails to capture the underlying relationships in the time series. On the other hand, too many epochs might result in overfitting, which causes the model to memorize the training data and perform badly on new data.

Input / Output Chunk Length : The Input and output chunk lengths represent; the amount of historical data used as input to make predictions, and the length of the predicted future sequence, respectively. As this length is a numerical variable a shorter input chunk length might capture finer-grained patterns but might miss longer-term trends and patterns, and vice versa. In the case of Output Chunk Length, a small value might provide fine predictions but might lack context, while a bigger/large output chunk value could provide more contextual predictions but might be less accurate for short-term trends.

Batch Size : Batch size determines the number of training samples/observations used in each iteration. The choice of batch size is considered on many things e.g., available resources, dataset characteristics, and the model architecture.

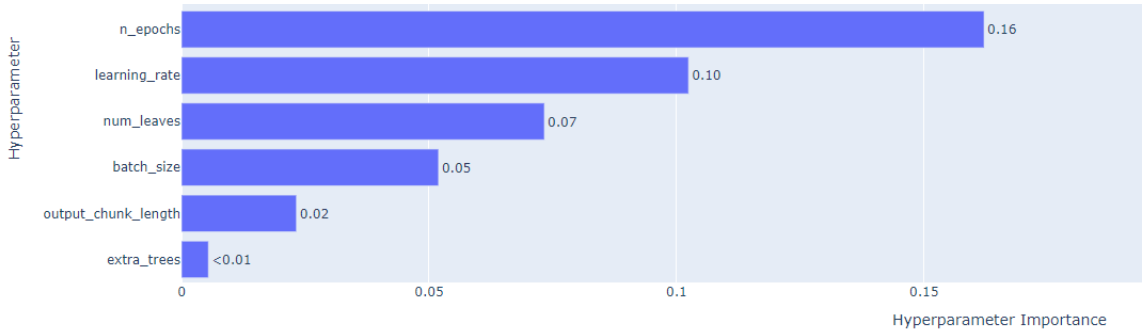

Figure 6: A Feature importance example visualization derived from the M4-Daily-Train dataset after optimization within the search space for N-BEATS model.

Therefore, from the standpoint of hyper-parameter optimization, the exploration of the Search Space of Optuna involves experimenting with various combinations of hyper-parameters, including Epochs, Input Length, Output Chunk Length, and batch sizes. In the optimization process with Optuna, the same training and validation sets as the base model run are used. The optimal model's performance is gauged by the error metric values, with lower values being better. This error metric is consistent with the one used for the dataset evaluation. The scope of the search space extends beyond the above-mentioned parameters and can accommodate other parameters of the models.

Cross Validation

In machine learning, cross-validation is a crucial technique used for evaluating the performance of a model during training and validation. It plays a significant role in time-series forecasting as it helps ascertain whether the model is overfitting or underfitting, and how effectively it captures the underlying patterns and dynamics of the time series. Unlike traditional cross-validation, where data points can be randomly shuffled, the order of data points is critical in time series. Therefore, time series cross-validation preserves the temporal order by partitioning the data into chronological groups. This approach evaluates the model's ability to predict future data based on past observations, providing insights into its effectiveness in anticipating future patterns in the series. In our research, the techniques we use are Walk Forward, Group Time Series, Purged Group Time Series, and Combinatorial purged group Cross Validation.

Walk Forward Cross Validation

The Walk Forward Cross Validation technique is a fundamental method used in time series cross-validation. This technique respects the temporal order of the data while splitting the dataset into folds, similar to how folds are created in traditional cross-validation methods. In each iteration, the validation set and the training set from the previous iteration are merged to form the new training set for the current iteration. This process continues until all folds have been processed.

The TimeSeriesSplit function from the sklearn.model_selection module in the Scikit-learn library is utilized to perform this technique on all three datasets. This function provides train/test indices to partition time series data samples that are observed at fixed time intervals. This ensures that the temporal structure of the data is preserved during the cross-validation process.

Figure 7 represents a 4-fold Walk Forward Cross Validation. In the first iteration, the first fold is taken as the training set (denoted in blue) and the second fold is taken as the validation set (denoted in green). The red line represents the predicted values, it is not mandatory for the prediction to start or end at the same time as the validation set; these parameters are user defined, In Darts, however the prediction starts after the training set ends. In the second iteration or repetition, the first and second folds are combined to form the new training set, and the third fold is used as the validation set. In the third iteration, the first, second, and third folds are merged to form an even larger training set, and the fourth fold is used as the validation set. Figure 7 also illustrates how the predicted values in red fit to the actual values in green.

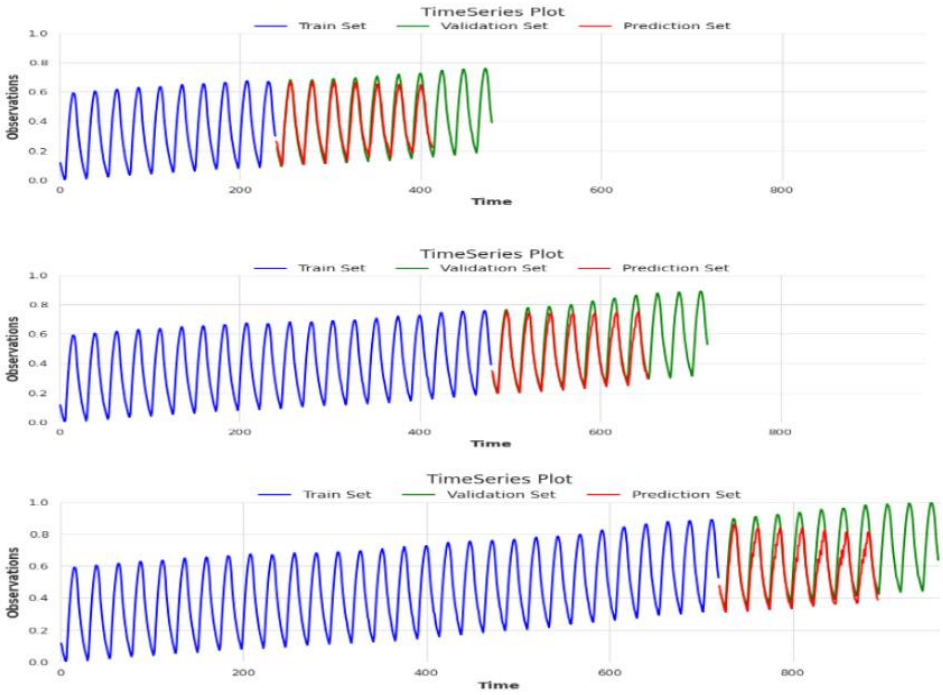

Figure 7: Walk Forward Cross-Validation plot of a sample time series with the training set (green), validation set (blue), and predicted values (red) of user-defined length.

While this method is advantageous in certain scenarios, it does have its limitations. For instance, as depicted in figure 7, the accuracy of forecasts decreases with each iteration. This may happen considering, with each iteration, the size of the training set increases, and could potentially lead to overfitting or underfitting, depending on the data.

To address the issue of increasing training data size in Walk Forward Cross Validation, a variation known as Rolling Cross Validation is employed. Figure 8 depicts this technique, operating similarly to its predecessor.

In the first iteration, similar to the Walk Forward method, the first fold is taken as the training set (denoted in blue) and the second fold is taken as the validation set (denoted in green). The red line represents the predicted values, which follow the same start time as the validation set but for user specified time steps. In the second iteration, instead of combining the first and second folds, the second fold becomes the new training set, and the third fold is used as the validation set. This is because the size of the training set is kept constant. In the third iteration, the third fold becomes the training set, and the fourth fold is used as the validation set and so on.

The training size limit at times allows the data from the previous fold to still be part of the training set in the subsequent iterations, along with the data from the respective current fold, as depicted in figure 8. This training size is not strictly tied to the fold size but is dependent on the training size limit that we set.

