NOVA IMS Information Management School

MDSAA

Master Degree Program in

Data Science and Advanced Analytics

A Transformer-based Approach to Time-Series Forecasting

The Galp Example

Gonçalo Rodrigues Lourenço

Master Thesis

presented as partial requirement for obtaining a Master's Degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

A Transformer-based Approach to Time-Series Forecasting

The Galp Example

by

Gonçalo Rodrigues Lourenço

Master Thesis presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics, with a specialization in Data Science

Supervised by

Mauro Castelli, PhD, NOVA Information Management School

July, 2024

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Lisbon, July 15th, 2024

DEDICATION

I would like to dedicate this work to my wonderful family and girlfriend, who remind me every day what a happy life looks like. To the ones I miss the most, both my grandfathers and my grandmother, this work is mostly dedicated to you.

ACKNOWLEDGEMENTS

I know for a fact I could not have developed this study without the help of some amazing people. I would like to start by thanking my supervisor, Professor Mauro Castelli, for believing in the potential of this study and for trusting that I could carry it out. Without Professor Mauro's disposition to hear new ideas and help me bring them to life, this study would have never happened. For that, I am thankful.

I would like to thank my co-supervisors, Frederico Cabral and Rúben Menezes, for their guidance towards success. They taught me how to structure, organize and develop a Data Science project with rigor and scientific relevance but, most of all, how to excel in my professional life. They have both truly been mentors to me and, for that I am thankful.

I would like to thank my friends, for their support and their friendship that means more to me than they can imagine. They know who they are. Honourable mention to Alexandra, Edgi and Daniel, because without you, this academic journey would not have been the same.

I would like to thank Neni, for being there even before her front teeth were. She will always hold a special place in my heart and there will never be a version of my life where she is not my best friend.

Now, I would like to thank my family and my girlfriend. They are the people I want to make the proudest. I would like to thank my family for providing me with an amazing life, with a loving family that supports me in every challenge I face. Thank you for teaching me the values I employ today as a student, a professional but, mostly, as a person. Your support made this work possible, and for that, I am thankful.

Last but not least, I would like to thank the person who holds my whole heart, my girlfriend. Throughout my lowest points of this journey, she would build me back up and make me believe I could do it, as she does every day. To her, I must thank for my happiness, my successes and my future. For all that, for making this life so exciting, and for always being my significant *otter*, I am deeply thankful.

ABSTRACT

Financial time-series forecasting, more concretely stock price forecasting, has been a highly studied problem since the beginning of trading. Throughout the decades, the evolution of time-series forecasting models has led to more precise and consistent solutions to this problem. However, the traditionally used models have only been able to sustain this precision on a shorter forecasting horizon. This study aims to assess the performance of Transformer-based models on the stock price forecasting context, compared to other most commonly used models such as RNN-based models or CNN-based models. For this specific experiment, five models were selected: Informer, Autoformer, PatchTST, TimesNet and LSTM. Using Galp Energia S.A.'s seventeen-year historic stock price data, these models will produce comparable results, evaluated using Mean Average Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). The final results are expected to provide valuable insights on whether the Transformer architecture is the next step on the time-series models' evolution.

KEYWORDS

Transformer; CNN; RNN; Long Time-Series Forecasting; Galp Energia.

Sustainable Development Goals (SDG):

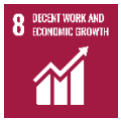

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE INDEX

1. INTRODUCTION

1.1. CONTEXT

The stock market encompasses the purchasing and selling of stocks, equities, currencies and other financial commodities by agents, denominated as traders. These aim to purchase stocks belonging to companies that were believed to be headed towards financial success and, simultaneously, sell the stocks of companies who were headed towards the opposite direction. However, the successful path that a company could be taking can rapidly shift and, with it, its stock prices. Hence, in order to maximize capital gains and minimize losses to a full extent, arose the need to study the indicators and factors that could influence a company's position in the stock market. These factors can vary from company to company, sector to sector and internal or external to companies (Sindhu, 2014) , making the stock market one of the most complex objects of study.

The stock market in the energy sector has at its core a trade-off between security and risk, driven by an extensive list of factors (such as oil and gas prices, exchange rates, or dividends disclosure) that, combined with the transition to clean energy by most of the companies, have been increasing its price volatility (Vrînceanu et al., 2020). Therefore, being able to precisely foresee stock price behaviour has become a growing necessity for the trading activity.

To this end, statistical models such as AR, ARMA, ARIMA (Menon et al., 2016; Pai & Lin, 2005), Support Vector Machines (SVM; Bao et al., 2004) and Logistic Regression (Gong & Sun, 2009) were first employed to, not only attempt to predict short-term stock prices, but also grasp the stock market's future trends. These models, while providing increased interpretability over complex models and being able to predict a short-term tendency for a specific company, struggle to capture non-linear relationships among the data and fail to adapt to the stock market's inherent volatility.

To address this limitation, advanced Machine Learning models based on Recurrent Neural Networks (RNN) such as Long Short-Term Memory (LSTM; Pawar et al., 2019; Wang et al., 2018) and Convolutional Neural Networks (CNN; Mehtab & Sen, 2020; Selvin et al., 2017) have been increasingly used.

RNN-based models introduced the capacity to store information regarding past values for longer periods of time, which enhanced the capacity to capture long-range dependencies among different time-steps and more accurately predict a stock price long-term trend. However, the longer time variation between two time-steps, the more prone most of these models became to suffer from exploding and vanishing gradients (Hochreiter & Schmidhuber, 1997), making them not suitable for Long Time-Series Forecasting (LTSF).

CNN-based models stand out from other approaches due to their capacity to find complex patterns hidden between data points, such as cyclical behaviors, local trends and anomalies. Furthermore, these models can identify these features on different time scales, for example daily, monthly and annual trends. These capabilities are visible in models such as TimesNet (Wu et al., 2023) and are instrumental in attempting to predict stock market's long-term trends. However, CNNs are designed to process spatial data, which can pose a problem when trying to capture complex relationships amongst distant time-steps.

In 2017, Vaswani and colleagues proposed the Transformer architecture, revolutionizing the performance capabilities of the Natural Language Processing (NLP) models by employing a self-attention mechanism to capture the dependencies between positions in long-range sequences, positional encoding to integrate the contextual information of each position in a sequence and a multi-head attention mechanism to process and apply the different operations to the sequence in parallel. These capabilities show an improvement in the efficiency and accuracy of the models on sequential data, including time series data.

Since the introduction of the Transformer architecture, multiple models have been developed in the field of NLP, but also in the Long Time Series Forecasting (LTSF) field. Models like Informer (Zhou et al., 2021), Autoformer (Wu et al., 2021), and PatchTST (Nie et al., 2022) succeeded in reducing the space and time complexity $O(L^2)$ of the original Transformer architecture, which resulted in a bottleneck for memory and computational effort. This was a primary limitation that, once surpassed, would provide these models with the capabilities to forecast a longer horizon of time series data with a higher accuracy than previously possible by the traditionally used models.

1.2. OBJECTIVES & EXPECTED CONTRIBUTIONS

Financial data forecasting is among the most complex and difficult time-series forecasting problems, due to its intricacies regarding volatility and non-linear relationships. Solutions inspired on Transformer mechanisms, such as Multi-head Attention and Self-Attention, have shown prevalence over solutions inspired in RNN and CNN for this specific setting (Zuo et al., 2020). This MSc Thesis seeks to explore and examinate the validity of these conclusions, by comparing actual Transformer-based models against models based on traditionally employed architectures, more specifically RNN and CNN. This comparison will be executed for the stock price forecasting problem, using Galp Energia S.A. as the subject of this experiment.

This study is expected to generate insights on how Transformer-based models sustain the volatility inherent to the stock price of an energy company and how it can affect predictions for a longer horizon, compared to Deep Learning models. Consequently, the outcome of this experiment is expected to produce evidence on whether Transformer-based models constitute the next generation of the time-series forecasting models.

2. RELATED WORK

Over the years, time-series forecasting has been evolving rapidly, including the stock price forecasting problem. This evolution encompasses the shift from statistical models to machine learning and deep learning models. Initially, models such as the Autoregressive Integrated Moving Average (ARIMA) were widely adopted for their effectiveness in representing linear relationships within a time-series. Additionally, their interpretability facilitated their usage and the results refinement. However, as the financial markets grew in complexity, these models developed limitations when attempting to capture non-linear patterns and accounting for the volatility brought by external factors.

These were limitations that Support Vector Machines (SVM) did not incorporate. This Machine Learning model offered increased capacity in capturing non-linear relationships compared to statistical models (Cortes & Vapnik, 1995; Smola & Schölkopf, 2004). Despite the capacity to do so, this model presented numerous limitations regarding efficiency in the training and testing stage, and the lack of computational efficiency when dealing with multiclass problems (Patle & Chouhan, 2013).

The introduction of Deep Learning models to the time-series context became a pivotal moment for stock price forecasting. Recurrent Neural Networks and RNN-based models, such as Long Short-Term Memory, were among the first to be applied, enhancing the capacity to capture long-range dependencies in sequential data (Gers et al., 2000; Hochreiter & Schmidhuber, 1997). However, when the forecasting horizon was too long, these models suffered from vanishing and exploding gradients.

These challenges were overcome by Convolutional Neural Networks, as well as hybrid models combining CNN and RNN, leveraging an improved capacity to identify complex patterns within stock price data (Kim & Kim, 2019; Patnaik et al., 2024; Wu et al., 2023. More recently, new architectures such as the Transformer, revolutionized the way sequential data is processed, introducing attention mechanisms capable of identifying complex long-term dependencies. This has created the necessity of using Transformer-based models to stock price forecasting, as well as hybrid models,

combining CNN and Transformer architectures (Vaswani et al., 2017; Zeng et al., 2023).

2.1. STATISTICAL MODELS

2.1.1. Auto Regressive Integrated Moving Average (ARIMA)

The Autoregressive Integrated Moving Average (ARIMA) model is a popular statistical method used in time series analysis and forecasting. It combines Autoregression (AR), Differencing (I), and Moving Average (MA) components to capture the linear relationships within time-series data (Box & Pierce, 1970). ARIMA models were firstly applied in stock price forecasting for their ability to handle stationary data and linear relationships among sequential data.

