Sebastião Vasconcelos Maia dos Santos Lessa
Curricular Internship Report

Transfer learning using deep learning for forecasting retail sales

Advisor: Luís Roque
Co-Advisor: Rafael Guedes

Departamento de Ciência de Computadores[2]
Faculdade de Ciências da Universidade do Porto[3]
July 2024

Abstract

This report presents the findings of a study on *Transfer learning using deep learning for forecasting retail sales*. The primary objective is to investigate the potential of developing a superior hybrid forecasting model that combines an accurate foundation model, which uses only historical data, with a multivariate model capable of incorporating external data that changes over time. In this study, *Chronos*[1] is used as the foundation model, while two multivariate models, *TiDE*[6] and *TSMixer*[9], are explored. Two different approaches were tested to combine the foundation model with the multivariate model. The first strategy involved calculating the residuals of the foundation model forecasts and using them, along with the original features, a the target to train the multivariate model. This method aimed to predict the future residuals of the foundation model, which, when added to its initial forecasts, could potentially yield a more accurate final forecast. The second strategy involved directly using the forecasts generated by the foundation model as dynamic features in the training data for the multivariate model. By integrating these forecasts into the training process, it was hypothesized that the multivariate model could leverage this additional information to enhance its forecasting performance. The results showed that both hybrid models outperformed the state-of-the-art *TiDE* and *TSMixer* models. However, *Chronos* on its own remained the best model on average. These findings significantly advance the application of hybrid deep learning techniques in retail forecasting, providing a robust methodology for integrating different types of covariates to enhance predictive accuracy.

Contents

List of Figures

Listings

Chapter 1

Introduction

The principal aim of this study is to investigate the potential of advanced hybrid forecasting models to enhance the accuracy and adaptability of time-series predictions. This research primarily focuses on evaluating the capabilities of hybrid models that combine a foundation model that only uses historical data with a multivariate model. Such exploration is crucial for advancing our understanding of how different data inputs can be effectively leveraged to improve forecasting methodologies. To assess the practical implications and effectiveness of these theoretical models, I applied them to the retail industry, a sector that greatly benefits from precise forecasting due to its dynamic nature and the direct impact of such forecasts on inventory management and strategic planning. For this purpose, we utilize a specific dataset tailored to retail sales data (see **Figure 1.1**), which provides a relevant environment for testing our models. The dataset, to be detailed later, includes a mix of constant and evolving characteristic of retail operations. For my foundation model, I will use the *Chronos*[1] model, which I will then try to combine with a multivariate model. I will test two multivariate models: *TiDE*[6] and *TSMixer*[9]. I explore two strategies in my study: the first strategy involves enhancing the forecasts of the foundation model by calculating and subsequently correcting its residuals with the multivariate model. The second strategy integrates the forecasts from the foundation model directly into the multivariate model as dynamic covariates, hypothesizing that this will provide a richer input and thus lead to improved forecasting performance.

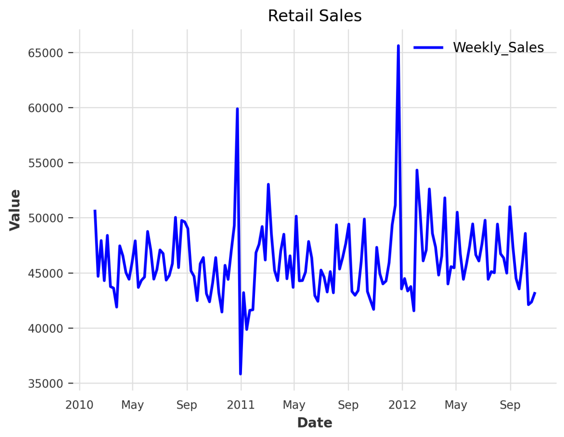

Figure 1.1: Weekly Retail Sales at a Sample Store from the Dataset

Chapter 2

State of the Art

In this chapter, we dive into the advanced methodologies and tools that underpin our investigation into using deep learning to forecast retail sales. As computing technology and algorithmic development continue to advance at a breakneck pace, our ability to conduct detailed data analysis and complex predictive modeling has improved dramatically. At the heart of our research are several pivotal technologies and models that have been instrumental in pushing the boundaries of machine learning and data science.

2.1 Python

For this study, we selected Python due to its widespread acclaim for versatility and robustness in scientific computing. Python's rich ecosystem, especially its libraries designed for data manipulation and machine learning, make it an ideal choice for in-depth research. A key component of this ecosystem that we utilized is the `Darts` library, which has been indispensable in our analysis.

2.1.1 Darts

In our study, Darts, as shown in **Figure 2.1**, played a crucial role in building and validating time series forecasting models for retail sales. I leveraged its extensive model library and preprocessing capabilities to streamline my workflow, from data preparation to model evaluation. The ease of integrating Darts with other Python libraries like Pandas for data manipulation and Matplotlib for visualization significantly enhanced my productivity and analytical depth.

Figure 2.1: Darts library logo

2.2 Jupyter Notebook

Jupyter Notebook stands as an indispensable tool in the modern data science workflow, particularly due to its ability to seamlessly integrate code, visualizations, and narrative text in a single, interactive document. Its fundamental role in my study cannot be overstated, as it provided a versatile platform for developing, documenting, and executing the complex analyses required. The Jupyter environment supported every phase of the research, from preliminary data exploration and preprocessing to sophisticated model development and evaluation, including the implementation of models like *TSMixer* and *Chronos*. This capability was crucial for fine-tuning models and instantly observing the effects of changes, fostering a deeper understanding and more rapid advancement of the research.