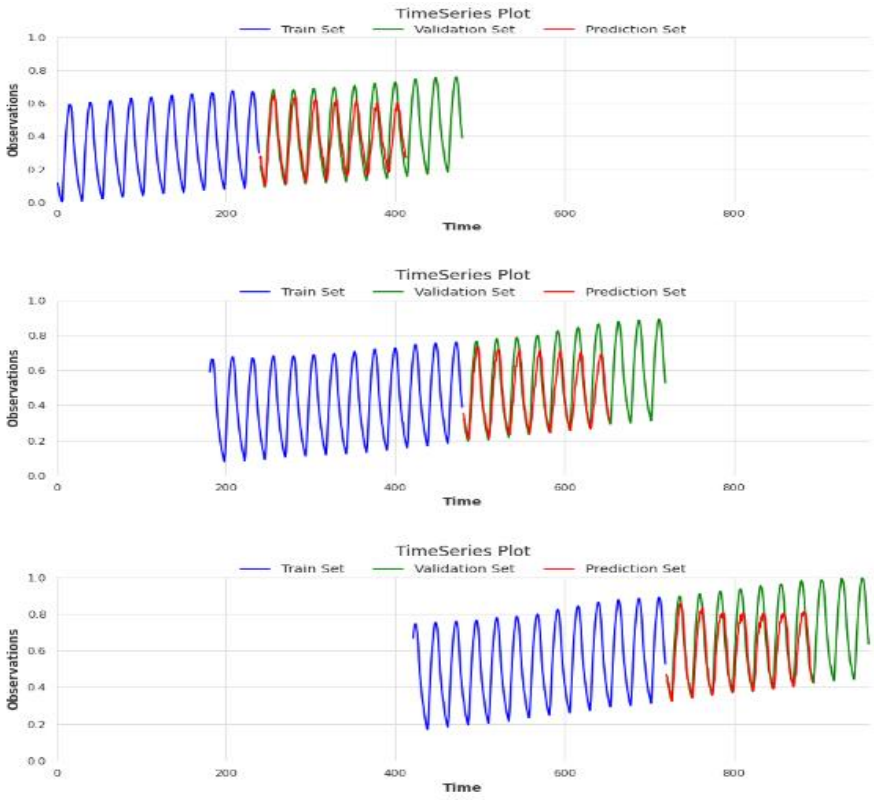

Figure 8: Rolling Window Cross-Validation plot of a sample time series with the training set (green), validation set (blue), and predicted values (red) of user-defined length.

While this modification addresses the issue, it does not guarantee better model performance. This method might not capture long-term temporal dependencies as effectively as compared to Walk Forward Cross Validation, due to smaller training size. However, this technique provides valuable insights into the consistency of the model's accuracy across different folds.

While the Walk Forward and Rolling Cross Validation techniques acknowledge the temporal order of the time series, they do not consider the frequency of the dataset. For instance, in the example dataset, cyclic patterns recur with a 24-hour periodicity. When these CV methods split the dataset between these 24 hours, the model may either over-train on specific group patterns or completely overlook them. This could potentially impact the model's prediction accuracy. We call this temporal dependency.

This is a common characteristic of time series data, where values at one time period can influence values at a future time period. for example, if the sales value of certain products during the day affects the products' sales value during the night, this would influence the model to be biased or overfit and would influence the prediction result.

Grouped-Time-Series Cross Validation

Addressing this problem, we implement Grouped timeseries Cross Validation. It is like a crossover between TimeSeriesSplit and GroupKFold, one that respects the temporal order of

the series and ensures that samples from the same group do not appear in both training and testing, reducing or eliminating temporal dependency.

Since a Grouped Time, Series Cross-Validation implementation is often not readily available in standard machine learning libraries, we had to work with a custom solution developed by a GitHub user named Gaurav Chawla (Chawla, 2023).This implementation requires an explicit 'group' column in our dataset, indicating the group membership of each data point or sample

The process, as figure 9 depicts, follows a similar pattern to Walk Forward Cross Validation. In the first iteration, the first fold is taken as the training set (denoted in blue)and the second fold as the validation set (denoted in green). In the second iteration, the first and second folds are combined to form the new training set, and the third fold becomes the validation set. The values (denoted in red) are the predicted values of the timeseries that occur at the same interval as the validation set but only for user specified time steps.

This process continues until the last fold has been used for validation. The area between dotted lines denotes the groups, and if we examine these groups in conjunction with the time series data, we can observe that the groups are formed for every 24 values of the hourly dataset.

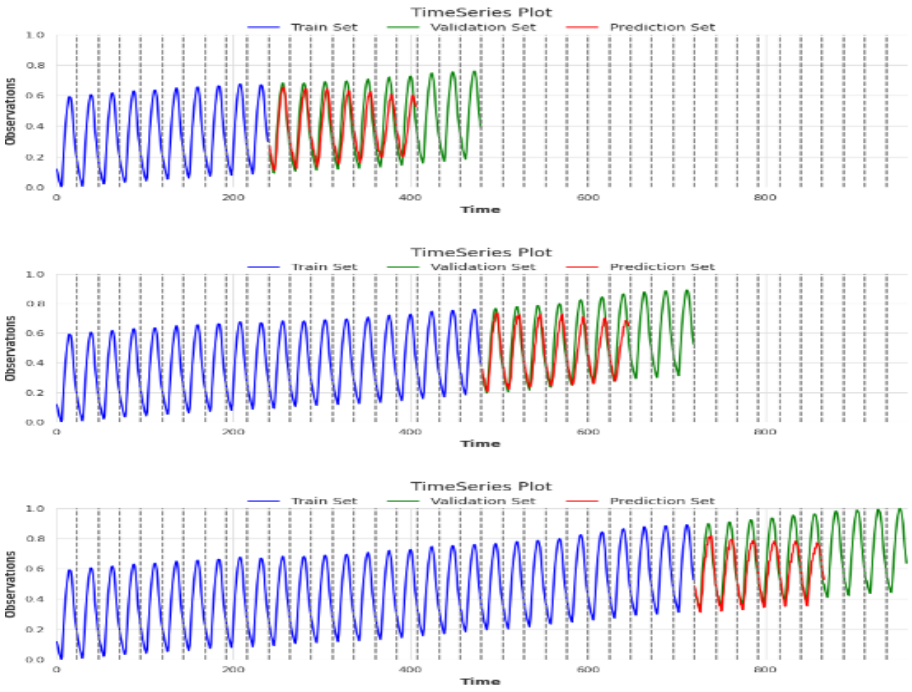

Figure 9: Grouped Time-series Cross-Validation plot of a sample time series with the training set (green), validation set (blue), and predicted values (red) of user-defined length.

In Group Cross Validation, data is grouped according to a certain periodicity before the cross-validation process begins. For instance, for the hourly data, we might group it into daily data (i.e., 24 hours are grouped as 1 day). This grouping is done before the training and validation sets are created for each fold.

While there is no definitive research suggesting the best way to group time series data, a common practice is to group the data according to recycling patterns. This approach can help

to reduce the noise in the data and reveal underlying patterns that are relevant on a daily, weekly, or other periodic basis. The utilization of group features may enhance the accuracy of the model compared to previously discussed cross-validation techniques, due to the absence or reduction of temporal dependencies, aids in mitigating overfitting or underfitting and would yield a more accurate estimation of the model's generalization performance on unseen data.

Purged Group Cross Validation

While the implementation of Group Time Series addresses the issue of grouped data, it is important to note that there exists a potential for subtle dependencies between successive groups within this dataset, for instance the present-day hourly sales influenced by previous day or stock prices influenced by election results happened the previous day.

These dependencies can sometimes lead to look ahead bias, while splitting the dataset. This can make the model predict values based on the information that is not known during the time of prediction. To mitigate this, it becomes important to create a buffer between the training and validation sets, this is called purging. The process of purging is helpful in addressing these issues by removing observations in the training set that are temporarily close to the validation set

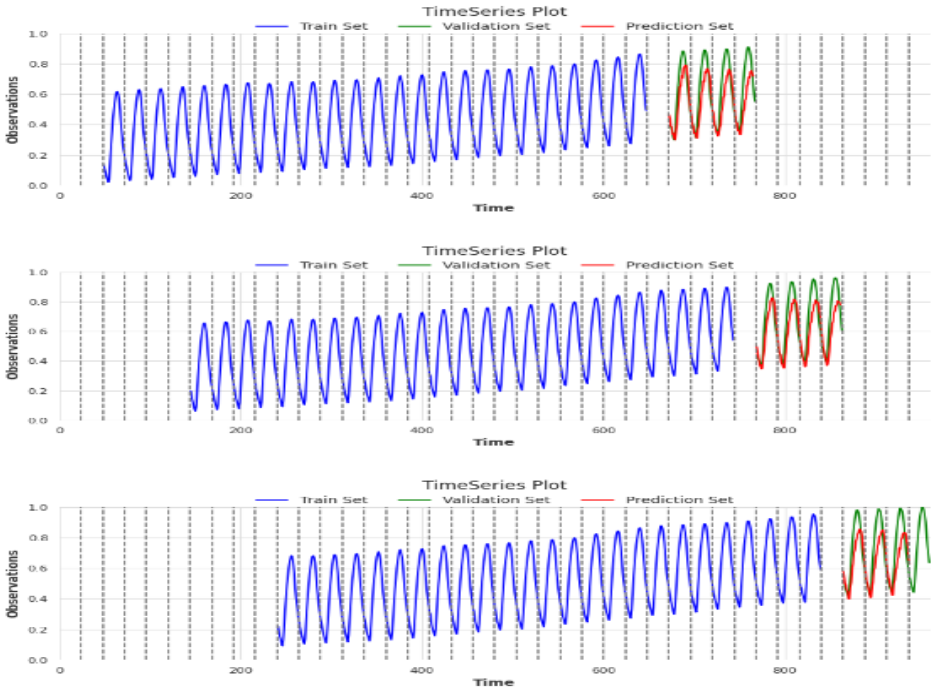

Figure 10: Purged Group Cross-Validation plot of a sample time series with the training set (green), validation set (blue), and predicted values (red) of user-defined length.

To implement Purged Group Cross Validation, we have worked on a custom variation of group cross-validation, developed by a user on Kaggle, called "marketneutral". This custom approach as illustrated in figure 10, addresses Look-Ahead Bias by introducing purging. This process effectively 'hides' the purged portion of the dataset, ensuring it is not considered for either

training or validation and also has the option of setting training size limit if needed. However the optimal purging size varies depending on multiple factors such as serial correlation of data, the forecasting horizon and also the model we use. For our research we have taken the size of a single group as the purging size as a starting point, but there is the option to set any size as required. This implementation also respects the inherent grouped structure of timeseries data by purging entire groups of data, rather than dividing the data within a group.

Combinatorial Purged Group Cross Validation

Combinatorial Purged Group Cross Validation works in a different way than above mentioned approaches, but it incorporates some of their principles to some extent. For Instance. It begins by dividing the entire dataset into folds, determined by the 'N' variable. From these divisions, 'k' segments are designated as the validation set with the rest forming the training set. When 'k' equals 1, this Cross Validation mirrors the K-Fold Cross Validation scenario. In Combinatorial Purged Cross Validation, 'I' iterations are performed, each with a unique training-validation fold combination, which is calculated by the below formula and finally the mean is taken to show the desired metric results.

$$Number\ of\ Iterations\ 'I' = \binom{N}{N-k}$$

We've applied the Combinatorial Purged Group Cross-Validation technique exclusively to the Visuelle 2.0 dataset due to its incompatibility with the Darts Timeseries object. This incompatibility arises because the Timeseries object requires a continuous time index in the training and validation sets and can only predict future values based on the end of the training set's time index.

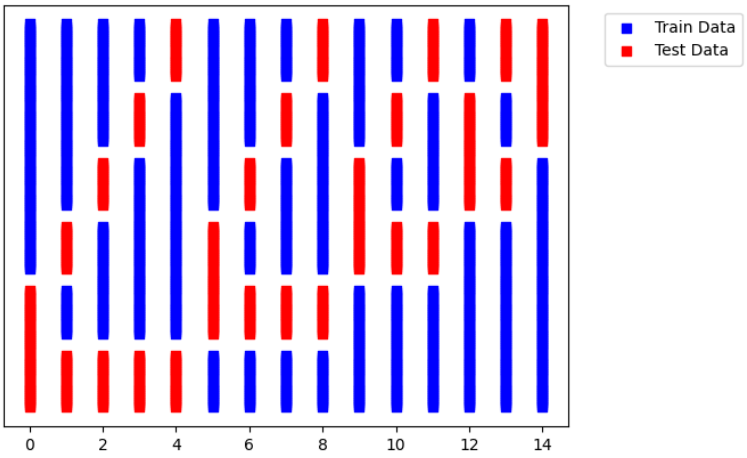

Figure 11: Combinatorial Cross Validation of a sample time series demonstrating 4-fold (blue) as training and 2 as test (red) at each iteration

It employs the Purging process to prevent information leakage, and also uses a method called Embargo (Berend, 2022). Similar to Purging, Embargo removes a portion of the data, but it does so at the end of the test set and before the start of the training set. Similar to purging, the optimal embargo size depends on the same factors and as a starting point, we have set equal

sizes to both purging and embargo, that is, the size is set to the size of 1 group. For Visuelle dataset, products are grouped based on the product release dates.

As shown in Figure 11, we divide the entire dataset into 'N' folds, with 'N' being 6 in this case. Out of these, 'K' folds (where 'K' equals 2) are selected as test sets. To prevent information leakage, we apply purging and embargo processes, both set to remove 'n = 1' group, as depicted in Figure 11. This approach, unlike other cross validations discussed before, utilizes all data in each iteration and for each iteration, different combinations of the data are selected as the training and testing sets, denoted by blue and red colors respectively.

Validity and Reliability:

In order to ensure the validity of our findings, we chose the error metrics recommended by prior studies discussed in our literature review. The error metrics are as per the characteristics of datasets, to help evaluate the model performance accurately. We used multiple cross-validation methods across all datasets to verify the precision (accuracy of predictions) and generalizability (ability to perform well on new data) of our models. We maintain reliability (consistency of results) in our approach by utilizing similar pre-processing techniques and Python libraries. Furthermore, the parameters of the chosen model remained largely consistent (stability of model parameters) throughout our study.

Results

This chapter is dedicated to the detailed presentation and interpretation of the results obtained from each dataset for all the implemented models and cross-validation techniques. It focuses on the computational efficiency and performance of the N-BEATS and LightGBM models across the Visuelle 2.0, M5, and M4 datasets.

M4 Hourly Dataset

In our study, we employed the M4 Hourly Dataset from the Makridakis Competition. This dataset consists of 414 unique time series, with some having 960 observations and others having 700 observations. The time series are synthetic and randomly selected from a larger dataset. Upon analyzing the correlation matrix, we discovered that the time series did not exhibit any significant correlation.

We used the Weighted Absolute Percentage Error (WAPE) as our evaluation metric because it is scale-independent and as it normalizes errors by the sum of realized values.

As the Darts library has various built-in models and Examining Darts library's refinement for timeseries forecasting was one of our intentions. We tested the m4 hourly dataset on 5 different models on the default parameters of those pre-built models from Darts (Herzen et al., 2022). A noteworthy advantage of the using Darts is its ability to handle time series data, which facilitated the use of the same data type as input across all models. This feature streamlined the process and allowed for a more consistent analysis across different models.

In our study, we work on the N-BEATS and LightGBM models, adjusting parameters such as output_chunk_length (the number of time steps to be predicted), max_depth (the maximum depth of the tree in the LightGBM model), n_estimators (the number of boosting stages or trees to learn sequentially from the residuals of prior trees), and learning_rate (a hyperparameter controlling the adjustment of model weights with respect to the loss gradient). We also explored other models available in Darts. However, their results are not included in our final analysis as they are not relevant to our research focus.

Model	WAPE	Training Time	Forecast Time
LightGBM	0.281459	**1.092325**	**0.399977**
N-BEATS	**0.234081**	398.246493	0.534523

Table 1: Comparison of metrics, training times for models on M4-hourly Dataset

The wape value is calculated between the validation set and the predicted set which has been explained in methodology. Darts considers the time index of the values in both sets. This ensures that the values from both sets are compared at the corresponding time points in the time series. Darts also calculates the desired metric of multiple time series (414 timeseries) and takes the mean of all the wape values to give the consolidated wape result.

In conclusion, The LightGBM model, a gradient-boosting decision tree model, was fast in both fitting and predicting and also had a respectable WAPE value. N-BEATS, a deep learning model, showed promising results, but training time was significantly higher when compared.

Hyper Parameter Optimization:

In the course of our research, we implemented Optuna, to fine-tune the Darts LightGBM model applied to the M4 hourly dataset. We tried performing Optuna on our Darts N-BEATS model. However, due to resource constraints and the observation that hyperparameter improvements did not yield a substantial increase in accuracy or performance, we deemed the long training times unjustifiable. It remains unclear whether the issue is inherent to the Darts models, as our custom-built N-BEATS model exhibited faster performance on a larger Visuelle dataset.

The primary objective of this exercise was to improve the predictive accuracy and overall performance of the model. The Weighted Absolute Percentage Error (WAPE) metric served as the benchmark for evaluating the model's accuracy. Within the Optuna framework, our objective function was explicitly defined to minimize the WAPE value, thereby guiding the selection of parameters for modification.

Trial	WAPE	Training Time (LightGBM)	Forecast Time (LightGBM)	Total Time (LightGBM)
1	0.74	2.04sec	0.73sec	2.78sec
2	0.66	1.67sec	0.50sec	2.17sec
3	0.26	0.82sec	0.56sec	1.38sec
4	0.42	1.07sec	0.38sec	1.46sec
5	0.34	0.96sec	0.43sec	1.39sec
6	0.33	3.58sec	0.41sec	3.99sec
7	0.28	1.59sec	0.44sec	2.03sec
8	0.41	1.08sec	0.39sec	1.47sec
9	0.51	0.74sec	0.56sec	1.29sec
10	**0.23**	**1.23sec**	**0.58sec**	**1.81sec**

Table 2: Optuna Hyperparameter Optimization of LightGBM model on M4-Hourly

In our research, we initiated an Optuna study on the LightGBM model. After conducting ten trials, each exploring different combinations of parameter values such as output_chunk_length, max_depth, n_estimators, learning_rate, we observed a decrease in the WAPE value in several instances. The optimal trial, although no considerable improvement over the base run, as indicated in Table 2, yielded a set of values that were subsequently used for cross validation.

Cross Validation

We perform 4 different types of time series cross validation on these two models to compare them in terms of their forecasting accuracy and training time.

Walk Forward Cross Validation

While Walking through the results of the walk-forward cross-validation from the perspective of each model, we observe the following:

Walk Forward Cross Validation	N-BEATS		LightGBM	
	WAPE	Training Time (seconds)	WAPE	Training Time (seconds)
Fold 1	0.37	132.65	0.45	0.3
Fold 2	0.3	268.81	0.37	1.47
Fold 3	0.34	417.27	0.35	1.93

Table 3: Walk Forward CV comparison of models on M4-hourly.

With N-BEATS, the model consistently achieves lower WAPE values across all folds, suggesting better accuracy. However, its longer training time is a drawback in scenarios where computational efficiency is a priority like predicting sales or stock prices etc. Despite not being as fast as LightGBM, our custom implementation of N-BEATS was able to achieve better speed with a larger dataset.

On the other hand, the LightGBM model, despite having slightly higher WAPE values, compensates with its remarkably fast training times. This suggests that LightGBM might be a more suitable choice for applications that require rapid model training without a significant sacrifice in accuracy.

Rolling Cross Validation:

While going through the results of the rolling cross-validation from the perspective of each model, we observe the following:

Rolling Cross Validation	N-BEATS		LightGBM	
	WAPE	Training Time (seconds)	WAPE	Training Time (seconds)
Fold 1	0.36	119.98	0.45	0.26
Fold 2	0.33	156.36	0.36	0.45
Fold 3	0.31	158.86	0.32	0.5

Table 4: Rolling Cross Validation comparison of models on M4-hourly.

With N-BEATS, the model consistently achieves lower WAPE values across all folds even here, However, its longer training time is a major setback in scenarios where computational efficiency is a priority.

On the other hand, the LightGBM model, despite having slightly higher WAPE values, compensates with its remarkably fast training times which makes it a more suitable choice for applications that require rapid model training without a significant sacrifice in accuracy.

Grouped TimeSeries Cross Validation:

Grouped Cross Validation	**N-BEATS**		**LightGBM**	
	WAPE	Training Time (seconds)	WAPE	Training Time (seconds)
Fold 1	0.35	117.02	0.41	0.34
Fold 2	0.35	266.83	0.36	0.74
Fold 3	0.31	414.69	0.34	1.41

Table 5: Grouped CV comparison of models on M4-hourly.

Upon evaluating the results of the Grouped TimeSeries Cross Validation from the perspective of each model, we find that the pattern of outcomes align with those of previous cross-validation methods.

Purged Group TimeSeries Cross Validation:

Purged Grouped Cross Validation	**N-BEATS**		**LightGBM**	
	WAPE	Training Time (seconds)	WAPE	Training Time (seconds)
Fold 1	0.33	118.42	0.36	0.37
Fold 2	0.37	117.72	0.33	0.32
Fold 3	0.37	118.27	0.35	0.33

Table 6: Purged Group CV comparison of models on M4-hourly

In the process of assessing the outcomes of the Purged Group TimeSeries Cross Validation, we observed that each folds' outcomes align with the previous cross-validation techniques.

M5 Dataset results

In our research, we employed the M5 Walmart Dataset from the Makridakis Competition. This dataset encompasses data for 3049 distinct products, categorized into 3 product types and 7 departments, retailed across 10 stores in 3 states. It inherently comprises 30,490 unique time series. However, given the hierarchical nature of this dataset, it's feasible to aggregate data across various levels and combine these timeseries, thereby yielding a total of 42,840 time series. Due to resource limitations, our study focused only on a subset of this dataset, as previously explained in our methodology.

We initially considered the Weighted Absolute Percentage Error (WAPE) as our evaluation metric. However, we encountered potential division by zero errors, making it unreliable in certain scenarios. We ultimately chose SMAPE for its scale-independent nature. Additionally, we used Root Mean Square Error (RMSE) as well because while SMAPE provides a relative error measure, RMSE helped understanding the magnitude of errors.

In our study, we evaluated the M5 Walmart dataset using our selected models from the Darts library, the N-BEATS and LightGBM. Our primary objective was to assess the capability of these models to handle intermittent time series data, where values are recorded sporadically. The M5 dataset exemplifies this, as product sales are not always recorded, and the value at these intervals is considered zero, denoting no sales. We incorporated various types of covariates to enhance the model's accuracy. The Darts library facilitated this process by seamlessly integrating these covariates as inputs.

Model	SMAPE	RMSE	Training Time (sec)	Forecast Time (sec)	Total Time (sec)
LightGBM Model	**144.59**	**0.1**	**2.67**	**1.5**	**4.17**
N-BEATS Model	147.18	0.1	2339.39	1.31	2340.7

Table 7: Comparison of metrics, training times for models on M5 Sample dataset

In broader context, neither of these 2 models emerged as the superior performer. However, we observed that the LightGBM model was the fastest, taking less than 3 seconds to train. The N-BEATS model underperformed, exhibiting a higher Symmetric Mean Absolute Percentage Error (SMAPE) value while requiring 2339 seconds to train (almost 1000 times slower than LightGBM).

Hyperparameter optimization:

In the course of our research, we implemented Optuna, to fine-tune the Darts LightGBM model applied to the M5 Walmart dataset. The primary objective of this exercise was to improve the predictive accuracy and overall performance of the model. The SMAPE and RMSE metrics served as the benchmark for evaluating the model's accuracy with more weightages given

SMAPE(80%). Within the Optuna framework, our objective function was explicitly defined to minimize the average metric value, thereby guiding the selection of parameters for modification.

We tried performing Optuna on our Darts N-BEATS model. However, similar to the M4 evaluation, due to resource constraints and also the first few hyperparameter improvements did not yield a substantial increase in accuracy or performance but had long training times and often led to premature termination of the model training process, attributable to insufficient RAM or processing capacity. It's worth noting that this challenge may be due to larger dataset size and the inherent characteristics of the few Darts models.

We initiated an Optuna study on the LightGBM model and, after conducting ten trials exploring different combinations of parameter values, we observed varying levels of the average metric value in these trails. The optimal trial, as indicated in the table, yielded a specific set of parameter values, 'n_epochs': 1, 'batch_size': 8, 'output_chunk_length': 2, 'max_depth': 5, 'n_estimators': 20, 'learning_rate': 0.1034149480909358, 'num_leaves': 44, 'extra_trees': True, that were consequently used in further experiments.

Trial Number	SMAPE	RMSE	Training Time	Forecast Time	Total Time
1	145.89	0.09	31.29sec	2.01sec	33.31sec
2	146.55	0.1	24.62sec	2.12sec	26.74sec
3	145.89	0.09	11.93sec	2.29sec	14.21sec
4	145.72	0.09	22.07sec	4.01sec	26.08sec
5	146.03	0.09	45.11sec	2.04sec	47.15sec
6	**145.48**	**0.09**	**18.04sec**	**2.83sec**	**20.88sec**
7	146.01	0.09	27.55sec	1.99sec	29.31sec
8	146.01	0.09	23.45sec	2.51sec	25.97sec
9	146.62	0.09	45.99sec	1.99sec	47.97sec
10	146.34	0.09	13.70sec	3.07sec	16.77sec

Table 8: Optuna Hyperparameter Optimization of LightGBM model on M5 Sample Dataset

Cross Validation:

We opted not to conduct other forms of cross-validation, as the methods detailed below offered a higher degree of sophistication than traditional walk-forward and rolling cross-validations. Given the larger dataset, characterized by a greater number of observations and time series, we employed a five-fold cross-validation strategy as opposed to the three-fold approach utilized in the M4 dataset.

Grouped TimeSeries Cross Validation:

Group Cross Validation	N-BEATS			LightGBM		
	SMAPE	RMSE	Training Time (seconds)	SMAPE	RMSE	Training Time (seconds)
Fold 1	161.94	1.31	458.13	158.39	1.53	3.89
Fold 2	162.06	1.26	846.91	159.43	1.4	2.26
Fold 3	163.66	1.2	1239.06	160.24	1.39	4.59
Fold 4	169.1	0.96	1573.57	163.98	1.18	3.83
Fold 5	154.95	1.23	1971.25	151.77	1.37	4.61

Table 9: Group Cross Validation Results for M5 Sample Dataset

In this cross-validation method, we can observe that.

- N-BEATS generally has lower RMSE scores than LightGBM, indicating better accuracy in terms of absolute errors.
- However, LightGBM often has lower SMAPE scores, especially in later runs, suggesting it performs better when considering proportional errors.
- The LightGBM is significantly faster to train and forecast compared to N-BEATS.

Purged Group TimeSeries Cross Validation

Purged Group Cross Validation	N-BEATS			LightGBM		
	SMAPE	RMSE	Training Time (seconds)	SMAPE	RMSE	Training Time (seconds)
Fold 1	166.53	1.1	456.96	160.25	1.38	1.88
Fold 2	164.57	0.93	461.12	165.43	1.12	1.84
Fold 3	172.61	0.98	461.64	166.64	1.15	1.83
Fold 4	156.82	1	459.83	156.96	1.1	2.44
Fold 5	154	1.16	458.43	150.95	1.27	1.8

Table 10: Purged Group Cross Validation Results for M5 Dataset

In this cross-validation method, similar to group cross validation, we can observe that.

- N-BEATS generally has lower RMSE scores than LightGBM, indicating better accuracy in terms of absolute errors.
- However, LightGBM often has lower SMAPE scores in 3 out of 5 folds, especially in later runs, suggesting it performs better when considering proportional errors.
- The LightGBM is significantly faster to train and forecast compared to N-BEATS.

In conclusion, the reason for nearly similar error metric values between N-BEATS and LightGBM is due to the M5 forecasting competition dataset being derived from real-world store sales data, offering a more complex and comprehensive challenge compared to benchmark dataset like M4.

This is also a reason for longer training time as the dataset is very large for complex ML models to train on google colab free tier resources. Introduction of more capable resources may result in better efficiency and favorable results.

Visuelle 2.0 Dataset Results

This table presents the results of the evaluation metrics, Mean Absolute Error (MAE), and Weighted Absolute Percentage Error (WAPE), for the Visuelle 2.0 dataset across two different forecasting models: LightGBM and the N-BEATS model consisting of 2 different implementations, the PyTorch implementation and custom implementation. The error metric is calculated on multiple time series and the mean is taken to consolidate them into a single value.

Models	N-BEATS PyTorch	LightGBM	N-BEATS custom
WAPE	110.61	110.50	113.60
MAE	0.71	0.71	0.73

Table 11: Comparison of Metric values between models on Visuelle2.0

Based on the table above, it's noticeable that the LightGBM model performs slightly better than the N-BEATS implementations. While the PyTorch implementation of N-BEATS shows better results compared to its custom counterpart, we have observed inconsistencies in the results of the PyTorch implementation. We have not included comparison of training time between these models because it is not relevant to our study at this stage, but in terms of training speed, LightGBM outperforms the other two models by a large margin, while both the implementations of N-BEATS were comparable in their training speed. The cross-validation techniques are applied to measure the performance of these models on being consistent with accuracy and the findings are discussed below:

LightGBM Results

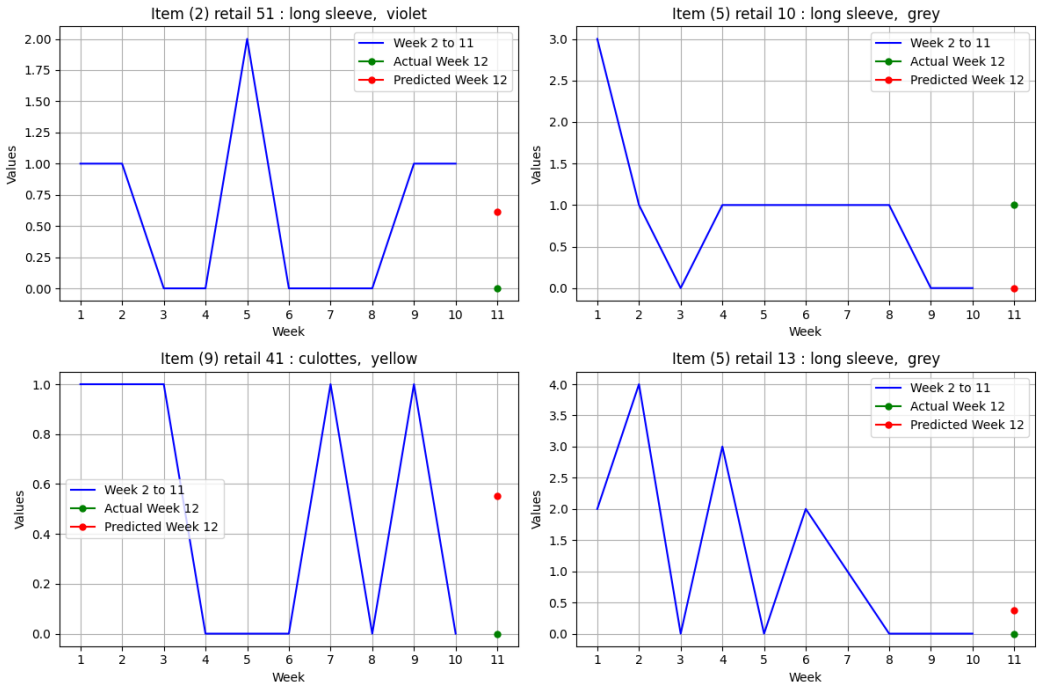

Figure 12: Time-series sales LightGBM prediction plots for four different retail items over a span of 12 weeks.

Figure 12 plots the weekly sales values of the first 4 timeseries. As explained in our methodology, we are predicting the final week and comparing it with the actual sales value of that week. In this figure, the blue line is the training set. The green dot is the actual value of that week and the red dot on the same line as green dot is the predicted value.

Model/ CV	MAE	WAPE
Model	0.71	110.50
Walk Forward CV	0.785	101.83
Group CV	0.816	108.86
Purged Group CV	0.849	112.06
Combinatorial Purged Group CV	0.81	104.88

Table 12: LightGBM Results for Visuelle 2.0 in terms of Cross Validation Techniques & Evaluation Metrics

Above Table 12 presents the results of the LightGBM model applied to the Visuelle 2.0 dataset, including cross-validation outcomes. For the single model run, training and prediction are conducted using the complete dataset, allowing to establish the baseline performance level of the model. Also, to maintain temporal order during cross-validation, the dataset is sorted in ascending order based on product release dates and for grouping, we have grouped it by the release date.

N-BEATS Results : Custom Implementation

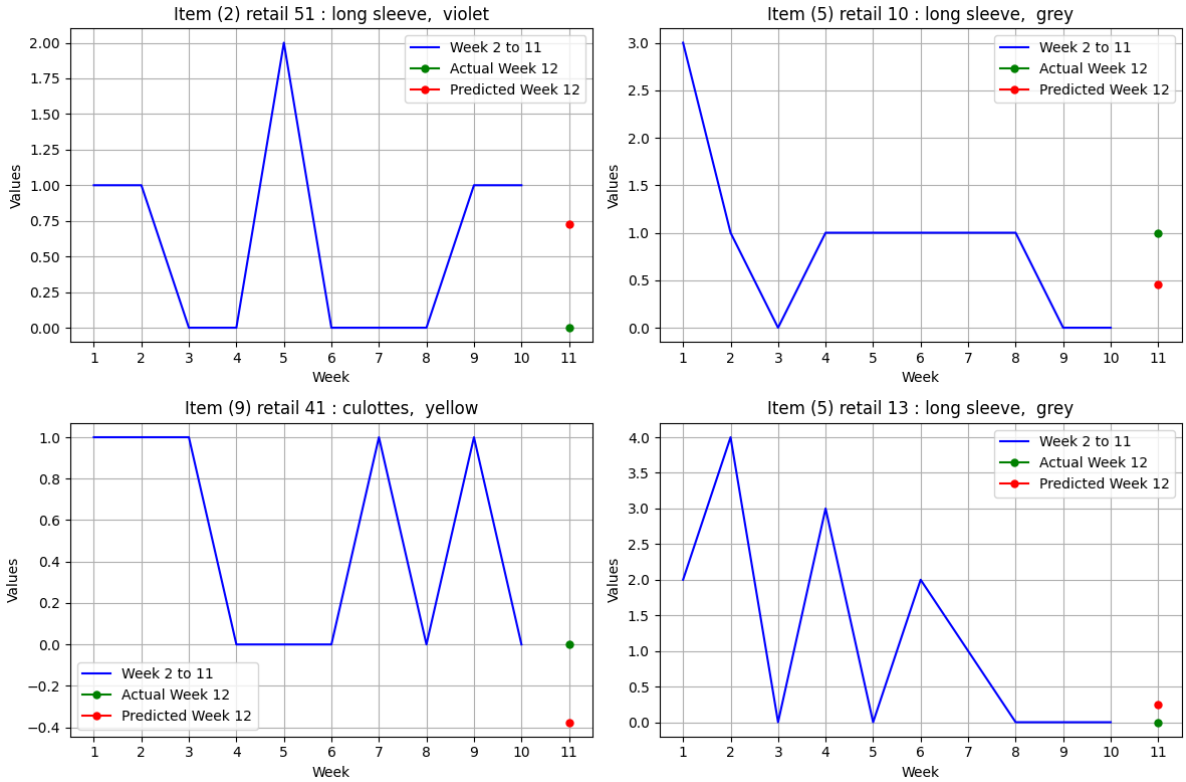

Figure 13: Time-series sales Custom N-BEATS prediction plots for four different retail items over a span of 12 weeks.

Figure 13 plots the weekly sales values of the first 4 timeseries. As explained in our methodology, we are predicting the final week and comparing it with the actual sales value of that week. In this figure, the blue line is the training set. The green dot is the actual value of that week and the red dot on the same line as green dot is the predicted value.

The Table 13 below presents the metric results of the N-BEATS model applied to the Visuelle 2.0 dataset, including cross-validation results. It follows the same single-model run as the above model, so the model's scores serve as a baseline.

Model/ CV	MAE	WAPE
Model	0.73	113.60
Walk Forward CV	0.82	106.63
Group Cross Validation	0.91	122.85
Purged Group CV	0.96	126.25
Combined Purged Group	0.83	108.56

Table 13: N-BEATS Results for Visuelle 2.0 in terms of Cross Validation Techniques & Evaluation Metrics

N-BEATS cross-validation is done similar to LightGBM and to maintain temporal order during cross-validation, the dataset is sorted in ascending order based on the product release dates. where older dates are at the start of dataset and newer releases are at the end of the dataset.

In conclusion, cross-validation explains the model's metric values, and the changes are due to how Cross-Validation works. but it is observed that the PyTorch implementation of N-BEATS is inconsistent with the results, as we did not observe this kind of high variance of metric values in the other two implementations.

N-BEATS Results : PyTorch Implementation

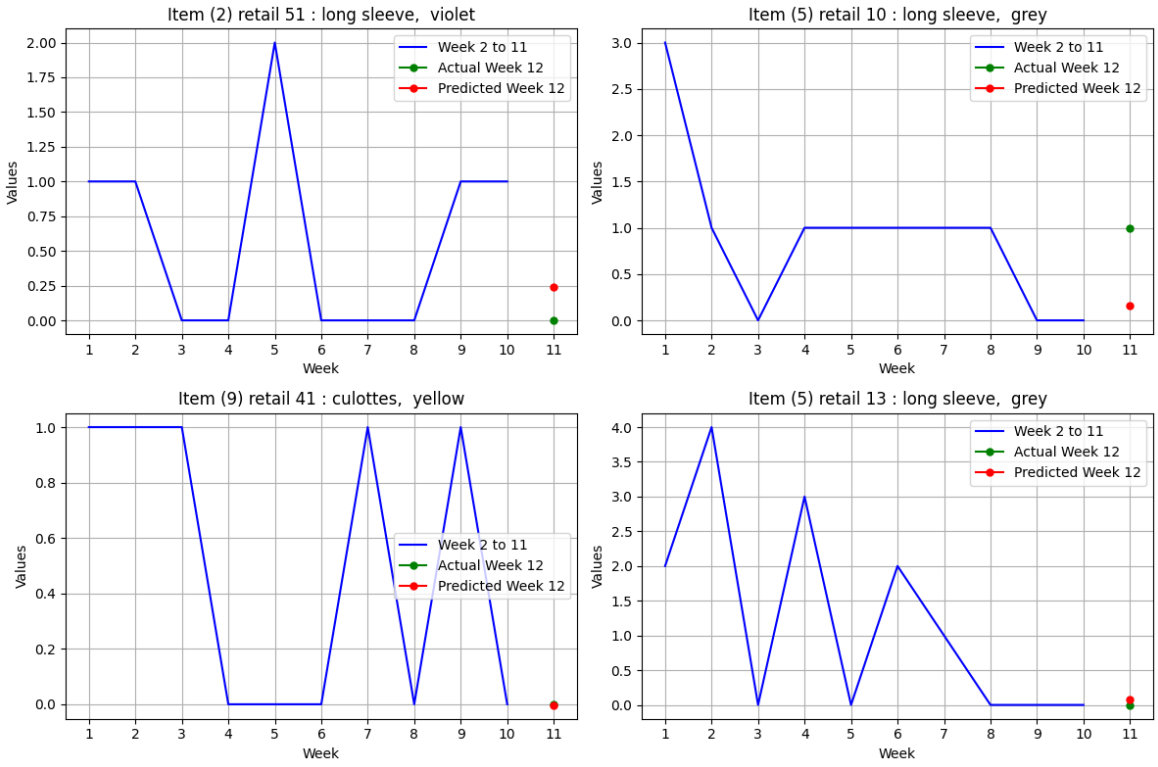

Figure 14: Time-series sales N-BEATS prediction plots for four different retail items over a span of 12 weeks.

Figure 14 plots the weekly sales values of the first 4 timeseries. As explained in our methodology, we are predicting the final week and comparing it with the actual sales value of that week. In this figure, the blue line is the training set. The green dot is the actual value of that week and the red dot on the same line as green dot is the predicted value.

Model/ CV	MAE	WAPE
Model	0.71	110.61
Walk Forward CV	0.91	118.82
Group Cross Validation	1.05	141.18
Purged Group CV	1.04	136.81
Combined Purged Group	0.82	106.40

Table 14: N-BEATS Results for Visuelle 2.0 in terms of Cross Validation Techniques & Evaluation Metrics

This Table 14 presents the metric results of the custom N-BEATS model applied to the Visuelle 2.0 dataset, including cross-validation results. It also follows the same single-model run as the above-mentioned model, so the model's scores serve as a baseline.

This cross-validation is also done similar to LightGBM and to maintain temporal order during cross-validation, the dataset is sorted in ascending order based on the product release dates. where older dates are at the start of dataset and newer releases are at the end of the dataset.

In conclusion, our custom implementation of N-BEATS is more consistent with the results compared to PyTorch implementation and the variance of metric values is similar to the variance of metric values observed in LightGBM model and the variance observed are due to how the cross-validation works.

Discussion

This chapter is dedicated to discussing each dataset, based on the findings from the results chapter. This chapter is where we connect the dots between our results and our initial research objectives, providing an understanding of what our findings mean.

M4 Dataset

In our study of time series forecasting on the M4 dataset, we evaluated the performance of various Machine Learning and Neural Network models, including traditional ensemble methods like LightGBM, deep learning architectures like N-BEATS, and recurrent neural network (RNN) models. Our findings provide valuable insights into the strengths and limitations of these approaches, shedding light on their suitability for different forecasting tasks.

Firstly, our analysis revealed that while LightGBM, a gradient-boosting machine-learning model, exhibited commendable speed in both fitting and prediction, it encountered challenges in analyzing patterns across multiple time series simultaneously. This limitation was evident in cases where the prediction set deviated noticeably from the validation set, particularly in time series with fewer observations. Nevertheless, LightGBM demonstrated competitive performance, as indicated by its respectable Weighted Absolute Percentage Error (WAPE) value.

Conversely, deep learning models such as N-BEATS showcased promising results, albeit with longer training times. N-BEATS displayed a superior ability to capture complex temporal dependencies and nonlinear patterns, making it particularly well-suited for datasets with high dimensionality and volatility. However, its longer training time and computational demands may pose practical challenges in certain contexts.

Furthermore, our exploration of hyperparameter optimization through an Optuna study on the LightGBM model underscored the importance of parameter tuning in enhancing forecasting accuracy. Through iterative trials and parameter adjustments, we were able to identify configurations that yielded lower WAPE values, thus improving the model's predictive performance.

Cross-validation analyses, including Walk Forward, Rolling, Group, and Purged Group Cross Validation, provided additional insights into the comparative performance of LightGBM and N-BEATS across different evaluation metrics and training times. While N-BEATS consistently demonstrated higher accuracy, LightGBM stood out for its efficiency and speed, particularly in scenarios where computational resources are limited, or real-time forecasting is imperative.

In conclusion, our study suggests that for time series datasets like the M4 dataset, which exhibit regular patterns and do not contain complex temporal dependencies, traditional machine learning models such as LightGBM and neural network models like N-BEATS can offer competitive performance, provided their hyperparameters are carefully optimized. However, the longer training times associated with deep learning models may render them less practical

in scenarios where computational efficiency is paramount. Therefore, selecting the most appropriate forecasting model entails a careful consideration of trade-offs between accuracy, computational resources, and real-world applicability.

M5 Dataset

Our investigation into time series forecasting on the M5 dataset delved into the performance of machine learning models, specifically LightGBM, and attempted to explore N-BEATS, albeit constrained by computational resources. Through hyperparameter optimization using Optuna and cross-validation analyses, we gained valuable insights into the comparative efficacy of the LightGBM model in handling the complexities of the M5 dataset.

Optuna results highlighted the significance of parameters such as n_estimators and max_depth in optimizing the LightGBM model for forecasting accuracy. Among the ten trials conducted, the fifth trial emerged as the best-performing configuration, which was then utilized for further evaluation.

In terms of cross-validation, our evaluation revealed intriguing patterns between LightGBM and N-BEATS. In the Cross Validation methods, N-BEATS consistently exhibited lower Root Mean Squared Error (RMSE) scores compared to LightGBM, indicating better accuracy in terms of absolute errors. However, LightGBM often outperformed N-BEATS in terms of symmetric Mean Absolute Percentage Error (SMAPE) scores, particularly in later runs. This suggests that while N-BEATS excelled in minimizing absolute errors, LightGBM demonstrated better performance when considering proportional errors. Additionally, LightGBM showcased significantly faster training and forecasting times compared to N-BEATS, which is crucial for real-time applications or large-scale forecasting tasks.

In conclusion, our analysis on the M5 dataset underscores the nuanced trade-offs between forecasting accuracy, computational resources, and model complexity. While N-BEATS displayed better accuracy in absolute error metrics, LightGBM showcased competitive performance, particularly in proportional error metrics, and offered the added advantage of faster training and forecasting times. As a result, choosing the best forecasting model relies on the particular needs of the application while weighing factors like scalability, accuracy, and efficiency.

Visuelle 2.0 Dataset

Both the LightGBM and N-BEATS models achieved comparative results in terms of Mean Absolute Error (MAE) and Weighted Absolute Percentage Error (WAPE). LightGBM slightly outperformed both N-BEATS implementations, demonstrating better accuracy and forecasting performance on the Visuelle 2.0 dataset. Moreover, the LightGBM model was significantly faster than both N-BEATS implementations.

Across various cross-validation techniques, LightGBM again consistently had better accuracy, and speed in its forecasting. While our custom N-BEATS implementation was consistent and slightly faster than the PyTorch implementation, it was considerably slower than LightGBM.

In our previous experiments, we observed that N-BEATS performs better when the data is high-dimensional, i.e., when there are more features to train on. However, with the Visuelle 2.0 dataset, which consists of a large number of features, we encountered some challenges. When we attempted to include all the values of google trends for textual tags, the WAPE was reduced, but having 156 columns (52*3) along with other features significantly slowed down the LightGBM model. Although N-BEATS was faster in this scenario, it failed to deliver good results and often struggled to extract the patterns in the dataset.

In conclusion, both the LightGBM and N-BEATS models are effective in forecasting the Visuelle 2.0 dataset, with LightGBM demonstrating slightly better performance in terms of accuracy and forecasting efficiency. Additionally, the choice of cross-validation technique can significantly impact the evaluation of model performance, with certain methods yielding better results than others.

Conclusion

In this final chapter, we summarize our study's key findings and answer our research questions.

Our study evaluated various Machine Learning and Neural Network models on three distinct datasets: M4, M5, and Visuelle 2.0. We found that both traditional ensemble methods like LightGBM and deep learning architectures like N-BEATS demonstrated competitive performance across all datasets, with slight variations in accuracy and computational efficiency depending on the dataset characteristics.

LightGBM consistently showed commendable speed and accuracy in fitting and prediction, while N-BEATS exhibited better capability in capturing complex temporal dependencies and nonlinear patterns, especially with short observation window in visuelle 2.0. However, the longer training times and computational demands associated with N-BEATS may limit its practical applicability in certain scenarios.

Implementing these cross-validation strategies was significantly challenging, especially given our focus on time series data and the unique characteristics of our datasets. While the implementation of walk-forward and rolling cross-validation was relatively straightforward, group cross-validation necessitated a deeper understanding of grouping and the optimal ways to segment the datasets. Once we grasped group cross-validation, implementing purged group cross-validation became easier. The most challenging aspect was comprehending combinatorial purged group cross-validation, which was incompatible with the Darts time series datatype due to its inability to join non-continuous time series. However, by opting for PyTorch instead of Darts, we were able to implement this cross-validation strategy, thanks to the additional flexibility it offered.

Our research does not conclusively proclaim that Neural Network is universally better than traditional machine learning or vice versa. Instead, it highlights the importance of selecting the most appropriate forecasting model based on the specific characteristics of the dataset, the forecasting task, and the available computational resources. For datasets with regular patterns and relatively simple temporal dependencies, traditional machine learning models like LightGBM may offer competitive performance with faster training times and lower computational demands. On the other hand, Neural Network models like N-BEATS excel in capturing complex temporal dynamics but may require longer training times and more computational resources.

The results of our study highlight how crucial it is to weigh the trade-offs between model complexity, computing efficiency, dataset size and accuracy when choosing a forecasting model for practical uses. Furthermore, the selection and evaluation of models must take great thought because the cross-validation approach used might have a considerable impact on the model's performance assessment.

Future Suggestions

One avenue for future research involves exploring the application of transformer-based models (Vaswani et al., 2023), such as the Transformer architecture, in time series forecasting tasks, particularly on datasets like Visuelle 2.0. Transformer models may provide promising results in capturing temporal dependencies and patterns in time series data, as they have proven very successful in a variety of sequential data processing applications.

Furthermore, more research into hybrid systems (Smyl, 2020) that combine the benefits of deep learning methodology with classic machine learning techniques may provide insightful information about improving the efficiency and accuracy of forecasting. Enhancing forecasting performance on a variety of datasets and forecasting tasks may be possible by experimenting with ensemble models that include classical and deep learning components.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mane, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viegas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan and Zheng, Xiaoqiang (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. http://arxiv.org/abs/1603.04467

Akiba, Takuya, Sano, Shotaro, Yanase, Toshihiko, Ohta, Takeru and Koyama, Masanori (2019). Optuna: A Next-generation Hyperparameter Optimization Framework.

Berend (2022). The Combinatorial Purged Cross-Validation method. Medium. https://pub.towardsai.net/the-combinatorial-purged-cross-validation-method-363eb378a9c5

Bergmeir, Christoph, Hyndman, Rob J. and Benítez, José M. (2016). Bagging exponential smoothing methods using STL decomposition and Box-Cox transformation. International Journal of Forecasting, 32(2), p. 303–312, doi:10.1016/J.IJFORECAST.2015.07.002

Bergmeir, Christoph, Hyndman, Rob J. and Koo, Bonsoo (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. Computational Statistics & Data Analysis, 120, p. 70–83, doi:10.1016/J.CSDA.2017.11.003.

Box, George E. P., Jenkins, Gwilym M. and Reinsel, Gregory C. (2013). Time series analysis: Forecasting and control: Fourth edition. Time Series Analysis: Forecasting and Control: Fourth Edition, p. 1–746, doi:10.1002/9781118619193.

Brockwell, Peter J. and Davis, Richard A. (2016). Introduction to Time Series and Forecasting, doi:10.1007/978-3-319-29854-2.

Cerqueira, Vitor, Torgo, Luis and Mozetič, Igor (2019). Evaluating time series forecasting models An empirical study on performance estimation methods.

Cerqueira, Vitor, Torgo, Luis and Mozetič, Igor (2020). Evaluating time series forecasting models: an empirical study on performance estimation methods. Machine Learning, 109(11), p. 1997–2028, doi:10.1007/S10994-020-05910-7/FIGURES/33.

Chen, Tianqi and Guestrin, Carlos (2016). XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 785–794.

Chiew, Ernest and Choong, Shin Siang (2022). A solution for M5 Forecasting-Uncertainty: Hybrid gradient boosting and autoregressive recurrent neural network for quantile estimation. International Journal of Forecasting, 38, p. 1442–1447, doi:10.1016/j.ijforecast.2022.01.009.

Chollet, François and others (2018). Keras: The Python Deep Learning library. Astrophysics Source Code Library, p. ascl:1806.022.

Darts Models, Unit8co (2020). Forecasting Models — darts documentation. https://unit8co.github.io/darts/generated_api/darts.models.forecasting.html

DelSole, Michael (2018). What is One Hot Encoding and How to Do It. Medium, 2018-04-24. https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179

Goodfellow, Ian, Bengio, Yoshua and Courville, Aaron (2016). Deep Learning. MIT Press. https://www.deeplearningbook.org/

Gort, Berend and Yang, Bruce (2022). The Combinatorial Purged Cross-Validation method. TowardsAI.

Harris, Charles R., Millman, K. Jarrod, van der Walt, Stéfan J., Gommers, Ralf, Virtanen, Pauli, Cournapeau, David, Wieser, Eric, Taylor, Julian, Berg, Sebastian, Smith, Nathaniel J., Kern, Robert, Picus, Matti, Hoyer, Stephan, van Kerkwijk, Marten H., Brett, Matthew, Haldane, Allan, del Río, Jaime Fernández, Wiebe, Mark, Peterson, Pearu, Gérard-Marchant, Pierre, Sheppard, Kevin, Reddy, Tyler, Weckesser, Warren, Abbasi, Hameer, Gohlke, Christoph and Oliphant, Travis E. (2020). Array programming with NumPy. Nature, 585(7825), p. 357–362, doi:10.1038/s41586-020-2649-2.

Herzen, Julien (2021). Time Series Forecasting Using Past and Future External Data with Darts. Unit8 - Big Data & AI, 2021-08-17. https://medium.com/unit8-machine-learning-publication/time-series-forecasting-using-past-and-future-external-data-with-darts-1f0539585993

Herzen, Julien, Lässig, Francesco, Piazzetta, Samuele Giuliano, Neuer, Thomas, Tafti, Léo, Raille, Guillaume, Van Pottelbergh, Tomas, Pasieka, Marek, Skrodzki, Andrzej, Huguenin, Nicolas, Ko´sciszko´scisz, Jan, Bader, Dennis, Gusset, Frédérick, Benheddi, Mounir, Williamson, Camila, Kosinski, Michal, Petrik, Matej and Grosch, Gaël (2022). Darts: User-Friendly Modern Machine Learning for Time Series Maxime Dumonal †. Journal of Machine Learning Research, 23, p. 1–6.

Hwang Tim (2018). Computational Power and the Social Impact of Artificial Intelligence. arxiv, doi:10.48550/arXiv.1803.08971

Jose, Divya (2021). Reshaping Data Frames using Pandas Melt and Pivot. Predmatic, 2021-04-21. https://medium.com/predmatic/reshaping-data-frames-using-pandas-melt-and-pivot-8fe65a30a252

Ke, Guolin, Meng, Qi, Finley, Thomas, Wang, Taifeng, Chen, Wei, Ma, Weidong, Ye, Qiwei and Liu, Tie-Yan (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems, 30.

Lainder, A. David and Wolfinger, Russell D. (2022). Forecasting with gradient boosted trees: augmentation, tuning, and cross-validation strategies: Winning solution to the M5 Uncertainty competition. International Journal of Forecasting, 38(4), p. 1426–1433, doi:10.1016/J.IJFORECAST.2021.12.003.

Lipton, Zachary C., Berkowitz, John and Elkan, Charles (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning.

Ma, Shaohui and Fildes, Robert (2022). The performance of the global bottom-up approach in the M5 accuracy competition: A robustness check. International Journal of Forecasting, 38(4), p. 1492–1499, doi:10.1016/J.IJFORECAST.2021.09.002.

Maćkiewicz, Andrzej and Ratajczak, Waldemar (1993). PRINCIPAL COMPONENTS ANALYSIS (PCA)*. Computers & Geosciences, 19(3), p. 303–303.

Makridakis, Spyros, Spiliotis, Evangelos and Assimakopoulos, Vassilios (2018). The M4 Competition: Results, findings, conclusion and way forward. International Journal of Forecasting, 34(4), p. 802–808, doi:10.1016/J.IJFORECAST.2018.06.001.

Makridakis, Spyros, Spiliotis, Evangelos and Assimakopoulos, Vassilios (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. International Journal of Forecasting, 36(1), p. 54–74, doi:10.1016/J.IJFORECAST.2019.04.014.

Makridakis, Spyros, Spiliotis, Evangelos and Assimakopoulos, Vassilios (2022a). The M5 competition: Background, organization, and implementation. International Journal of Forecasting, 38(4), p. 1325–1336, doi:10.1016/J.IJFORECAST.2021.07.007.

Makridakis, Spyros, Spiliotis, Evangelos and Assimakopoulos, Vassilios (2022b). M5 accuracy competition: Results, findings, and conclusions. International Journal of Forecasting, 38, p. 1346–1364, doi:10.1016/j.ijforecast.2021.11.013.

Makridakis, Spyros, Spiliotis, Evangelos, Assimakopoulos, Vassilios, Chen, Zhi, Gaba, Anil, Tsetlin, Ilia and Winkler, Robert L. (2022). The M5 uncertainty competition: Results, findings and conclusions. International Journal of Forecasting, 38(4), p. 1365–1385, doi:10.1016/J.IJFORECAST.2021.10.009.

MARKETNEUTRAL (2020). Purged Time Series CV, XGBoost, Optuna. Kaggle.com

Oreshkin, Boris N., Carpov, Dmitri, Chapados, Nicolas and Bengio, Yoshua (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting.

Pahwa, Harjot (2024). Harnessing the Power of Hash Encoding for Categorical Data in Data Science. Medium. https://ai.plainenglish.io/harnessing-the-power-of-hash-encoding-for-categorical-data-in-data-science-d5fd3cff6673

Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, Desmaison, Alban, Köpf, Andreas, Yang, Edward, DeVito, Zach, Raison, Martin, Tejani, Alykhan, Chilamkurthy, Sasank, Steiner, Benoit, Fang, Lu, Bai, Junjie and Chintala, Soumith (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. http://arxiv.org/abs/1912.01703

Pedregosa, Fabian, Varoquaux, Gael, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, Vanderplas, Jake, Passos, Alexandre and Cournapeau, David (n.d.). Scikit-learn: Machine Learning in Python. MACHINE LEARNING IN PYTHON.

Peralta, Daniel, Del Río, Sara, Ramírez-Gallego, Sergio, Triguero, Isaac, Benitez, Jose M. and Herrera, Francisco (2015). Evolutionary Feature Selection for Big Data Classification: A MapReduce Approach. Mathematical Problems in Engineering, 2015, doi:10.1155/2015/246139.

Prokhorenkova, Liudmila, Gusev, Gleb, Vorobev, Aleksandr, Dorogush, Anna Veronika and Gulin, Andrey (2019). CatBoost: unbiased boosting with categorical features. http://arxiv.org/abs/1706.09516

Radečić, Dario (2021). Stop Using CSVs for Storage — Pickle is an 80 Times Faster Alternative. Medium. https://towardsdatascience.com/stop-using-csvs-for-storage-pickle-is-an-80-times-faster-alternative-832041bbc199

Sagiroglu, Seref and Sinanc, Duygu (2013). Big data: A review. Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013, p. 42–47, doi:10.1109/CTS.2013.6567202.

Salinas, David, Flunkert, Valentin and Gasthaus, Jan (2019). DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. http://arxiv.org/abs/1704.04110

Sbrana, Attilio, Debiaso Rossi, Andre Luis and Coelho Naldi, Murilo (2020). N-BEATS-RNN: Deep learning for time series forecasting. Proceedings - 19th IEEE International Conference on Machine Learning and Applications, ICMLA 2020, p. 765–768, doi:10.1109/ICMLA51294.2020.00125.

Simonyan, Karen and Zisserman, Andrew (2015). VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION.

Skenderi, Geri, Joppi, Christian, Denitto, Matteo, Scarpa, Berniero and Cristani, Marco (2022). The multi-modal universe of fast-fashion: the Visuelle 2.0 benchmark. arxiv, doi:10.48550/arXiv.2204.06972.

Smit J.J.H. (2020). Found the Holy Grail: GroupTimeSeriesSplit. Kaggle.com.

Smyl, Slawek (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. International Journal of Forecasting, 36(1), p. 75–85, doi:10.1016/J.IJFORECAST.2019.03.017.

Spiliotis, Evangelos, Kouloumos, Andreas, Assimakopoulos, Vassilios and Makridakis, Spyros (2020). Are forecasting competitions data representative of the reality? International Journal of Forecasting, 36(1), p. 37–53, doi:10.1016/j.ijforecast.2018.12.007.

Tashman, Leonard J. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. International Journal of Forecasting, 16(4), p. 437–450, doi:10.1016/S0169-2070(00)00065-0.

Theodorou, Evangelos, Wang, Shengjie, Kang, Yanfei, Spiliotis, Evangelos, Makridakis, Spyros and Assimakopoulos, Vassilios (2021). Exploring the representativeness of the M5 competition data.

Unit8 SA, Darts (n.d.). Static Covariates — darts documentation. https://unit8co.github.io/darts/examples/15-static-covariates.html

Vaswani, Ashish, Brain, Google, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Łukasz and Polosukhin, Illia (2023). Attention Is All You Need.

Wellens, Arnoud P., Udenio, Maxi and Boute, Robert N. (2022). Transfer learning for hierarchical forecasting: Reducing computational efforts of M5 winning methods. International Journal of Forecasting, 38(4), p. 1482–1491, doi:10.1016/J.IJFORECAST.2021.09.011.

Zhang, G. Peter and Kline, Douglas M. (2007). Quarterly Time-Series Forecasting With Neural Networks. IEEE TRANSACTIONS ON NEURAL NETWORKS, 18(6), doi:10.1109/TNN.2007.896859.

Chawla, Gaurav (2023) GroupTimeSeriesSplit, GitHub. https://github.com/getgaurav2/scikit-learn/blob/d4a3af5cc9da3a76f0266932644b884c99724c57/sklearn/model_selection/_split.py