2.1.1.1. Model Components

Autoregressive – Indicates that the target variable is regressed on its own values, which means that it is assumed there is a dependency between the current time-step and the previous time-steps (lags). It is controlled by the order parameter p, which represents the number of lagged observations.

Integrated – This component represents the differencing of the observations, to meet the stationarity assumption of ARIMA. Differencing is the process of subtracting the previous observation from the current observation, ensuring that statistical properties, such as mean or variance, do not shift over time. The notation $I(d)$ denotes the differencing of order i.

Moving Average – This component represents the dependency between an observation and the residual error of the past forecast values (lagged values), using moving average. The notation $MA(q)$ denotes the moving average model of order q, where q specifies the number of lagged forecast errors.

2.1.1.2. Limitations

One of the defining features of this model is the stationarity assumption, which is not fulfilled due to the volatility present in financial data (Petrică et al., 2016). Additionally, while it is able to produce accurate short-term predictions, it struggles with long-term predictions. This may be due to the assumption of linear relationships among different

observations, which becomes invalid during market crashes or influence of external factors, common in the financial scenario (Liu, 2024).

2.2. MACHINE LEARNING MODELS

2.2.1. Support Vector Machines (SVM)

Support Vector Machines (SVM; Cortes & Vapnik, 1995) are machine learning models specialized in classification and regression problems by discovering a hyperplane that optimally separates classes or predicts sequential values (Smola & Schölkopf, 2004). SVM have been adapted for stock price forecasting due to their ability to capture non-linear relationships and generalize using unseen data.

2.2.1.1. Model Components

Kernel Function – SVM use a kernel function, responsible for mapping non-linear vectors into a higher-dimensional structure, making separation easier to perform. This kernel can commonly be linear, polynomial or Radial Basis Function (RBF).

Hyperplane – The Hyperplane is a decision surface represented in the high-dimensional feature space, responsible for separating classes by maximizing the margin between them (classification problems) or fitting the data (regression problems), while finding the function that minimizes error.

Support Vectors – These vectors are the data points closest to the Hyperplane and they play a critical role in finding its most suitable position and orientation.

2.2.1.2. Applications to Stock Price Forecasting

The inherent non-linearity, noise and volatility of the stock market make stock price forecasting a challenging and complex task. SVM can be a well-suited solution to this task. The usage of kernel functions allows the model to capture non-linear patterns in a more effective manner, even when compared to Deep Learning models (Kim, 2003). The flexibility of classification or regression provided by this model can be useful to predict future values (regression) and predict market trends (classification).

2.3. DEEP LEARNING MODELS

2.3.1. Long Short-Term Memory

The LSTM model, published in 1997 by Hochreiter and Schmidhuber, is a Recurrent Neural Network (RNN) based model, specifically designed to address the vanishing and exploding gradients issues exposed on the traditional RNN model. This model is characterized by the usage of memory cells and gate units, responsible for controlling the flow of information.

2.3.1.1. Memory Cells and Gate Units

Memory Cells - This is the core structure of the LSTM model. It is responsible for storing the temporal state of the network and, combined with gates, control the flow of information across it.

Input Gate - This gate controls whether new information should flow into the memory cell. This decision is executed by the input activation function and the previous hidden state.

Forget Gate - This gate controls which amount of past information should be removed from the cell state, resetting some parts of the memory cell.

Output Gate - This gate controls the amount of information present in the cell state that flow to the rest of the network.

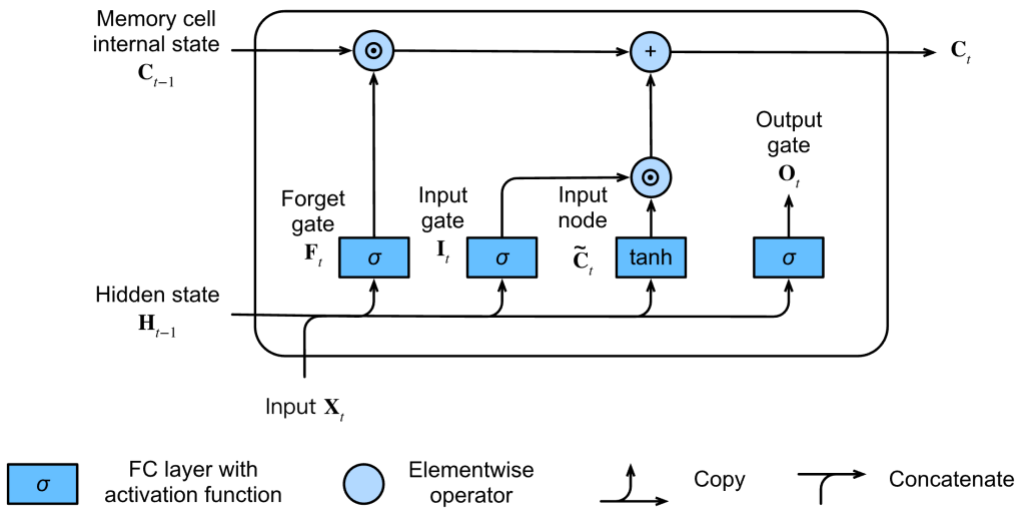

Figure 1 – LSTM Architecture (Hochreiter & Schmidhuber, 1997).

These multiplicative gate units allow the model to decide when to accept new information, when to forget old information and when to output information. This phenomenon enhances the LSTM capacity to capture long-term dependencies.

Furthermore, RNNs suffer from exponential growth or diminishing of error signals due to the continuous multiplication of weights (exploding and vanishing gradients, respectively). LSTMs employ a Constant Error Carousel (CEC), maintaining an effective learning across long series.

Overall, this model can outperform many traditional RNN architectures regarding the capacity to capture long-range dependencies. However, it can be more computationally expensive and requires careful hyperparameter tuning.

2.3.2. TimesNet

TimesNet is a model published in 2023 by Wu and colleagues, designed to execute various time-series tasks, including time-series forecasting. It leverages the multiple unique temporal patterns among the data by transforming it into a two-dimensional representation. By doing so, the model is able to capture intra-period and inter-period behaviours, providing a distinct approach to the identification of dependencies among different observations.

2.3.2.1. Temporal 2D-Variation Transformations

One of TimesNet fundamentals is the conversion of the one-dimensional time-series sequence into a two-dimensional representation. The underlying idea behind this process is to expose the cyclic patterns within the data. By reshaping the data into a 2D tensor, the model can identify the patterns within each period (intra-period) and between different periods (inter-period). This process is carried out in the TimesBlock structure.

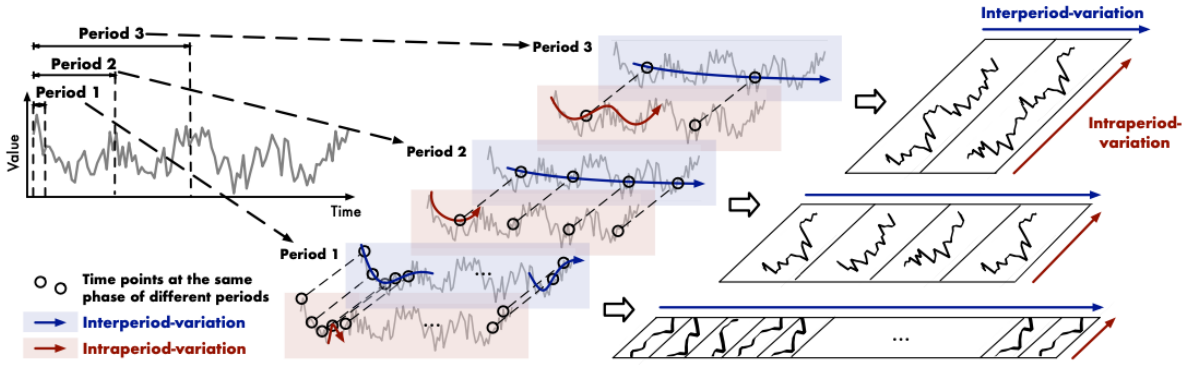

Figure 2 - Example of TimesNet 1D structure reshape into a 2D tensor (Wu et al., 2023). This operation allows for the identification of the intra-period and inter-period behaviors.

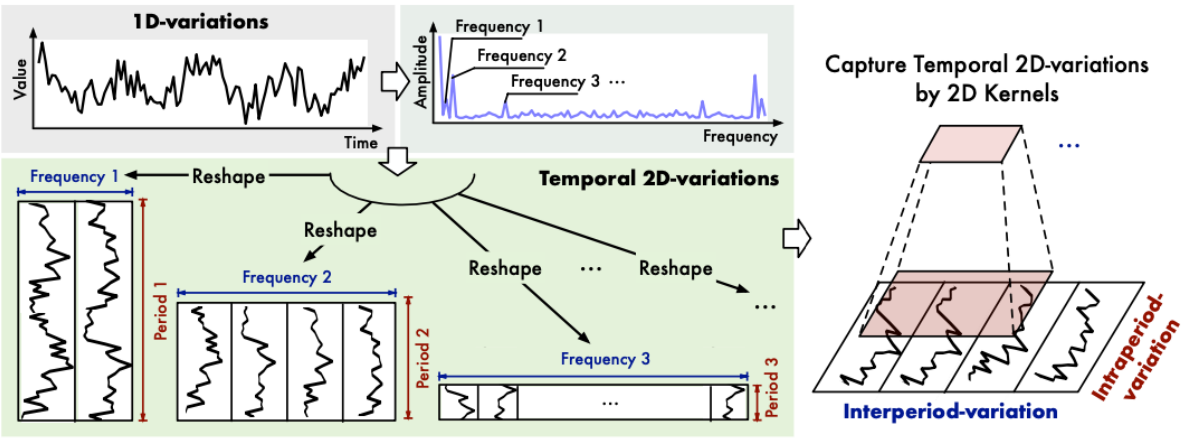

Figure 3 - Identification of the temporal variations by TimesNet (Wu et al., 2023).