2.3 Time Series Models

Time series modeling is essential for extracting meaningful statistics and characteristics from data indexed in time order. These models are crucial for forecasting future values based on previously observed data, aiding numerous applications across finance, retail, meteorology, and more. By understanding patterns in historical data, time series models enable predictions and insights that inform strategic planning and operational improvements.

In this study, as mentioned before, I selected *Chronos*[1] as the foundation model and *TiDE*[6] and *TSMixer*[9] as the multivariate models. The multivariate models attempt to incorporate the forecasts of *Chronos* along with external factors that change over time. This approach aims to leverage both static and dynamic covariates to enhance the accuracy and adaptability of the a hybrid forecasting model.

2.3.1 Chronos

In the realm of time series forecasting, *Chronos* emerges as a transformative framework, ingeniously adapting the proven mechanisms of language models to the unique challenges of time series data. Developed by *AWS AI Labs*, *Chronos* redefines traditional forecasting methods by tokenizing time series into a fixed vocabulary through simple scaling and quantization techniques, thereby enabling the application of existing transformer-based language model architectures, notably those from the T5 family, directly to time series forecasting. This innovative approach, detailed in their comprehensive study, leverages a large corpus of both public and synthetic datasets to train these models, resulting in a tool that not only excels in benchmark datasets but also demonstrates robust zero-shot forecasting capabilities. Chronos's success underscores its potential to substantially simplify forecasting pipelines by employing pre-trained models that adapt seamlessly across diverse domains, offering a significant leap forward from conventional forecasting models that require task-specific adjustments.

Figure 2.2 illustrates the high-level process of how *Chronos* operates. (Left) The input time series is scaled and quantized to obtain a sequence of tokens. (Center) These tokens are then fed into a language model, which may either be an encoder-decoder or a decoder-only model. The model is trained using the cross-entropy loss to ensure effective learning. (Right) During inference, the system autoregressively samples tokens from the model and maps them back to numerical values. Multiple trajectories are sampled to obtain a predictive distribution, showcasing the model's capacity to generate accurate and diverse forecasts.

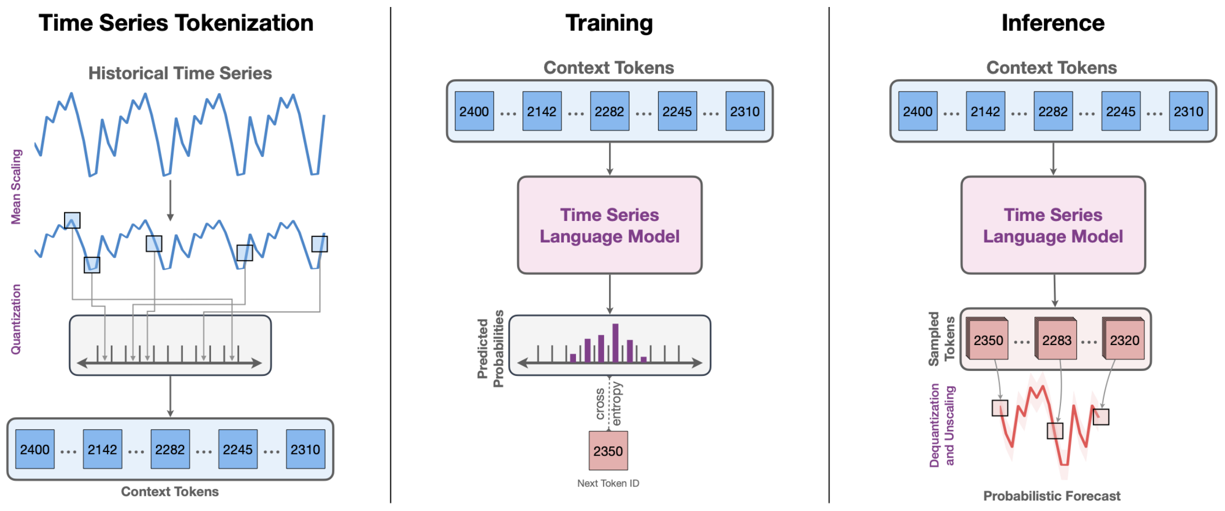

Figure 2.2: Chronos Learning Process

2.3.2 TiDE

In the rapidly evolving domain of time series forecasting, the *Time-series Dense Encoder (TiDE)* model stands out as a notable advancement, particularly in the context of long-term forecasting. Developed by researchers at Google, TiDE offers a novel approach by integrating Multi-layer Perceptrons (MLPs) into an encoder-decoder framework, which simplifies the forecasting process while maintaining high accuracy and computational efficiency. Unlike traditional models that often rely on complex mechanisms like Transformers, *TiDE's* architecture is remarkably straightforward, leveraging dense MLP layers to handle both past and future covariates effectively. This simplicity allows it to achieve near-optimal error rates in scenarios modeled as linear dynamical systems, and empirical results demonstrate its superior performance on benchmark datasets, consistently outperforming more complex Transformer-based models. *TiDE's architecture* as you can visualize on **Figure 2.3** facilitates significantly faster training and inference, making it a practical choice for real-world applications where speed and accuracy are paramount.

