Universiteit Leiden

Master Computer Science

Transfer Learning in Time-Series Forecasting

Name:	Kenneth Fargose
Student ID:	3281353
Date:	10/08/2023
Specialisation:	Data Science
1st supervisor:	Dr. E.M. Bakker
2nd supervisor:	Prof. Dr. M.S.K.Lew

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

Acknowledgements

I would like to express my heartfelt gratitude to all the individuals who have contributed to the completion of this thesis.

First and foremost, I extend my sincere appreciation to my supervisor, Erwin Bakker, for his support, guidance, and valuable insights throughout this research journey. His expertise and encouragement have been instrumental in shaping this thesis and enhancing my understanding of the subject matter.

I am also deeply grateful to the Wolf Vos, my supervisor from Heineken for providing a conducive academic environment and resources that facilitated my research endeavors. I would like to thank Heineken for giving me the opportunity to work on this thesis

I am indebted to my friends and colleagues for their camaraderie and encouragement during challenging times. Their support and constructive feedback have been invaluable in refining my ideas and improving the quality of this thesis.

I extend my heartfelt thanks to my mom Vandana Fargose, dad Stephen Fargose, sisters Sara Fargose and Joan Fargose and girlfriend Remiya Dodti for their unwavering love, understanding, and encouragement. Their belief in me and constant motivation have been the driving force behind my academic pursuits.

The completion of this thesis would not have been possible without the collective support and contributions from all those mentioned above. I am truly grateful for their invaluable assistance and encouragement, and I humbly dedicate this work to them.

Abstract

Transfer Learning has emerged as a promising technique in machine learning, allowing pre-trained models to be applied to new datasets to improve their general capabilities. In recent years, there has been a growing interest in applying transfer learning to time-series data, especially in Time-Series forecasting (TSF) using deep learning methods such as Convolutional Neural Networks (CNNs), transformers and Long Short-Term Memory (LSTM) networks. However, N-BEATS (Neural basis expansion analysis for interpretable TSF) has not yet been tested for transfer learning, despite its superiority over existing TSF models.

This paper performs an in-depth analysis of transfer learning in TSF by utilizing N-BEATS and RNN deep learning models as well two important classical TSF models Prophet and AutoArima. In this thesis, various TSF models, including Prophet, AutoArima, and transfer learning using RNN and N-BEATS, are extensively investigated. Transfer learning, especially the model-based approach, emerges as a promising technique for enhancing forecasting accuracy. Prophet showcases robust performance, while N-BEATS holds potential with hyperparameter tuning. The findings strongly recommend adopting model-based transfer learning, highlighting its potential across various scenarios, including zero-shot learning. The research also underscores future directions, like multivariate forecasting . Transfer learning emerges as a dynamic tool for precise and adaptable time-series forecasts, holding significant implications for practical implementations, including the feasibility of zero-shot learning for N-BEATS and RNN

1 Introduction

Time-series analysis and forecasting (TSF) has become increasingly important in various industrial applications, ranging from financial market predictions to energy consumption estimation. This is due to their ability to support decision making and enhance efficiency. With the growing amount of data being generated, there is a pressing need for developing highly accurate and robust models for TSF.
Forecasting future trends and patterns in time-series data can provide valuable insights for businesses to make strategic decisions, optimize their resources. For example, in the field of energy management, accurate forecasting of energy demand can help to optimize energy usage and reduce costs. In the field of finance, accurate forecasting of stock prices can help investors make informed decisions about buying and selling stocks.

However, accurate forecasting of time-series data is often challenging due to the complexity and uncertainty of the underlying processes. Traditional forecasting methods often rely on statistical models that assume stationary and linear relationships between the variables. Time-series data is often non-stationary, meaning that the statistical properties of the data change over time. Moreover, time-series data is often subject to various external factors that can cause abrupt changes in the data, such as natural disasters, economic events, and social trends. Furthermore, the data generated for time-series is often very erratic. It may contain a lot of noise. Common time-series data type is health data generated by a fitness device, such devices have a tendency to capture data which is noisy and therefore is very ambiguous. For example a fitness watch can capture data when the user is not moving and classify it as an activity.
One of the key challenges in TSF is limited and noisy data. In many cases, historical data is not sufficient to capture the underlying patterns and trends in the time-series, leading to inaccurate and unreliable forecasts. Moreover, time-series data is often subject to various external factors such as seasonality, trends, and abrupt changes, making it even more difficult to model and predict. Time-series data can be classified depending on the dimensionality of the time-series samples into two types, univariate and multivaraite data . Over the years, many models have been proposed and employed to tackle both univariate and multivaraite time-series problems with significant improvements in prediction accuracy like the classical ARIMA , AutoArima [8] and more recently Facebook's Prophet [28]. Classical models are usually only used to forecast univariate time-series data. Prophet which is state-of-the-art for classical models can handle small amount of data as well as a huge dataset without using a lot of computation time.

Whereas recently deep learning techniques such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) [12] networks, and Transformers have demonstrated their capability to capture complex temporal patterns and dependencies present in time-series data.
Deep learning models can handle multivariate data this is at an expense of processing time and requiring a lot of data to learn the complexity of the problem. Even the recent state-of-the-art deep learning model N-BEATS [20] requires a lot of data to get good predictions. This can be an issue for applications industry as there can be limited data available due to a variety of reasons.This can be addressed by a technique commonly used in machine learning known as Transfer Learning.
Transfer learning addresses these challenges by pre-training a flexible model on a large dataset, which can then be applied to a typically smaller dataset with minimal or no additional training [33] Transfer learning is a machine learning technique that leverages knowledge learned from one task to improve performance on another related task.

As such it an effective approach that leverages pre-trained models to solve emerging problems with limited data or limited computational resources. It allows fine-tuning of a model on a target task, effectively reducing the deployment time and computational costs. This can be particularly beneficial in TSF, as obtaining large amounts of labeled data may not always be feasible It can help to overcome the limitations of limited and noisy data by leveraging knowledge learned from other related time-series datasets.

Transfer learning in TSF has already shown promising results in various applications. For example, in the field of energy management, transfer learning has been used to improve the accuracy of load forecasting for different buildings by leveraging knowledge learned from other similar buildings [6]. Similarly, in the field of finance, transfer learning has been used to improve the accuracy of stock price forecasting by leveraging knowledge learned from other related stocks [22].

Despite the potential benefits of transfer learning in TSF, there remain several challenges that need to be addressed. One of the main challenges is the selection of the appropriate source time-series for transfer learning. The source time-series should be similar enough to the target time-series to be useful, but not too similar to avoid overfitting. Another challenge is the design of appropriate transfer learning algorithms that can effectively leverage the knowledge from the source time-series to improve the forecasting performance on the target time-series. Another essential factor to consider when implementing transfer learning in TSF is having the proper computational infrastructure, such as GPUs, when working with deep learning models. Although, utilizing pre-trained models can reduce the need for such computational resources, making forecasting tasks more manageable for organizations with limited computational resources .

In this thesis, the aim is to explore the potential of different methods of transfer learning in time-series analysis and forecasting by investigating various deep learning models such N-BEATS and RNN-LSTM and their capabilities to adapt to the given data. The challenges and opportunities associated with applying transfer learning to time-series data will be discussed, and a comparison of different models in terms of their adaptability will be made. Ultimately,the research aims to provide a comprehensive understanding of the benefits and limitations of transfer learning in the context of TSF and contribute to the advancement of this field in industrial applications.

The main goal of this thesis is to investigate the effectiveness of transfer learning in TSF using the state-of-the-art models and investigate if that can improve the forecasting performance on specific target time-series. In particular, the following research questions will be addressed:

- **What are the best time-series forecasting models?**
- **Is transfer learning possible for time-series data?**
- **What technique of transfer learning should be used for further ease of time-series forecasting ?**
- **Does transfer learning give a better performance on time-series forecasting than classical approaches?**

This thesis addresses a series of research questions centered around TSF and transfer learning. Leveraging real-world time-series datasets, specifically the M5 dataset for univariate time-series, the research embarks on a comprehensive set of experiments. The core objective is twofold: firstly, to pinpoint the optimal forecasting model among the contenders - Prophet, AutoARIMA, N-BEATS, and RNN-LSTM - within the confines of the M5 dataset. A comparison of their performances utilizing established evaluation metrics such as MAPE, SMAPE, and R2.

Moreover, the investigation encompasses an exploration of two distinct transfer learning methodologies tailored for time-series data: zero-shot learning and model-based learning using deep learning models, specifically RNN and N-BEATS, as they serve as base for investigating the efficiency of these transfer learning techniques. The goal of the research is to get the best forecasting model for univariate time-series but also to find transfer learning methodology tailored for time-series data.

To address the research questions effectively, the research designs to implement two transfer learning techniques on deep learning models. It delves into the novel concept of zero-shot learning, evaluating its applicability on both N-BEATS and RNN models. Concurrently, it also uses a distinct approach to model-based transfer learning. These techniques are systematically evaluated against classical models, aiming to check if they outperform classical methodologies in TSF. Through this investigation, the research endeavors to pave the way for informed decisions and promising avenues in the fields of TSF and transfer learning.

The rest of the paper is organized as follows: In Section 2, related research on TSF and transfer learning in TSF is discussed. Section 3, provides background information on the different classical and deep learning algorithms used in the experiments, as well as the definitions of the measures used to evaluate the performance of the algorithms. Section 4 describes the methodology of the experiments, including the setup and training process, it also provides information on the M5 dataset and analysis, and preprocessing. Section 5 describes the different experiments using Prophet, AutoArima, RNN and N-BEATS, RNN and N-BEATS in Zero-shot transfer learning and m o01del-based transfer leaning. In Section 6, the results will be discussed, answering our research questions. In Section 7, the conclusions are given Finally in **??**, future works are discussed. Section 8 lists the contribution done.

2 Related research

In recent years, deep learning models have been widely used for TSF applications. However, the use of these models is challenging when dealing with irregularly sampled time-series data and hierarchical structures within the data. This has led to the development of novel deep learning architectures that can handle these challenges.

One such architecture, introduced by Lim et al.[17] , makes use of dilated convolutions in a convolutional neural network (CNN) to reduce computational complexity. The paper uses LSTM to overcome the exploding and vanishing gradients problem together with an Attention mechanism model to aggregate temporal features using dynamically generated weights, enabling the network to focus on significant time steps in the past, even if they are very far back in the look-back window.

In [29], Torres et al. investigated a diverse set of deep neural network architectures for TSF. Their research not only covered established architectures like Bidirectional RNNs (BRNNs) and Deep RNNs (DRNNs) but also explored Convolutional Neural Networks (CNNs) for TSF. Of particular significance was the examination of RNN, showcasing its capacity to capture temporal dependencies with its context layer. This in-depth analysis offers valuable insights into selecting suitable deep learning models for various TSF applications. Notably, they emphasized the effectiveness of Recurrent Neural Networks (RNNs) in this domain, with LSTM and GRU variants standing out due to their ability to address gradient-related issues and maintain long-term memory efficiently.

N-BEATS [20] by N. Oreshkin et al. is a deep learning model designed for TSF , renowned for its state-of-the-art nature and unique features. By using basis functions to capture meaningful patterns in time-series data, such as trend and seasonality, N-BEATS enables interpretable insights into the driving factors behind forecasts. The model's adaptability and flexibility, achieved through easy customization of basis functions, make it effective for various time-series scenarios, including those with long-term dependencies and irregular patterns. In comparison to RNN-based models, N-BEATS offers advantages such as a fully feedforward architecture, interpretability, scalability, and superior forecasting accuracy. N-BEATS outperforms classical methods like ARIMA and ETS in univariate time-series forecasting tasks. N-BEATS demonstrates superior performance on various benchmark datasets across different forecasting horizons. The model's interpretable architecture and flexibility make it a strong contender for accurate and efficient time-series forecasting.. Cawood et al. [4] also confirm N-BEATS to be a state-of-the-art model by using it as a baseline to compare their study on. The primary objective of their study was to compare ensemble methods with two forecasting methods, ES-RNN and N-BEATS. The goal was to assess whether ensembling could improve the accuracy of contemporary hybrid models, and to identify a leading ensembling technique that could serve as the state-of-the-art method for future research endeavors. In their study [32], Wang et al. tackle the challenge of short-term macroeconomic forecasting by introducing EcoForecast1, an innovative and interpretable data-driven approach. Built upon the foundation of N-BEATS, a state-of-the-art model, EcoForecast1 offers a unified framework for macroeconomic prediction, demonstrating superior performance and interpretability compared to traditional statistics-based and deep learning-based methods. EcoForecast1 stands out for its transparent and understandable forecasting process, making it a groundbreaking solution in the field. The integration of N-BEATS and interpretability opens up avenues for accurate and meaningful macroeconomic predictions, benefiting analysts, decision-makers, and researchers alike While transfer learning has seen significant success in computer vision, recent advancements have spurred research on applying transfer learning to time series data. This approach holds promise for en-

hancing learning in diverse time series domains, including those based on sensor values. To explore the state of the field, the authors of this paper Weber et al. [33] conduct a systematic mapping study, thoroughly analyzing 223 relevant publications on transfer learning with time series data. The findings of the study suggest that the most prevalent method for transfer learning with time series data is the model-based approach, with Convolutional Neural Networks (CNN) and Recurrent Neural Networks with Long Short-Term Memory (RNN-LSTM) being the preferred models.

Rui Ye et al. [35] investigate deep neural networks, including RNNs and CNNs, for TSF. The paper specifically explores the use of the LSTM model and the GRU model, as well as bidirectional RNNs (BRNNs) and deep RNNs (DRNNs).

They find that DTr-CNN a deep transfer learning method using a CNN architecture is effective at leveraging useful knowledge to improve time-series prediction, particularly when labeled data is limited. The research makes use of the model-based approach for transfer learning from one dataset to another. He et al [10] propose a novel approach to enhance the accuracy of TSF using deep learning models in the context of short time-series. They address the issue of poor performance of deep learning models when applied to short time-series data. The proposed solution involves transfer learning, which utilizes knowledge learned from two source datasets for the prediction task in the target dataset. The authors evaluate the effectiveness of their approach on financial time-series extracted from stock markets. The proposed training strategy involves using two source datasets and introducing additional hidden layers to the network architecture. The authors experiment with financial time-series extracted from stock markets to evaluate the effectiveness of their approach. The results show that transfer learning has a positive impact on financial TSF. Using more source datasets for transfer learning outperforms using a single source dataset. The proposed strategy yields good results in the majority of cases, indicating its efficacy.

They also propose a similarity-based approach using Dynamic Time Warping (DTW) to select source and target datasets for training the deep learning models. They find that transfer learning with similar source datasets, i.e., those with smaller DTW distance, from the same industry performs better than selecting source datasets from different industries. This suggests that knowledge from related financial data can be effectively transferred to improve forecasting accuracy. Their approach of transfer learning positively impacts financial TSF. Using more source datasets for transfer learning outperforms using a single source dataset. This research also makes use of the model based approach for transfer learning. Although the study is conducted with only two source datasets, the authors suggest that the strategy in their paper [10] can be extended to accommodate more datasets by incorporating additional hidden layers in the neural network architecture.

The papers were helpful to get significant insights. Notably, two key deep learning models, N-BEATS and RNN-LSTM, emerged as central models due to their established efficacy in time-series forecasting. Moreover, the decision to incorporate model-based transfer learning stemmed from its widespread adoption as one of the most utilized and promising methods in the field. This approach was guided by the extensive application of model-based transfer learning, further reinforcing the foundation for the experimentation process. These choices, grounded in the well-established success of N-BEATS, RNN-LSTM, and model-based transfer learning, for the research exploration into advanced time-series forecasting techniques

3 Fundamentals

3.1 Time-series

A time-series is a collection of well-defined data item observations obtained through repeated measurements over time, often exhibiting trends or seasonality. Time-series data encompasses various domains such as sales, weather, and finance, among others. The use of this data for predicting future events has been a subject of interest in recent years. Forecasting involves analyzing past observations to identify patterns and trends, enabling accurate predictions about future points in the series. Time-series analysis [9] and forecasting has significant implications for decision-making, risk management, and planning. It can be applied to predict heatwaves based on temperature data, anticipate electricity consumption, and even forecast events like volcanic eruptions or landslides using time-series data. This data can classified into two types, univariate and multivariate. In univariate time-series, there is only one variable (e.g., temperature) that changes over time, and each data point represents a single value at a specific time.

On the other hand, in multivariate time-series, there are multiple variables (e.g., x, y, z accelerations) that change simultaneously over time. Each data point in a multivariate time-series consists of values for all the variables at a particular time.

3.2 Models in Time-Series Forecasting

In TSF, different classical models are commonly employed. AutoRegressive [36] models utilize past values of the series to predict future values, while Moving Average [31] models adjust forecasts based on the error term from previous predictions. AutoRegressive Integrated Moving Average (ARIMA) [2] models combine AR and MA models with differentiation to handle non-stationary time-series. The **I** component involves differencing the time-series to make it stationary. Stationarity implies that the statistical properties of the series, such as mean and variance, do not change over time. Differencing helps remove trends and seasonality, making the data suitable for modeling. Exponential smoothing (ES) [7] models make predictions by considering weighted averages of past observations. Seasonal Autoregressive Integrated Moving Average (SARIMA) [2] models incorporate seasonality into the forecasting process. Facebook Prophet (FB Prophet) [28], a popular open-source tool, utilizes a time-series model with customizable parameters for trend, seasonality, and holidays.

Forecasting no longer uses just classical models many neural network models that [37] are used , including RNNs [34], LSTM [12] networks, and Neural basis expansion analysis for interpretable time-series (N-BEATS), these models use neural networks to model time-series data. Transformer models, originally developed for natural language processing, have also been adapted for TSF by treating the series as a sequence of inputs and capturing temporal patterns using self-attention mechanisms.

Each model type has its own strengths and weaknesses, making them more suitable for specific types of time-series data or forecasting problems.

This research will focus on the current state-of-the-art models for TSF namely, Prophet and AutoArima for the classical approaches and N-BEATS and RNN-LSTM for the deep learning methods. All these models will be discussed in details in next sections.

3.2.1 Evaluation Metrics

The papers mentioned in 2 typically utilize Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (SMAPE), and R-squared (R2) as evaluation metrics due to their established effectiveness. MAPE and SMAPE offer an intuitive representation of forecasting accuracy in percentage terms, accommodating diverse data scales. They also exhibit robustness against outliers, ensuring reliable performance assessment. R2, commonly used in regression tasks, quantifies variance predictability and indicates how well the model fits the data. This choice ensures comprehensive evaluation aligned with existing literature and facilitates clear interpretation and communication to the scientific community.

Evaluating the performance of time-series models is crucial for accurate forecasting and analysis. Several evaluation metrics are used to assess the goodness of fit and accuracy of these models. All the research papers have predominately have used mostly used the same evaluation metrics. In this thesis the commonly used metrics are: MAPE, SMAPE and R^2,. Each metric provides valuable insights into the model's performance

- R^2**(Coefficient of Determination)**: When dealing with time series data, the dependent variable is what we're trying to figure out or predict over time. The independent variables are the factors that could affect the changes in the dependent variable. The R^2 metric, which is like a measuring tool, helps us understand how well the things we know (independent variables) explain the changes we see (dependent variable) in the data. It is widely used to determine how well the model fits the data.

  $$R^2 = 1 - (SSR/SST) \tag{1}$$

  SSR: Sum of the squared differences between the actual values observed in the time series and the values that the model predicts. It helps measure how well our model fits the actual data.

  SST: Sum of squared differences between each individual data point in the time series and the mean value of the whole finite time series. It gives a sense of the total variability in the data.

  Interpretation:
  The R^2 score ranges from $-\infty$ to 1 with 1 indicating a perfect fit and anything below 0 suggesting poor model fit. Higher R^2 values indicate that a model is well fit. It's important to note that R^2 does not derive from squaring a specific value i.e, SSR ¿¿ SST it can evaluate to be a negative value. Specifically, R^2 becomes negative when the chosen model deviates from the data's trend, resulting in a less favorable fit.

- **MAPE (Mean Absolute Percentage Error)**: The MAPE metric measures the average percentage difference between predicted and actual values. It provides a relative measure of the model's accuracy, making it useful for comparing performance across different datasets and time-series.

  $$MAPE = (100/n) * \sum_{i=1}^{u} |(test_i - predicted_i)/test_i| \tag{2}$$

  Where:
  n is equal to the number of time-series observations
  $test_i$ is equal to the actual value, where $i \in \{1, 2, ...u\}$
  $predicted_i$ is equal to the respective predicated value, where $i \in \{1, 2, ...u\}$

Interpretation: MAPE represents the average percentage deviation between predicted and actual values. However, MAPE has limitations when the actual values are close to zero or contain zero values, leading to undefined or infinite values. Therefore, caution should be exercised when interpreting MAPE results. Lower the values of MAPE shows that the model fits the data well

- **SMAPE (Symmetric Mean Absolute Percentage Error)**: SMAPE is a modified version of MAPE that overcomes the asymmetry issue when the actual values are close to zero. It provides a balanced and symmetric measure of the percentage difference between predicted and actual values.

$$SMAPE = (200/n) * \sum(|test_i - predicted_i|/(|test_i + predicted_i|) \quad (3)$$

Where:
n is equal to the number of time-series observations
$test_i$ is equal to the actual value, where $i \in \{1, 2, ...u\}$
$predicted_i$ is equal to the respective predicated value, where $i \in \{1, 2, ...u\}$

Interpretation:Interpretation: SMAPE provides a symmetric measure of the percentage deviation between predicted and actual values. It is preferred when dealing with time-series data that may contain zero or close-to-zero values. Lower SMAPE values indicate a better model fit

Table 1: Mertics used for evaultion

Metric	Interpretation
R^2	A value close to 1 indicates good fit
MAPE	A lower value indicates a better fit
SMAPE	A lower value indicates a better fit

3.2.2 Facebook Prophet

Facebook Prophet [28], is a widely used open-source TSF library, that has been developed by Facebook's Core Data Science team. This model simplifies the process of predicting time-series data while making it easy to use in Python.

Prophet incorporates key components that make forecasting accessible to both data scientists and domain experts. It utilizes an additive model to break down time-series into trend, seasonality, and holiday components. Prophet's decomposition approach significantly enhances its modeling and forecasting capabilities, enabling the extraction of intricate time-series patterns. By effectively addressing missing data and outliers, Prophet accommodates irregularities frequently encountered in real-world datasets. This contributes to a simplified yet robust framework for predicting future values in univariate time-series data, streamlining forecasting and alleviating the need for laborious manual parameter tuning. Through its automated trend detection, seasonality estimation, and model selection, Prophet offers a streamlined and efficient process for accurate predictions.

In practical applications, Facebook Prophet has gained widespread adoption across industries, proving valuable for tasks such as demand forecasting, inventory optimization, and financial planning. However, it is essential to acknowledge certain limitations. Prophet's primary design caters to univariate time-series forecasting, potentially restricting its support for intricate multivariate scenarios involving multiple interrelated variables. Additionally, Prophet heavily relies on the accurate identification of trends, seasonality, and holiday effects. The presence of outliers or inadequate capturing of these

patterns can undermine its forecasting accuracy. Furthermore, while Prophet attempts to handle outliers, the model's performance may be sensitive to extreme or influential outliers that can disrupt its ability to capture meaningful patterns and generate reliable forecasts.

Prophet's inherent strengths, particularly its decomposition of seasonalities, trends, and holidays, align well with the chosen dataset for this research. A comprehensive comparison will be conducted between Facebook Prophet and other state-of-the-art univariate forecasting models, such as AutoArima, using the same univariate dataset. Through rigorous evaluations of Prophet's performance metrics, a thorough assessment will be made by comparing it to classical models in terms of simplicity, accuracy, and forecasting effectiveness. The Prophet model will be executed using its default settings [13], without alterations to hyperparameters, to ensure a consistent baseline evaluation. This approach is justifiable based on Prophet's proven ability to autonomously optimize its hyperparameters, contributing to a more comprehensive and unbiased model assessment[14]

3.2.3 AutoArima

AutoArima [8] short for Automatic AutoRegressive Integrated Moving Average, is a widely adopted univariate time-series forecasting model recognized for its automated parameter selection within the ARIMA framework. This model is renowned for its ability to predict future values by autonomously determining optimal lag orders and differencing parameters. Unlike traditional ARIMA models that require manual parameter selection, AutoArima employs an algorithmic approach to explore various combinations of lag orders (p), differencing (d), and moving average orders (q), simplifying the model selection process and alleviating the challenges of manual tuning. Its automation facilitates integration into data analysis workflows and accommodates users with varying levels of time-series expertise. AutoArima's selection as a forecasting model for the dataset is motivated by its automation, which adapts to unique dataset characteristics and patterns. Through its implementation, this research aims to evaluate its forecasting performance and assess its applicability in this specific context.

However, it's important to acknowledge the limitations of AutoArima. It assumes stationarity of time-series data, potentially disregarding trends or seasonality present in real-world data. Handling outliers and anomalies poses a challenge for AutoArima, as they can lead to suboptimal model fits. The model also does not consider external factors or exogenous variables that can influence the time-series. Despite these limitations, AutoArima's automated parameter selection mechanism streamlines the forecasting process, which aligns with the goal of this research to assess its performance against other models.

Incorporating AutoArima alongside Facebook Prophet enables a comprehensive comparison, evaluating their forecasting accuracy, simplicity, computational efficiency, and overall effectiveness. By understanding the strengths and limitations of both models, researchers and practitioners can make informed decisions when selecting the most suitable model for accurate and reliable predictions in univariate time-series analysis. The findings of this research contribute to the broader understanding of univariate time-series forecasting models and provide insights that have practical implications for industries relying on accurate forecasts for decision-making. Just as with Prophet, AutoArima is utilized at its default settings to ensure a consistent and unbiased evaluation of its performance.

Utilizing forecasting models at their default settings ensures a consistent and unbi-

ased evaluation of their performance. This approach aligns with best practices in the field of time-series forecasting, as it provides a fair comparison of models' inherent capabilities without introducing potential bias arising from manual parameter tuning. The application of models at their default settings has been used for evaluating forecasting performance [13]. This approach aids in showcasing the models' out-of-the-box effectiveness, which is particularly important when comparing their inherent abilities across different datasets. It eliminates the influence of subjective parameter choices that might vary based on the analyst's familiarity with the model, ensuring a more objective and reliable assessment.[14]

3.2.4 RNN-LSTM

RNNs have emerged as one of the most powerful deep learning methods for time-series analysis. Unlike traditional feedforward neural networks, RNNs have recurrent connections that allow them to retain information from previous time steps and use it to inform predictions at subsequent steps. This capability enables RNNs to capture long-term dependencies, which is crucial for accurate TSF. RNNs process data sequentially, taking into account the current input and the information stored from previous time steps. This recurrent nature makes RNNs capable of modeling time dependencies and capturing long-term dependencies in the data. Each step in the sequence involves feeding the current input along with the hidden state from the previous step into the RNN, producing an output and updating the hidden state for the next step.

Additionally, RNNs demonstrate great flexibility in handling variable-length sequences commonly found in time-series data. They can process sequences of different lengths, making them adaptable to various time-series lengths without the need for fixed-size inputs. This flexibility is particularly advantageous when dealing with irregularly sampled or sparse time-series data, where the time intervals between data points are irregular.

Furthermore, RNNs can be enhanced by incorporating specialized units such as LSTM. LSTMs address the vanishing gradient problem by selectively retaining or forgetting information over long sequences, allowing the model to capture long-term dependencies more effectively. The combination of RNNs with LSTM units, known as RNN-LSTM, has shown competative performance in TSF tasks [26].

Overall, the inherent characteristics of RNNs, including their sequential data processing, ability to capture long-term dependencies, adaptability to variable-length sequences, dynamic modeling of temporal dynamics, and integration with specialized units like LSTM, contribute as one of the best deep learning methods for time-series analysis [26]. Their application in various industries and research domains showcases their effectiveness in extracting valuable insights and making accurate predictions from time-series data.

RNNs suffer from the vanishing gradient problem, which limits their ability to capture long-term dependencies in time-series data. LSTM[12], a variant of RNN, was introduced to overcome this issue. LSTM networks have an additional memory cell that allows them to selectively retain or forget information over long sequences, making them well-suited for capturing long-term dependencies.

Furthermore, LSTM's ability to learn and model complex temporal patterns is highly advantageous for TSF. By analyzing historical time-series data, LSTM can discover underlying patterns, relationships, and dependencies. The network can learn to recognize seasonality, trends, irregularities, and other intricate patterns that contribute to accurate forecasting.

Additionally, LSTM's architectural design makes it well-suited for handling various

challenges in time-series analysis. It can handle irregular time intervals, missing data, and noisy observations. Its inherent adaptability enables it to dynamically adjust to changing patterns and capture evolving trends in the data.

RNN-LSTM model is powerful for time-series analysis. They excel at capturing temporal dependencies and long-term patterns, making them valuable for tasks such as TSF, anomaly detection, and sequence prediction. Over the years RNN with LSTM have been considered as the benchmarked model[33] for TSF for transfer learning after CNN models which convert the time-series in image representation. When comparing these methods, several considerations come into play. Facebook Prophet's strength lies in its decomposition approach, which effectively captures patterns via trend, seasonality, and holidays. This suits datasets with varying patterns and missing values. AutoARIMA's automation minimizes manual intervention, enhancing simplicity and speed, albeit potentially struggling with complex patterns. LSTM's deep learning capabilities make it well-suited for intricate data with nonlinear relationships, but its performance is tied to proper architecture setup. Since RNN-LSTM is better than the most other flavours of RNN especially for time-series [33] LSTM will be used for the research

3.2.5 N-BEATS

N-BEATS [20] (Neural basis expansion analysis for interpretable TSF) is s a more recent innovation that has gained attention for its impressive performance in time series forecasting for both multivariate and univariate

N-BEATS distinguishes itself from RNNs by employing a fully feedforward architecture, making it computationally efficient and easier to train. The model consists of a stack of fully connected layers called basis functions, which learn to transform the input time-series into a set of interpretable patterns. These patterns capture different types of temporal dynamics, allowing N-BEATS to accurately forecast future time steps.

One key advantage of N-BEATS over RNNs is its ability to provide interpretable forecasts. The basis functions in N-BEATS capture meaningful patterns such as trend, seasonality, and other temporal dynamics inherent in the time-series. By decomposing the time-series into interpretable components, N-BEATS allows users to gain insights into the driving factors behind the forecasts. This interpretability empowers analysts and domain experts to make informed decisions and better understand the underlying data.

The feedforward nature of N-BEATS enables parallelization, resulting in faster training and inference times. Unlike RNNs, which require sequential processing of data, N-BEATS can process multiple time steps simultaneously. This scalability is particularly valuable when working with large-scale time-series datasets, where RNNs may face computational limitations due to their recurrent connections.

Furthermore, N-BEATS exhibits state-of-the-art forecasting accuracy compared to RNNs. Numerous benchmark evaluations and comparative studies have consistently demonstrated that N-BEATS outperforms RNN-based models [4], including LSTM, in terms of forecasting error. The ability of N-BEATS to capture complex temporal patterns and dependencies without relying on recurrent connections contributes to its superior accuracy. The basis functions in N-BEATS can capture various types of temporal dynamics, allowing the model to adapt to different patterns present in the time-series data.

Moreover, N-BEATS offers greater flexibility and adaptability to different time-series domains and forecasting tasks. The model can be easily customized by adjusting the number and size of the basis functions to suit specific data characteristics.

Parameters	N-BEATS	RNN-LSTM
input_chunk_length	14	14
num_layers/n_rnn_layers	2	2
training_length	NA	24
batch_size	32	32
output_chunk_length	1	1
generic_architecture	True	NA
num_stacks	30	NA
activation	ReLu	NA
dropout	0	0
n_epochs	100	100
layer_widths	256	NA
hidden_dim	NA	25
expansion_coefficient_dim	5	NA
optimizer_kwargs	NA	None
trend_polynomial_degree	2	NA
force_reset	True	True

Table 2: Parameters for deep learning models

3.2.6 Parameters and hyperparmeter optimizzation

Both N-BEATS and RNN-LSTM have been tuned on parameters as similar to each other as possible.They are tuned in way to fit the dataset. Parameters have been changed from their default values like the input chunk length see the table 2. The parameters are arranged on the basis of their influence on the models. Controlled experiments with similar parameters are a common practice in machine learning research to ensure fair comparisons between different models [19]. This approach helps isolate the impact of individual model components on performance.[30] Stable comparisons involve keeping variables consistent to avoid introducing unnecessary variability that could lead to biased comparisons [24]. Running experiments with similar parameters allows for in-depth analysis of how each model architecture learns, predicts, and behaves under comparable conditions.

The hyperparameters are also optimized to see, the their impact on both the models. Grid search is used as the hyperparameter tuning technique. In this, we define a grid of possible values for each hyperparameter, and then iterate through all possible combinations of hyperparameters to find the best set of hyperparameters that results in the optimal performance for your model. From the table 2 the first five parameters are selected for this grid search since these parameters are common in both RNN-LSTM and N-BEATS model. Table 3 shows the hyperparameters that are optimized and the grid space for these parameters. These parameters have a direct impact on the model's architecture, learning process, and prediction horizon. By systematically varying these parameters, we can effectively explore different configurations and identify the combination that yields the best performance.

Table 3: Parameters for deep learning models

Parameters	**N-BEATS**	**Tuned Parameters Range**	**RNN-LSTM**	**Tuned Parameters range**
input_chunk_length	14	[10-40]	14	[10-40]
num_layers/n_rnn_layers	2	[1-5]	2	[1-5]
training_length	NA	NA	24	[12,24,36]
batch_size	32	[16,32,64]	32	[16,32,64]
output_chunk_length	1	[1,10]	1	[1,10]

3.3 Transfer Learning

Transfer learning [3] is a powerful and popular technique in machine learning. The basic idea of transfer learning is to reuse knowledge gained from solving one problem, and apply it to a different, but related problem. This holds the promise of improving generalization and reducing the need for large amounts of labeled data. By leveraging the knowledge gained during training on the source task, the model can enhance its performance on the target task. This strategy proves particularly beneficial when the target task has limited data or exhibits similarities with the source task.

Transfer learning is done in a specific domain using a particular task. A domain is defined by its features and probability distribution, while a task consists of a label space and a predictive function. Domains can be images, text, or speech, while tasks include classification, regression, and generation. Transfer learning can save time and

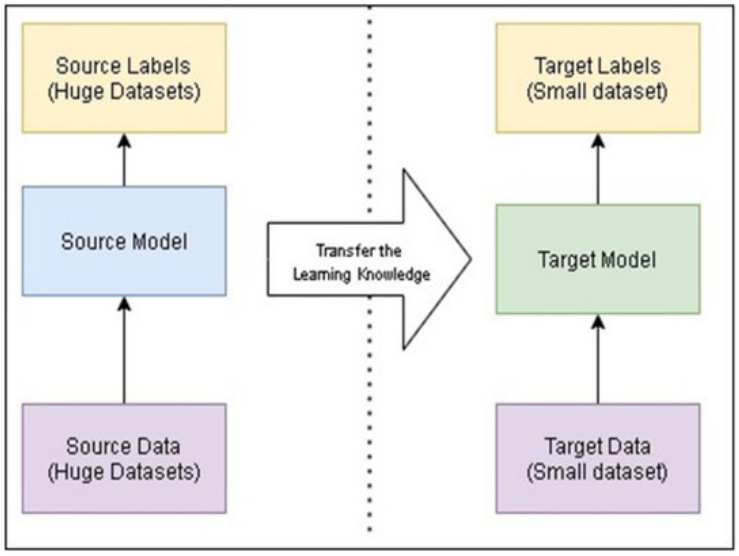


Figure 1: Transfer Learning high level [25]

resources by allowing us to reuse existing models rather than starting from scratch, and it can improve the performance of models on new tasks, especially when there is limited data available. In the context of TSF, transfer learning can be used to improve the accuracy of predictions by leveraging knowledge from related time-series. For example, a model trained on weather data from one region can be adapted to make predictions for a different region with similar weather patterns. Transfer learning can also be used to improve the efficiency of training by allowing the model to start with pre-trained weights, reducing the amount of data required for training a new task. Transfer learning aims to improve learning of the target task using information from the source task. For example,

one might train a model on a large dataset of images, and then use the model for a different task, such as object detection in a new set of images. The goal is to improve the learning rate or accuracy on the new task by transferring the relevant knowledge from the source. Transfer learning can be categorized into problem categorization and solution categorization as discussed in the paper [38] Problem categorization further divides into labeled settings-based and space settings-based approaches.

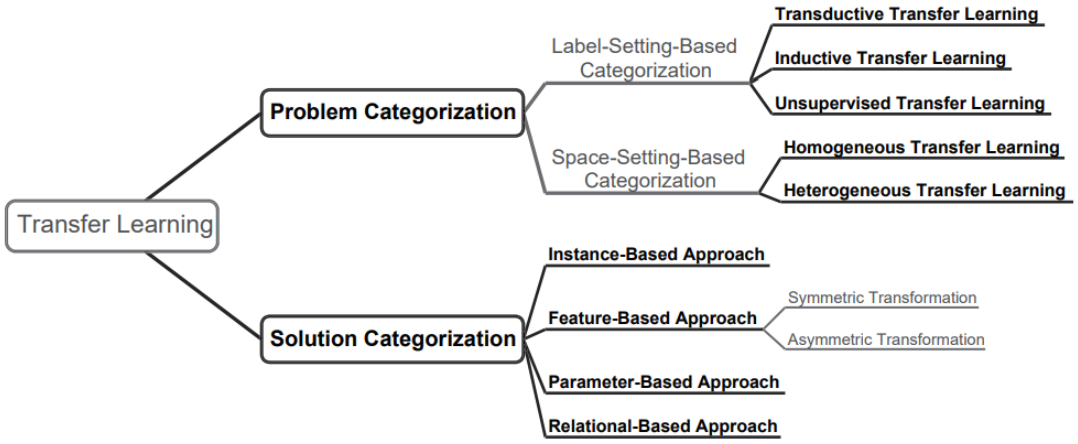


Figure 2: Transfer Learning categories [38]

Labeled settings can be further classified into inductive transfer learning, transductive transfer learning, and unsupervised transfer learning [23]. Inductive transfer learning refers to scenarios where the source and target domains have different feature spaces but share the same label space. Transductive transfer learning involves scenarios where the source and target domains share the same feature space but have different label spaces. Unsupervised transfer learning deals with scenarios where the source and target domains have different feature and label spaces.

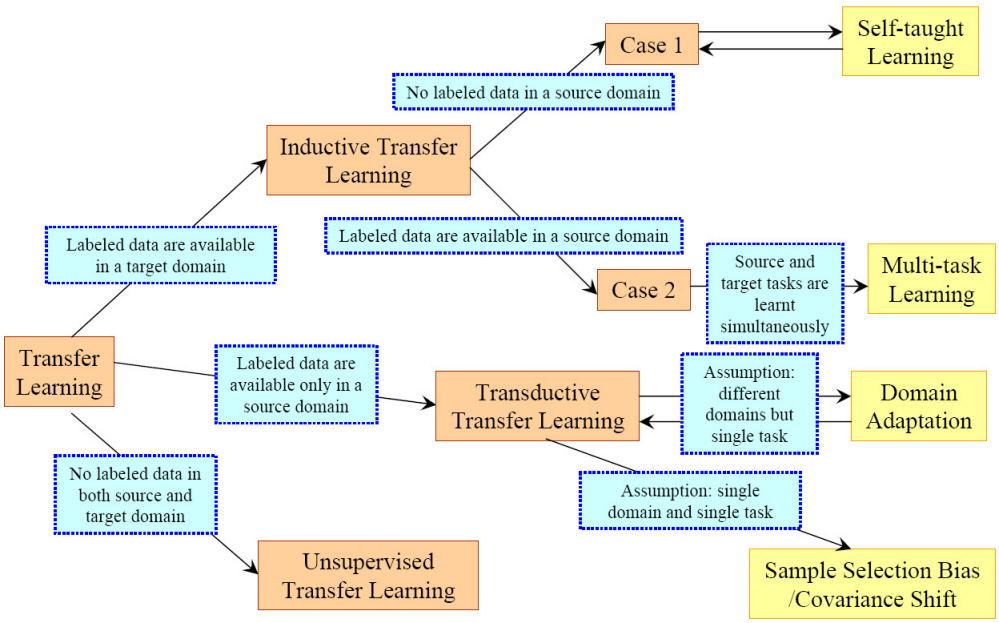


Figure 3: Transfer Learning methods [23]

Space settings categorize transfer learning into homogeneous and heterogeneous transfer learning. Homogeneous transfer learning occurs when the source and target domains have the same feature space but different marginal distributions or conditional distributions. Heterogeneous transfer learning, on the other hand, deals with scenarios where the source and target domains have different feature spaces.

Solution categorization transfer learning approaches can be classified as model-based, feature-based, parameter-based, and relation-based methods[23].

Model-based transfer learning involves using a pre-trained model on the source domain and fine-tuning it on the target domain. The pre-trained model acts as the base model, which is then adapted to the target domain using gradient descent. This approach has been found to be helpful in cases where the target dataset is small because it allows us to leverage the learned knowledge from the source domain in the target domain.

Feature-based transfer learning involves extracting meaningful features from the source domain, which are then used in the target domain. These features can be either general or specific to the source domain, and are leveraged to improve the learning of the target domain.

Parameter-based transfer learning shares the parameters between the source model and the target model, allowing them to learn from each other. This approach has the advantage of reducing number of parameters that need to be learned in the target domain, which can be beneficial when the target dataset is small.

Relation-based transfer learning involves learning the relationship or mapping between the source and target domains. This approach has the advantage of being flexible and adaptable to various tasks. It can be used for tasks like cross-domain sentiment analysis and cross-lingual word embeddings.

These different transfer learning approaches can be used based on the specific task at hand. Some tasks may benefit more from a model-based approach, while others may require a feature-based or parameter-based approach In recent years, transfer learning has evolved, leading to sophisticated techniques such as zero-shot learning [15] and one-shot learning. One-shot [5] and zero-shot learning represent advanced techniques in transfer learning that aim to tackle the challenge of recognizing or classifying objects or instances with limited or no labeled training data.

One-Shot Learning In conventional machine learning or deep learning approaches, substantial labeled training data is typically necessary to effectively train models. However, acquiring abundant labeled data for every class or category may be impractical or costly in real-world scenarios. One-shot learning surpasses this limitation by enabling models to recognize or classify new instances based on a single example per class. To achieve one-shot learning, various techniques can be employed, including similarity-based methods that allow models to compare and match features or characteristics of the given example with the test instances. Metric learning approaches, such as Siamese networks or prototypical networks, strive to learn a similarity metric to differentiate between different classes based on a limited number of examples.

Zero-shot learning takes a step further by addressing the challenge of recognizing or classifying objects or instances that were not encountered during training. In traditional learning approaches, the model can only recognize classes or categories that are present in the training data. However, in zero-shot learning, the model demonstrates the ability to generalize to unseen or novel classes during testing. Zero-shot learning [27] involves learning a mapping between the input data and a semantic space where classes or categories are represented by attributes or textual descriptions. During training, the model learns to associate visual features with semantic representations of classes. At

test time, the model can classify instances into classes it has never encountered before by leveraging the learned associations between visual and semantic features.

Both one-shot learning and zero-shot learning are challenging techniques in transfer learning as they require models to generalize effectively from limited or no training data. These techniques have been successfully applied to various domains, including image recognition, object detection, and natural language processing, to overcome the limitations of traditional supervised learning approaches and empower models with enhanced flexibility and adaptability.

Transfer learning has many applications in real-world settings, including image recognition, natural language processing, and speech recognition. For example, a model trained to recognize handwritten digits can be used to recognize other handwriting styles, such as printed letters. This technique could be used in numerous applications, from recognizing check amounts on scanned images to recognizing customer signatures.

However, transfer learning also poses several challenges. One of the main challenges is determining which information from the source domain is useful and transferable to the target domain. Another challenge is avoiding transferring information that is detrimental to the desired outcome. Furthermore, transfer learning also has limitations, such as the need for similar feature spaces, models, and tasks in the source and target domains.

3.3.1 Transfer Learning in Time-series

With the increase in time dependent data, it has become crucial in to investigate new methods to get better forecast for this time-series data, it is prevalent in variety of domain from finance to health care. However, collecting labeled data in such domains can be time-consuming and expensive, making transfer learning a promising approach. Transfer learning in time-series has increasingly studied in the last few years [33]. In many time-series applications obtaining labeled data can be costly or time-consuming, which limits the amount of data available for training. Transfer learning holds significant importance in time-series analysis due to a multitude of compelling reasons.

- Limited labeled data: Time-series datasets often pose a challenge with their scarcity of labeled data, making it arduous to train accurate models from scratch. Transfer learning offers a viable solution by harnessing knowledge acquired from a source dataset rich in labeled data. This enables us to enhance model performance on a target dataset that suffers from limited labeled data.
- Domain adaptation: Time-series data exhibits variations across different domains or contexts. Transfer learning empowers us to adapt models trained on a source domain effectively, capturing relevant patterns and characteristics specific to the target domain. This adaptability is crucial in ensuring optimal performance in diverse contexts.
- Generalization: Transfer learning plays a pivotal role in increasing the generalization capabilities of models. By transferring knowledge gained from a source dataset, models can better comprehend and predict the target dataset. This proves especially advantageous in TSF, where the target dataset might be limited in samples or possess distinct characteristics.
- Model efficiency: Utilizing pre-trained models obtained from a source dataset offers significant efficiency gains. These models serve as a robust foundation for training models on a target dataset, resulting in reduced training time and computational

resources. This efficiency is particularly valuable when dealing with large-scale time-series datasets.

The previous section 3.3 has already delved in the intricacies of the different methods of transfer learning which also encompasses the different methods of transfer learning in time-series. Of the many transfer learning methods the two most commonly used transfer learning methods are feature-based transfer learning and model-based transfer learning with respect to TSF [33]

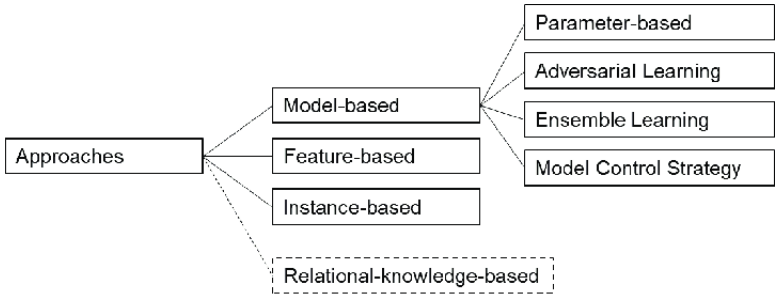


Figure 4: Transfer Learning methods [33]

Feature-Based Transfer Learning: Feature-based transfer learning involves extracting relevant features from the source time-series and using them to enhance the prediction of the target time-series. This approach focuses on capturing domain-specific patterns and characteristics that can be generalized across different time-series datasets.

This thesis will further delve into the model-based approach of transfer learning in the next section 3.3.2

3.3.2 Model-Based Transfer Learning:

Model-based transfer learning involves training a base model on the source time-series dataset and transferring the learned model parameters to the target time-series dataset. This approach assumes that the learned representations or parameters of the base model capture relevant temporal patterns that can be useful for predicting the target time-series. After training a base model on the source time-series, the learned weights or parameters are transferred to the target time-series model. The transferred parameters are then fine-tuned or adapted on the target time-series dataset to improve its predictive performance. Since the majority of the research is done with model-based transfer learning for time-series, it will be one of the methods used in this thesis. In the model-based approach, instead of freezing weights, this research will freeze part of the dataset. This innovative approach holds promising potential, particularly for various industrial applications. . The research aims to see the effect of transfer learning on RNN-LSTM which is shown as one of the most used baseline methods for transfer learning. [33].

Moreover, to explore novel avenues in time-series transfer learning, this research will also delve into zero-shot learning. Zero-shot learning is a relatively underexplored technique in the field of TSF [33]. It enables the model to generalize to unseen target tasks by leveraging knowledge from related source tasks without direct training on target data. By leveraging models such as N-BEATS and RNN-LSTM, this thesis aims to apply zero-shot learning techniques to TSF and evaluate their effectiveness.

Currently time-series transfer learning is an active and evolving research area. As a result, various transfer learning methods and combinations of approaches are continuously being explored. For instance, recent studies have highlighted the potential of zero-shot

learning in TSF tasks, indicating its relevance and promising outcomes [21]. Until now, the exploration of zero-shot learning within the context of N-BEATS and RNN-LSTM has been limited as to best of our knowledge. This research endeavors to address this gap by investigating the potential application of zero-shot learning. Transfer learning in TSF poses unique challenges compared to different types of data. Time-series data has temporal dependencies, and these dependencies can vary significantly between different time-series. Therefore, it can be challenging to identify related time-series with similar patterns and dependencies. Furthermore, transfer learning methods designed for other types of data may not be suitable for time-series data.When applying transfer learning to univariate or multivariate TSF, the pre-trained model can capture patterns and relationships in the data, which can be beneficial for predicting future values or patterns in similar time-series data. Thus, in this thesis, the aim will be to explore the challenges and opportunities of zero-shot and model-based transfer learning in TSF, and evaluate its effectiveness compared to traditional TSF techniques. The central focus will be on assessing the practical efficacy of these transfer learning methods and their comparison to established traditional TSF techniques.

4 Experimental Setup

4.1 Dataset

Time-series forecasting research has been around for a several decades, this has lead to development of benchmark datasets. Well-known TSF benchmark datasets used in research are LORENZ, M4,M5, international airline passenger. The dataset used for this thesis will be the M5 dataset [18], as it has been used to validate the state-of-the-art time-series forecasting methods and forecasting competitions and is large enough to train deep learning models. It was released in 2020 as part of the M5 forecasting competition. The dataset includes 3049 products classified into three categories (Hobbies, Foods, and Household) and seven departments within these categories. The products are sold across 10 stores located in three states: California (CA), Texas (TX), and Wisconsin (WI).

The data can be merged on different levels of aggregation. One way is by *location*, considering the store and state information. Another way is by *product*-related information, considering the department and category. These different levels of aggregation allow for analyzing the sales patterns at various levels of granularity.

The selection of stores and states by Walmart was done deliberately to represent different characteristics, shopping habits, and dynamics. Similarly, the product categories and departments were chosen to represent a mix of consumables and durable goods, as well as products with varying sales velocities.

To capture the diverse hierarchy within the M5 dataset, multiple cross-sectional levels of aggregation are considered for evaluation. The dataset employs a numbering system to categorize levels of aggregation. This systematic approach enables comprehensive analysis of sales patterns, yielding valuable insights into consumer behavior, emerging market trends, and product performance. Considering the various levels of aggregation, the M5 dataset enables researchers to analyze and forecast sales patterns at different levels of granularity, providing insights into consumer behavior, market trends, and product performance.

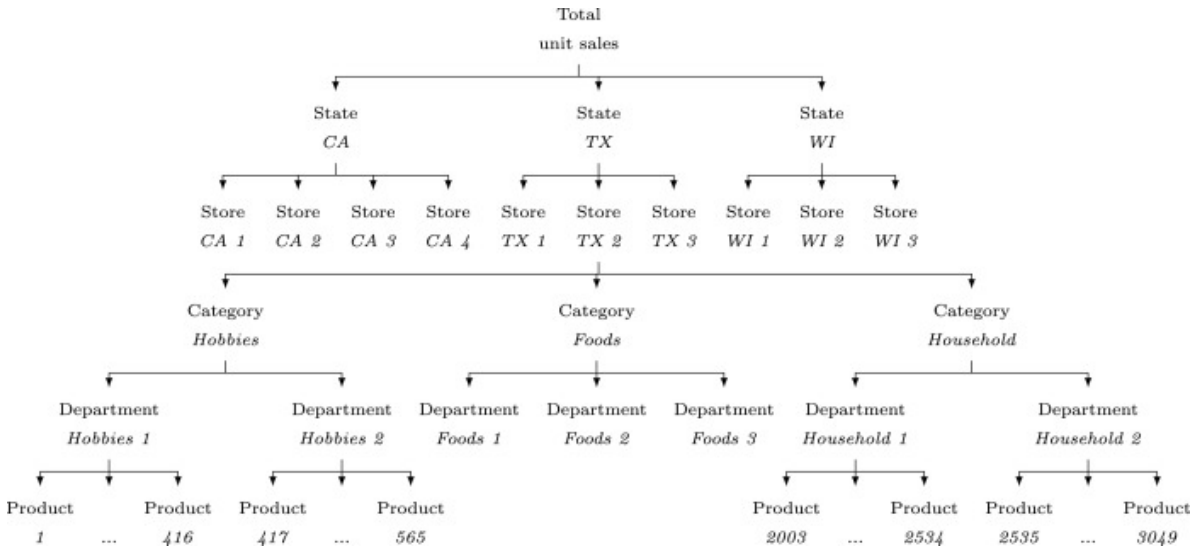


Figure 5: M5 dataset hierarchy details [18]

The dataset itself utilizes hierarchical sales data provided by Walmart. The data is available at item level and aggregated at higher levels starting from department, store, product and finally in three geographical areas of the United States: California, Texas, and Wisconsin. In addition to the time-series data, the dataset also included explanatory variables such as price, promotions, day of the week, and special events (e.g. Super Bowl, Valentine's Day, and Orthodox Easter) that can affect sales and improve forecasting accuracy. The dataset is divided into sales_train_validation, sales_train_evaluation, calendar, prices. One of the datasets sales_train_validation or sales_train_evaluation can be used for analysis and modelling. Furthermore, The M5 dataset consists of daily data spanning a period of approximately 5.4 years, from January 29, 2011, to June 19, 2016, comprising a total of 1969 days but the evaluation date is set to May 22,2016 and there are 1941 in for evaluation.

Level id	Level description	Aggregation level	Number of series
1	Unit sales of all products, aggregated for all stores/states	Total	1
2	Unit sales of all products, aggregated for each state	State	3
3	Unit sales of all products, aggregated for each store	Store	10
4	Unit sales of all products, aggregated for each category	Category	3
5	Unit sales of all products, aggregated for each department	Department	7
6	Unit sales of all products, aggregated for each state and category	State-category	9
7	Unit sales of all products, aggregated for each state and department	State-department	21
8	Unit sales of all products, aggregated for each store and category	Store-category	30
9	Unit sales of all products, aggregated for each store and department	Store-department	70
10	Unit sales of product i, aggregated for all stores/states	Product	3,049
11	Unit sales of product i, aggregated for each state	Product–state	9,147
12	Unit sales of product i, aggregated for each store	Product–store	30,490
Total			**42,840**

Table 4: M5 dataset original

4.1.1 Data-Preprocessing

The preprocessing step plays a crucial role in handling large datasets like M5 to optimize computational resources and reduce complexity. Given the vast number of potential time-series in the M5 dataset (42,840), it is reasonable to select a subset of time-series for analysis in the context of this thesis.

By carefully selecting a subset of time-series, the computational requirements can be managed more effectively. It allows for a focused analysis on a representative sample of

time-series, reducing the complexity of pretraining and handling the data. This selection process enables researchers to concentrate on specific aspects and achieve meaningful insights without overwhelming computational constraints.

The preprocessing step in this thesis involves selecting a subset of time-series from the M5 dataset based on a specific criterion. In this case, the selection criterion is to focus on the top four levels of aggregation in the dataset.

The M5 dataset offers multiple levels of aggregation, with different levels representing varying degrees of granularity. By choosing the top four levels of aggregation, the analysis can capture a comprehensive view of the dataset while reducing computational complexity.

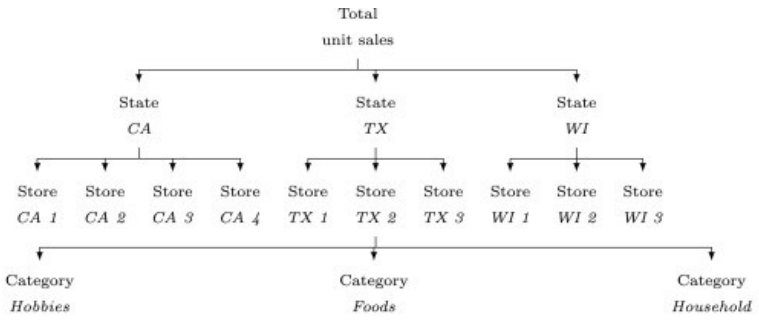


Figure 6: M5 dataset hierarchy details [18]

The top four levels of aggregation include higher-level categories, regions, or the states and stores that provide meaningful insights into the dataset as seen in 6. This selection criterion allows for a manageable subset of time-series while still maintaining the representativeness and diversity of the data.

The series are selected based on their hierarchy as seen in figure 6. Of 42,840 time-series 56 time series are choosen based on the top 4 levels of hierarchy. Table 5 shows how and which time-series combination are selected. These time-series are selected because they capture the most important details of the dataset.

Level id	Level description	Aggregation level	Number of series
1	Unit sales of all products, aggregated for all stores/states	Total	1
2	Unit sales of all products, aggregated for each state	State	3
3	Unit sales of all products, aggregated for each store	Store	10
4	Unit sales of all products, aggregated for each category	Category	3
5	Unit sales of all products, aggregated for each store and category	Store-category	30
6	Unit sales of all products, aggregated for each state and category	State-category	9
Total univariate			**56**

Table 5: M5 dataset with time-series after pre-processing

This approach ensures that the analysis remains feasible while still capturing the important patterns and characteristics of the dataset.

By selecting a subset of time-series from the M5 dataset, this thesis aims to strike a balance between the available computational resources and the complexity associated with handling the entire dataset.

First, three distinct CSV files were imported, encompassing calendars with all dates, sales data, and item prices. These datasets were transformed into data frames. To optimize memory usage while maintaining data integrity, downsizing techniques were applied due to the dataset's size. Subsequently, a merged dataset was generated by combining these data frames using a shared identifier, 'd'.

Following this, columns like $'ww - mm - yyyy'$ were eliminated from the refined dataset to enhance memory efficiency. This process was pivotal in managing the dataset effectively, facilitating analysis while upholding accuracy.

To obtain multiple time-series for model training, the dataset was filtered by region based on its total sales for different stores. The three regions considered were California, Texas, and Wisconsin. Further, the data was filtered to obtain sales per store per region, resulting in four stores for California, three stores for Wisconsin, and three stores for Texas. This filtering was necessary to obtain multiple time-series that could be used for model training and prediction.

Algorithm 1 Data Preprocessing for Time-Series Analysis

Input: The M5 CSV with calendar dataset, sales dataset and sell prices dataset of California, Texas and Wisconsin

Output: A collection of multiple time-series data formatted for Darts.

procedure DATAPREPROCESSING
    Downsize the datasets to reduce memory usage from int64 to int4.
    Merge the three datasets using a common key, 'd', to create a dataframe.
    Remove unused columns such as 'ww-mm-yyyy'
    Filter the dataframe by region
    Filter the dataframe on per store per region.
    Divide the data based on the category of items, namely food, household, and hobbies.
    Filter each category per store of that region.
    Remove zero values by replacing them with interpolated values
    Convert this dataframe into a time-series datatype by using the **Date** column as index.
    return a collection of multiple time-series data, structured to be suitable for Darts analysis.
end procedure

To facilitate transfer learning, the data was divided based on the category of items, namely food, household, and hobbies. Each of these categories was filtered per store of the respective region. This step allowed the transfer of knowledge between stores and regions, which is essential in developing effective time-series models.

After merging and filtering the datasets, the resulting dataframes are further processed to convert them into time-series format compatible with the Darts library. However, before performing the conversion, a careful observation is made regarding certain days, such as Christmas days (e.g., 2013-12-25, 2014-12-25), where very low or no sales are recorded. It is evident that on those specific days, the stores were closed, and any sales recorded during those periods are likely erroneous.

To address this issue and avoid having extremely low values in the dataset, which can negatively impact evaluation metrics such as MAPE (Mean Absolute Percentage Error), a technique called interpolation [16] is applied. Interpolation helps in filling in the missing values or low sales observations with reasonable estimates based on the neighboring data points. The initial step involves replacing all instances of zero with NA values. To address the resulting gaps, the interpolate function is applied. This function strategically utilizes the entire time series' min-max range to effectively fill the missing values, enhancing the dataset's completeness and accuracy. [1].

Through interpolation, the dataset is smoothed, generating more accurate values for days with low or no sales like Christmas. This preprocessing step enhances time-series authenticity, reducing outlier influence. Interpolation preserves data integrity, boosting reliability in analysis and forecasting. Addressing low sales days enhances metric accuracy, ensuring precise model evaluation.

4.2 Darts Library

Darts[11] is a library that is focused on time-series analysis. It is widely regarded as one of the best libraries for time-series analysis, offering a comprehensive set of tools and capabilities. Its versatility makes it suitable for various applications in finance, healthcare, engineering, environmental science, and beyond.

One of the strengths of Darts lies in its support for both univariate and multivariate time-series data. It offers a diverse range of models, including classical methods like ARIMA, Prophet, as well as deep learning architectures such as RNN and N-BEATS. The Darts implementation these models will be used for the research.

Majority of the tasks performed in this research is performed using Darts. It is used perform some of the pre-processing tasks along with Numpy and it is primarily used for everything after pre-processing from training to validating to mertics evaluation.

4.3 Methodology

After completing the preprocessing steps, a total of 56 univariate time-series were generated see table 5. The chosen models for this study include Prophet and AutoARIMA for classical approaches, and N-BEATS and RNN (specifically LSTM) for deep learning methods. It is important to note that all models will be used with their default parameters, ensuring no bias or partiality in the evaluation.

The Time-Series Modeling process involves distinct steps for both classical and deep learning models. For classical models, the initial time-series data is converted into an appropriate univariate format suitable for modeling. Next, a classical forecasting model is chosen based on the specific requirements of the analysis. The time-series is then divided into training and test sets, with the split being determined by a predefined validation date. The collection of 56 time-series is meticulously partitioned into separate sets dedicated to training, testing, and validation, each serving the purpose of deep learning analysis. This partitioning adheres to the widely accepted distribution of **70%** for training, **15%** for testing, and **15%** for validation. The rationale underlying this partitioning strategy lies in allocating dates occurring before 2014-04-30 exclusively to the training subset, while the subsequent dates are distributed between the testing and validation subsets For deep learning models, similar preprocessing steps are followed, converting the time-series data into an appropriate format for the selected deep learning model. However, in this case, an additional validation set is created during the data split. The time-series data is divided into training, validation, and test sets.

In both cases, the models are trained on the training data and subsequently used to make predictions on the test set.

Notably, in the case of deep learning models, data scaling is an essential step. Deep learning models benefit from scaled data to mitigate bias and enhance model performance. As a result, the data is scaled before training the deep learning models and the predictions obtained are unscaled to reflect the original data format which is **2014-04-30**.

The entire process aims to evaluate and compare the performance of different models on the given time-series data, providing valuable insights into the forecasting capabilities of both classical and deep learning methods.

Out of the complete set of 56 time-series see table 6, a subset of 30 time-series is meticulously chosen for the purpose of conducting transfer learning experiments. This selection process involves picking one time-series from each hierarchical level depicted in Figure 6. In instances where only three options are available at a given level, all three are considered. To illustrate, from the initial "State" level, all three options—CA, TX, and WI—are included. The selection process continues through subsequent levels, encompassing state-store, category, state-category, and store-category combinations. After selecting the subset of 15 time-series for experimentation, the training and testing series are swapped to create a balanced setup. This means that the data originally designated for training the models is now used for testing, and vice versa. This exchange ensures that each time-series is evaluated as both a training and testing dataset, providing a comprehensive assessment of the transfer learning techniques' performance across different scenarios. This process results in a final set of 30 time-series, where each series serves as both training and testing data according to the level.

Table 6: Selected series for Transfer Learning

No	**TIME SERIES SELECTED**
1	CA_total
2	TX_total
3	WI_total
4	TX_1
5	CA_2
6	WI_3
7	Hobby_combined
8	Food_combined
9	Household_combined
10	CA_full_state_Food sale
11	TX-full_state_Household sale
12	WI_full_state_Hobby sale
13	CA_4Household_ sale
14	TX_1Food_sale
15	WI_2Hobby_sale

In zero-shot transfer learning, the focus is on training a model on one time-series and then leveraging that learning to make predictions on another, entirely different time-series. For instance, the model is initially trained on a specific time-series, such as California (CA) Store 1, and then it is utilized to generate predictions for Texas (TX) Store 2, and so forth, moving across hierarchical levels. The objective here is to examine the ability of the model to generalize its learning from one time-series to another, without any specific training on the target time-series. For example, training on the total sales of CA and predicting on TX or Wisconsin (WI), and vice versa. The process continues by training on a specific store in CA and predicting on a different store in WI or TX, gradually moving from higher levels of aggregation to lower levels from state to store to category 6, such as training on CA Store 1 - food category and predicting on TX Store 3 - hobbies category. The aim is to evaluate the effectiveness of zero-shot transfer learning in this context.

Algorithm 2 Zero-Shot Learning for Time-Series

Input: A hierarchical structure of time-series with a total of 56 time-series
Output: Zero-shot results (MAPE, SMAPE, R^2) on the test set
Initialize an empty table for evaluation results
Select one of the 30 time-series from the hierarchy for training from Section 6
for each time-series in the same hierarchy **do**
    Train N-BEATS and RNN models on the training sets of the selected time-series
    Calculate MAPE, SMAPE, and R^2 results on the test set of the selected time-series
end for
Repeat the process for the remaining time-series
return Zero-shot results

Conversely, model-based transfer learning involves a different approach. First, a model is trained on a particular time-series dataset, as with the deep learning time-series modeling process. However, instead of merely evaluating the model's performance on the same dataset, a portion of a different time-series dataset is introduced to the already-trained model. This additional dataset is used for further fine-tuning the model, effectively transferring its knowledge to new data. Following the fine-tuning process, the model is then utilized to make predictions on other time-series, such as applying the model trained on CA_1 to make predictions for TX_2.

In both transfer learning methods, the goal is to investigate the effectiveness of knowledge transfer and the ability of the model to adapt its learning to new datasets. Zero-shot learning assesses how well the model can generalize without explicit training on the target data, while model-based transfer learning explores the potential of leveraging pre-trained models to enhance performance on new datasets with some context given during retraining. These methodologies provide valuable insights into the transferability and adaptability of TSF models. For instance, training on CA and then using the trained model to train on a selected portion of TX, followed by using this model to predict on the remaining TX data. This is highly relevant to industry right now since data isn't readily availalbe and it can help data scientist to decide if transfer learning can be used in industry.

Algorithm 3 Model-Based Transfer Learning for Time-Series

Input: A hierarchical structure of time-series with a total of 56 time-series
Output: Model-based results (MAPE, SMAPE, R^2) on the test set
Initialize an empty table for evaluation results
Select one of the 30 time-series from the hierarchy for training from Section 6
for each time-series in the same hierarchy **do**
    Train N-BEATS and RNN models on the training sets of the selected time-series
    Select 5% of the other time-series for training
    Retrain the N-BEATS and RNN models on this new selected time-series
    Calculate MAPE, SMAPE, and R^2 results on the test set of the selected time-series
end for
Repeat the process for the remaining time-series
return Model-based results

5 Results

The Results section presents a comprehensive analysis of the experimental findings, shedding light on the performance of various forecasting models across different scenarios. It discusses the outcomes of classical models like Prophet and AutoArima, as well as deep learning models like RNN-LSTM and N-BEATS. Additionally, the section delves into the results of zero-shot learning and model-based transfer learning approaches. The findings highlight the strengths and limitations of each model, offering insights into their predictive capabilities, adaptability, and computational efficiency. The comparison between classical models and deep learning models underscores the significance of leveraging transfer learning techniques to enhance forecasting accuracy. This section will delve into the top results obtained from classical models,deep learning models and the comparisons between them. It also shows the top zero-shot transfer learning results for the deep learning models as well as the best model-based transfer learning results. This section also shows the top results of a comparison between model-based transfer learning and the classical model i.e Prophet.

5.1 Classical Models

The analysis of the experimental results reveals intriguing insights regarding the performance of two classical forecasting models, Prophet and AutoArima.

Prophet's consistent outperformance in 26 out of 56 instances reflects its remarkable predictive capabilities, reinforcing its stature as a resilient forecasting model. This overarching trend underscores its adaptability across diverse forecasting scenarios, aligning with the broader goal of delivering accurate predictions across a range of data-driven contexts.

When examining the well-fitted models based on the positive R^2 value, it was observed that Prophet generally performed well. The best-fitted Prophet model achieved a MAPE of 4.952695 , a SMAPE of 5.068750 and a R^2 score of 0.788715,. In comparison, the best-fitted AutoArima model achieved a MAPE of 4.599789, a SMAPE of 4.654468 and a R^2 score of 0.833042.

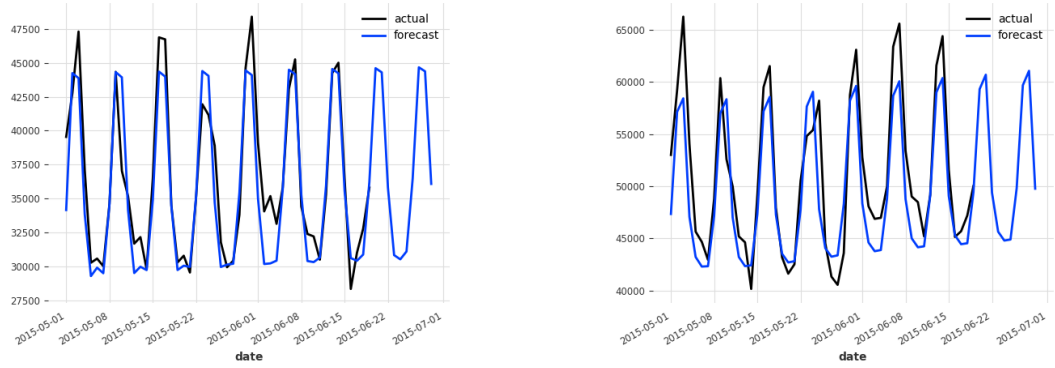
Note: The bold text in the tables denoted the best times-series with the best overall evaluation metrics score.

Table 7: Prophet vs AutoArima. For detailed results see table 16 and table 15

PROPHET MODEL					AUTOARIMA MODEL			
TIME-SERIES NAME	**MAPE**	**SMAPE**	**R^2**	**TIME (s)**	**MAPE**	**SMAPE**	**R^2**	**TIME (s)**
Household_combined	5.92	6.18	0.71	0.38	**4.59**	**4.65**	**0.83**	77.28
CA_full_state_Hobby _sale	5.95	5.82	0.55	0.42	**5.44**	**5.35**	**0.58**	88.33
CA_total	**4.95**	**5.06**	**0.78**	0.36	6.25	6.46	0.71	94.46
CA_full_state_Househ _sale	**6.86**	**7.21**	**0.7**	0.29	**9.20**	**9.85**	**0.51**	71.66
WI_3Hobby_sale	13.60	13.26	0.25	0.34	13.99	13.53	0.28	71.83
CA_1Food_sale	8.83	8.44	0.65	0.29	8.37	8.62	0.68	112.63
Total sale	6.97	7.04	0.63	0.37	7.90	8.32	0.48	85.84
TX_ 1	7.55	7.51	0.55	0.32	10.70	11.48	0.27	73.51

Upon a thorough analysis of these results, it can be concluded that Prophet consistently demonstrates strong performance. The best-fitted Prophet model showcased competitive performance, although slightly lower than the best-fitted AutoArima model in terms of MAPE, ,SMAPE and R^2, .

An intriguing situation arises when observing Prophet's performance in less-than-optimal fitting scenarios. Here, Prophet's remarkable ability to maintain predictive accuracy shines. This capacity is of utmost importance in real-world applications where data often exhibits irregularities and deviations. Prophet's capacity to generate accurate forecasts under suboptimal conditions showcases its robustness, making it a valuable asset for practical decision-making. A crucial consideration pertains to the default parameters of the models and their applicability across different data types. It's noteworthy that while default parameters may suit one kind of data well, they might be better suited for another. The comparative analysis sheds light on this aspect: Prophet, with its default parameters, consistently demonstrates strong performance, underscoring the efficacy of its out-of-the-box settings for a range of datasets.



(a) AutoARIMA best Household_combined (b) Prophet best CA_total

Figure 7: Prophet vs AutoArima

These results have significant implications for researchers and practitioners seeking an effective forecasting approach. Incorporating the Prophet model into their decision-making processes can enhance the accuracy and reliability of predictions, particularly in cases where the model fit may be less optimal.

Furthermore, Prophet showcased an additional advantage in terms of computation time. The time required for Prophet was at least 10 times less than that of AutoArima. This factor of computational efficiency enhances the overall effectiveness of Prophet as a forecasting model.

5.2 Deep Learning models

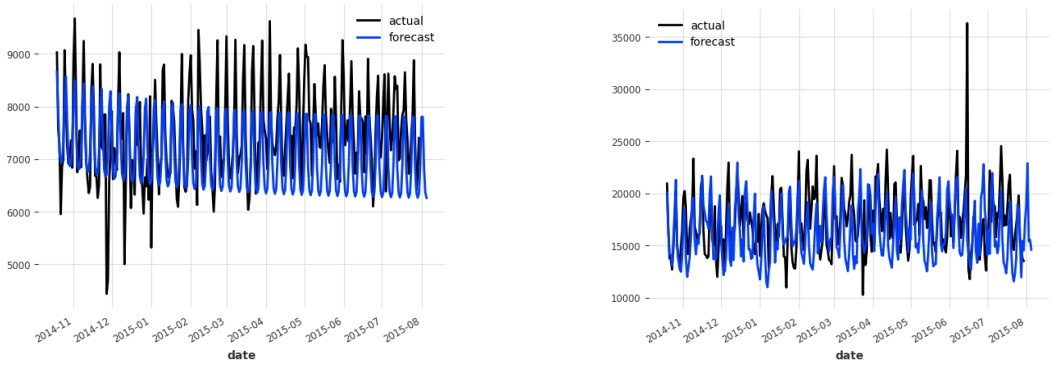
The results obtained from the deep learning models, RNN-LSTM and N-BEATS, on the M5 dataset is quite interesting. Both the RNN-LSTM and N-BEATS models demonstrated suboptimal fits with the data. RNN-LSTM exhibited a marginal advantage over N-BEATS. The performance profiles of these models across the 56 evaluated time-series offer insights of significance. Notably, N-BEATS excelled in only 12 instances, whereas RNN-LSTM showcased superior performance in 17 instances where both models achieved positive R2 scores. Further delving into the specifics of MAPE and SMAPE scores, RNN-LSTM delivered lower values compared to N-BEATS. This emphasizes

RNN-LSTM's potential superiority in forecasting accuracy for this particular dataset with the default parameters.

Table 8: RNN vs N-BEATS. For detailed results see table 17 and table 18

RNN-LSTM MODEL					N-BEATS MODEL			
TIME-SERIES NAME	**MAPE**	**SMAPE**	**R^2**	**TIME (s)**	**MAPE**	**SMAPE**	**R^2**	**TIME (s)**
CA_4	**7.22**	**7.38**	**0.38**	127.06	11.29	10.53	-0.25	705.57
CA_4Food_sale	8.10	7.85	0.35	108.73	10.66	10.64	-0.06	657.54
CA_total	7.98	8.08	0.66	130.89	**8.85**	**9.09**	**0.58**	691.5
TX_full_state_Food_sale	**9.40**	**9.58**	**0.4**	125.20	**9.20**	**9.17**	**0.38**	701.95
Household_combined	12.40	13.22	0.39	127.15	12.09	12.92	0.38	696.71

While examining individual case studies, such as the performance of RNN-LSTM and N-BEATS in the context of CA_4, has the best overall score. Here, RNN-LSTM's MAPE score significantly outperforms N-BEATS, indicating the context-dependence of model effectiveness. Whereas even where N-BEATS has it's best score i.e CA_total,where N-BEATS scored 8.85, RNN-LSTM achieved a better MAPE score of 8.08. This contextual variation underscores the intricate interplay between model performance and dataset characteristics. This shows that in the overall context of the RNN performs better with it's default parameters



(a) RNN best best CA_4

(b) N-BEATS best CA_total

Figure 8: RNN vs N-BEATS best

RNN-LSTM's performance seems to hinge more acutely on the quality of data fitting. Its reliance on capturing temporal dependencies and intricate data patterns is indicative of its nature as a deep learning model sensitive to nuances in the data. This highlights the model's reliance on capturing temporal dependencies within the dataset to generate accurate predictions. This assertion, grounded in data evidence, avoids unfounded attributions.

N-BEATS displays the ability to deliver competitive performance even with less-than-ideal fits. This characteristic underscores its inherent modeling capacity, enabling accurate forecasts even in scenarios with suboptimal model-data alignment. Furthermore, the results underscore the imperative of hyperparameter tuning for N-BEATS. R,

N-BEATS exhibited better initial performance in 12 instances compared to RNN-LSTM's 17 instances.

In essence, the analysis traverses the intricate landscape of model selection, parameterization, and performance.

To further see if N-BEATS is perfoms well when the hyperparameters are optimized, the time-series from table are hypertuned. With optimization N-BEATS is better and performs like state of the art all the time series get better than the original results obtained in 8, in fact it performs better than RNN-LSTM with the hyperparamters optimized. This table 9 shows N-BEATS perfomance with optimization and how well it does with the scores with hyperparameters optimized

The study delved into optimizing the N-BEATS model's performance by tuning its hyperparameters. This empirical process was carried out on the time-series data previously analyzed in table 8 to gauge the model's potential under enhanced configurations. The results revealed a significant boost in forecasting accuracy across all instances, showcasing the model's untapped capabilities when equipped with optimal parameters. Particularly noteworthy was N-BEATS' consistent outperformance against both its original configuration and the optimized RNN-LSTM model, highlighting its potential as a state-of-the-art forecasting solution.

Table 9 summarizes the evaluation scores from the optimized N-BEATS and RNN-LSTM models, illustrating N-BEATS' consistent performance improvement and superiority over RNN-LSTM in most cases.It underscores the importance of hyperparameter tuning in unlocking the full potential of forecasting models. By maintaining continuity with prior analysis and substantiating findings with empirical evidence, the study contributes a robust understanding of the intricate relationship between model architecture, hyperparameters, and forecasting precision.

Table 9: RNN-LSTM with hyperparamters optimized

	RNN-LSTM MODEL			**N-BEATS MODEL**		
TIME-SERIES NAME	**MAPE**	**SMAPE**	**R^2**	**MAPE**	**SMAPE**	**R^2**
CA_4Food_sale	9.96	10.45	0.04	8.87	8.40	0.38
CA_4	10.11	10.23	0.02	7.78	7.34	0.40
CA_total	6.86	6.77	0.77	6.48	6.21	0.79
CA_full_state_food _sale	18.35	17.89	-0.88	9.57	8.98	0.59
TX_full_state_Food_sale	8.00	7.86	0.61	7.46	7.54	0.68
Household_combined	9.39	9.37	0.73	8.50	8.91	0.61

5.2.1 Deep Learning model vs Classical Models

From the analysis of table 7 and Table 8, it appears that the classical TSF models maintain superiority in terms of predictions when compared to their deep learning counterparts. On average, the classical models perform better in various metrics and are also more time-efficient.This could be due to

- Data Availability and Size: Classical TSF models, such as AutoARIMA and Prophet, have been extensively used and developed over the years. As a result, there might

be more data available for these models, enabling them to learn and generalize well from historical patterns.

- Simplicity and Interpretability: Classical models often have simpler architectures and are easier to interpret. This simplicity can be advantageous when the dataset is relatively small or when the forecasting task requires human understanding and interpretability.
- Domain-specific Knowledge: Classical models often incorporate domain-specific knowledge and assumptions, which can be beneficial in scenarios where certain characteristics of the time series are known in advance. Deep learning models, on the other hand, are more data-driven and may require a larger dataset to learn these characteristics effectively.
- Model Robustness: Classical models have been tested and refined over time, making them robust and reliable for various TSF tasks. Deep learning models may require more experimentation and tuning to achieve optimal performance for specific datasets.
- Training Complexity: Deep learning models typically have a higher number of parameters and may require more computational resources and time to train. In contrast, classical models like AutoARIMA and Prophet are often quicker to train and can provide satisfactory results with less computational overhead.
- Data Quality and Noise: Deep learning models can be sensitive to noisy data, and a relatively smaller dataset with noise may impact their performance. Classical models may be more resilient to noise and able to handle smaller datasets more effectively.

Overall, while deep learning models like N-BEATS and RNN-LSTM show great promise and have achieved remarkable results in various domains, classical models like AutoARIMA and Prophet still possess an advantage in terms of prediction accuracy and time efficiency, particularly in certain TSF scenarios with limited data or a preference for interpretability. It is essential to carefully consider the characteristics of the dataset and the specific forecasting requirements when choosing between deep learning and classical models for TSF tasks.

5.2.2 Model-Based Transfer Learning

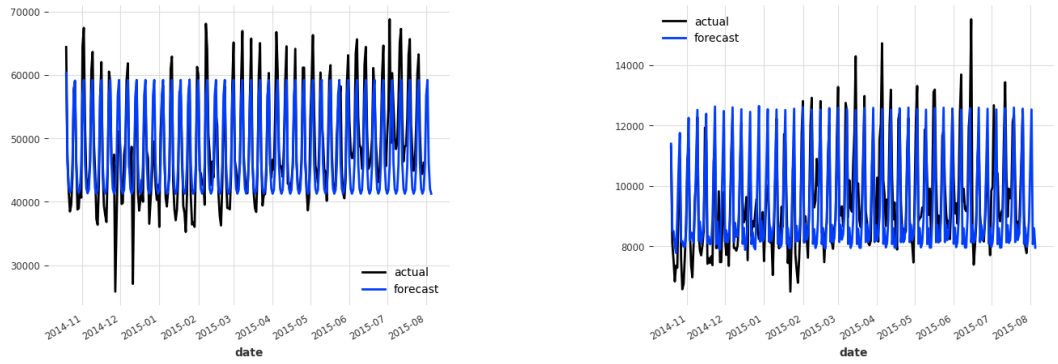
Encouraging results are observed in model-based learning for both N-BEATS and RNN-LSTM, especially when some part of the test dataset is used for training, highlight the potential of leveraging context and additional information to improve the models' performance. Among the time-series given to the models For N-BEATS more than half (20 out of 30) of the time-series were positively fitted, indicating that N-BEATS effectively adapted to the underlying patterns and relationships in the data when provided with relevant context. Several time-series here achieved comparable or even better scores to those obtained without transfer learning, highlighting the potential of transfer learning to enhance forecasting capabilities even with limited data. This result underscores the value of leveraging knowledge from diverse domains to ensure robust and accurate forecasts.

RNN-LSTM was similar to N-BEATS, over half (20 out of 30) of the time-series demonstrated positive fitting, showing that RNN-LSTM also benefited from incorporating context in its training process. As with N-BEATS, certain time-series in RNN-LSTM exhibited performance on par with or close to their non-transfer learning counterparts.

Table 10: Model Based learning RNN-LSTM vs N-BEATS models.For detailed results see table 21 and table 22

		RNN-LSTM MODEL				N-BEATS MODEL			
TRAINED ON	**TESTED ON**	**MAPE**	**SMAPE**	**R^2**	**TIME (s)**	**MAPE**	**SMAPE**	**R^2**	**TIME (s)**
TX _total	CA_total	**7.82**	**7.89**	**0.67**	194.7	**9.10**	**9.10**	**0.57**	1562.75
WI _full_state _Hobby_sale	CA_ full_state _Food_sale	**8.38**	**8.15**	**0.63**	162.91	9.17	8.73	0.55	2356.27
CA_ 2	TX_1	10.59	11.33	0.36	163.8	**9.10**	**9.31**	**0.51**	1839.71
TX_full _Househ_sale	CA_full_state _Hobby_sale	11.96	11.58	0.2	163.93	**9.26**	**8.88**	**0.53**	2193.87

When the model is provided with additional context or partial information about the test dataset during training, it can better adapt its internal parameters and representations to capture the data's specific characteristics. This additional context acts as a form of regularization, guiding the model to generalize better to unseen data.



(a) RNN best best TX_total to CA_total

(b) N-BEATS best CA_2 to TX_1

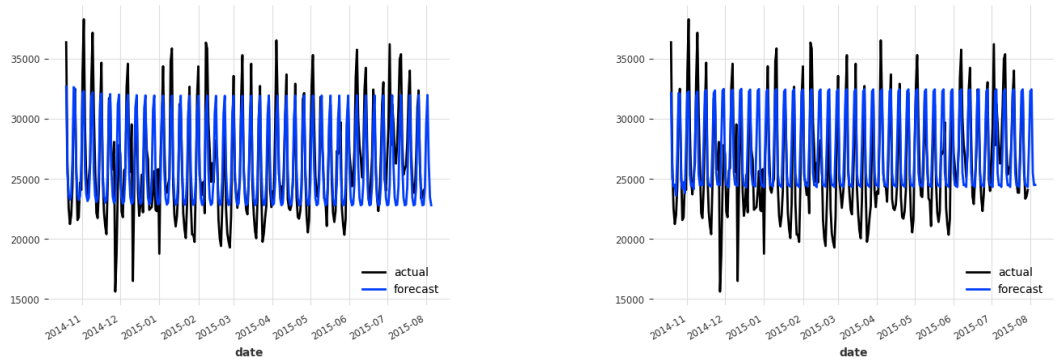
Figure 9: RNN vs N-BEATS best model based learning

The promising results of model-based learning suggest that leveraging context and transfer learning can be an effective strategy for enhancing the performance of TSF models like N-BEATS and RNN-LSTM. It showcases the potential of these models to achieve strong predictive capabilities without extensive manual tuning, making them valuable tools for various real-world forecasting applications.

5.2.3 Zero-Shot Transfer Learning

The results for zero-shot learning in both RNN-LSTM and N-BEATS were unexpected, as only a limited number of hierarchical levels out of the 30 time-series as seen in 6 attempted were well-fitted to the data, and even fewer yielded good evaluation scores. Specifically, out of the 30 different hierarchical levels, RNN-LSTM got only 8 out of 30 hierarchical levels were well-fitted to the data.

And N-BEATS got only 5 out of 30 hierarchical levels were well-fitted to the data. Moreover, among these well-fitted hierarchical levels, only a couple of them demonstrated good evaluation scores, as indicated in the table



(a) RNN best best TX_full_stateHouseh_sale to CA_full_state_Food_sale

(b) N-BEATS best TX_full_stateHouseh_sale to CA_full_state_Food_sale

Figure 10: RNN vs N-BEATS best zero shot learning

Table 12: Zero_shot learning RNN-LSTM vs N-BEATS models.For detailed results see table 19 and table 20

		RNN-LSTM MODEL				N-BEATS MODEL			
TRAINED ON	**TESTED ON**	**MAPE**	**SMAPE**	**R^2**	**TIME (s)**	**MAPE**	**SMAPE**	**R^2**	**TIME (s)**
Hobby_ combined	Food_ combined	11.07	10.55	0.34	152.78	30.9	38.2	-4.13	3006
Household_ combined	Food_ combined	9.67	9.58	0.46	155.99	10.46	10.8	0.32	1710
TX_full state _Househ_sale	CA_full state _Food_sale	**8.01**	**7.99**	**0.64**	157.38	**9.5**	**9.01**	**0.55**	1296
TX_full state _Househ_sale	WI_full_state _Hobby_sale	**16.94**	**18.15**	**-0.16**	153.0	**12.71**	**12.57**	**0.3**	1434

The varied performance of RNN-LSTM and N-BEATS in zero-shot learning can be attributed to their different architectural characteristics and how they approach handling sequential data. While both models are capable of processing sequential information, they do so in distinct ways, which may impact their performance in zero-shot learning scenarios.

RNN-LSTM, as a recurrent neural network, inherently possesses sequential memory, which allows it to retain information about past time points and effectively capture temporal patterns and hierarchies in sequential data. In zero-shot learning, where the model must generalize to unseen contexts, RNN-LSTM's ability to remember long-term dependencies becomes advantageous. Its recurrent connections enable it to maintain contextual information from earlier time steps, enabling accurate predictions without direct access to future data.

Conversely, N-BEATS relies on fully connected layers and does not have the recurrent memory of RNN-LSTM. Although N-BEATS is designed for TSF and can capture patterns in sequential data through its stacking of basis functions, it may not inherently handle complex temporal relationships and long-term dependencies as effectively as RNN-LSTM.

5.3 Classical models vs Transfer learning based Deep learning models

The model-based transfer learning approach has emerged as the most prominent technique in this research, exhibiting good performance compared to the best classical model, Prophet. After training the Prophet model using the transfer learning data, surprising results were observed. The model-based approach consistently outperformed Prophet in more than half of the 15 time series studied. This compelling finding demonstrates the effectiveness of model-based transfer learning in enhancing forecasting accuracy across various time series domains.

The table 14 lists some of the prominent instances where model-based transfer learning showed improved perfomance over Prophet. These results compared to the potential of this approach in improving forecasting outcomes and its ability to leverage knowledge from a source domain to better capture unique patterns and dynamics in the target time series. The strong performance of the model-based approach in a significant number of cases signifies its practical significance and positions it as a promising technique for accurate and efficient TSF in real-world applications.

Table 14: Prophet vs with model based RNN and N-BEATS time-series.For detailed results see table 23

	Prophet			**RNN-LSTM**			**N-BEATS**		
TIME-SERIES NAME	**MAPE**	**SMAPE**	**R^2**	**MAPE**	**SMAPE**	**R^2**	**MAPE**	**SMAPE**	**R^2**
CA_4 Househ _sale	45.51	35.59	-9.35	13.78	13.54	-0.25	11.15	11.35	0.17
TX_1Food _sale	17.65	19.82	-0.24	12.06	12.07	0.25	11.5	11.72	0.27
CA_total	7.59	7.24	0.76	7.82	7.89	0.67	13.55	14.68	0.07
Food_ combined	9.6	9.57	0.58	9.35	9.54	0.47	12.59	12.76	0.16
CA_full_ state_ Food_sale	**9.45**	**8.88**	0.6	**8.38**	**8.15**	0.63	**9.17**	**8.73**	**0.55**

6 Discussion

This thesis delved into a comprehensive analysis of two prominent TSF models, N-BEATS and RNN-LSTM, with a primary focus on evaluating their performance across various scenarios and datasets. While classical models, namely Prophet and AutoArima were used , to gauge the effectiveness of the subsequent deep learning models. Subsequently, N-BEATS and RNN-LSTM were thoroughly examined under different experimental setups, shedding light on their strengths and weaknesses in diverse forecasting contexts.

- **Effectiveness of Classical Models:** The initial evaluation of classical models was instrumental in setting a foundation for the subsequent analysis. The findings revealed that Prophet consistently outperformed AutoArima in the majority of cases, demonstrating its superior forecasting accuracy.

  The inherent modeling capabilities of Prophet, along with its flexibility in capturing seasonal and trend patterns, played a pivotal role in achieving better evaluation metrics, such as MAE, MAPE, and SMAPE.

- **Comparing N-BEATS and RNN-LSTM:** The subsequent analysis involved an in-depth comparison between N-BEATS and RNN-LSTM, two cutting-edge deep learning models for TSF. Both models showcased distinctive characteristics, excelling in specific scenarios. N-BEATS demonstrated remarkable robustness by delivering good evaluation scores even under suboptimal fitting conditions In contrast, RNN-LSTM exhibited a heavy reliance on proper fitting to attain good evaluation metrics. The impact of dataset complexity and hierarchical structures was evident in influencing the models performance. When tuned to the proper hyperparameters N-BEATS shows its real strength, since it heavily relies on the parameters. With optimization RNN doesn't show any significant improvements infact it becomes worse it some cases, whereas N-BEATS consistently gives better results

- **Transfer Learning:** An integral part of the study focused on exploring the potential of model-based transfer learning and zero-shot learning in elevating the forecasting models' performance. The results highlighted both the techinques are worth a shot for professionals as well as researchers. Zero-shot learning is possible and can be done but to have a positive results or make proper forecasting hyper-parameters will have to be optimized.Zero-shot learning approach is still a relatively novel and developing area within TSF. While it holds great promise for knowledge transfer and efficiency, it is still in its early stages of exploration. As such, there might be inherent limitations and challenges that need to be addressed through further research and advancements in the field.
  Model-based on the other hand is the go to method for transfer learning.Model-based transfer learning proves to be a highly effective and reliable method for TSF using transfer learninng .Model-based transfer learning leverages pre-existing knowledge obtained from training on a source time-series to enhance the learning process for a target time-series. By utilizing a model that has already been trained on a similar dataset, the target model can benefit from the patterns, temporal dependencies, and underlying structures learned from the source data. This enables the target model to start with a better initialization, reducing the amount of data needed for training and potentially speeding up convergence. Model-based transfer learning allows for knowledge transfer and adaptation to target data through fine-tuning, making it a robust technique for handling complexities and capturing

relationships in time-series data. Model-based transfer learning outperforms zero-shot learning in TSF due to its ability to leverage pre-existing knowledge, adapt to target data through fine-tuning, handle complexities in time-series data, and enable a more guided and controlled learning process. The successful application of model-based transfer learning in this thesis underscores its potential as a powerful technique for improving forecasting performance and knowledge transfer across related time-series domains. The findings of the study underscore the prominence and effectiveness of the model-based transfer learning approach in TSF. This technique showcases its superiority over the best classical model, Prophet, which highlights its ability to leverage knowledge from a source domain and adapt to unique patterns in the target time series. The successful performance of the model-based transfer learning approach in a significant number of cases indicates its potential to enhance forecasting accuracy and efficiency in various time series domains.

In scenarios where data is scarce or unrelated, model-based transfer learning stands out as a superior approach compared to zero-shot learning. It empowers forecasting models to leverage existing knowledge and adapt it to the target domain, enabling more guided and controlled learning. As demonstrated in this thesis, model-based transfer learning can yield impressive results, making it a go-to method for practitioners and researchers seeking to enhance forecasting performance in challenging settings.

Transfer learning in TSF holds great promise and opens up new opportunities for improving forecasting accuracy, even when data is limited or unrelated. Both model-based transfer learning and zero-shot learning have their merits, but model-based transfer learning emerges as the more effective and robust technique, showcasing its potential to revolutionize the field of TSF and knowledge transfer across diverse time-series domains.

7 Conclusions

In conclusion, the thesis presents a comprehensive investigation into the effectiveness of different TSF models like Prophet, AutoArima and transfer learning within the time-series domain, particularly focusing on deep learning models like RNN and N-BEATS. The empirical evidence gathered from the experiments strongly supports the robust applicability of transfer learning, showcasing its potential to enhance forecasting accuracy and generalization.

Prophet's superior predictive capabilities, and its computational efficiency, it can be concluded that Prophet is the best model for forecasting the target variable. Its robust performance, even in cases where it is not as well-fitted, further highlights its effectiveness in generating accurate predictions. Future forecasting tasks in similar contexts are recommended to utilize Prophet due to its superior performance and efficiency.

Despite its initial underperformance compared to default RNN-LSTM, N-BEATS demonstrates its true potential through hyperparameter tuning, consistently outperforming both its own default configuration and RNN-LSTM. This highlights the significance of optimization in N-BEATS and highlights it as a powerful and competitive choice for accurate time-series forecasting.

While the success rate of zero-shot learning appears modest, with 8/30 for RNN and 5/30 for N-BEATS, it is important to emphasize that this result does not undermine the promise of transfer learning. Instead, it highlights the need for careful hyperparameter tuning and fine-tuning of models to fully capitalize on the potential of this approach.

The thesis also uncovers an intriguing finding that model-based transfer learning outperforms other transfer learning methods when handling time-series data. Both RNN and N-BEATS exhibited success in 20 out of the 30 time-series when provided with context and supplementary information during training. This result marks a remarkable advancement from the model-based learning approach and emphasizes the significance of incorporating prior knowledge and domain-specific context in the transfer learning process.

Based on the findings from this thesis, practitioners and researchers looking to leverage transfer learning on time-series data are strongly encouraged to adopt the model-based approach. This method showcases the most promising strategy for improving forecasting accuracy, making deep learning models like RNN and N-BEATS more adaptable and flexible in handling complex time-series datasets.

Furthermore, the thesis opens up exciting avenues for future research. One such direction involves exploring the application of transfer learning in multivariate TSF, as this could further enhance the models' ability to capture intricate interdependencies between multiple variables. Additionally, fine-tuning hyperparameters and exploring real-world applications can offer valuable insights into the practical implications of transfer learning in various domains.

This thesis underscores the importance of transfer learning as a powerful tool in for TSF which can lead to significant advancements in the TSF domain. By adopting the model-based approach and exploring various extensions and optimizations, researchers can unlock the full potential of transfer learning, making it a very effective method for more accurate and reliable time-series forecasts.It offers valuable insights into the advantages of leveraging pre-trained models for improved time-series predictions, paving the way for future research and practical applications in real-world forecasting scenarios. Moreover, the promise of model-based learning and transfer learning unveils exciting prospects for improving forecasting accuracy in real-world settings. By integrating relevant context and harnessing transfer learning techniques, forecasting models can adapt adeptly to varying data conditions and achieve heightened generalization. This aspect holds particular importance in scenarios where obtaining large amounts of labeled data for training poses challenges.

8 Contribution

This research makes the following key contributions to the field of TSF and transfer learning:

- **Comprehensive Comparison of Models:** The research conducts a thorough comparison of Deep learning models with classical models offering insights into how different models impact forecasting performance.
- **Focus on N-BEATS Model:** While previous research has primarily concentrated on RNN-LSTM models, this work uniquely investigates the effectiveness of transfer learning using the N-BEATS model for improved forecasting accuracy.
- **Investigation of Transfer Learning:** This study examines the applicability of transfer learning in TSF, particularly in scenarios with limited data availability in industrial applications.
- **Focus zero-shot transfer learning:** While previous research has concentrated on other methods of transfer learning , this work uniquely investigates the effectiveness of zero-shot transfer learning using the RNN-LSTM as well as N-BEATS models.
- **Focus different method of model-based transfer learning:** While previous research has concentrated on just using freezing weights for model-based transfer learning , this work uniquely investigates the effectiveness freezing parts of dataset transfer learning in time-series using the RNN-LSTM as well as N-BEATS models.
- **Performance Evaluation of model-based transfer learning with Classical model:** Through experimentation, the study evaluates the performance of model-based transfer transfer learning approaches with prophet model, providing empirical evidence of the benefits and limitations of these techniques.
- **Practical Applications:** The findings of this research highlight the potential of transfer learning for enhancing forecasting accuracy and efficiency in industrial contexts, offering practical insights for industries seeking to optimize their forecasting processes.

  Overall, this thesis enhances the understanding of transfer learning's role in TSF and contributes valuable knowledge to assist industries in making informed decisions about leveraging transfer learning techniques for improved forecasting outcomes.

9 Future Work

Despite the valuable insights gained from this research, certain limitations warrant consideration. The evaluation was conducted on specific datasets, potentially limiting the generalizability of the results to all types of time-series data. As an avenue for future research, the incorporation of multivariate TSF holds great potential for further advancing forecasting accuracy and capturing complex interdependencies between multiple variables. Currently, the analysis primarily focused on univariate time-series data, and expanding the investigation to include multivariate time-series datasets would provide valuable insights into real-world applications where multiple variables influence the target variable's behavior.

- **Multivariate TSF:** Exploring multivariate TSF entails analyzing datasets with multiple related variables that interact with one another. By incorporating additional features and interrelationships, models can capture more nuanced patterns and dependencies, resulting in more accurate and comprehensive forecasts. The models such as N-BEATS and RNN-LSTM can be extended to accommodate multivariate inputs, and their performance can be evaluated on diverse and complex datasets.
- **Transfer Learning for Multivariate Time-Series:** Applying transfer learning techniques to multivariate TSF is an intriguing direction for future research. Similar to the model-based learning explored in this thesis, transferring knowledge from one domain or dataset to another could significantly improve forecasting performance for multivariate time-series. The success of transfer learning in the context of univariate TSF encourages its application in multivariate settings, where the interactions and dependencies between variables can be leveraged to enhance generalization.
- **Further Fine-tuning Hyperparameters:** In both univariate and multivariate TSF, thorough hyperparameter optimization can significantly impact model performance. Further fine-tuning the hyperparameters of the deep learning models, especially when dealing with complex multivariate datasets, can unlock their full potential and yield superior forecasting accuracy.

References

[1] Casper Solheim Bojer and Jens Peder Meldgaard. Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2):587–603, 2021.

[2] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, and G.M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2015.

[3] Stevo Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44, 09 2020.

[4] Pieter Cawood and Terence Van Zyl. Evaluating state-of-the-art, forecasting ensembles and meta-learning strategies for model fusion. *Forecasting*, 4(3):732–751, 2022.

[5] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, apr 2006.

[6] Yuan Gao, Yingjun Ruan, Chengkuan Fang, and Shuai Yin. Deep learning and transfer learning models of energy consumption forecasting for a building with poor information data. *Energy and Buildings*, 223:110156, 2020.

[7] Everette S. Gardner. Exponential smoothing: The state of the art. *Journal of Forecasting*, 4, 1985.

[8] Phillip G. Gould, Anne B. Koehler, J. Keith Ord, Ralph D. Snyder, Rob J. Hyndman, and Farshid Vahid-Araghi. Forecasting time series with multiple seasonal patterns. *European Journal of Operational Research*, 191(1):207–222, 2008.

[9] James Douglas Hamilton. *Time series analysis*. Princeton university press, 2020.

[10] Qi-Qiao He, Patrick Cheong-Iao Pang, and Yain-Whar Si. Transfer learning for financial time series forecasting. In Abhaya C. Nayak and Alok Sharma, editors, *PRICAI 2019: Trends in Artificial Intelligence*, pages 24–36, Cham, 2019. Springer International Publishing.

[11] Julien Herzen, Francesco LÃ¤ssig, Samuele Giuliano Piazzetta, Thomas Neuer, LÃ©o Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan KoÅcisz, Dennis Bader, FrÃ©dÃ©rick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and GaÃl Grosch. Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124):1–6, 2022.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[13] Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, 27(3):1–22, 2008.

[14] Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 2nd edition, 2018.

[15] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, 2009.

[16] Mathieu Lepot, Jean-Baptiste Aubin, and François H.L.R. Clemens. Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. *Water*, 9(10), 2017.

[17] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209, feb 2021.

[18] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022. Special Issue: M5 competition.

[19] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *ArXiv*, abs/1707.05589, 2017.

[20] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020.

[21] Bernardo P'erez Orozco and Stephen J. Roberts. Zero-shot and few-shot time series forecasting with ordinal regression recurrent neural networks. In *The European Symposium on Artificial Neural Networks*, 2020.

[22] Shanoli Samui Pal and Samarjit Kar. Fuzzy transfer learning in time series forecasting for stock market prices. *Soft Computing*, 26(14):6941–6952, Jul 2022.

[23] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[24] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965, 2019.

[25] Sudheer Babu Punuri, Sanjay Kumar Kuanar, Manjur Kolhar, Tusar Kanti Mishra, Abdalla Alameen, Hitesh Mohapatra, and Soumya Ranjan Mishra. Efficient net-xgboost: An implementation for facial emotion recognition using transfer learning. *Mathematics*, 11(3), 2023.

[26] Bibhuti Bhusan Sahoo, Ramakar Jha, Anshuman Singh, and Deepak Kumar. Long short-term memory (lstm) recurrent neural network for low-flow hydrological time series forecasting. *Acta Geophysica*, 67(5):1471–1481, 2019.

[27] Richard Socher, Milind Ganjoo, Christopher D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013.

[28] Sean J. Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[29] José Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. Deep learning for time series forecasting: A survey. *Big Data*, 9, 12 2020.

[30] Jan N Van Rijn and Frank Hutter. Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2367–2376, 2018.

[31] Gilbert Walker. On Periodicity in Series of Related Terms. *Proceedings of the Royal Society of London Series A*, 131(818):518–532, June 1931.

[32] Xuanzheng Wang, Changwang Li, Chengqi Yi, Xinan Xu, Jiandong Wang, and Youhui Zhang. Ecoforecast: An interpretable data-driven approach for short-term macroeconomic forecasting using n-beats neural network. *Engineering Applications of Artificial Intelligence*, 114:105072, 2022.

[33] Manuel Weber, Maximilian Auch, Christoph Doblander, Peter Mandl, and Hansarno Jacobsen. Transfer learning with time series data: A systematic mapping study. *IEEE Access*, 9:165409–165432, 12 2021.

[34] Ronald J. Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, 06 1989.

[35] Rui Ye and Qun Dai. Implementing transfer learning across different datasets for time series forecasting. *Pattern Recognition*, 109:107617, 2021.

[36] G. Udny Yule. Why do we sometimes get nonsense-correlations between time-series?–a study in sampling and the nature of time-series. *Journal of the Royal Statistical Society*, 89(1):1–63, 1926.

[37] Peter Zhang, Eddy Patuwo, and Michael Hu. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14:35–62, 03 1998.

[38] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive on transfer learning. *Proceedings of the Institute of Radio Engineers*, 109(1):43–76, January 2021.

A Results in detailed for normal models

In this section presents a comprehensive overview of all the results obtained from the conducted experiments. The detailed information encompasses the performance metrics, such as Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (SMAPE), and R-squared (R^2), for each model.

NOTE: The order of R^2, MAPE and SMAPE is changed here it doesn't signfy any change of priority of the mertics it's MAPE, SMAPE and R^2.

A.1 AutoArima

An automated TSF model that is part of the ARIMA (AutoRegressive Integrated Moving Average) family. It uses an intelligent algorithm to automatically select the optimal order of differencing (d), autoregressive (p), and moving average (q) terms based on the data's characteristics. By automatically determining the best combination of these parameters, AutoARIMA simplifies the forecasting process, making it accessible to users without extensive expertise in time-series modeling. The model's ability to automatically handle seasonal and trend components makes it a popular choice for quick and accurate TSF tasks. Below are the results for AutoArima. R^2, MAPE, SMAPE have been used as the evaluation metrics. The results have been presented in the ascending order of the MAPE score.

Table 15: Detailed results of AutoArima Model

No	TIME SERIES NAME	R^2	MAPE	SMAPE	TIME (s)
1	Hobby_combined	0.71588	4.50278	4.52851	84.4827
2	Household_combined	0.83304	4.59979	4.65447	77.28763
3	CA_full_state_Hobby_sale	0.5807	5.44943	5.35141	88.33441
4	CA_4	0.43802	5.67666	5.6254	111.882
5	CA_1	0.80681	5.86243	5.94787	93.53639
6	CA_total	0.7158	6.25144	6.46847	94.46475
7	TX-full_state_Househ_sale	0.73494	6.33169	6.07042	79.76967
8	CA_3	0.59054	6.47905	6.6479	70.7475
9	WI_full_state_Househ_sale	0.62357	6.66324	6.864	91.85632
10	CA_4Househ_sale	0.18565	7.69172	7.85174	105.5147
11	TX_3Househ_sale	0.54446	7.88028	7.87133	101.8543
12	Total_sale	0.48844	7.9029	8.32234	85.84386
13	WI_full_state_Hobby_sale	0.47738	8.20651	7.80803	83.21339
14	CA_3Food_sale	0.38953	8.23729	8.5539	35.84086
15	TX_2	0.48958	8.31152	8.2939	74.62666
16	CA_1Food_sale	0.68634	8.37937	8.62776	112.6318
17	TX_total	0.45024	8.44553	8.66334	65.94738
18	CA_full_state_Food_sale	0.48709	8.47895	8.91528	114.4139
19	WI_1	0.63412	8.52408	8.58964	85.97526
20	CA_1Hobby_sale	0.54588	8.69653	8.55868	95.83658
21	WI_3Househ_sale	0.56298	8.79277	9.12795	87.36886
22	TX_1Househ_sale	0.53692	9.04687	9.44294	57.32037
23	CA_1Househ_sale	0.69646	9.0763	9.24069	39.45512
24	WI_1Food_sale	0.52362	9.09703	8.97311	86.95212

25	CA_full_state_Househ_sale	0.51752	9.20719	9.85614	71.66782
26	CA_3Hobby_sale	0.0609	9.35154	9.68765	52.01659
27	CA_3Househ_sale	0.31001	9.9983	10.7331	66.54544
28	TX_3	0.3058	10.0678	10.2711	71.83625
29	CA_4Food_sale	-0.1178	10.1229	9.49278	55.2769
30	CA_4Hobby_sale	-0.0161	10.1974	10.1239	11.88748
31	Food_combined	0.14047	10.2934	11.0769	96.88343
32	TX_2Househ_sale	0.52821	10.5035	11.0336	55.70396
33	TX_2Food_sale	0.19252	10.6423	11.254	89.25186
34	WI_1Househ_sale	0.47977	10.6909	11.5667	62.76513
35	TX_1	0.27784	10.7019	11.4862	73.51879
36	CA_2Househ_sale	0.66028	11.1344	11.8761	90.44123
37	TX_2Hobby_sale	0.43517	11.2472	10.736	50.31506
38	CA_2Hobby_sale	0.41535	11.694	11.7613	50.49286
39	TX_full_state_Hobby_sale	0.37461	12.6182	11.7847	81.37191
40	TX_1Hobby_sale	-0.0464	12.683	12.9377	10.26024
41	WI_2Househ_sale	0.02558	13.1662	13.949	32.63976
42	WI_3	-0.0542	13.1721	14.4519	57.24171
43	TX_3Food_sale	0.1092	13.2228	13.6276	48.71504
44	WI_total	-0.0488	13.2502	12.7838	15.43657
45	WI_1Hobby_sale	0.49655	13.5092	12.83	76.23592
46	TX_1Food_sale	-0.0362	13.7284	15.0813	62.35116
47	TX_full_state_Food_sale	-0.009	13.7563	13.4695	31.69085
48	WI_2Hobby_sale	-0.0037	13.8274	13.4475	29.54223
49	WI_3Hobby_sale	0.28765	13.9987	13.5361	71.83498
50	WI_full_state_Food_sale	-0.1531	15.0754	16.4585	78.07876
51	WI_2	-0.5573	16.3744	18.392	53.77023
52	CA_2	0.18093	16.3929	18.3034	92.77944
53	WI_3Food_sale	-0.3234	17.3977	19.6228	73.41984
54	TX_3Hobby_sale	0.00524	18.7519	17.6215	34.56707
55	WI_2Food_sale	-0.5859	21.2888	24.6979	67.80978
56	CA_2Food_sale	-0.3483	22.9489	27.1638	56.80171

A.2 Prophet

Prophet is a TSF model created by facebook designed to handle time-series data with strong seasonal patterns and irregularities. Prophet combines a decomposable time-series model with a set of user-defined seasonal components, allowing it to capture various seasonalities, holidays, and other recurring events in the data. One of its notable features is its ability to handle missing data and outliers effectively, making it robust in real-world scenarios Below are the results for Prophet. R^2, MAPE, SMAPE have been used as the evaluation metrics. The results have been presented in the ascending order of the MAPE score.

Table 16: Detailed Results Prophet Model

No	TIME SERIES NAME	R^2	MAPE	SMAPE	TIME (s)
1	CA_total	0.788715	4.952695	5.06875	0.363561
2	CA_full_state_Hobby_sale	0.559423	5.953144	5.829172	0.421101
3	CA_4	0.341106	6.225255	6.067383	0.377633
4	Household_combined	0.718158	5.920006	6.181023	0.382055
5	CA_full_state_Food_sale	0.719448	6.227759	6.226539	0.438288
6	CA_3	0.622595	6.3295	6.379581	0.38761
7	CA_1	0.761724	6.691583	6.487925	0.442431
8	CA_4Househ_sale	0.359668	6.668757	6.774604	0.310705
9	CA_1Househ_sale	0.781132	7.030442	7.02402	0.422428
10	TX_1Househ_sale	0.695803	6.843002	7.027202	0.323343
11	Total_sale	0.638125	6.977922	7.045904	0.372602
12	TX-full_state_Househ_sale	0.670156	6.92566	7.066283	0.360049
13	CA_full_state_Househ_sale	0.702455	6.860998	7.214876	0.29689
14	TX_1	0.553173	7.554889	7.516515	0.327981
15	Hobby_combined	0.461322	8.078232	7.789649	0.70122
16	WI_full_state_Hobby_sale	0.556852	8.406109	8.136832	0.296226
17	WI_1Househ_sale	0.689476	8.297998	8.201786	0.303757
18	WI_full_state_Househ_sale	0.494937	7.781059	8.228878	0.394194
19	WI_1	0.628511	8.753927	8.442125	0.395628
20	CA_3Food_sale	0.449619	8.628278	8.446116	0.308298
21	CA_1Food_sale	0.657367	8.839975	8.449203	0.29659
22	CA_3Hobby_sale	0.189065	8.766398	8.492831	0.53098
23	CA_4Food_sale	0.02664	9.34881	8.857988	0.400859
24	CA_3Househ_sale	0.451991	8.390807	8.873399	0.286021
25	TX_3Househ_sale	0.432166	8.76367	8.976992	0.370426
26	TX_2Househ_sale	0.611207	8.823302	8.978433	0.311534
27	TX_total	0.392859	9.207782	9.061344	0.325328
28	TX_2	0.389148	9.59879	9.133341	0.445289
29	CA_4Hobby_sale	0.059801	9.333179	9.523822	0.28947
30	TX_1Hobby_sale	0.411973	10.19067	9.798561	0.389845
31	Food_combined	0.381437	10.19862	9.971897	0.517742
32	WI_1Food_sale	0.458168	10.55445	10.04494	0.387321
33	CA_1Hobby_sale	0.385316	10.93179	10.32759	0.388235
34	WI_total	0.313964	10.99086	10.88234	0.451796
35	TX_1Food_sale	0.281358	11.40715	10.99316	0.372931
36	WI_1Hobby_sale	0.605725	12.24113	11.32912	0.465218

37	TX_3	0.181576	11.45263	11.43647	0.403341
38	CA_2Hobby_sale	0.472856	11.21245	11.49622	0.33637
39	WI_2Househ_sale	0.234649	11.26473	11.5918	0.324227
40	TX_full_state_Food_sale	0.14056	12.7528	12.38308	0.432811
41	WI_3	0.135117	12.15962	12.89974	0.300654
42	WI_3Househ_sale	0.206297	11.94752	12.92379	0.818465
43	TX_2Food_sale	0.078753	14.00805	13.10421	0.467335
44	WI_3Hobby_sale	0.258158	13.60577	13.26525	0.348948
45	TX_2Hobby_sale	0.163926	15.06818	13.30123	0.474188
46	WI_2Hobby_sale	0.079436	14.30299	13.31612	0.286644
47	TX_full_state_Hobby_sale	0.159957	14.62091	13.43338	0.483078
48	CA_2Househ_sale	0.635332	12.84515	13.86716	0.359873
49	TX_3Food_sale	-0.00268	15.61547	15.26305	0.336075
50	WI_full_state_Food_sale	0.080861	16.28047	15.77096	0.293934
51	WI_2	-0.11302	17.67366	16.90272	0.349955
52	WI_3Food_sale	0.025472	16.76165	17.48353	0.404247
53	TX_3Hobby_sale	0.034166	19.9034	18.29801	0.388307
54	CA_2	0.114099	18.3471	20.71828	0.359412
55	WI_2Food_sale	-0.20156	26.63158	24.11357	0.311317
56	CA_2Food_sale	-0.28317	23.57434	27.93092	0.33992

A.3 RNN-LSTM

RNN-LSTM model widely used for TSF. It excels in capturing sequential patterns and dependencies present in time-series data due to its recurrent architecture and memory cells called LSTM units. These LSTM units allow the model to retain and utilize information from earlier time steps, enabling it to learn long-term dependencies effectively. RNN-LSTM is adept at handling irregular and complex temporal patterns, making it suitable for a wide range of time-series tasks. Its ability to learn from historical data and make accurate predictions at future time points has made RNN-LSTM a popular choice for TSF applications in various domains.

Below are the results for RNN-LSTM. R^2, MAPE, SMAPE have been used as the evaluation metrics. The results have been presented in the ascending order of the MAPE score.

Table 17: Detailed results RNN-LSTM model

No	TIME SERIES NAME	R^2	MAPE	SMAPE	TIME (s)
1	CA_4	0.388643	7.221248	7.300368	127.066
2	CA_total	0.669826	7.980882	8.088233	130.8938
3	CA_4Food_sale	0.354828	8.104657	7.856501	108.739
4	CA_full_state_Food_sale	0.630896	8.21515	8.035641	117.8658
5	TX_full_state_Food_sale	0.40068	9.409407	9.584195	125.2063
6	CA_3Food_sale	0.455806	9.674494	9.24722	126.0705
7	WI_1Food_sale	0.590121	10.16529	10.31818	119.3094
8	CA_full_state_Househ_sale	0.667502	10.58466	10.72027	123.5377
9	TX_1Food_sale	0.254123	11.44465	12.1273	127.3835
10	TX_total	0.18361	12.08376	13.01882	129.5359
11	Household_combined	0.394762	12.4081	13.22103	127.1585
12	CA_1Househ_sale	0.554162	13.06596	13.25177	119.8633
13	CA_2Househ_sale	0.781877	13.08371	12.15541	108.2068
14	TX_3	-0.0203	13.69062	14.66707	129.7183
15	WI_full_state_Househ_sale	0.379354	13.70266	14.14399	128.0045
16	Food_combined	-0.22529	13.81673	15.19008	127.7306
17	WI_3	0.112058	13.85051	15.04472	123.5911
18	TX_1	-0.0228	14.27527	15.71975	127.2429
19	CA_2Hobby_sale	0.436396	14.63814	14.07514	109.2901
20	CA_4Househ_sale	-0.49471	14.68536	14.66376	114.8193
21	WI_full_state_Hobby_sale	0.095886	14.82615	16.10614	124.3482
22	TX_1Househ_sale	0.158858	15.21857	16.80161	128.5651
23	TX_3Househ_sale	-0.02651	15.762	16.94109	116.1865
24	WI_3Hobby_sale	0.263912	15.80519	15.57857	122.0185
25	CA_3Househ_sale	0.081572	16.55867	16.01058	128.3259
26	TX_3Food_sale	-0.60018	16.56427	18.58241	112.5084
27	Hobby_combined	-0.52916	16.9831	18.92533	127.6717
28	CA_full_state_Hobby_sale	-0.58344	17.08161	18.88056	120.6734
29	TX_2	0.351723	17.46827	11.76938	127.9381
30	WI_total	-0.56134	17.70377	19.2728	128.0533
31	CA_2	0.063744	17.79921	20.57768	125.8129
32	CA_4Hobby_sale	-1.32831	17.80834	19.77335	111.3197
33	CA_1Food_sale	-0.35427	18.31708	19.96406	115.8828

34	CA_3Hobby_sale	-1.23274	18.77862	21.17483	127.8161
35	CA_1Hobby_sale	-1.02258	20.31583	22.93738	116.966
36	CA_2Food_sale	0.321765	20.73719	20.05517	110.8917
37	Total_sale	-1.39663	21.55097	24.83628	109.5879
38	WI_2Hobby_sale	-1.14198	21.70942	24.70461	128.203
39	TX_2Food_sale	-1.1E-05	21.96943	14.25371	115.9927
40	WI_2	-1.04577	22.75788	24.42265	126.2892
41	WI_1	-0.47598	23.0754	26.51462	126.8143
42	CA_1	-0.76384	23.09561	27.2211	125.0783
43	WI_2Househ_sale	-0.65084	23.29906	24.49943	127.0319
44	TX_full_state_Hobby_sale	-0.79257	24.04229	28.42225	123.1232
45	WI_full_state_Food_sale	-1.24605	24.22218	26.96626	124.2796
46	TX_2Househ_sale	-0.39723	26.03287	25.02377	108.4096
47	WI_3Househ_sale	-0.72377	26.18525	24.51793	121.7871
48	WI_2Food_sale	-0.70176	26.20513	28.76077	127.9111
49	TX_2Hobby_sale	-1.34054	27.32074	32.64041	109.2168
50	TX-full_state_Househ_sale	-2.24667	29.40002	36.25854	124.3257
51	WI_1Hobby_sale	-0.89702	29.65405	29.31208	121.358
52	CA_3	-4.04289	29.97295	37.34307	126.976
53	WI_3Food_sale	-1.48789	31.33938	25.24264	122.2344
54	TX_1Hobby_sale	-1.82802	31.79223	39.36066	128.0362
55	WI_1Househ_sale	-1.37495	33.54008	41.96524	121.112
56	TX_3Hobby_sale	-2.14328	37.43217	49.0619	116.613

A.4 N-BEATS

N-BEATS is a deep learning model designed specifically for TSF. It employs a fully feedforward architecture, making it efficient and parallelizable. Its decomposes time-series data into interpretable basis functions, capturing meaningful patterns such as trend and seasonality. This interpretability empowers users to gain insights into the driving factors behind forecasts. Its adaptability to various time-series scenarios, including long-term dependencies and irregular patterns, has contributed to its recognition as a powerful and versatile tool for accurate and interpretable TSF.

Below are the results for N-BEATS. R^2, MAPE, SMAPE have been used as the evaluation metrics. The results have been presented in the ascending order of the MAPE score.

Table 18: Detailed results N-BEATS model

No	TIME SERIES NAME	R^2	MAPE	SMAPE	TIME (s)
1	CA_total	0.58973	8.85414	9.09375	691.504
2	TX_full_state_Food_sale	0.38005	9.2094	9.17187	701.96
3	CA_4Food_sale	-0.0637	10.6699	10.6489	657.547
4	CA_4	-0.2536	11.2951	10.5337	705.577
5	CA_1Househ_sale	0.64848	11.3234	11.8428	636.111
6	WI_1Food_sale	0.5016	11.497	11.9183	644.303
7	TX_1	0.24612	11.7753	12.7107	692.892
8	Household_combined	0.38446	12.0988	12.9227	696.718
9	CA_1Food_sale	0.47538	12.0989	12.7886	650.372
10	CA_3Househ_sale	0.50255	12.569	11.7259	710.465
11	TX-full_state_Househ_sale	0.38208	12.9461	13.6793	705.389
12	CA_full_state_Food_sale	-0.2831	13.0775	14.3469	681.81
13	CA_2	0.48307	13.7282	14.3682	700.233
14	WI_full_state_Hobby_sale	0.09854	14.1707	15.3612	719.839
15	CA_3Food_sale	-0.5048	15.0716	16.5152	689.202
16	TX_1Food_sale	-0.2403	15.1151	15.4286	713.786
17	Total_sale	-0.4144	15.4588	17.2605	659.103
18	WI_1Hobby_sale	0.3651	15.6444	16.6027	643.45
19	CA_full_state_Hobby_sale	-0.7029	15.6688	17.2825	709.419
20	CA_2Hobby_sale	0.19189	16.265	17.3909	668.966
21	TX_total	-0.3462	16.3934	18.207	690.426
22	TX_3	-0.702	16.4793	18.0577	700.296
23	Food_combined	-0.5671	16.9231	18.7477	700.129
24	TX_2Hobby_sale	0.21399	17.1145	17.6156	642.783
25	WI_2Hobby_sale	-0.2062	17.3219	17.4293	698.246
26	CA_2Househ_sale	0.36274	17.9609	19.6546	664.346
27	WI_3Food_sale	-0.1302	18.8535	19.5424	664.192
28	Hobby_combined	-0.759	18.8742	21.2581	688.202
29	WI_total	-0.6658	18.9382	20.4107	693.398
30	WI_full_state_Househ_sale	-0.3948	19.4365	21.3935	716.3
31	TX_3Food_sale	-1.042	19.4858	22.1358	687.229
32	TX_1Househ_sale	-0.365	19.947	22.7694	690.047
33	WI_full_state_Food_sale	-0.4804	20.1453	21.1162	712.424
34	WI_2	-0.6858	20.3823	21.6771	705.92

35	CA_3Hobby_sale	-1.5283	20.3922	23.1415	701.105
36	TX_3Househ_sale	-0.6553	20.4464	21.0647	653.213
37	TX_2Househ_sale	0.3907	20.8838	14.3972	658.253
38	WI_3Househ_sale	-0.1119	20.9524	19.8551	673.796
39	WI_1	-0.4121	21.597	24.7407	722.213
40	CA_3	-2.6847	22.2008	18.8287	705.023
41	WI_3Hobby_sale	-0.2365	22.2512	19.3291	678.385
42	WI_2Househ_sale	-0.5366	23.3986	22.6027	692.158
43	WI_3	-0.7825	24.2862	21.8114	716.544
44	CA_1Hobby_sale	-1.0892	24.2906	28.1909	647.398
45	WI_1Househ_sale	-0.293	24.7815	28.6315	635.416
46	CA_2Food_sale	-0.7628	25.0261	30.7774	683.933
47	TX_full_state_Hobby_sale	-1.1022	26.587	32.0496	688.894
48	CA_1	-2.2582	26.6614	33.189	704.382
49	TX_2Food_sale	-0.4526	28.0037	17.594	669.068
50	CA_4Househ_sale	-4.7429	28.524	35.422	663.902
51	WI_2Food_sale	-0.6926	29.6641	28.6306	697.425
52	TX_2	-1.9697	30.5794	30.6415	686.123
53	TX_1Hobby_sale	-2.2537	33.4815	42.1698	699.743
54	TX_3Hobby_sale	-1.8562	36.1883	47.0143	692.236
55	CA_full_state_Househ_sale	-5.344	46.0057	65.378	689.694
56	CA_4Hobby_sale	-14.046	49.1554	70.6568	662.072

B Results in detailed for Transfer Learning

In this section presents a comprehensive overview of all the results obtained from the conducted experiments. The detailed information encompasses the performance metrics, such as Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (SMAPE), and R-squared (R^2), for each model

B.1 Zero-shot learning results

In time-series zero-shot transfer learning refers to the application of transfer learning techniques in TSF tasks where the target domain lacks labeled data for the specific time points of interest. In traditional transfer learning, a pre-trained model is fine-tuned on a related source domain with available labeled data before being applied to the target domain. However, in zero-shot transfer learning, the model is directly utilized for forecasting without any fine-tuning or access to labeled target domain data.

Below are the results for zero-shot transfer learning RNN-LSTM. R^2, MAPE, SMAPE have been used as the evaluation metrics. The results have been presented in the ascending order of the MAPE score.

Table 19: Zero-shot learning RNN-LSTM

No	TIME SERIES TRAINED ON	TIME SERIES TESTED ON	R^2	MAPE	SMAPE	TIME (s)
1	CA_total	WI_total	-1.15	22.35	23.18	250.446
2	CA_total	TX_total	-0.69	16.57	17.50	174.65
3	TX_total	CA_total	-0.20	16.52	18.32	164.23
4	WI_total	CA_total	-0.44	14.34	15.69	168.63
5	TX_total	WI_total	-1.06	22.93	26.57	141.43
6	WI_total	TX_total	-0.63	15.23	16.81	151.89
7	TX_1	CA_2	0.31	14.38	15.93	167.14
8	TX_1	WI_3	0.23	14.48	13.7	149.72
9	WI_3	TX_1	-0.9	18.12	20.08	151.09
10	CA_2	TX_1	-1.25	22.73	26.3	153.03
11	CA_2	WI_3	0.19	14.23	14.59	154.4
12	WI_3	CA_2	-0.25	26.5	26.19	151.49
13	Hobby_combined	Food_combined	0.34	11.07	10.55	152.78
14	Food_combined	Hobby_combined	-3.75	30.1	37.18	155.1
15	Food_combined	Household_combined	-1.15	22.42	25.63	150.54
16	Hobby_combined	Household_combined	0.31	12.2	12.68	155.57
17	Household_combined	Hobby_combined	-1.22	22.55	26.04	160.84
18	Household_combined	Food_combined	0.46	9.67	9.58	155.99
19	CA_full_state_Food_sale	TX-full_state_Househ_sale	-1.35	21.4	24.64	157.93
20	CA_full_state_Food_sale	WI_full_state_Hobby_sale	-1.73	26.08	28.61	158.52
21	TX-full_state_Househ_sale	WI_full_state_Hobby_sale	-0.16	16.94	18.15	153.07
22	TX-full_state_Househ_sale	CA_full_state_Food_sale	0.64	8.01	7.99	157.38

23	WI_full_state_Hobby_sale	CA_full_state_Food_sale	0.29	10.86	10.9375	151.105
24	WI_full_state_Hobby_sale	TX_full_state_Househ_sale	-0.29	17.4	19.416	149.19
25	CA_4Househ_sale	TX_1Food_sale	-0.69	22.4	19.47	146.67
26	CA_4Househ_sale	WI_2Hobby_sale	-0.37	19.3	17.84	136.22
27	TX_1Food_sale	CA_4Househ_sale	-4.43	28.0	34.33	178.74
28	TX_1Food_sale	WI_2Hobby_sale	-2.82	30.5	37.54	257.42
29	WI_2Hobby_sale	CA_4Househ_sale	-1.48	18.12	20.2	256.07
30	WI_2Hobby_sale	TX_1Food_sale	-0.54	19.54	17.94	266.73

Below are the results for zero-shot transfer learning N-BEATS. R^2, MAPE, SMAPE have been used as the evaluation metrics. The results have been presented in the ascending order of the MAPE score.

Table 20: Zero-shot learning N-BEATS model

TIME SERIES TRAINED ON	**TIME SERIES TESTED ON**	**R^2**	**MAPE**	**SMAPE**	**TIME (s)**
CA_total	WI_total	-7.53	49.7	70.4	2474
CA_total	TX_total	-5.28	36.1	46.4	2683
TX_total	CA_total	-6.75	42.0	55.1	2740
WI_total	CA_total	-4.4	34.1	42.5	2673
TX_total	WI_total	-9.20	52.9	76.1	2701
WI_total	TX_total	-4.27	32.1	40.	3447
TX_1	**CA_2**	**0.50**	**16.28**	**15.7**	**3670**
TX_1	WI_3	-0.41	21.27	18.3	3677
WI_3	TX_1	-1.05	19.1	20.1	5071
CA_2	TX_1	-0.99	20.8	24.0	4322
CA_2	**WI_3**	**0.07**	**15.6**	**15.5**	**4167**
WI_3	CA_2	-0.5	29.4	27.4	4292
Hobby_combined	Food_combined	-4.13	30.9	38.2	3006
Food_combined	Hobby_combined	-12.0	57.7	84.2	3232
Food_combined	Household_combined	-5.40	47.19	63.8	3015
Hobby_combined	Household_combined	-4.5	38.69	51.2	3622
Household_combined	Hobby_combined	-2.10	26.83	31.9	2927
Household_combined	**Food_combined**	**0.32**	**10.46**	**10.8**	**1710**
CA_full_state_Food_sale	TX-full_state_Househ_sale	-0.25	18.45	20.9	1322
CA_full_state_Food_sale	WI_full_state_Hobby_sale	-0.69	22.07	24.5	1877
TX-full_state_Househ_sale	**WI_full_state_Hobby_sale**	**0.3**	**12.71**	**12.57**	**1434**
TX-full_state_Househ_sale	**CA_full_state_Food_sale**	**0.55**	**9.5**	**9.01**	**1296**
WI_full_state_Hobby_sale	CA_full_state_Food_sale	-0.87	20.1	18.38	1356
WI_full_state_Hobby_sale	TX_full_state_Househ_sale	-0.5	18.9	19.6	1340
CA_4Househ_sale	TX_1Food_sale	-0.21	16.7	15.12	1440
CA_4Househ_sale	WI_2Hobby_sale	-0.30	17.95	18.77	1364
TX_1Food_sale	CA_4Househ_sale	-8.7	42.88	56.7	1540
TX_1Food_sale	WI_2Hobby_sale	-6.97	47.21	65.6	1409
WI_2Hobby_sale	CA_4Househ_sale	-1.44	19.1	21.46	1353
WI_2Hobby_sale	TX_1Food_sale	-0.17	15.71	14.98	1381

B.2 Model Based learning

In time-series model-based transfer learning involves leveraging pre-trained models on a source time-series domain and adapting them to a target time-series domain for improved forecasting performance. In this approach, the pre-trained model is typically fine-tuned or adapted using a small amount of labeled data from the target domain.

Below are the results for model-based transfer learning RNN-LSTM. R^2, MAPE, SMAPE have been used as the evaluation metrics. The results have been presented in the ascending order of the MAPE score.

Table 21: Model based learning RNN-LSTM model

No	TIME SERIES TRAINED ON	TIME SERIES TESTED ON	R^2	MAPE	SMAPE	TIME (s)
1	CA_total	WI_total	-0.02	14.98	15.36	200.47
2	CA_total	TX_total	0.15	11.72	12.61	248.44
3	TX_total	CA_total	0.67	7.82	7.89	194.73
4	WI_total	CA_total	0.48	9.75	9.97	187.83
5	TX_total	WI_total	0.33	11.87	12.28	184.93
6	WI_total	TX_total	0.29	11	11.70	185.1
7	TX_1	CA_2	0.49	13.	13.72	185.83
8	TX_1	WI_3	-0.56	18.5	20.51	179.9
9	WI_3	TX_1	-0.21	14.16	15.13	167.1
10	CA_2	TX_1	0.36	10.59	11.33	163.8
11	CA_2	WI_3	-0.001	15.21	16.61	163.16
12	WI_3	CA_2	0.37	14.08	14.95	205.9
13	Hobby_combined	Food_combined	0.07	12.59	12.73	167.81
14	Food_combined	Hobby_combined	-1.09	20.7	23.70	168.69
15	Food_combined	Household_combined	0.64	10.14	10.21	163.83
16	Hobby_combined	Household_combined	0.60	10.5	10.84	166.88
17	Household_combined	Hobby_combined	-0.86	19.27	21.79	165.29
18	Household_combined	Food_combined	0.47	9.35	9.54	165.11
19	CA_full_state_Food_sale	TX-full_state_Househ_sale	0.41	12.76	13.40	166.3
20	CA_full_state_Food_sale	WI_full_state_Hobby_sale	0.25	13.16	13.97	162.66
21	TX-full_state_Househ_sale	WI_full_state_Hobby_sale	0.27	12.82	13.56	166.27
22	TX-full_state_Househ_sale	CA_full_state_Food_sale	0.20	11.96	11.58	163.93
23	WI_full_state_Hobby_sale	CA_full_state_Food_sale	0.63	8.38	8.15	162.91
24	WI_full_state_Hobby_sale	TX_full_state_Househ_sale	0.37	13.08	13.81	165.71
25	CA_4Househ_sale	TX_1Food_sale	-0.61	17.37	17.31	249.95
26	CA_4Househ_sale	WI_2Hobby_sale	-0.067	15.91	16.38	174.0
27	TX_1Food_sale	CA_4Househ_sale	0.30	10.11	10.38	181.9
28	TX_1Food_sale	WI_2Hobby_sale	-0.33	17.34	17.98	186.7
29	WI_2Hobby_sale	CA_4Househ_sale	-0.25	13.78	13.54	187.6
30	WI_2Hobby_sale	TX_1Food_sale	0.25	12.06	12.07	184.05

Below are the results for model-based transfer learning N-BEATS. R^2, MAPE, SMAPE have been used as the evaluation metrics. The results have been presented in the ascending order of the MAPE score.

Table 22: Model based learning N-BEATS model

TIME SERIES TRAINED ON	TIME SERIES TESTED ON	R^2	MAPE	SMAPE	TIME (s)
CA_total	WI_total	0.07	13.55	14.68	1355.72
CA_total	TX_total	0.11	12.53	13.51	1485.73
TX_total	CA_total	0.57	9.10	9.10	1562.75
WI_total	CA_total	0.28	12.29	13.28	1377.20
TX_total	WI_total	-0.97	19.50	22.09	1524.82
WI_total	TX_total	-4.27	32.17	40.04	3447.18
TX_1	CA_2	0.40	13.64	14.73	1526.72
TX_1	WI_3	-0.36	17.72	19.45	1744.46
WI_3	TX_1	0.43	9.83	10.43	1619.31
CA_2	TX_1	0.51	9.10	9.31	1839.71
CA_2	WI_3	-0.04	15.20	16.53	1820.80
WI_3	CA_2	0.41	13.41	14.42	2040.4
Hobby_combined	Food_combined	0.16	12.59	12.76	1760.68
Food_combined	Hobby_combined	-1.05	20.76	23.69	2445.84
Food_combined	Household_combined	0.57	11.07	11.52	2186.96
Hobby_combined	Household_combined	0.15	13.11	14.05	1833.65
Household_combined	Hobby_combined	-2.62	29.23	34.94	1769.22
Household_combined	Food_combined	0.04	13.62	13.65	1969.36
CA_full_state_Food_sale	TX-full_state_Househ_sale	0.36	12.88	13.57	2715.89
CA_full_state_Food_sale	WI_full_state_Hobby_sale	-1.06	20.98	23.17	2043.33
TX-full_state_Househ_sale	WI_full_state_Hobby_sale	-0.09	16.87	17.73	2070.31
TX-full_state_Househ_sale	CA_full_state_Food_sale	0.53	9.26	8.88	2193.87
WI_full_state_Hobby_sale	CA_full_state_Food_sale	0.55	9.17	8.73	2356.27
WI_full_state_Hobby_sale	TX_full_state_Househ_sale	0.35	13.00	13.80	2638.56
CA_4Househ_sale	TX_1Food_sale	0.21	12.52	12.47	2098.75
CA_4Househ_sale	WI_2Hobby_sale	-0.53	19.55	20.18	2287.05
TX_1Food_sale	CA_4Househ_sale	0.17	11.15	11.35	2027.69
TX_1Food_sale	WI_2Hobby_sale	-0.41	18.26	19.24	1926.03
WI_2Hobby_sale	CA_4Househ_sale	0.16	11.52	11.39	1755.91
WI_2Hobby_sale	TX_1Food_sale	0.27	11.50	11.72	1562.32

B.3 Prophet with model-based time-series data

Table 23: Prophet with model based time-series retraining data

TIME-SERIES NAME	R^2	MAPE	SMAPE	TIME (s)
CA_4Househ_sale	-9.35	45.51	35.59	0.29
TX_1Food_sale	-0.24	17.65	19.82	0.45
WI_2Hobby_sale	0.14	22.36	22.12	0.56
CA_2	-0.49	28.74	35.83	0.35
WI_3	-0.08	23.53	20.1	0.45
TX_1	0.34	10.84	11.15	0.65
CA_total	0.76	7.59	7.24	0.74
TX_total	0.49	9.27	9.42	0.62
WI_total	0.13	17.89	15.74	0.32
Food_combined	0.58	9.6	9.57	0.81
Hobby_combined	-0.004	13.771887	14.95	0.51
Household_combined	0.78	8.048152	7.72	0.53
CA_full_state_Food_sale	0.6	9.454188	8.88	0.32
TX-full_state_Househ_sale	0.54	10.67	11.14	0.56
WI_full_state_Hobby_sale	0.26	16.05	14.38	0.69