2.3.2.2. TimesBlock Architecture Overview

The TimesNet model is composed by a stack of TimesBlock units, operating with residual connections, to promote the flow of information. There are three main processes involved in each TimesBlock unit:

Significant Period Identification - After reshaping the time series sequence to a 2D tensor, the model applies a Fast Fourier Transform (FFT). This operation is used to plot a signal into a representation of frequency and amplitude. After obtaining the frequency and amplitude of each period, the largest ones are considered the most relevant, since they represent the long-range dependencies present in the data.

Inception Block - Once the most relevant periods are identified, the 2D tensor is fed to the Inception Block. This module is the building block behind the well-acknowledged

computer vision model GoogLeNet, published in 2015. This CNN was employed due to its capacity to parse 2D data, such as images or, in this case, 2D tensors. The idea behind its use is to keep data sparse enough, so that the intra-periods and inter-periods become more easily identifiable.

Adaptive Aggregation - The outputs of the Inception Block are then reshaped back to a 1D representation. This is made possible by the Adaptive Aggregation mechanism that, inspired by Autoformer's Auto-Correlation Mechanism, aggregates the patterns with higher amplitude (representative of the relative importance of the periods) into a one-dimensional representation.

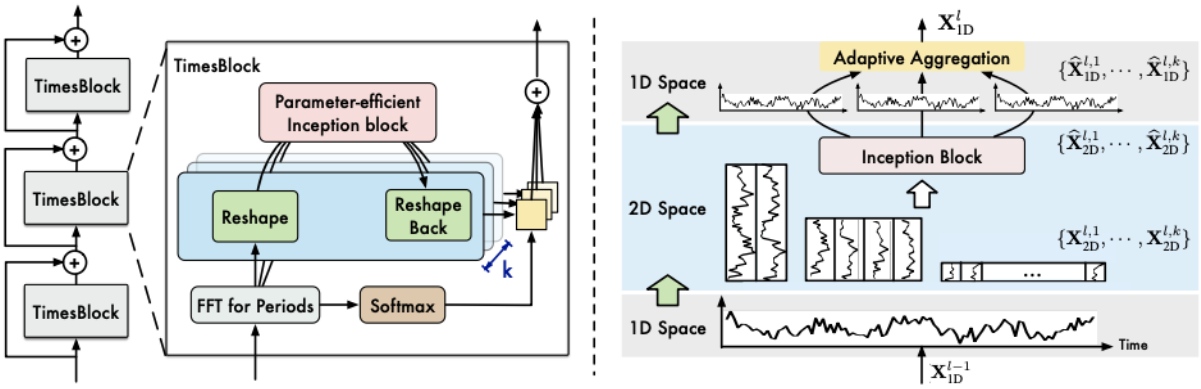

Figure 4 – TimesNet Architecture (Wu et al., 2023).

Additionally, the residual connections between TimesBlock units allow the model to adaptively reshape the 1D representations into 2D tensors based on learned periods. This design reduces the memory usage considerably, making TimesNet suitable for the LTSF problem setting.

2.4. TRANSFORMER MODELS

2.4.1. The Original Transformer

Introduced in 2017 by Vaswani and colleagues, the original transformer architecture is a *seq2seq* model designed for the problem settings of the Natural Language Processing field. *Seq2seq* is the nomenclature used for models that receive sequential data as input and generate sequential data as output. This is possible due to an encoder-decoder approach, where the encoder is responsible for mapping out the input, and the decoder is responsible for generating the output values of the model.

2.4.1.1. Architecture Overview

Encoder

The encoder is composed by a Multi-head Attention mechanism sub-layer and a simple feed-forward network sub-layer. Residual connections are applied around each sub-layer, immediately followed by layer normalization.

Decoder

The decoder structure is similar to the encoder structure, with the addition of a Multi-head Attention sublayer, performing over the encoded output from the encoder. The first Multi-head attention sub-layer is modified, using masking, ensuring that the predictions for a position can only be made using the encoder outputs of the earlier positions.

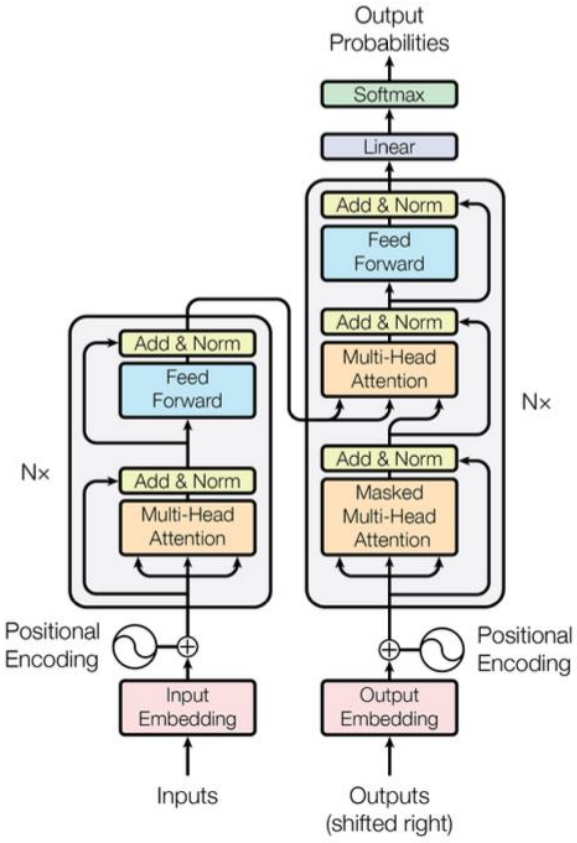

Figure 5 - Original Transformer Architecture (Vaswani et al., 2017).

2.4.1.2. Positional Encoding

The Transformer architecture does not use either convolution or recurrence, both of which are sequential approaches to deal with data. Hence, Positional Encoding is a method used to provide the model with the relative or absolute positions of input values. This process is added to the beginning of the encoder and decoder to provide context over the inputs of these stacks.

2.4.1.3. Self-Attention Mechanism

Contrarily to RNN-based models, the Transformer model does not consider each position to have the same relevance to the overall context of the sequence. As such, a Scaled Dot-Product Attention function is applied to the data. This function assigns a weight from 0 to 1 to each position, being 1 representative of the most relevance to the model.

2.4.1.4. Multi-head Attention

This mechanism allows the model to attend to different subsets of the data and compute the attention scores for those positions, in parallel. This abruptly decreases the computational effort, compared to traditionally used models, such as RNN or LSTM.

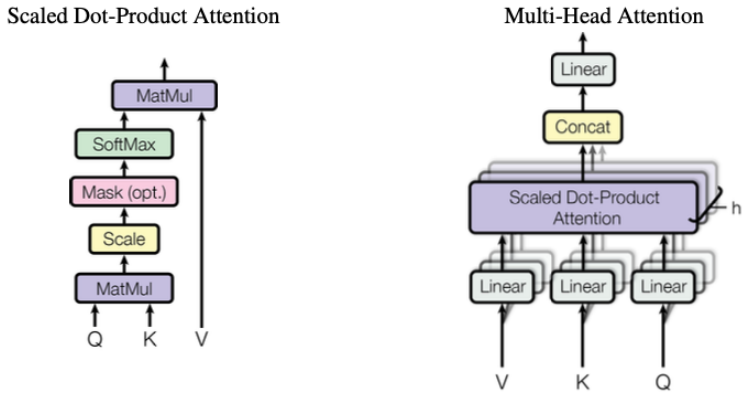

Figure 6 - Scaled Dot-Product Attention Mechanism and Multi-head Attention incorporated in the Original Transformer (Vaswani et al., 2017).

2.4.2. Informer

The Informer model was published in 2021 by Zhou and colleagues, and it was designed to forecast time-series data, more specifically in the LTSF problem setting. This model shares a similar architecture with the original Transformer model, using an encoder-decoder architecture. The original transformer Self-Attention mechanism

uses a Scaled Dot-Product operation with a time and memory complexity of $O = (L^2)$. The Informer model seeks to avoid this quadratic computation constraint by introducing the ProbSparse Attention mechanism. At the same time, the stack of layers on the encoder and decoder creates a memory usage bottleneck of $O(J * L^2)$. This constraint is addressed by the Self-Attention Distilling process.

2.4.2.1. ProbSparse Attention

The underlying idea behind ProbSparse attention is to maximize the efficiency of the attention mechanism by reducing the number of queries and keys involved in the process. This is possible due to the selection of the “dominant” queries.
A query is considered dominant when its attention probability is higher compared to its neighbouring queries.

2.4.2.2. Self-Attention Distilling

This technique aims to progressively consider solely the higher attention scores while simultaneously reducing the input sequence length. This effectively compresses the attention maps and further reduces computational effort and memory usage.
The self-attention distilling mechanism ensures that the model maintains significant performance levels even with reduced network complexity, thereby enabling the processing of considerably longer sequences than traditional models.

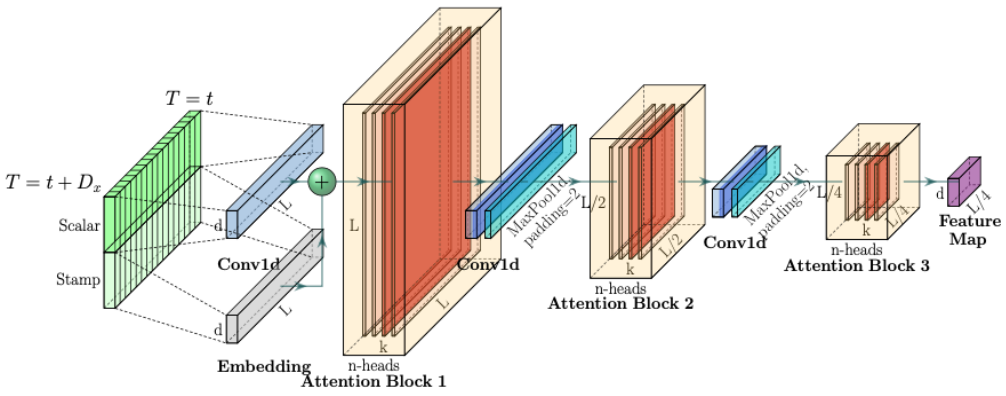

Figure 7 - Informer Encoder & Self-Distilling Process (Zhou et al., 2021).