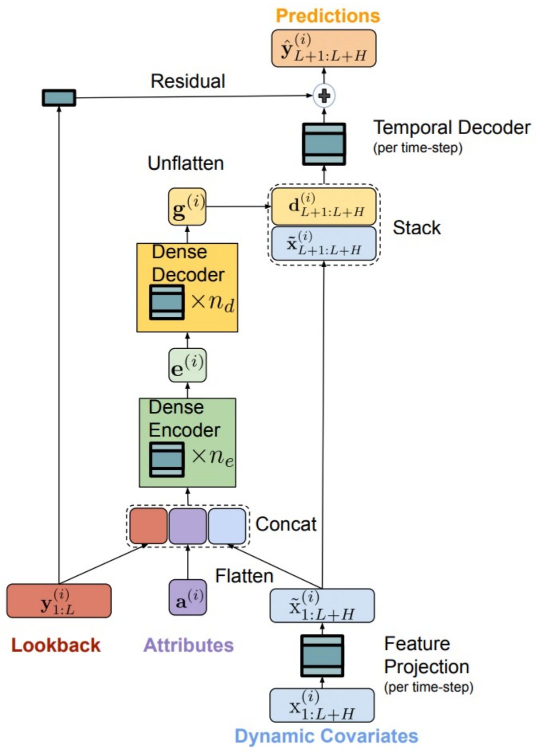

Figure 2.3: TiDE Architecture

2.3.3 TSMixer

The *TSMixer* model represents a breakthrough in time series forecasting with its distinctive ”mixer” architecture, adept at handling complex multivariate data. As illustrated in **Figure 2.4**, the TSMixer employs a dual-path setup using two types of Multi-layer Perceptrons (MLPs): one for mixing over time and another for mixing features. This structure allows each layer to focus on different aspects of the data—the Time Mixing layer processes sequential dependencies, while the Feature Mixing layer deals with the interactions among various features.

Repeatedly applying these mixer layers allows for comprehensive integration of both temporal and feature information, starting with batch normalization to standardize input data before it undergoes the intricate mixing process. The culmination of this process is a temporal projection that synthesizes all mixed data into a well-defined forecast, enhancing the model's ability to detect and adapt to complex patterns and dependencies effectively.

This innovative approach not only boosts the TSMixer's capability to dissect and understand complex data intricacies but also significantly improves its forecasting accuracy. This makes the TSMixer an invaluable asset for diverse forecasting tasks across different fields, providing insights and predictions that are crucial for informed decision-making.

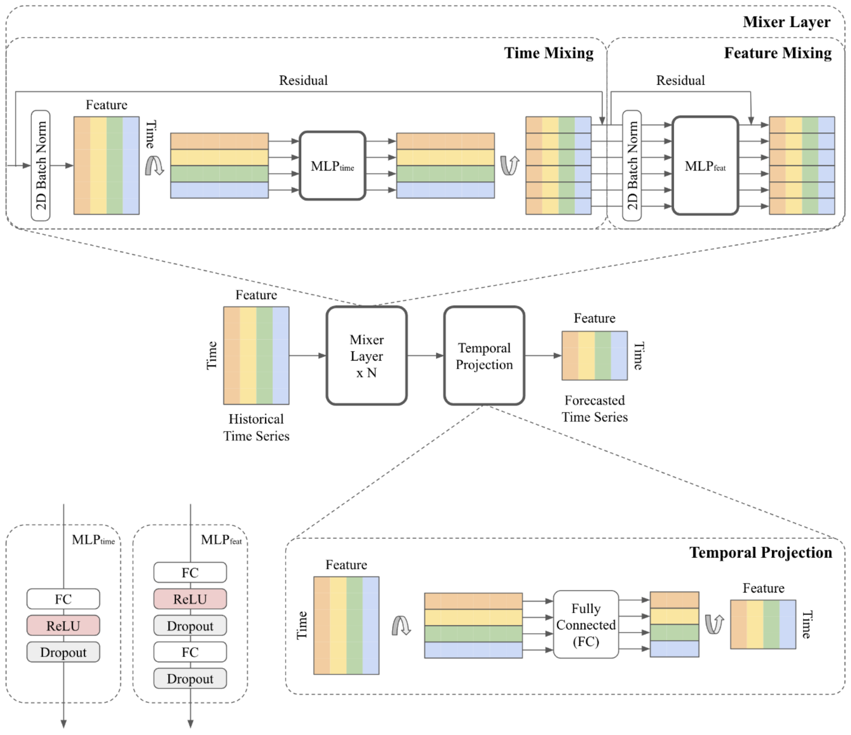

Figure 2.4: TSMixer Architecture

Chapter 3

Methodology

3.1 Dataset

To test my hypothesis, I utilized the **Walmart Sales Forecast dataset**[10]. Spanning from February 5, 2010, to October 19, 2012, this dataset includes Store ID and Department ID, which together serve as the primary key for each store entry. It features both constant attributes such as *Type* and *Size*, and dynamic variables including *IsHoliday*, *Temperature*, *Fuel Price*, *Consumer Price Index (CPI)*, and *Unemployment* rates, providing a rich source of time-variant data for predictive modeling.