2.4.2.3. Architecture Overview

Encoder

The encoder leverages the ProbSparse self-attention mechanism to handle long sequence inputs efficiently. It stacks multiple layers of self-attention and feed-forward

networks, with each layer distilling the input further to reduce dimensionality while holding the most relevant information.

Decoder

The decoder employs a Generative Inference mechanism responsible for generating a complete output sequence from a segment of the input sequence in a single time-step. This mechanism is capable of, not only reaching a higher prediction efficiency, but also mitigating the error accumulation issue commonly found in step-by-step approaches.

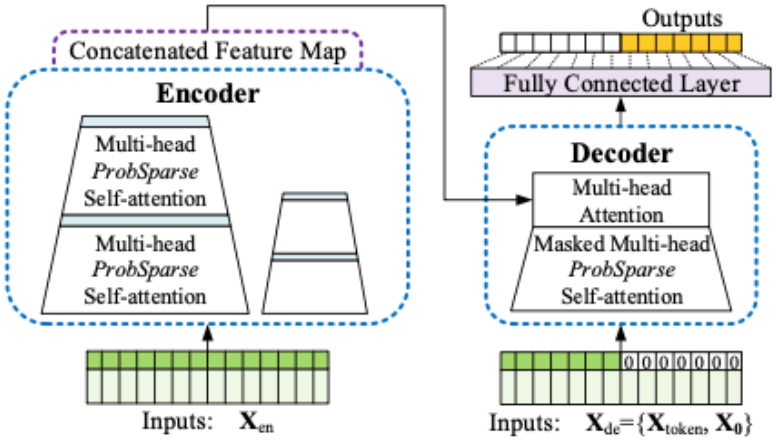

Figure 8 – Informer Architecture (Zhou et al., 2021).

2.4.3. Autoformer

The Autoformer is an encoder-decoder model published in 2021 by Wu and colleagues to address the long-range dependencies limitation, visible in the traditionally used in the time-series forecasting scenario. Traditional self-attention mechanisms, while effective for short-range dependencies, struggle to identify temporal patterns in longer sequences and to maintain computational efficiency in this context. This model employs a decomposition architecture and an auto-correlation mechanism to address these limitations.

2.4.3.1. Decomposition Block

The Autoformer model incorporates a Decomposition Block, responsible for separating the seasonal and trend-cyclical components from the series. It leverages the use of a

moving average to smooth out periodic fluctuations in the data, highlighting the long-term trends.

2.4.3.2. Auto-Correlation Mechanism

This mechanism was designed to address the memory and computational constraints of the attention mechanisms seen in other models. It takes into consideration the seasonal component separated in the Decomposition Block and computes the attention scores from data points of sub-series with the similar seasonal behaviours. This method can improve the accuracy of the model by capturing complex long-range dependencies more effectively, particularly when periodic patterns can be identifiable in the training data. Additionally, it achieves a memory and time complexity of $O(L\ log\ L)$ making it more efficient compared to the original attention mechanism.

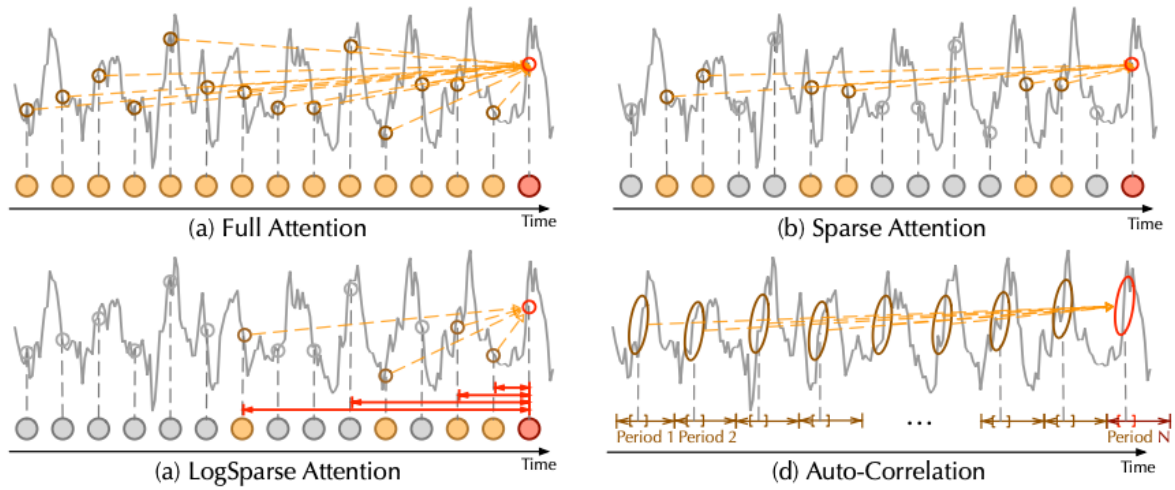

Figure 9 - Comparison between Autoformer Autocorrelation Mechanism (Wu et al., 2021) and Self-Attention Mechanisms used by other Transformer-based models.

2.4.3.3. Architecture Overview

Encoder and Decoder

The encoder takes as input the historical data from the series and uses Decomposition Blocks to separate the seasonal component from the trend-cyclical component, eliminating the latter. The decoder takes as input both the trend-cyclical and seasonal components from the series, refined through Decomposition Blocks, and utilizes the season information from the encoder to compute the attention scores in the Auto-

Correlation blocks. The refining process continues throughout the process to ensure that the seasonal information in clear of any trend-cyclical noise.

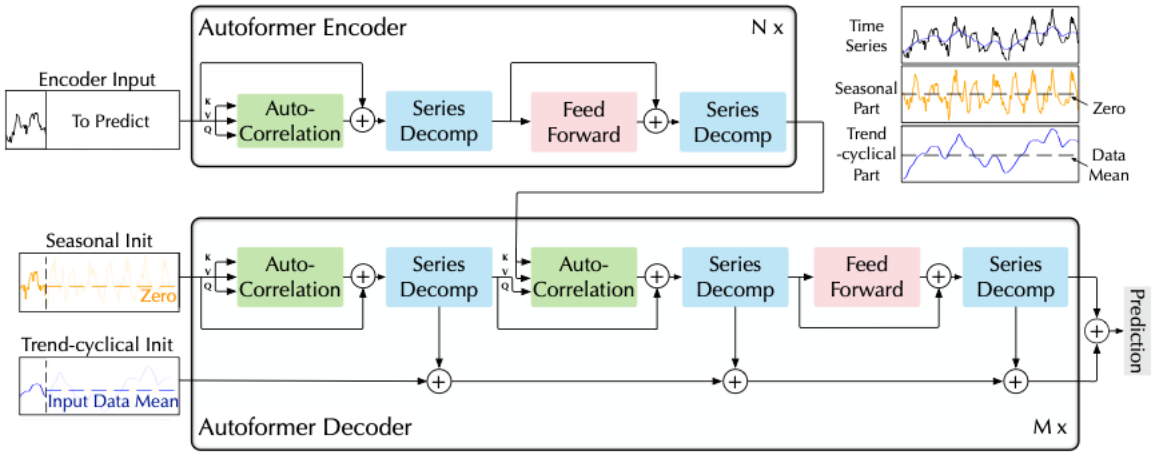

Figure 10 – Autoformer Architecture (Wu et al., 2021).

2.4.4. Patch Time-Series Transformer (PatchTST)

The PatchTST model, published in 2022 by Nie and colleagues, was developed based on the Transformer architecture, although with a diverse approach to the original attention mechanism. This model incorporates a Transformer encoder in its backbone and introduces the concept of Channel Independence and Patching to the time-series forecasting context.

2.4.4.1. Channel Independence

PatchTST employs a Channel Independence structure, where each time-series is independently processed by the Transformer backbone. This means that each input token fed to the Transformer Backbone contains only the information from a single channel.

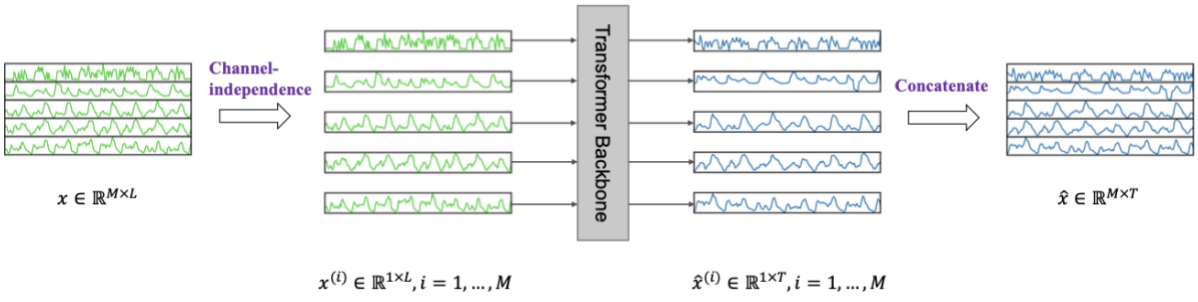

Figure 11 - PatchTST Channel Independence (Nie et al., 2022)

2.4.4.2. Patching

This concept involves dividing the series into sub-series and feeding them as input tokens to the Transformer encoder. This allows the model to consider the local semantic surrounding each data point, contrarily to the point-wise input tokens used in other Transformer-based models.

Patching not only enhances semantic comprehension and the capacity to capture dependencies between data points more effectively, but also reduces the number of input tokens. This ultimately reduces the memory and time constraints of the attention mechanism. Simultaneously, the model is allowed to access a longer historical sequence, which can be instrumental in the LTSF problem setting.

2.4.4.3. Transformer Backbone

The Transformer Backbone takes as input a time series, or channel, that will be processed through normalization and divided into patches (Patching). A position encoding is applied to the patches to maintain their temporal order before feeding them to the Transformer encoder. Afterwards, the encoder will compute the attention scores through the Multi-Head Attention block. These scores will later be used by a flatten linear head to produce the predictions.

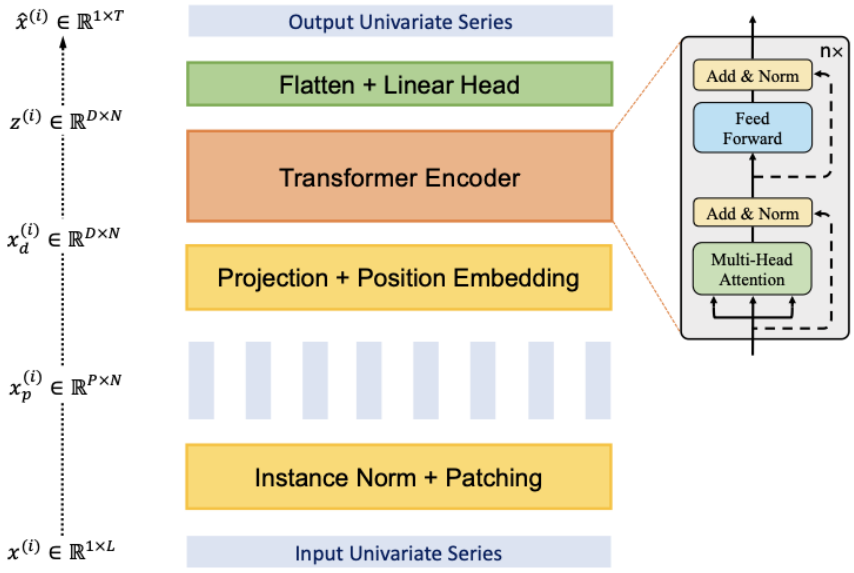

Figure 12 – PatchTST Transformer Backbone (Nie et al., 2022).

3. METHODOLOGY & EXPERIMENTAL SETUP

This chapter entails the process behind collecting, exploring and processing the data, with the finality of leveraging the potential of these models.

The data collected to develop this experiment consisted of Galp Energia S.A. closing stock price. It was obtained from Yahoo Finance, spanning the period from October 24th, 2006, to October 31st, 2023.

3.1. Data Collection and Initial Exploration

The data source for this research is Yahoo Finance, an established platform often used for historical financial data collection. The stock price data for Galp Energia S.A. was downloaded in CSV format, which included the following fields: Date, Open, High, Low, Close, Adjusted Close, and Volume. For the purposes of this study, only the 'Close' price was considered, since this is the target variable in a univariate forecasting scenario.

The historical data covers a span of seventeen years, since the company started being publicly traded, providing a comprehensive view of the stock's performance over time. This adds up to 4354 rows of stock price values.

3.2. Data Exploration and Processing

3.2.1. Missing Values and Outliers

Upon data importation, the time series was checked for missing values and outliers. While no missing values were found, it was observed that there were gaps corresponding to non-trading days, such as weekends and holidays. These time gaps could pose challenges for time series analysis and forecasting models, which require continuous data. To address this issue, a forward fill technique was employed. This method involves replacing missing values with the most recent available data point, thereby creating a continuous time series suitable for analysis.

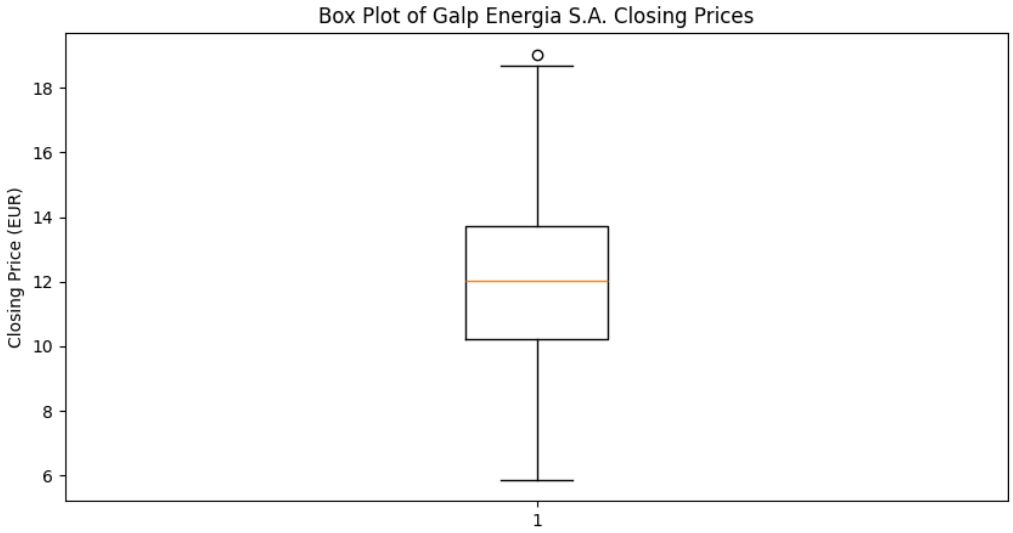

Figure 13 - Boxplot of target variable.

After plotting a boxplot for the target variable, it was possible to observe a single outlier. Since the data represents real variations of stock price and this data point is unlikely to be associated to a measuring error, it was not removed.

3.2.2. Tendency and Seasonality

Initial exploration on this subject involved plotting the variation of Galp Energia S.A.'s stock price over time, aimed to identify any obvious trends or seasonal patterns in the data. Furthermore, a Fast Fourier Transform (FFT) was applied to the time series data, to analyze the presence of tendencies and seasonality. The FFT technique transforms the time-series data into the frequency domain, simplifying the process of detecting any periodic patterns (Brigham & Morrow, 1967).

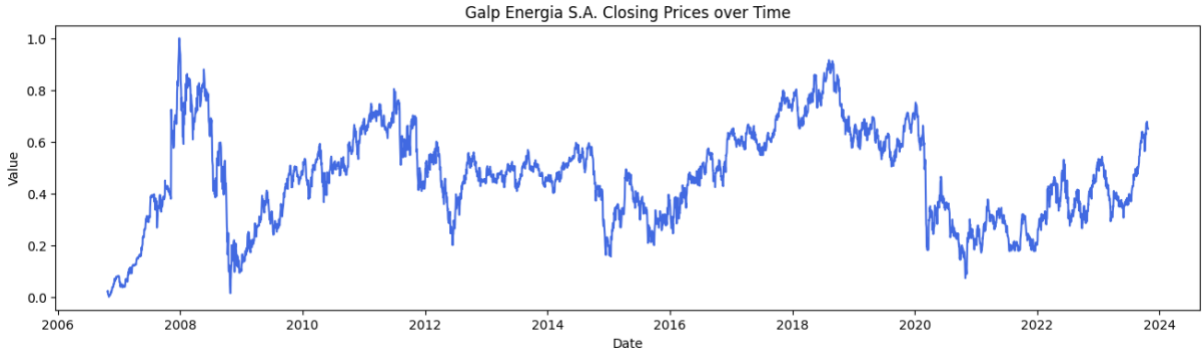

Figure 14 - Target variable distributed over time.

The figure represents the variation of the time series over time and does not indicate any signs of periodic behavior or clear tendencies. Hence, arose the necessity to further test the presence of these, using the FFT technique.

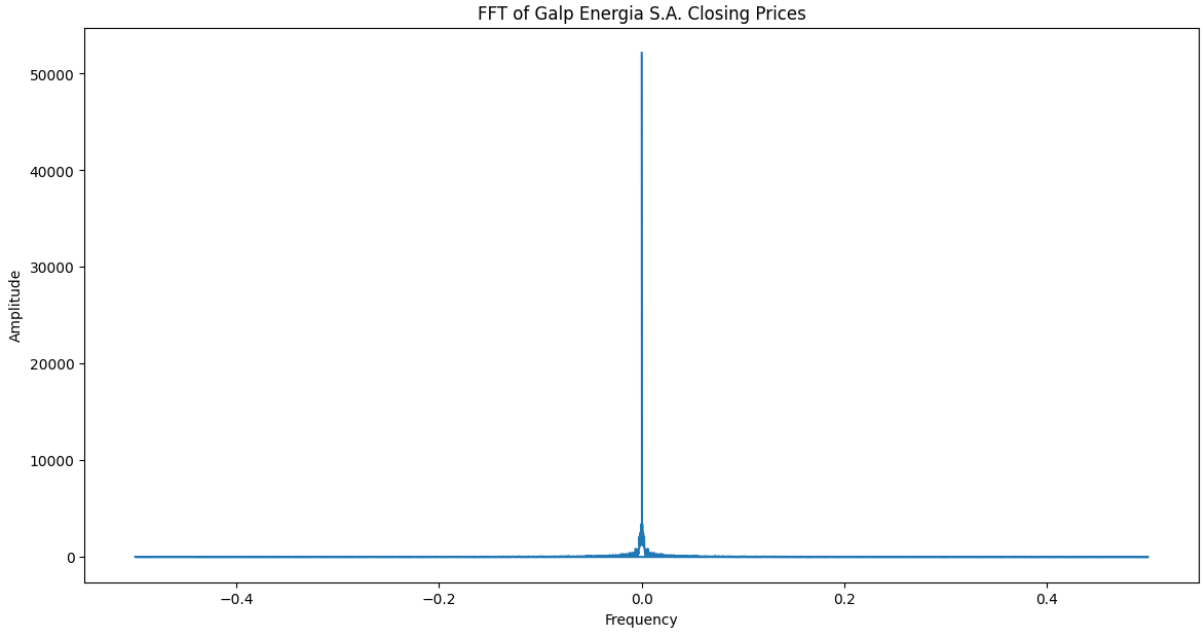

Figure 15 - FFT technique results.

The results represented in the figure show a large spike in the frequencies close to 0, while the rest of the frequencies remain flat. This suggests a dominancy of the mean level in the stock price, with no periodic fluctuations.

Consequently, the horizon settings considered in this study did not follow any behavioral features present in the data. Attending to the daily frequency of the target variable, the models' performance will be compared for the forecasting horizons of 14, 30 and 60 days (two weeks, one month and two months, respectively). These predefined horizons allow this study to converge on conclusions regarding long time-series forecasting without sacrificing computational capacity and falling into memory bottlenecks.