3.2 Approaches

I initiated the analysis by providing context to the *Chronos* model, using data from February 5, 2010, through June 10, 2011. Predictions were generated from June 17, 2011, to October 12, 2012, utilizing a **expanding window forecasting technique**, as illustrated in **Figure 3.1**. This method involves sequentially updating the training and testing datasets, moving the prediction start date forward by one week at a time. Each cycle involved defining a training set that included all data up to the current prediction start date and a test set spanning a forecast horizon beginning at this date. The model was retrained weekly with the updated training set and then used to forecast the next period, effectively mimicking real-world forecasting scenarios by incrementally incorporating new data as it becomes available. In this case, the forecast horizon was set to one week, and at the end of each prediction cycle, the forecast for that week was saved, creating a dataset of weekly forecasts. This accumulated dataset represents a series of optimally reduced error predictions, providing a comprehensive view of the model's performance across the entire forecasting period. This approach is key in assessing the model's adaptability and accuracy over time.

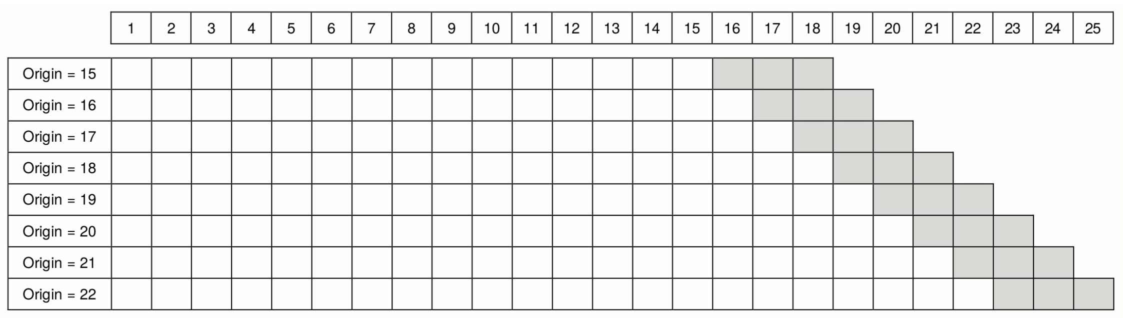

Figure 3.1: Expanding Window Forecasting

3.2.1 Residual Training

Following the generation of *Chronos* forecasts, I calculated the residuals by subtracting the *Chronos* predicted values from the actual values. These residuals were then merged with the non-target attributes of the original dataset for corresponding dates, resulting in a new enriched dataset, as visualized in **Figure 3.2**. This enriched dataset was used to train the multivariate models, *TiDE* and *TSMixer*, employing a **cross-validation technique** [4] as illustrated in **Figure 3.5**. For each validation fold, the predicted residuals were added back to the original *Chronos* forecasts. This hybrid approach theoretically aims to yield a final forecast that is superior to the initial *Chronos* predictions by compensating for initially unaccounted variances. You can visualize the training process for the residuals in **Figure 3.3**.

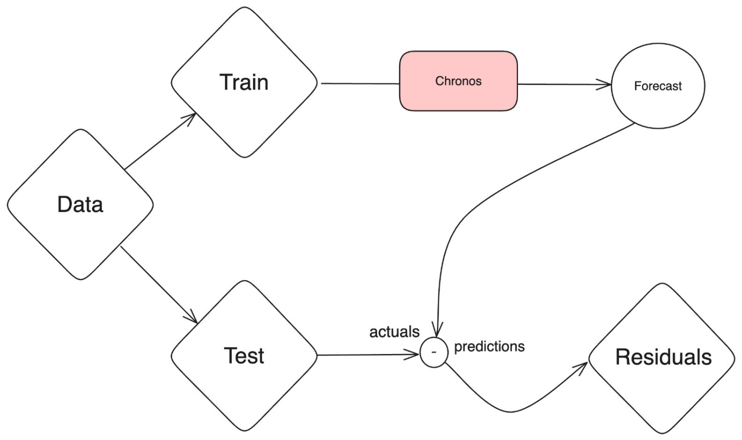

Figure 3.2: Residuals Creation

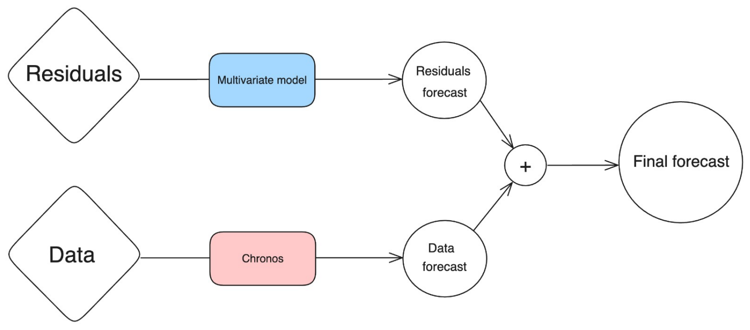

Figure 3.3: Residual Training

3.2.2 Forecasts as Dynamic Covariates

Another method tested involved integrating *Chronos* forecasts directly into the training dataset, using these forecasts as dynamic covariates, as illustrated in **Figure 3.4**. This approach aimed to enhance the learning process for the *TiDE* and *TSMixer* models, enabling them to utilize both historical data and near-term forecast insights to more accurately predict future outcomes.