3.2.3. Data Normalization

The data exhibited large amplitude variations, which could adversely affect the model training process. To control these variations and promote an effective learning from the model, the MinMax Scaler was applied for normalization. The MinMax Scaler transforms the data by scaling each feature to a given range, in this case [0, 1]. This

scaling guarantees that all the data points are within a consistent range, facilitating the training process and improving the performance of the forecasting models.

3.3. DATA MODELLING

Before entering the modelling stage of this study, the data was subject to a widely employed Train-Test split method (Tan et al., 2021), where the Test dataset size corresponded to the forecast horizon and the Train dataset size corresponded to the remaining length of the initial dataset. Simultaneously, the Train dataset was split, using 80% for training and 20% for validation after training.

The models used in this study were tested using Nixtla's platform. This platform allows any user to train and test a variety of models open source, including Transformer-based models. For this study, the selected models were: Informer, Autoformer, PatchTST, TimesNet and LSTM.

The comparison of these models encompasses the consideration of multiple factors, such as computational effort, runtime, and configurability. Since these models can be computationally and time challenging while training and the configuration needed step-by-step adjustments to attain the best performance from each model, a structure of three rounds of testing was developed.

The first round of testing included three transformer-based models: Informer, Autoformer and PatchTST. This round had an exploratory and auxiliary purpose, where the results produced a "champion" model. This model served as a comparison standard for the next rounds of testing. Here, the models' configuration was gradually adjusted, attempting to reach an optimum stage between computational effort and performance, while simultaneously comparing the models according to the selected metrics.

The second round of testing included three models: TimesNet, LSTM and the "champion" model from the first round. This round had the purpose of comparing the models that do not benefit from the Transformer-based architecture with the models that benefit from such and arrive at a conclusion regarding their performances. The underlying expectation behind this round of experiments was to understand if the "champion" model, that outperformed the models in its class, is capable of continuing to outperform these models that benefit from a more traditional architecture.

The third round of testing included five models: Informer, Autoformer, PatchTST, TimesNet, LSTM. This final round of experiments had the purpose of an overall model comparison and reach final answers to the initial research questions. Additionally, the results from this comparison can provide a clearer picture of the discrepancies between the Transformer-based models as a class and the rest of the models included in this study.

4. RESULTS & DISCUSSION

4.1. First Round of Testing

The results portrayed in the first round of comparisons served as a starting point and benchmark for the potential of these models. Their configuration was iteratively adjusted so that, gradually, they could improve their performances.

For a forecasting horizon of 14 days, there is a noticeable difficulty of the models to adjust to the real values of the target variable. However, despite the overall poor results, Autoformer was able to mimic a similar pattern to the actual values in its prediction.

For a forecasting horizon of 30 days, the models struggled to make predictions close to the actual target values, especially the Informer and Autoformer models. PatchTST was capable of predicting the low-amplitude variations in the data, but showed clear limitations regarding the high-amplitude. Contrarily, the Autoformer showed a higher capacity to predict high-amplitude variations in the data, while low-amplitude variations showed to be more difficult to predict.

For a forecasting horizon of 60 days, there is an overall difficulty from the models to follow the high-amplitude variation seen on the target variable. However, the three models were able to produce accurate predictions for the first horizon values, constituted by low-amplitude variations.

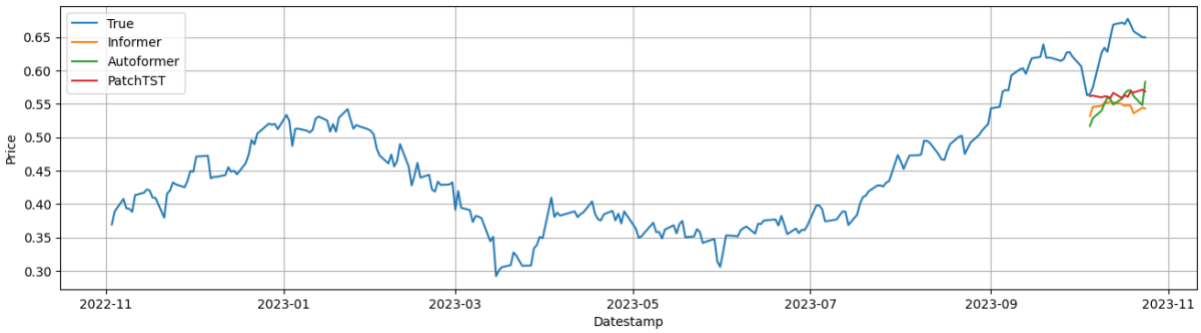

Figure 16 - Forecasting results for a horizon of 14 days (First Round).

Model	RMSE	MAE	MSE
Informer	0.100968	0.095949	0.010195
Autoformer	0.090922	0.088143	0.008267
PatchTST	0.085375	0.078755	0.007289

Table 1 - Computed metrics for a horizon of 14 days (First Round).

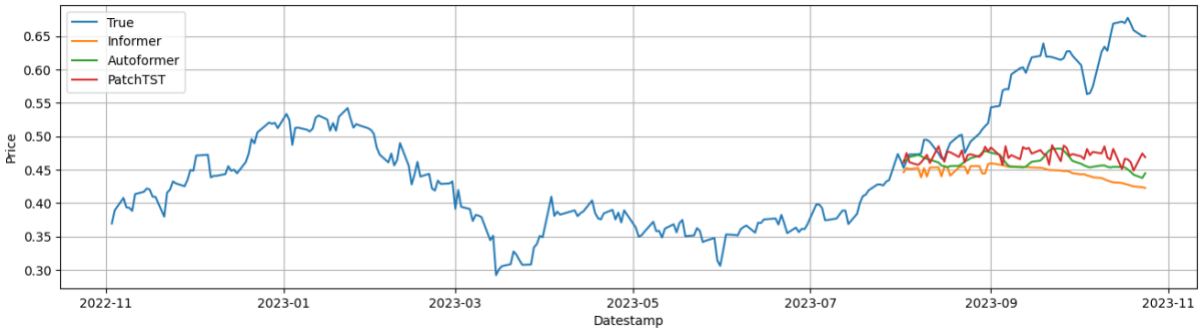

Figure 17 - Forecasting results for a horizon of 30 days (First Round).

Model	RMSE	MAE	MSE
Informer	0.106449	0.102356	0.01133
Autoformer	0.178102	0.175249	0.031720
PatchTST	0.045339	0.038716	0.002056

Table 2 - Computed metrics for a horizon of 30 days (First Round).

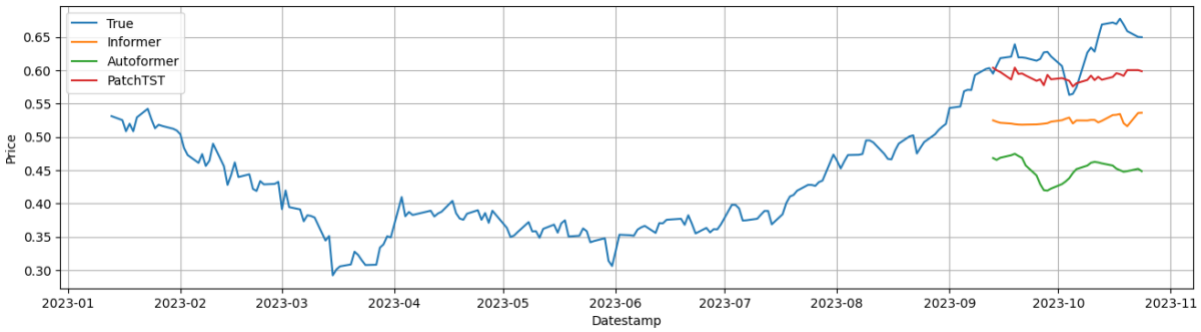

Figure 18 - Forecasting results for a horizon of 60 days (First Round).

Model	RMSE	MAE	MSE
Informer	0.143295	0.122198	0.020533
Autoformer	0.128846	0.106654	0.016601
PatchTST	0.119087	0.097784	0.014182

Table 3 - Computed metrics for a horizon of 60 days (First Round).

After carefully examining the performance results presented in the figures and the computed metrics for each scenario, it was clear that PatchTST was the model that, under these premature conditions, performed best. Consequently, the “champion” model used as a Transformer-based standard for the next round was PatchTST.

4.2. SECOND ROUND OF TESTING

In this round, PatchTST was compared to models based on other architectures. More specifically, LSTM (RNN-based) and TimesNet (CNN-based). In this stage, it is already possible to start retrieving valuable insights on whether there is a benefit to the usage of the Transformer architecture in time-series forecasting, compared to other architectures already traditionally used. It is, yet, worthwhile mentioning that the configuration of these models was being gradually updated and, for that reason, the results might not capture the potential of these models, to their full extent.

For the forecasting horizon of 14 days, TimesNet was able to produce predictions significantly close to actual values of the target variable, for the majority of the horizon. Contrarily, the LSTM and PatchTST models had clear difficulties producing accurate predictions and adapt to the high-amplitude variation period described in the data.

For the forecasting horizon of 30 days, there was an attempt to follow the target variable’s behavior by PatchTST that, while not being able to produce identical predictions to the real values, was able to capture and follow the trend of the target variable. The LSTM and TimesNet models revealed a considerable difficulty in capturing any type of the target variable’s behavior and produced inaccurate predictions.

For the forecasting horizon of 60 days, PatchTST yielded results very close to the real values, for the portion of the horizon with low-amplitude variations. However, it

revealed ineffective to capture the high-amplitude variations of the target variable. The LSTM and TimesNet models were unable to produce accurate predictions in any part of the horizon.

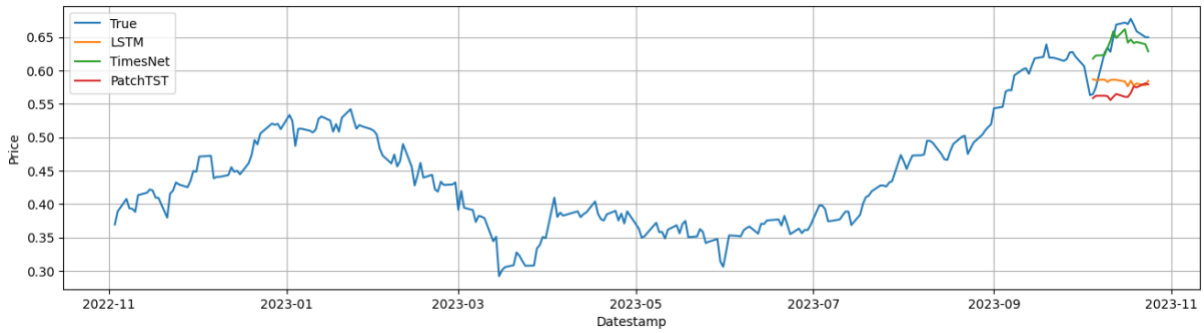

Figure 19 - Forecasting results for a horizon of 14 days (Second Round).

Model	RMSE	MAE	MSE
LSTM	0.068871	0.063827	0.004743
TimesNet	0.025762	0.021027	0.000664
PatchTST	0.082606	0.076295	0.006824

Table 4 - Computed metrics for a horizon of 14 days (Second Round).

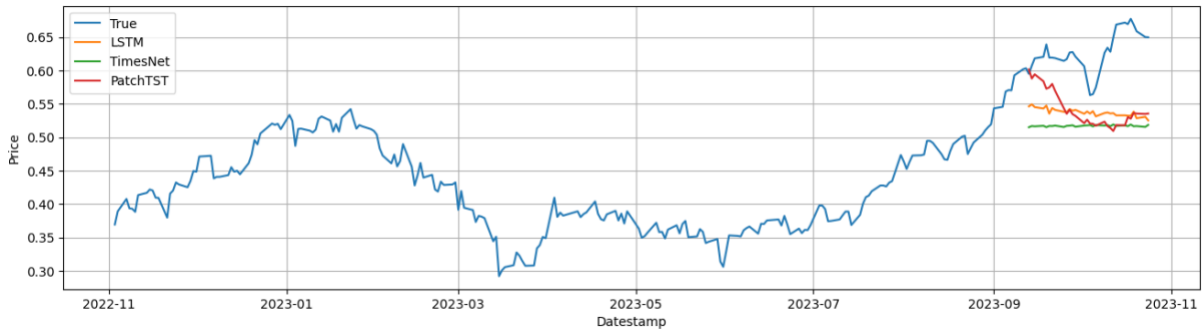

Figure 20 - Forecasting results for a horizon of 30 days (Second Round).

Model	RMSE	MAE	MSE
LSTM	0.094603	0.088756	0.008950
TimesNet	0.113374	0.109271	0.012854
PatchTST	0.094903	0.084899	0.009007

Table 5 - Computed metrics for a horizon of 30 days (Second Round)

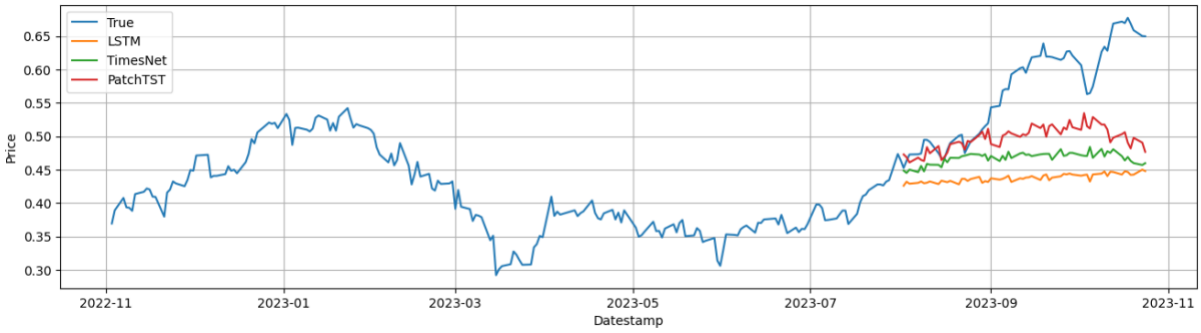

Figure 21 - Forecasting results for a horizon of 60 days (Second Round).

Model	RMSE	MAE	MSE
LSTM	0.145415	0.130364	0.021146
TimesNet	0.119879	0.100550	0.014371
PatchTST	0.093146	0.072802	0.008676

Table 6 - Computed metrics for a horizon of 60 days (Second Round).

4.3. THIRD ROUND OF TESTING

This was the final stage of testing, aiming to retrieve the final conclusions regarding the research questions of this study. This comparison incorporated all five models included in this study so far, stacking all models against each other, after manually tuning each model's configuration towards an improved performance.

For the forecasting horizon of 14 days, TimesNet and LSTM were the models with the most accurate predictions. TimesNet, specifically, had the capacity to successfully follow the variations of the target variable, as opposed to the rest of the models that could not predict a variation in the data.

For the forecasting horizon of 30 days, there was a visible difficulty of the TimesNet and LSTM models to represent any type of variation or tendency in their predictions. Contrarily, the Informer model was able to predict the overall tendency of the target variable and generate accurate predictions. The PatchTST and Autoformer models, while not producing the most accurate predictions, were effective to predict the variation of the target variable.

For the forecasting horizon of 60 days, Informer and LSTM showed the greatest difficulties to generate accurate predictions of the target variable, compared to the rest of the models. Simultaneously, Autoformer, TimesNet and PatchTST were effective to predict the target variable values in the portion of the data characterized by low-amplitude variations. Regarding the high-variation portion of the data, it is possible to identify an effort to predict the overall tendency of the target variable by PatchTST and TimesNet.

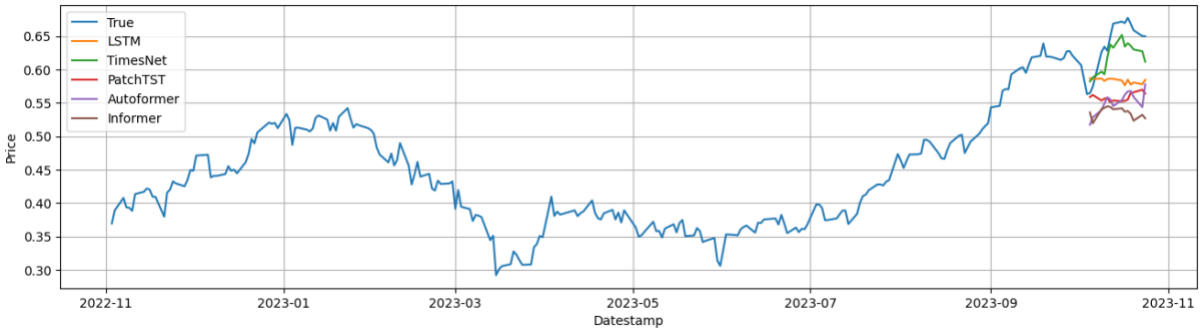

Figure 22 - Forecasting results for a horizon of 14 days (Third Round).

Model	RMSE	MAE	MSE
LSTM	0.068871	0.063827	0.004743
TimesNet	0.028778	0.026823	0.000828
PatchTST	0.090869	0.083921	0.008257
Autoformer	0.093463	0.090509	0.008735
Informer	0.111381	0.106560	0.012406

Table 7 - Computed metrics for a horizon of 14 days (Third Round).

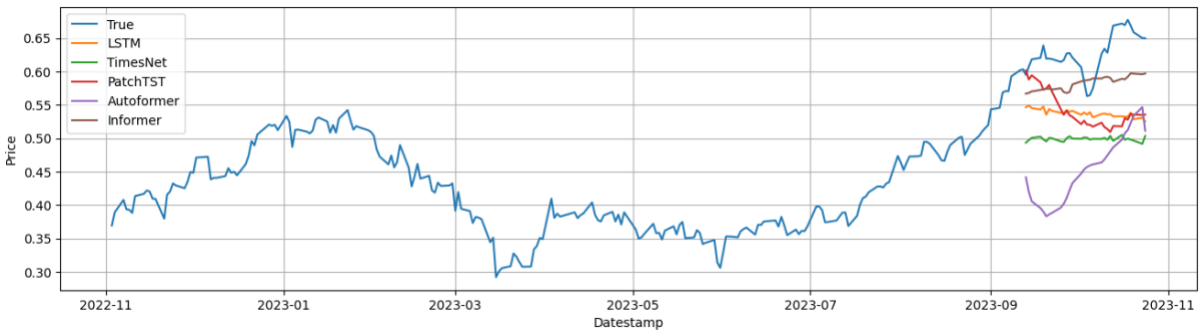

Figure 23 - Forecasting results for a horizon of 30 days (Third Round).

Model	RMSE	MAE	MSE
LSTM	0.094603	0.088756	0.008950
TimesNet	0.130981	0.127436	0.017156
PatchTST	0.094903	0.084899	0.009007
Autoformer	0.179449	0.174480	0.032202
Informer	0.052282	0.048108	0.002733

Table 7 - Computed metrics for a horizon of 30 Days (Third Round)

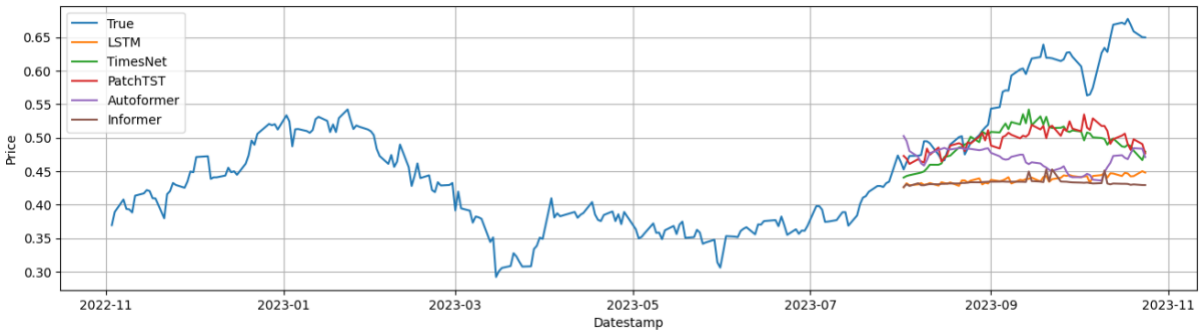

Figure 24 - Forecasting results for a horizon of 60 days (Third Round).