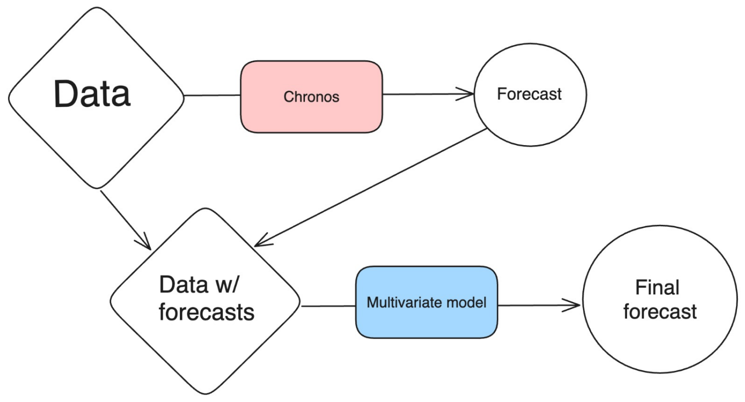

Figure 3.4: Forecasts as Dynamic Covariates Training

3.3 Hyper Parameter Tuning

Hyperparameter tuning is a critical step in optimizing the performance of machine learning models. For our analysis, this involved fine-tuning the hyperparameters of the *TiDE* and *TSMixer* models on the hybrid model combinations to maximize forecasting accuracy. The technique employed for both approaches, described in *Section 3.2.1* and *Section 3.2.2*, was **Random Search**[5]. This method explored various combinations of hyperparameters, as illustrated in **Listing 3.1**. The random search was conducted through a while loop, randomly adjusting hyperparameters such as *input chunk length*, *hidden sizes*, *dropout rates*, and *learning rates*, among others. The models' performance was evaluated based on the mean of the **mean root mean square error (RMSE)**[8] of a set of predefined folds using the **cross-validation technique**[4], as visualized in **Figure 3.5**, each with a forecast horizon of 10 weeks.

RMSE was chosen because it is sensitive to data outliers. Large errors have an excessively negative effect on the total value since squaring the errors magnifies their impact. This sensitivity can be helpful in recognizing and understanding the effects of extreme forecasts or outliers on the model's performance. The configuration yielding the lowest **RMSE** was then saved. This process was designed to run continuously until manually stopped, allowing for extensive exploration of the parameter space. It ran for approximately 50 hours.

```python
def get_tsmixer_params():
    return {
        "input_chunk_length": random.choice([2, 4, 8, 10]),
        "output_chunk_length": FORECAST_HORIZON,
        "hidden_size": random.choice([2, 4, 8, 16]),
        "ff_size": random.choice([2, 4, 8, 16]),
        "num_blocks": random.choice([1, 2, 3, 4]),
        "activation": random.choice(["ELU", "ReLU", "LeakyReLU", "GELU"]),
        "dropout": random.choice([0.1, 0.15, 0.3, 0.35]),
        "normalize_before": random.choice([True, False]),
        "batch_size": random.choice([8, 16, 32, 64]),
        "n_epochs": random.choice([10, 15, 20, 25, 30]),
        "likelihood": QuantileRegression(quantiles=[0.25, 0.5, 0.75]),
        "random_state": 42,
        "use_static_covariates": True,
        "optimizer_kwargs": {"lr": random.choice([1e-3, 1e-4, 1e-5, 1e-6])},
        "use_reversible_instance_norm": random.choice([True, False]),
    }

def get_tide_params():
    return {
        "input_chunk_length": random.choice([2, 3, 4, 6, 7]),
        "output_chunk_length": FORECAST_HORIZON,
        "num_encoder_layers": random.choice([2, 4, 6, 8]),
        "num_decoder_layers": random.choice([2, 4, 6, 8]),
        "decoder_output_dim": random.choice([6, 8, 10, 15, 16]),
        "hidden_size": random.choice([2, 4, 8, 16]),
        "temporal_width_past": random.choice([2, 4, 8]),
        "temporal_width_future": random.choice([4, 8, 10, 12]),
        "temporal_decoder_hidden": random.choice([16, 23, 26, 32]),
        "dropout": random.choice([0.1, 0.15, 0.3]),
        "batch_size": random.choice([8, 16, 32, 64]),
        "n_epochs": random.choice([10, 15, 20, 25, 30]),
        "likelihood": QuantileRegression(quantiles=[0.25, 0.5, 0.75]),
        "random_state": 42,
        "use_static_covariates": True,
        "optimizer_kwargs": {"lr": random.choice([1e-3, 1e-4, 1e-5, 1e-6])},
        "use_reversible_instance_norm": random.choice([True, False]),
    }
```

Listing 3.1: Hyperparameter configuration functions for TSMixer and TiDE models

3.4 Results

3.4.1 Evaluation Metrics

To evaluate the effectiveness of both approaches, described in *Section 3.2.1* and *Section 3.2.2*, the same **cross-validation technique**[4] seen in **Figure 3.5** was employed for both, enabling a comprehensive comparison of results across different modeling strategies. The metrics involved calculating the mean values of the **Mean Absolute Percentage Error (MAPE)**[7] for each forecast horizon iteration across all folds, synthesizing the data into a comprehensive measure of model performance. **MAPE** was chosen for visualization because it is more intuitive and easier to interpret, as it expresses the error as a percentage, making it straightforward to understand the relative magnitude of forecasting errors.

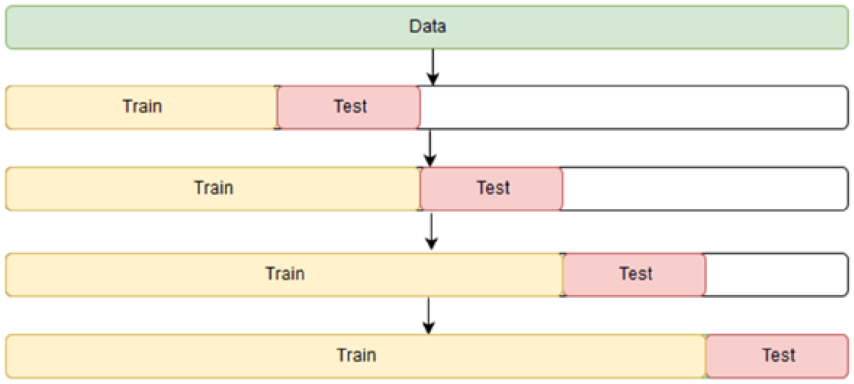

Figure 3.5: Cross Validation in Time Series

3.4.2 Visualization and Analysis of Forecasting Results

This analysis culminated in a series of bar charts, each representing the mean percentage error for each forecast horizon prediction. I performed this measurement twice: once using tuned hyperparameters and once with default settings. Additionally, I selected the top 10 stores to observe their predictions, both with and without tuning, for comparative analysis. Consequently, I produced four graphs: two illustrating the overall **MAPE** results for the top 500 stores and the top 10 stores without tuning, and two more illustrating the **MAPE** results with tuning for the same groups.

Untuned Graphs

The following figures display the performance outcomes for models without hyperparameter tuning. They illustrate the **MAPE** across the top 500 stores in **Figure 3.6** and the top 10 stores in **Figure 3.7**, providing insights into the baseline performance of our forecasting approach under standard conditions.

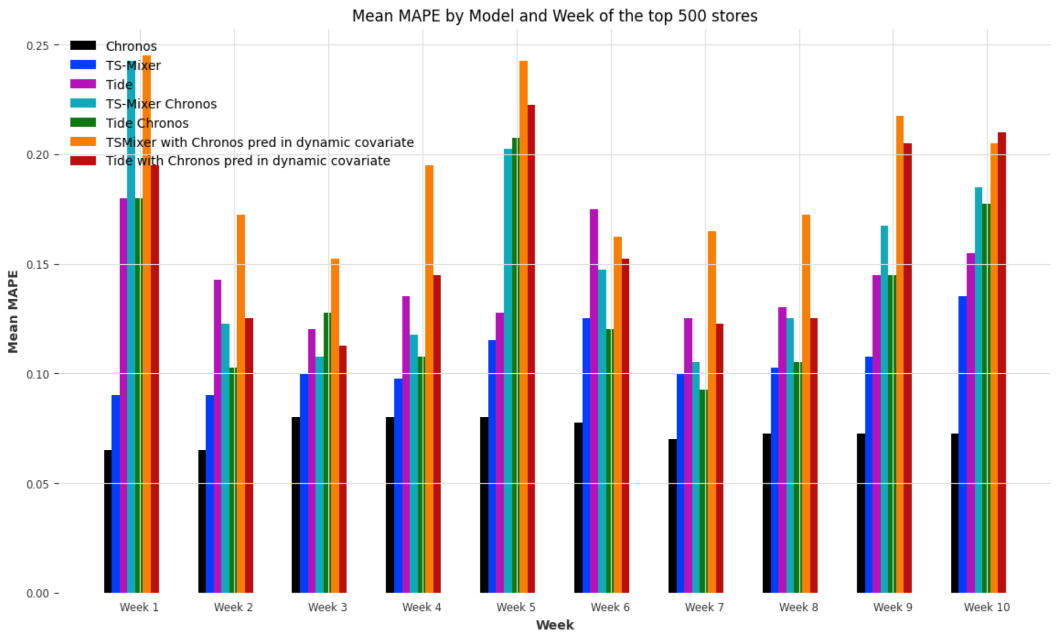

Figure 3.6: Top 500 Stores without Tuning

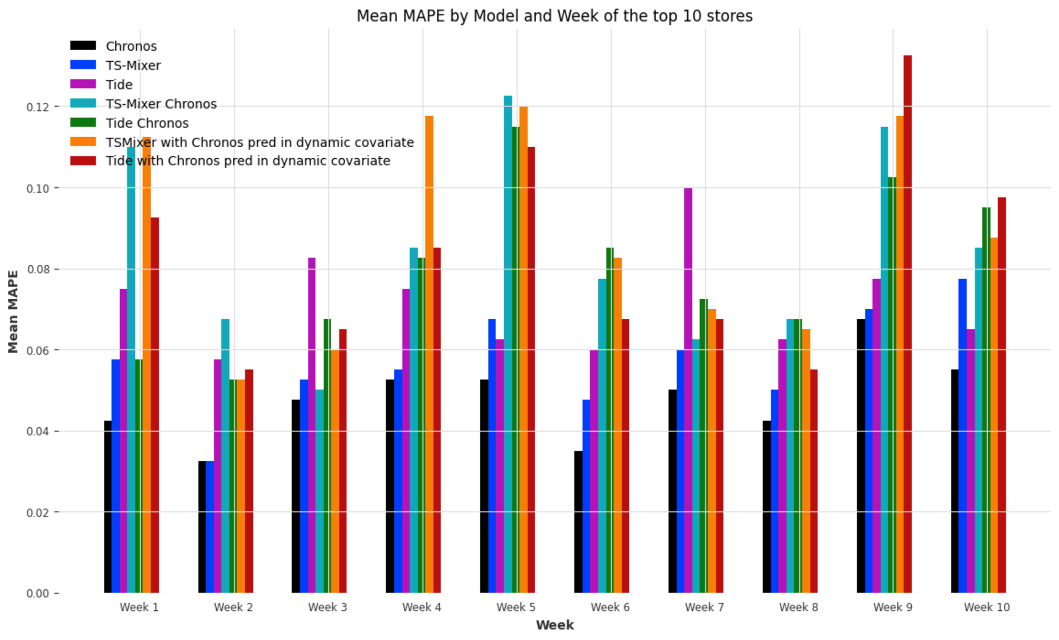

Figure 3.7: Top 10 Stores without Tuning

Tuned Graphs

This section presents figures that compare the effectiveness of the models after the application of optimized hyperparameters. The graphs provide a visual representation of the improvements in forecasting accuracy for both the top 500 stores in **Figure 3.8** and the top 10 stores in **Figure 3.9**, demonstrating the impact of tuning on model performance.

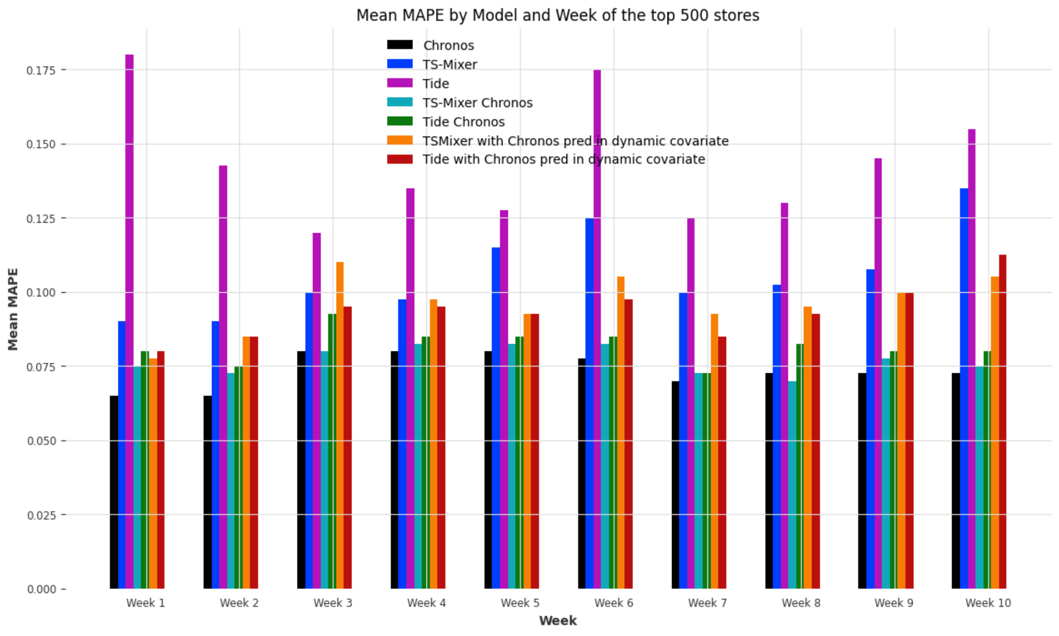

Figure 3.8: Top 500 Stores with Tuning

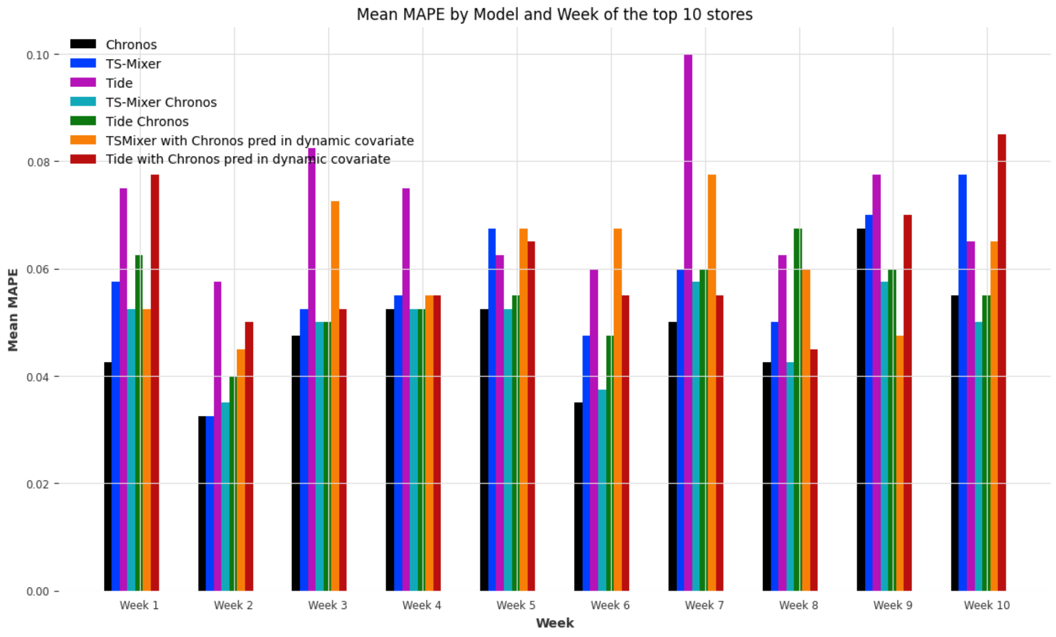

Figure 3.9: Top 10 Stores with Tuning

Discussion of Results

It is important to note that our efforts to improve the *Chronos* model's forecast did not yield significant enhancements. This is likely due to the macro nature of the external regressors, which may not have added substantial specific information to the model.

However, despite not improving the *Chronos* forecasts, our approach did enhance the performance of the state-of-the-art models: *TiDE* and *TSMixer*. They showed marked improvement, demonstrating the effectiveness of our strategy in leveraging dynamic covariates and residual training to boost forecasting accuracy. This highlights the potential of advanced deep learning techniques in enhancing time series forecasting for complex datasets.

Chapter 4

Conclusions

This study has embarked on an in-depth exploration of advanced time series forecasting methods, strategically leveraging a foundation model with a multivariate model to enhance the final prediction accuracy. Central to our inquiry were sophisticated models like *Chronos*, which served as our foundation model, and multivariate models such as *TiDE* and *TSMixer*, which were explored in two contexts: residual training and the use of forecasts as dynamic covariates.

Our research demonstrated that incorporating *Chronos* forecasts as dynamic covariates into the training of *TiDE* and *TSMixer* significantly enriched the input data, providing these models not only with historical data but also with foresight into potential future trends. This method proved particularly effective, enhancing the models' ability to anticipate and adapt to changes in the dataset, thereby yielding more accurate and reliable predictions. The addition of the forecasts as dynamic covariates made the singular multivariate models obsolete when matched against their hybrid combinations.

Furthermore, the strategy of residual training allowed us to fine-tune the multivariate models. This approach helped in minimizing forecast errors by adjusting the model parameters more precisely based on the residuals, thus significantly improving the overall forecast accuracy. Residual training involves a feedback mechanism where the discrepancies between predicted and actual values are used to iteratively refine model forecasts. By continuously adjusting to these errors, the models can evolve to become more sensitive to subtle patterns and anomalies in the data, which might otherwise be overlooked. This method proved to be highly effective, even almost always surpassing the results obtained by incorporating forecasts as dynamic covariates, as evidenced in **Figure 3.8** and **Figure 3.9**. The efficacy of residual training underscores the potential of iterative learning approaches in machine learning landscapes. By capitalizing on the errors as learning opportunities, residual training not only enhances the models' performance but also contributes to a more robust understanding of the underlying data dynamics. This strategy, therefore, not only improves forecast accuracy but also deepens our insights into the mechanisms driving the observed trends, leading to more informed decision-making and strategic planning in practical applications.

Both approaches were underpinned by rigorous hyperparameter tuning, which was crucial in optimizing model performance. Through the integration of hyperparameter tuning, we were able to systematically explore a vast parameter space, thereby identifying the optimal configurations that maximized forecasting accuracy. Our findings are visually substantiated in the comparison of tuned versus untuned models, as shown in **Figures 3.6**, **3.7**, **3.8**, and **3.9**, where tuned models consistently outperformed their untuned counterparts.

In conclusion, the integration of dynamic covariates derived from *Chronos* forecasts into multivariate models like *TiDE* and *TSMixer*, combined with the strategic use of residual corrections, has marked a significant advancement in the field of time series forecasting. This study not only highlights the potential of hybrid modeling techniques but also sets a precedent for future research in employing complex model integrations to enhance predictive performance in various applications. The implications of this research extend beyond academic interests, offering valuable insights into practical applications in industries where accurate forecasting is critical, such as retail, finance, and beyond.

As we continue to explore and innovate within the realm of time series forecasting, the horizon broadens for the development of even more sophisticated techniques that will undoubtedly revolutionize our approach to predictive analytics in the coming years.

Code Availability

The source code and supplementary materials used in this study are available on my GitHub profile. This includes the main notebook, all custom scripts, and additional documentation necessary to replicate the study's findings and analyses. The specific repository can be accessed directly at `https://github.com/seblessa/ZAAI`.

Bibliography

[1] "Chronos: Learning the Language of Time Series". In: (2024). URL: https://arxiv.org/pdf/2403.07815.

[2] Departamento de Ciência de Computadores. URL: https://www.dcc.fc.up.pt/.

[3] Faculdade de Ciências da Universidade do Porto. URL: https://www.up.pt/fcup/.

[4] "Evaluating time series forecasting models". In: (2019). URL: https://arxiv.org/pdf/1905.11744.

[5] *Hyper-Parameter Random Search*. URL: https://dl.acm.org/doi/pdf/10.5555/2188385.2188395.

[6] "Long-term Forecasting with TiDE: Time-series Dense Encoder". In: (2024). URL: https://arxiv.org/pdf/2304.08424.

[7] *Mean absolute percentage error*. URL: https://arxiv.org/pdf/1605.02541.

[8] *Root Mean Squared Error*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.root_mean_squared_error.html.

[9] "TSMixer: An All-MLP Architecture for Time Series Forecasting". In: (2023). URL: https://arxiv.org/pdf/2303.06053.

[10] *Walmart Sales Forecast*. URL: https://www.kaggle.com/datasets/aslanahmedov/walmart-sales-forecast/data?select=train.csv.