Model	RMSE	MAE	MSE
LSTM	0.145415	0.130364	0.021146
TimesNet	0.094295	0.074814	0.008892
PatchTST	0.093146	0.072802	0.008676
Autoformer	0.127230	0.104899	0.016187
Informer	0.150786	0.134554	0.022736

Table 8 - Computed metrics for a horizon of 60 days (Third Round).

5. CONCLUSIONS & FUTURE WORK

This study compared Transformer-based models to more commonly used Deep Learning models, aiming to understand whether the Transformer architecture could sustain the volatility and complex dependencies present in stock price data with a higher effectiveness, compared to the architectures used so far.

Despite the difficulty to separate these models in a shorter horizon context, it was possible to identify a clear growing discrepancy between the Transformer-based models and the traditionally used models regarding performance, once the horizon extended. PatchTST was particularly able to consistently maintain its accuracy, regardless of the forecasting length. TimesNet's results are also worth noticing, as they were comparable to the ones produced by PatchTST, though not as consistent as shown in the second round of testing. LSTM and TimesNet often produced more accurate results in shorter forecasting horizons, but the error metrics increased exponentially as the forecasting horizon grew, making them less suitable for long time-series forecasting.

Another objective of this study was to gain insights on whether the Transformer-based model class was the next step on the evolution of the time-series forecasting models. This would be verified if, throughout this experiment, the performance of Informer, Autoformer and PatchTST were comparable. Although the Informer and Autoformer models produced results comparable to PatchTST, this was only true for shorter horizons. As a class, there was a definite performance gap between PatchTST and the remaining two, as represented in the first and third round of testing. Furthermore, both TimesNet and LSTM have shown to perform above Informer and Autoformer metrics-wise.

Overall, it was possible to observe the superiority of PatchTST (Transformer-based) on the task of stock price forecasting, compared to TimesNet (CNN-based) and LSTM (RNN-based). However, this does not translate in a superiority of the Transformer architecture over the most commonly employed.

In the future, it would also be pertinent to compare these models on their efficiency, specifically on their runtime and memory usage. It is known that the Transformer-based models try to address this issue by reducing the memory time complexity, originated

by the original transformer. However, to do so, it creates a tradeoff between efficiency and performance, where a highly efficient model may not be able to identify complex patterns in data as effectively as a less efficient model. Additionally, this comparison could potentially be pertinent in an environmental context, by calculating the amount of Carbon Dioxide (CO2) released in the training process of each model.

In order to broaden the scope of this comparison even more and assert the Transformer-based models place amongst the time-series forecasting models, there are some model classes to be added to a study such as this one. Statistical models like ARIMA and Machine Learning models like SVM are merely examples of models used in time-series forecasting scenario that could be helpful in comprehending how the Transformer-based models stack against their competition in the time-series forecasting context.

6. LIMITATIONS

Despite being possible to test and compare these models and successfully arrive at conclusions, there were some limitations to the robustness of this study.

According to the literature, these models could be tested in the context of multivariate time-series forecasting. However, this idea became difficult to replicate in a practical scenario. Most of these models' original repositories were not available to the public, apart from the Nixtla framework (from where all the models included in this study were retrieved). Nixtla´s platform, Nixtlaverse, includes an extensive variety of models available open source, but it is constantly under development. Initial model documentation research revealed the possibility of including exogenous variables in the study, but after testing the models with exogenous variables that could improve the models' performance, it became known that these models were pending an update to make this feature available.

The models researched in this study would also benefit from a higher volume of data and the presence of tendency and seasonality. This would allow the models to more easily identify complex patterns and relationships between data points which would, ultimately, result in better predictions in a longer forecasting horizon. This would be verifiable especially for architectures that rely on finding patterns by isolating frequencies and seasonal patterns, such as the Autoformer and the TimesNet models.

Additionally, the computational power available for this experiment did not allow me to extend the forecasting horizon as initially intended. The longer the horizon, the higher the number of parameters of each model. As such, this study had to consider horizon values that could prove the benefit of the Transformer-based architecture, without compromising computational capacity. Furthermore, the testing stage of this study would significantly benefit from a hyperparameter tuning, that would also generate memory and computational constraints.

This research would also considerably benefit from the broadening of the models' comparison, by including other model classes, such as statistical models, such as ARIMA, and machine learning models, such as SVM. However, the inclusion of additional models would potentially result in memory and computational constraints.

BIBLIOGRAPHICAL REFERENCES

Bao, Y., Lu, Y., & Zhang, J. (2004). Forecasting Stock Price by SVMs Regression. In C. Bussler & D. Fensel (Eds.), *Artificial Intelligence: Methodology, Systems, and Applications.* Springer. https://doi.org/10.1007/978-3-540-30106-6_30

Box, G. E. P., & Pierce, D. A. (1970). Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models. *Journal of the American Statistical Association*, *65*(332), 1509–1526. https://doi.org/10.1080/01621459.1970.10481180

Brigham, E. O., & Morrow, R. E. (1967). The fast Fourier transform. *IEEE Spectrum*, *4*(12), 63–70. https://doi.org/10.1109/MSPEC.1967.5217220

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, *12*(10), 2451–2471. https://doi.org/10.1162/089976600300015015

Gong, J., & Sun, S. (2009). A New Approach of Stock Price Prediction Based on Logistic Regression Model. *2009 International Conference on New Trends in Information and Service Science*, 1366–1371. https://doi.org/10.1109/NISS.2009.267

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Kim, K. J. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1–2), 307–319. https://doi.org/10.1016/S0925-2312(03)00372-2

Kim, T., & Kim, H. Y. (2019). Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. *PLOS ONE*, *14*(2), e0212320. https://doi.org/10.1371/journal.pone.0212320

Liu, J. (2024). Navigating the Financial Landscape: The Power and Limitations of the ARIMA Model. Highlights in *Science, Engineering and Technology*, 88, 747-752. https://doi.org/10.54097/9zf6kd91

Mehtab, S., & Sen, J. (2020). Stock Price Prediction Using CNN and LSTM-Based Deep Learning Models. *2020 International Conference on Decision Aid Sciences and Application (DASA)*, 447–453. https://doi.org/10.1109/DASA51403.2020.9317207

Menon, V. K., Chekravarthi Vasireddy, N., Jami, S. A., Pedamallu, V. T. N., Sureshkumar, V., & Soman, K. P. (2016). Bulk price forecasting using spark over NSE data set. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9714 LNCS*, 137–146. https://doi.org/10.1007/978-3-319-40973-3_13

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2022). *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers* [Conference Paper]. ICLR 2022. https://doi.org/10.48550/arXiv.2211.14730

Pai, P. F., & Lin, C. S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*, *33*(6), 497–505. https://doi.org/10.1016/j.omega.2004.07.024

Patle, A., & Chouhan, D.S. (2013). SVM kernel functions for classification. *2013 International Conference on Advances in Technology and Engineering (ICATE)*, 1-9. doi: 10.1109/ICAdTE.2013.6524743

Patnaik, D., Rao, N. V. J., Padhiari, B., & Patnaik, S. (2024). Optimised hybrid CNN-LSTM model for stock price prediction. *International Journal of Management and Decision Making*, *23*(4), 438–460. https://doi.org/10.1504/IJMDM.2024.139387

Pawar, K., Jalem, R.S., Tiwari, V. (2019). Stock Market Price Prediction Using LSTM RNN. In V Rathore, M. Worring, D. Mishra, A. Joshi & S. Maheshwari (Eds.), *Emerging Trends in Expert Applications and Security. Advances in Intelligent Systems and Computing*. Springer. https://doi.org/10.1007/978-981-13-2285-3_58

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1643–1647. https://doi.org/10.1109/ICACCI.2017.8126078

Sindhu, M.I. (2014). Macroeconomic Factors do Influencing Stock Price: A Case Study on Karachi Stock Exchange. *Journal of economics and sustainable development, 5*, 114-124. https://iiste.org/Journals/index.php/JEDS/article/view/12581/12899

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222. https://doi.org/10.1023/B:STCO.0000035301.49549.88

Tan, J., Yang, J., Wu, S., Chen, G., & Zhao, J. (2021). *A critical look at the current train/test split in machine learning.* Arxiv. https://doi.org/10.48550/arXiv.2106.04525

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 6000-6010). Curran Associates, Inc. https://doi.org/10.48550/arXiv.1706.03762

Vrînceanu, G., Horobeţ, A., Popescu, C., & Belaşcu, L. (2020). The Influence of Oil Price on Renewable Energy Stock Prices: An Analysis for Entrepreneurs. *Studia Universitatis Vasile Goldis Arad, Economics Series*, *30*(2), 24–35. https://doi.org/10.2478/sues-2020-0010

Wang, Y., Liu, Y., Wang, M., & Liu, R. (2018). LSTM Model Optimization on Stock Price Forecasting. *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, 173–177. https://doi.org/10.1109/DCABES.2018.00052

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., & Long, M. (2023). *Timesnet: Temporal 2D-Variation Modeling for General Time Series Analysis* [Conference Paper]. ICLR 2023, Rwanda. https://openreview.net/pdf?id=ju_Uqw384Oq

Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang & J. Wortman-Vaughan (Eds.), *NIPS '21: Proceedings of the 35th International Conference on Neural Information Processing Systems* (pp.22419-22430). Curran Associates Inc. https://dl.acm.org/doi/10.5555/3540261.3541978

Zeng, Z., Kaur, R., Siddagangappa, S., Rahimi, S., Balch, T., & Veloso, M. (2023). *Financial Time Series Forecasting using CNN and Transformer* [Conference Paper]. AAAI 2023, Washington. https://doi.org/10.48550/arXiv.2304.04912

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(12), 11106-11115. https://doi.org/10.1609/aaai.v35i12.17325

Zuo, S., Jiang, H., Li, Z., Zhao, T., & Zha, H. (2020). Transformer Hawkes Process. In H. Daumé & A. Singh (Eds.), *ICML'20: Proceedings of the 37th International Conference on Machine Learning* (pp.11692–11702). JMLR.org. https://dl.acm.org/doi/10.5555/3524938.3526022

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa