Lisbon School of Economics & Management
Universidade de Lisboa

MESTRADO EM

MÉTODOS QUANTITATIVOS PARA A DECISÃO ECONÓMICA E EMPRESARIAL

TRABALHO FINAL DE MESTRADO

RELATÓRIO DE ESTÁGIO

BITCOIN TIME SERIES FORECASTING WITH EXOGENOUS FACTORS: DEEP LEARNING APPROACH

ANDRÉ CATARINO

ORIENTAÇÃO:

PROF. CARLOS J. COSTA

JOÃO PEDRO ALVES CRUZ LIMA

OUTUBRO-2023

Acknowledgments

I would like to acknowledge everyone who has played an important role during my academic accomplishments.

Firstly, I would like to thank my family, who have always been there with unconditional support.

Secondly, I would like to thank my supervisors, Professor Carlos Costa, and João Pedro Lima, for their guidance and support.

Last but not least, I would like to thank my closest friends who have been with me all these years.

Thank you all.

Resumo

Ao longo dos últimos anos, a tecnologia subjacente a ativos digitais tem sido alvo de grandes desenvolvimentos, o que resultou no aumento da sua presença em transações financeiras mundiais, proporcionando um valor crescente para a sociedade, servindo como meio de troca e como uma classe de ativos de investimento.

A modelação dos preços de ativos digitais é uma tarefa relevante para os investidores institucionais e de retalho, contribuindo para uma informada tomada de decisão numa classe de ativos altamente volátil. Apesar da relevância em prever as fortes flutuações observadas nesta classe de ativos, esta tarefa é extremamente complexa e depende de múltiplos fatores exógenos, tais como a rede blockchain, tendências de mercados financeiros e dados macroeconómicos.

Como simples métodos estatísticos não são capazes de capturar a complexidade das dependências temporais, investigadores a recorrem a algoritmos avançados de aprendizagem de máquina e aprendizagem profunda para resolver este problema séries temporais não estacionárias.

Este trabalho resulta de um estágio realizado na Klever em parceria com o ISEG no âmbito do Mestrado em Métodos Quantitativos para a Decisão Económica e Empresarial. Apresentamos uma metodologia para a construção de modelos de aprendizagem profunda *sequence-to-vector* para a previsão do preço, do retorno e do estado direcional do Bitcoin. Recorrendo a um completo sistema de engenharia de atributos, esta investigação alcança uma precisão de até 90,9% para o estado direcional, um *MAPE* de até 1.74% para a previsão do preço e 0.11 de *MAE* para a previsão do retorno, alcançando melhores resultados que o modelo de referência respetivo a cada problema.

Palavras-chave: aprendizagem profunda, séries temporais, bitcoin, fatores exógenos, engenharia de atributos.

Abstract:

Over the last years, major developments have been made in cryptocurrency technology, resulting in their increased presence in worldwide financial transactions, providing increasing value for society by serving as a means of exchange and investment asset class.

Modeling cryptocurrency prices is relevant for institutional and retail investors, contributing to informed decision-making in a highly volatile asset class. Despite the importance of forecasting the steep fluctuations observed in this asset class, this task is extremely complex and relies on multiple exogenous factors such as the blockchain network, market trends, and macroeconomic data.

As simple statistical methods are unable to capture the complexity of temporal dependencies, researchers are turning to advanced machine learning and deep learning algorithms to tackle this non-stationary time series problem.

This work is the result from an internship carried out at Klever in collaboration with ISEG within the scope of the Master of Quantitative Methods for Economic and Business Decision. We present a methodology for building sequence-to-vector deep learning models to predict the price, return, and directional state of Bitcoin. Leveraging a comprehensive feature engineering system, this research achieves an accuracy up to 90.9%, a MAPE up to 1.74% for price prediction, and up to 0.11 MAE for return prediction, surpassing each task's respective baseline model.

Keywords: deep learning, time series, bitcoin, exogenous factors, feature engineering.

Table of Contents

Acknowledgments I

Resumo II

Abstract: IV

Table of Contents V

Glossary VIII

List of Figures X

List of Tables XII

1 Introduction 1

1.2 Background 1

1.3 Problem Statement 1

1.4 Research Methodology 3

1.5 Outline 3

2 Literature Review 4

2.2 Time Series Forecasting 4

2.3 Related Work 5

2.4 Feature Engineering for Time Series 7

2.5 Gradient-Based Learning 9

2.6 Activation Functions 10

2.7 Deep Learning Forecasting Models 11

- RNN (Recurrent Neural Network) 11
- LSTM (Long Short-Term Memory) 12
- GRU (gated recurrent unit) 13
- BiLSTM (Bidirectional Long Short-Term Memory) 14
- CNN (Convolutional neural networks) 15
- Attention Mechanism 16

- TCN (Temporal Convolutional Network) 17
- Transformer 18
- Stacking Ensemble 20
- Bias-Variance Trade-Off 20
- Overfitting Mitigation 21

2.8 Evaluation Metrics 23

3 Methodology 25

3.1 Data Collection 26

3.2 Data Preprocessing 27

3.3 Impact of Feature Engineering in LSTM Predictive Performance 31

3.4 Data Preparation 33

3.5 Deep Learning Models 35

3.6 10.6 Model Training 44

4 Experimental Results 45

4.1 Model Performance 45

5 Discussion 53

6 Conclusion 54

6.1 Findings 54

6.2 Implications 55

6.3 Limitations 55

6.4 Future Work 56

References 58

Appendix 65

Glossary

ADAM	Adaptive Moment Estimation
ADF	Augmented Dickey-Fuller
AE	Autoencoder
ALSTM	Attention-based Long Short-Term Memory
ANN	Artificial Neural Network
ARIMA	Auto Regressive Integrated Moving Average
BiLSTM	Bidirectional Long Short-Term Memory
BTC	Bitcoin
CNN	Convolutional Neural Network
CRISP-DM	Cross-Industry Standard Process for Data Mining
GRU	Gated Recurrent Unit
KNN	K-nearest neighbor
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
PC	Principal Component
PCA	Principal Component Analysis
RFE	Recursive Feature Elimination
RNN	Recurrent Neural Network
TCN	Temporal Convolutional Network

List of Figures

Figure 1: BTC autocorrelation visualization with a lags value of 50 2

Figure 2: Autoencoder representation 9

Figure 3: Linear vs nonlinear dimensionality reduction 9

Figure 4: Gradient descent algorithm 10

Figure 5: Activation functions 11

Figure 6: Recurrent neural network architecture. Source: (Biswas et al., 2021) 12

Figure 7: Many-to-one relationship suitable to predict one step ahead. 12

Figure 8: LSTM cell. 13

Figure 9: GRU cell. 14

Figure 10: Unfolded architecture of Bidirectional LSTM with three consecutive steps. 15

Figure 11: (a) Simple scheme of a one-dimension (1D) convolutional operation. (b) Full representation of a 1D convolutional neural network for a SNP-matrix. 16

Figure 12: Bahdanau attention mechanism. 17

Figure 13: Dilated causal convolutional structure of TCN. 18

Figure 14: Transformer architecture 19

Figure 15: Stacking ensemble high level architecture. 20

Figure 16: Bias-variance trade-off in supervised learning. 21

Figure 17: Dropout technique dropping units along with respective connections. 22

Figure 18: Early stopping regularization technique to prevent overfitting. 22

Figure 19: High-level solution architecture 26

Figure 20: Example illustration of principal component analysis 29

Figure 21: Cumulative explained variance by the number of PCs retained. 29

Figure 22: Learning curve of autoencoder 30

Figure 23: Distinct LSTM predictive performance by training with different feature transformations. 32

Figure 24: Splitting BTC price series into train, validation, and test periods. 33

Figure 25: Normal input vectors compared to recurrent neural networks input. 34

Figure 26: Data preparation function “sequential_window_dataset”. 35

Figure 27: Transformer encoder architecture 41

Figure 28: High level stacking ensemble architecture for regression predictive modeling to predicting the price on test dataset. 43

Figure 29: High level stacking ensemble architecture for classification predictive modeling to predict the trend on test dataset. 43

Figure 30: High level stacking ensemble architecture for regression predictive modeling to predict the return on test dataset. 44

Figure 31: Accuracy scores of all models on test dataset 47

Figure 32: TCN confusion matrix on test dataset. 47

Figure 33: Learning curve of TCN classifier 48

Figure 34: Predicted vs ground truth with stacking ensemble on test dataset. 50

Figure 35: MAE score across all models on test dataset. 51

Figure 36: Predicted vs ground truth with BiLSTM on test dataset. 52

Figure 37: Learning curve of BiLSTM. 52

Figure 38: Hybrid CNN-LSTM implementation 67

Figure 39: CNN-BiLSTM-Attention implementation. 69

List of Tables

Table 1: Parameter specification for LSTM 36

Table 2: Parameter specification for GRU. 37

Table 3: Parameter specification for BiLSTM 37

Table 4: Parameter specification for hybrid architectures 39

Table 5: Parameter specification for CNN-BiLSTM-Attention 40

Table 6: Parameter specification for TCN. 40

Table 7: Parameter specification for Transformer Encoder 42

Table 8: Evaluation results of BTC directional status classification on test dataset 46

Table 9: Evaluation of stacking ensemble classifier for trend forecasting on test dataset 46

Table 10: Evaluation of models for BTC price forecasting 49

Table 11: Evaluation of stacking ensemble for BTC price forecasting 49

Table 12: MAPE score across all models on test dataset. 49

Table 13: Evaluation of BTC returns forecasting. 50

Table 14:Evaluation of stacking ensemble for BTC return forecasting. 51

Table 15: Collected features set. 67

1 Introduction

1.2 Background

Since their initial appearance in 2008, Cryptocurrencies have emerged as an alternative investment asset, generally considered one of the most promising types of profitable investments, with Bitcoin reaching its highest market capitalization of $ 1.28 trillion in November of 2021. Nevertheless, this constantly increasing financial market is characterized by significant volatility and strong price fluctuations over time. Nowadays, cryptocurrency forecasting is generally considered one of the most challenging time-series prediction problems due to the large number of unpredictable factors involved and the significant volatility of cryptocurrencies' prices, resulting in complex temporal dependencies.

1.3 Problem Statement

In the volatile cryptocurrency market context, this study focuses on researching forecasting methods to predict daily BTC price, return, and directional state. We aim to achieve this through developing and evaluating advanced deep learning models and a robust feature engineering process, as the BTC price time series is non-stationary and exhibits autocorrelation as shown in Figure 1, revealing the lack of independence between values.

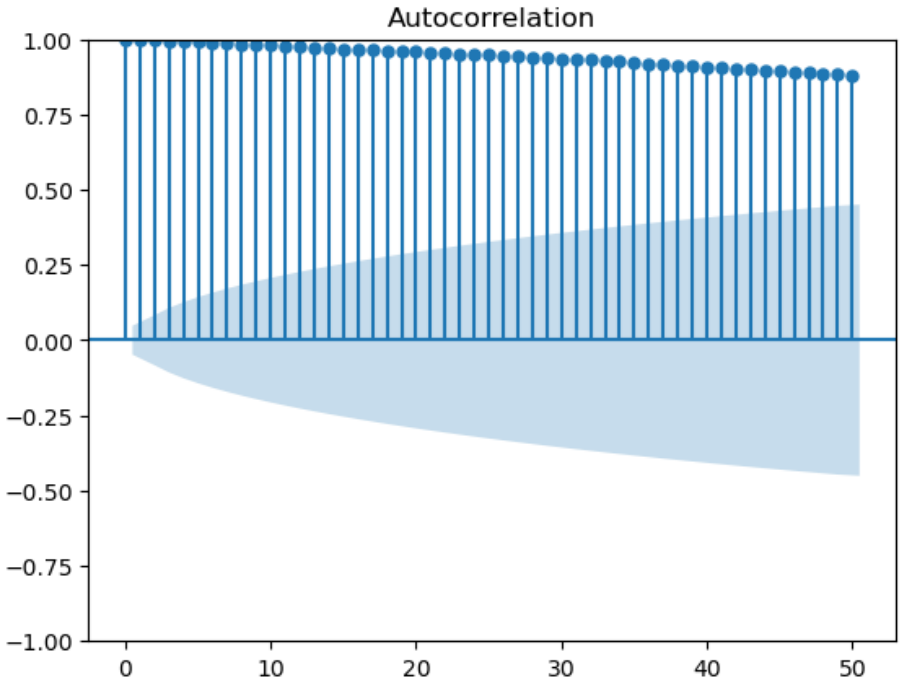

Figure 1: BTC autocorrelation visualization with a lags value of 50

This research addresses the need for more precise forecasting methods in the digital asset domain, which is vital for informed decision-making and market comprehension. This work aims to research three different problems, namely a binary classification problem, to provide insights into the future direction of BTC price trend, and two regression problems, bitcoin price prediction and return forecasting.

In this research, the predictive modeling for the day-ahead price and trend also encompasses both a univariate and multivariate approach. In a univariate approach, only the previous price of bitcoin is considered by the predictive models, whereas in a multivariate approach, multiple features are considered to forecast the target variable. These approaches were included in this research to test if adding more features would increase the models' predictive power or would increase the noise in the data and, therefore, decrease the performance of the explored models.

With the detailed design and evaluation of the forecasting results, data preprocessing methods, and feature engineering, this research contributes to the financial time series research domain.

1.4 Research Methodology

A methodology based on CRISP-DM (Costa & Aparicio, 2020, 2021)was employed, with a particular emphasis on the identification of the most suitable methods. This approach involved a comprehensive exploration of deep learning algorithms for both classification and regression predictive modeling, with the primary goal of improving prediction performance by uncovering concealed patterns within the datasets:

- Feature engineering: our research encompasses both univariate and multivariate approaches, utilizing a dataset consisting of 51 features related to market data, macroeconomic factors, BTC network data, social popularity, and newly generated features from the collected ones. We emphasize the importance of comprehensive feature engineering, including techniques like Granger Causality Test for feature selection, Principal Component Analysis, and an Autoencoder for feature extraction. These techniques enhance the quality of our input data and contribute to improved forecasting results.
- Selection of deep learning models: we employ a diverse set of deep learning models tailored to both classification and regression scenarios based on the literature review. We incorporate models like Long Short-Term Memory, Gated Recurrent Units, Temporal Convolutional Networks, Transformers, and ensembles, known for their robust performance in forecasting financial time series.
- Advanced architectural designs: our models incorporate advanced architectural designs, including an additive attention mechanism and hybrid structures that combine one-dimensional convolutional layers for noise reduction with Long Short-Term Memory (LSTM) layers for capturing sequence patterns and both long and short-term dependencies. These designs are adapted to both classification and regression-supervised tasks as needed.
- Evaluation metrics: to assess the performance of our deep learning algorithms, we employ a range of evaluation metrics tailored to the specific task. For classification, we consider accuracy, precision, and recall. For regression, we evaluate Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE) and R^2.

By employing this methodology, we aim to not only implement a wide range of deep learning models but also optimize their performance through careful model selection, evaluation, and feature engineering. This comprehensive approach enables us to extract meaningful insights from the data and advance the fields of financial time series forecasting and classification.

1.5 Outline

In this section, we provide a thematic overview of the chapters in this thesis, outlining the main objectives and contributions of each chapter. As we delve deeper into bitcoin

price forecasting and deep learning models, each chapter serves a unique purpose in building a comprehensive understanding of our research.

First, we set the stage by introducing the volatile world of cryptocurrencies, and bitcoin in particular. We identify the critical problem of short-term price prediction and articulate our research objectives to address this challenge.

Second, we provide a guided tour of the literature relevant to our study. We explore the rich field of time series forecasting, deep learning, and the unique dynamics of bitcoin prices. This section provides a solid foundation for understanding the context of our research.

This research continues in the following section, where we detail the data collection and preprocessing processes. Here, we introduce the various datasets that serve as input to our models, revealing their sources and characteristics. We discuss the data cleaning and feature engineering processes that refine our input.

After detailing the preprocessing pipeline, we present our deep-learning exploration methodology. We discuss the blueprints of our predictive models, explaining the architectural designs and hyperparameter choices behind our models.

As our research unfolds, the next section puts our models through rigorous testing. We present the results of our experiments and evaluate their predictive performance against various metrics. Here, we discuss the lessons learned and analyze the implications of our findings.

The final section concludes our research. It summarizes our key findings, highlights their implications for investors, and outlines future directions for improving bitcoin or other asset price forecasting.

2 Literature Review

2.2 Time Series Forecasting

Mainly, it consists of predicting future events by applying models to time series. It is an extremely important area of machine learning because there are several prediction

problems involving time components. Every time series problem is very specific and has its own data characteristics, and according to that, the most fitting model(s) must be chosen to analyze and model the data. Its significance stems from its applicability to a wide array of prediction problems involving temporal components. Within the realm of time series forecasting, diverse tasks exist, including univariate and multivariate forecasting, point and interval predictions, and distinctions between short-term and long-term horizons.

Historically, time series forecasting has evolved through significant milestones shaped by seminal contributions. Notably, traditional statistical techniques such as ARIMA and Exponential Smoothing have long been employed for their interpretability and performance (Tomas et al, 2018). However, the advent of machine learning has introduced alternative approaches, encompassing regression-based models and tree-based methods. In recent years, deep learning models, including Long Short-Term Memory (LSTM) networks and Temporal Convolutional Networks (TCNs), have gained prominence for their ability to capture complex temporal dependencies and have further elevated the state-of-the-art in time series forecasting. These advances have enabled more accurate predictions and greater adaptability to complex data patterns. Nevertheless, despite these recent developments, challenges persist, encompassing issues like irregularly sampled data and the interpretability of deep learning models.

2.3 Related Work

Several research studies have contributed valuable insights and methodologies in cryptocurrency price forecasting using deep learning techniques. In this section, we delve into the related work, summarizing key findings and contributions from various papers in the field. (e.g.

Politis et al. (2021) explore the use of deep learning techniques for cryptocurrency price forecasting. They emphasize the importance of selecting relevant features, such as historical price data, trading volumes, and other market indicators, for improved forecasting accuracy. Short-term forecasts achieve an accuracy of up to 84.2%. In Dimitriadou & Gregoriou, (2023) the authors compile a dataset consisting of 24 potential explanatory variables that are commonly used in financial literature. The dataset includes daily data from December 2nd, 2014, to July 8th, 2019. The paper explores various

forecasting models that leverage historical Bitcoin prices, data from other cryptocurrencies, exchange rates, and macroeconomic variables. The study compares the performance of different machine learning algorithms, including logistic regression and linear support vector machines. Empirical results indicate that the traditional logistic regression model outperforms the other algorithms, achieving an accuracy rate of 66%. This suggests that the logistic regression model is more effective at predicting Bitcoin price movements in the given dataset. The authors (J. Shen & Shafiq, 2020) introduce a comprehensive framework for predicting stock market price trends, emphasizing feature engineering, deep learning, and extensive model evaluation. Their study encompasses a dataset extracted from the Chinese stock market, spanning two years. The authors assess a range of machine learning models commonly applied in short-term stock market price trend prediction, including support vector machine (SVM), artificial neural network (ANN), stacked artificial neural network (SANN), and long short-term memory (LSTM). Their comparison focuses on bi-weekly price trend prediction and retains all 29 features selected through the Recursive Feature Elimination (RFE) algorithm. The evaluation results demonstrate the capabilities of these models, with an emphasis on accuracy, training efficiency, and precision. The proposed Long Short-Term Memory (LSTM) model achieves a binary accuracy of 93.25%. Furthermore, data pre-processing through Principal Component Analysis (PCA) extracts five principal components, contributing to the overall efficacy of the solution. The authors (Livieris et al., 2020) propose three ensemble learning strategies: ensemble averaging, bagging, and stacking with deep learning models for forecasting cryptocurrency hourly prices. Base learners are comprised of LSTM, Bi-directional LSTM, and convolutional layers, concluding that stacking with KNN as meta-learner was considered the best forecasting model, being able to identify the redundant and non-informative base models and "weight them" to filter out noise. The authors achieved an accuracy of 54.52%. The paper (Greaves & Au, 2015) investigates the predictive power of blockchain network-based feature engineering on the future price of bitcoin and machine learning optimization to obtain a Bitcoin price movement classification accuracy of 55%. They found that only a limited amount of predictive information was embedded in the network features, not serving as a good proxy for Bitcoin exchange behavior. (Adcock & Gradojevic, 2019) proposed a non-parametric model based on technical analysis, producing point and density forecasts of Bitcoin returns with a feedforward neural network. They found that backpropagation neural networks dominate various competing models in terms of their forecast accuracy and

achieve an accuracy of 65%. McNally et al., (2018) attempt to predict the direction of Bitcoin price, implementing a Bayesian-optimized RNN and LSTM network. They utilized data ranging from August 2013 until July 2016, regarding open, high, low, and close of Bitcoin prices as well as the block difficulty and hash rate. The author states that ARIMA model was also implemented as a comparison to the deep learning models, performing poorly. After evaluating performance, LSTM achieves the highest classification accuracy of 52%. (Ortu et al., 2022) present a comprehensive analysis of the predictability of price movements comparing four different deep learning algorithms, MLP, CNN, LSTM and ALSTM. By using three classes of features, comprising technical, trading, and social data, they found that including trading and social indicators to the technical data yields a significant improvement in the prediction and accuracy, increasing the performance for the daily classification from a range of 51% to 55% to 67% to 84%.

2.4 Feature Engineering for Time Series

In this subchapter, we explore the fundamental concepts and techniques of feature engineering in the context of time series data. Feature engineering is a critical step in preparing data for modeling, as it involves the creation, selection, and transformation of variables to enhance the predictive power of machine learning models.

There are several feature selection methods in the literature such as the Random Forest Feature Importance, which provides information about the contribution of each feature to the model's predictive accuracy, determining which variables have the most substantial on the target variable (Breiman, 2001). Another method is RFE, recursive feature elimination, a systematic technique that iteratively prunes the least important features while evaluating model performance at each step (Guyon et al., 2002). RFE starts with the full feature set, computing an importance score for each predictor, and progressively removes variables that contribute the least to low error forecasts. This technique continues iteratively until the desired number of features is reached. Another technique is the Granger causality test, a statistical method generally used in time series analysis, consisting of the exploration of causal relationships between variables It determines whether past values of one variable help predict future values of another. To apply this test effectively, the data must be stationary, ensuring that statistical properties remain constant over time (Granger, 1969).

Feature extraction also plays a pivotal role in transforming raw data into informative representations. It involves the process of deriving new features from existing ones, aiming to reduce the dimensionality of the feature space while preserving essential information. Two prominent feature extraction techniques are explored in this section, namely PCA and AE.

Principal Component Analysis primary objective is to simplify complex datasets by projecting them onto a lower-dimensional subspace while retaining as much variance as possible. This procedure converts a set of possibly correlated variables into a set of linearly uncorrelated variables (Pearson, 1901). PCA accomplishes this by identifying linear combinations of the original features, known as principal components, that capture the most significant variability in the data.

One last mention goes to autoencoders as they are neural network architectures designed for unsupervised learning, specifically tailored to encode, and decode data. Their primary purpose is to learn compact representations of input data by encoding it into a lower-dimensional space and then decoding it back to the original format. This process is achieved using an encoder and a decoder network, often referred to as the "bottleneck" structure. The encoder network encodes the input data into a compressed representation, known as the latent space or encoding layer. This layer typically contains a reduced number of neurons compared to the input layer. The decoder network then takes this compressed representation and attempts to reconstruct the original input data. The key objective of the autoencoder is to minimize the reconstruction error, ensuring that the decoded output closely resembles the input. By learning a compressed representation of the data, autoencoders can effectively reduce the dimensionality of the feature space, resulting in a more concise set of features. Also, the process of encoding and decoding inherently filters out noise present in the data, leading to cleaner and more informative features (Vincent et al., 2008).

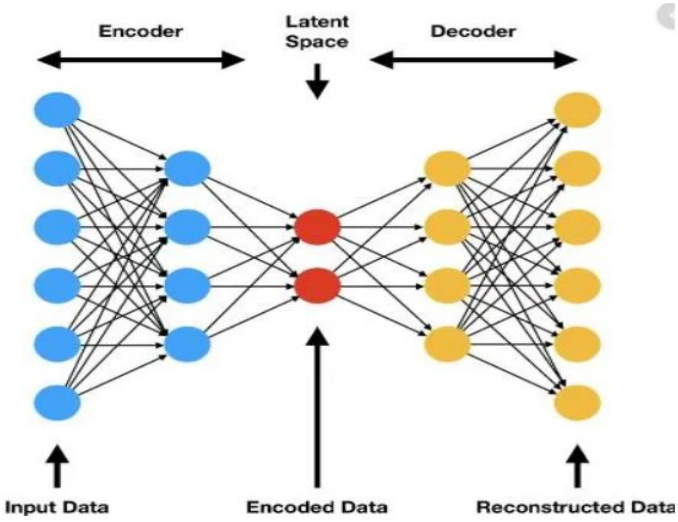

Figure 2: Autoencoder representation

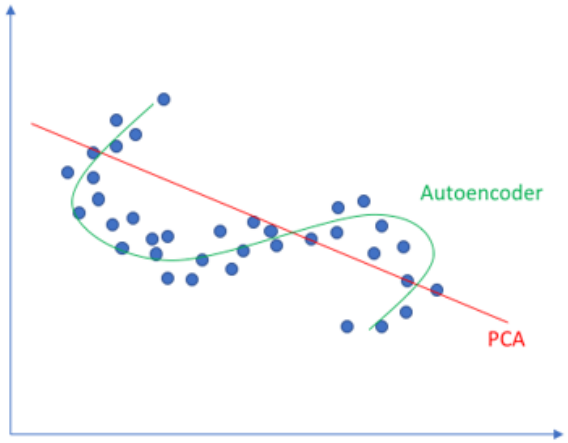

Figure 3: Linear vs nonlinear dimensionality reduction

Source: (Nugroho, 2020)

2.5 Gradient-Based Learning

Neural networks are usually trained using an iterative gradient-based optimizer that minimizes the network's cost function, aiming for lower values. This optimization process often involves backpropagation, a fundamental technique in neural network training. Backpropagation calculates the gradient of the cost function with respect to the model's weights and biases. It efficiently propagates the error backward through the network, enabling the adjustment of parameters in earlier layers based on the error observed in later layers.

However, while gradient descent-based algorithms are commonly applied to non-convex loss functions in neural networks, there's no guarantee of global convergence. Additionally, these algorithms are sensitive to the initial parameter values. The learning rate, a key hyperparameter, controls the step size by which the optimizer updates the model's parameters during each iteration of training. Properly setting this parameter is crucial for enhancing convergence and training stability. A high learning rate might lead to overshooting optimal parameter values, causing divergence in training, whereas a very low learning rate might trap the optimizer in a local minimum.

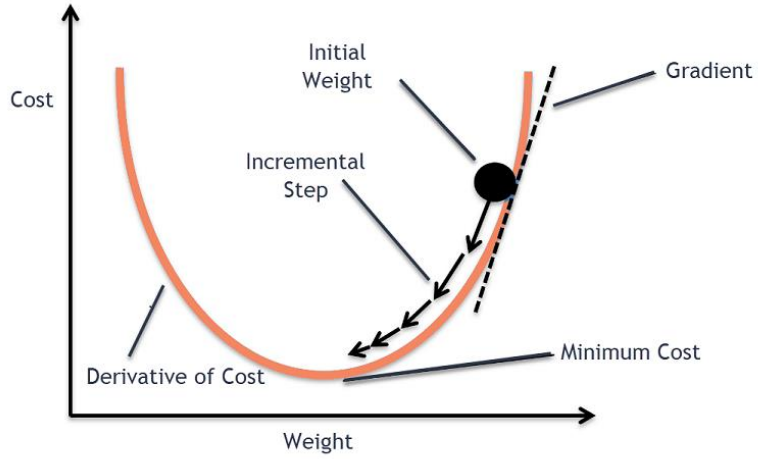

Figure 4: Gradient descent algorithm

Source: Clairvoyant

2.6 Activation Functions

In a neural network, an activation function normalizes the input and produces an output which is then passed forward into the subsequent layer. Activation functions add non-linearity to the output which enables neural networks to solve non-linear problems. Nonlinear functions usually transform a neuron's output to a number between 0 and 1 or -1 and 1. Common activation functions include Linear, Sigmoid, Tanh (Hyperbolic Tangent), and ReLU (rectified linear unit).

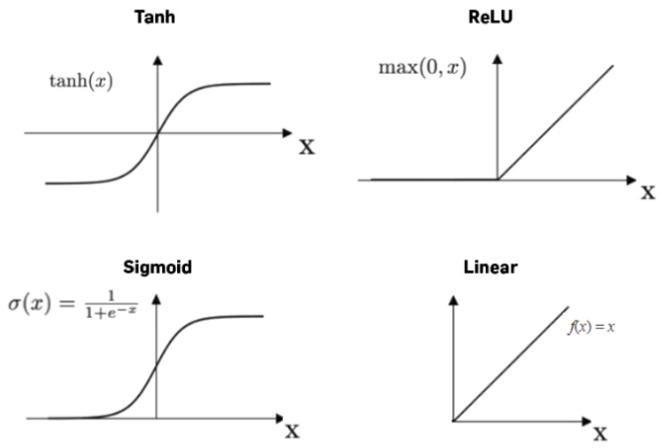

Figure 5: Activation functions

Source: AI Wiki

2.7 Deep Learning Forecasting Models

- ### RNN (Recurrent Neural Network)

RNNs were the first algorithm introduced with an internal memory that remembers its input, a type of neural network that has hidden states and allows past outputs to be used as inputs to better understand sequences.

In RNN models, each time step generates a hidden state, representing the information learned up to that point in the sequence. The last hidden state often contains the most refined representation of the entire input sequence because it encodes information from all previous time steps, although sometimes earlier hidden states might capture critical information, especially in sequences with complex patterns.

There is a wide variety of RNNs relationships that is possible to create with the idea of graph unrolling and parameter sharing across different time steps. It is possible to create a RNN with a one-to-one, one-to-many, many-to-one, or many-to-many relation.

RNNs have traditionally treated input information in a uniform manner This means that no explicit connections are made between individual tokens to fully capture their complex relationships. In essence, each token, represented as t_i, receives uniform information from all preceding tokens $t_1 ... t_{i-1}$ in a uniform fashion. However, as i increases, this method often results in a loss of information and challenges related to

vanishing or exploding gradients. This issue is mitigated in more advanced recurrent neural networks such as LSTM and GRU.

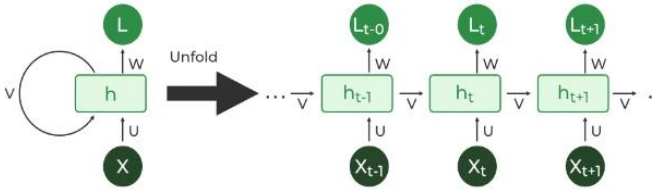

Figure 6: Recurrent neural network architecture.
Source: (Biswas et al., 2021)

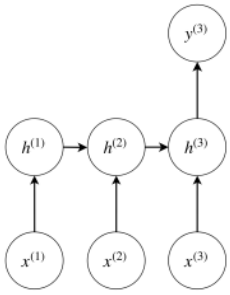

Figure 7: Many-to-one relationship suitable to predict one step ahead.
Source: (Jiao et al., 2016)

- ## LSTM (Long Short-Term Memory)

LSTMs are a type of recurrent neural network that excels at capturing long-range dependencies in sequential data. The core point behind the LSTM success is the memory cell that can preserve its state over time and its nonlinear gating units that thoroughly regulate the information flow (Greff et al., 2017).

Their unique architecture incorporates specialized gating mechanisms, including input, forget, and output gates, allowing them to selectively store and retrieve information over extended time steps. These gates learn what information is pertinent to keep or forget during training. By incorporating these mechanisms, LSTM models aim to address the RNN issues mentioned above and enhance the handling of information within the sequence.

This inherent memory capability makes LSTMs particularly suitable for tasks related to sequential data modeling, such as natural language processing, speech recognition, and

time series forecasting. They have become a cornerstone of deep learning in modeling complex temporal relationships (Hochreiter & Schmidhuber, 1997).

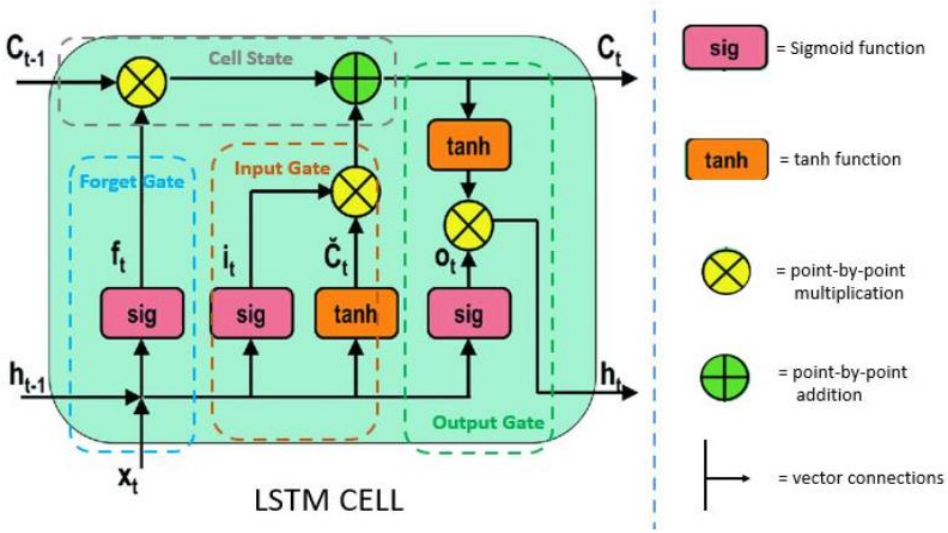

Figure 8: LSTM cell.

Source: (Mukhanov et al., 2023)

- GRU (gated recurrent unit)

A gated recurrent unit (GRU) was proposed by (Cho et al., 2014) to make each recurrent unit to adaptively capture dependencies of different time scales. Similar to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, although without having a separate memory cell. This results in the exposure of the full hidden content without any control because GRU does not have any mechanism to control the degree to which its state is exposed.

Because GRU has a less complex structure, it is computationally more efficient than LSTM.

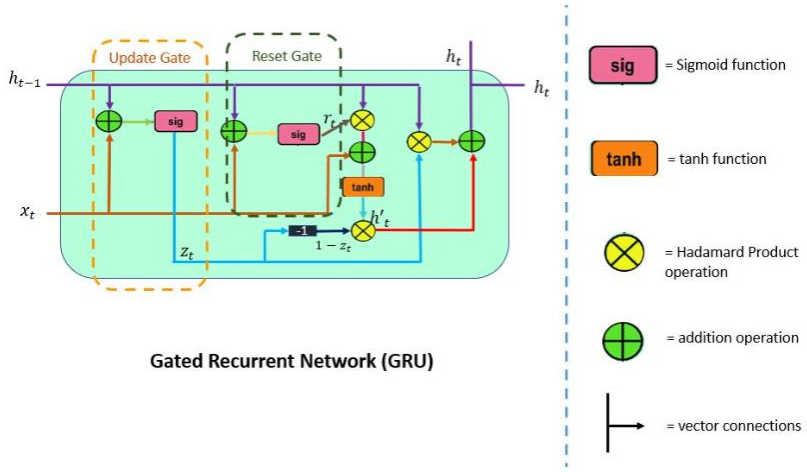

Figure 9: GRU cell.

Source: (Ahmad et al., 2023)

- ## BiLSTM (Bidirectional Long Short-Term Memory)

Bidirectional LSTMs are a type of recurrent neural network (RNN) that can capture both past and future context by processing sequences in both forward and backward directions, enhancing the model's ability to capture long-range dependencies in the data. They have been successful in various sequence prediction tasks, including time series forecasting.

By employing bidirectional recurrent layers, the model process sequences in both forward and backward directions during training, allowing the model to learn patterns and dependencies that are both forward-looking and backward-looking.

Prior study proved that the bidirectional networks are significantly better than the standard ones in many fields (Graves & Schmidhuber, 2005).

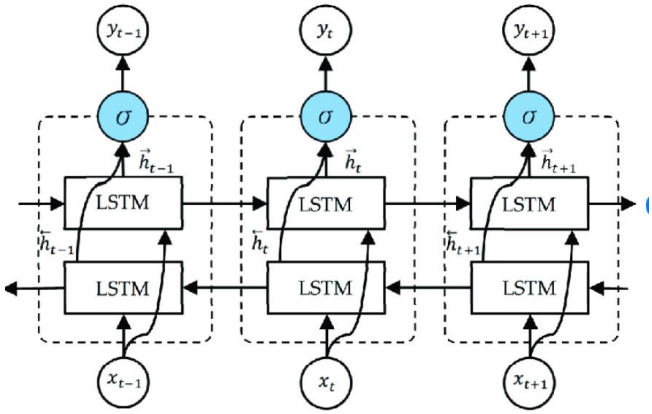

Figure 10: Unfolded architecture of Bidirectional LSTM with three consecutive steps.

Source: (Li et al., 2020)

- CNN (Convolutional neural networks)

Convolutional neural networks, or ConvNets, are a family of neural networks that uses convolution operation in place of general matrix multiplication, in at least one of their layers. This approach was significantly inspired by (Fukushima, 1980) resulting in very efficient models for image processing tasks, such as image classification as demonstrated by various research papers like (Yu et al., 2020) where the presented CNN achieves an higher accuracy than other state-of-the-art methods for classifying remote sensing images. Convolutional neural networks are a specially designed to deal with data that has a grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid with a regular sample interval, and image data, which can be seen as a 2-D grid of pixels.

As illustrated in figure 11, extracted from research in (Pérez-Enciso & Zingaretti, 2019) regarding the use of deep learning for complex trait genomic prediction, and focusing on the study of CNNs, we can observe representations of both 1D convolutional operation and a 1D CNN for a SNP-matrix, where the convolution outputs are represented in yellow. Pooling layers after convolutional operations combining the output of the previous layer at certain locations into a single neuron are represented in green. The final output is a standard MLP.

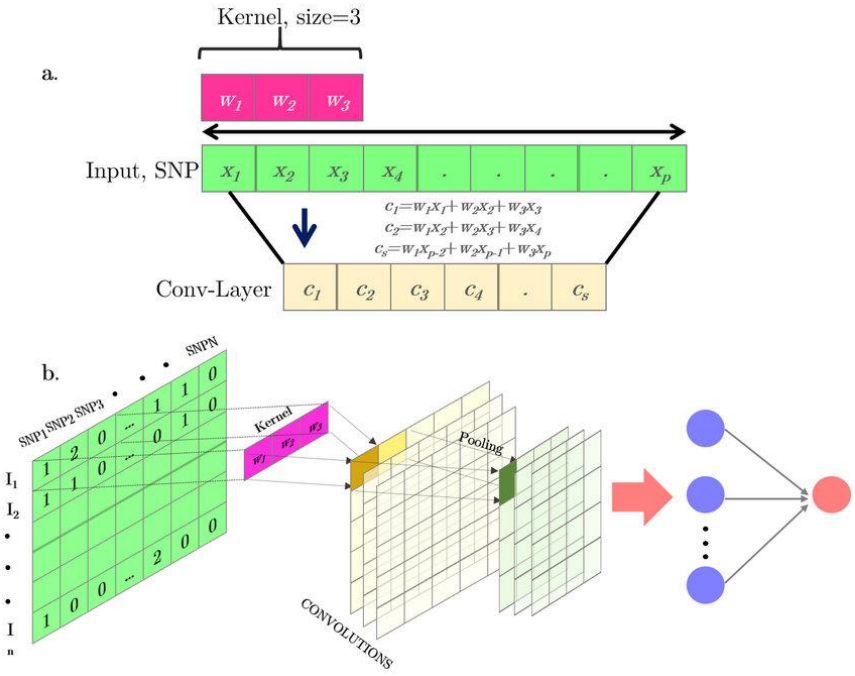

Figure 11: (a) Simple scheme of a one-dimension (1D) convolutional operation. (b) Full representation of a 1D convolutional neural network for a SNP-matrix.

Source: (Pérez-Enciso & Zingaretti, 2019)

- ## Attention Mechanism

A new approach to encoder-decoder models, introduced in 2014 (Bahdanau et al., 2016), exploits the bottleneck of encoding an arbitrary sequence length into a fixed length vector. To overcome this limitation in the encoder-decoder architecture, the attention mechanism was introduced to enable modeling relationships between elements in input or output sequences, regardless of their positional distance. The objective is to allow the decoder to learn which is the hidden state, from the encoder that is most valuable to predict the next output. This enables the decoder output to choose the context vector C, based on the set of positions where the input sequence carries the most relevant information. In this sense, the most significant difference between the attention mechanism and traditional RNN or LSTM models is that the attention mechanism focuses directly on specific parts of the sequence rather than treating them equally. The attention mechanism in neural networks assigns weights to the hidden states based on their relevance or importance for the specified task, emphasizing the more relevant states while attenuating the influence of less relevant ones. As a result, the attention mechanism will help the model to have a better understating of the context of the sequence.

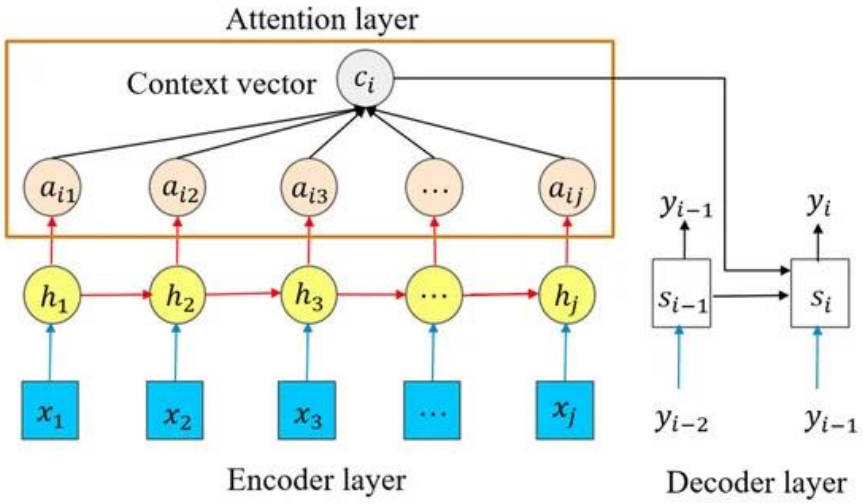

Figure 12: Bahdanau attention mechanism.

Source: (Yang et al., 2021)

- TCN (Temporal Convolutional Network)

Temporal Convolutional Network is a variant of the CNN architecture that is specially designed for time series forecasting. This architecture introduced in (Bai et al., 2018), showed a convolution neural network architecture outperforming several recurrent architectures for sequence modeling and it was inspired in recent works about sequential tasks. The authors stated that convolution networks should be considered as a starting point for sequential tasks. TCNs exhibit longer memory than recurrent architectures with the same capacity and offer advantages such as parallelism (convolutions can be done in parallel as the same filter is used in each layer), and stable gradients during training.

A key component of TCN that differ from CNN architecture is employing causal and dilated convolutions. Causal convolutions force the model to learn the dependence between the steps without violating the natural order of time and the dilated technique helps it process an increasingly larger portion of the time series steps as it advances to the deeper layers.

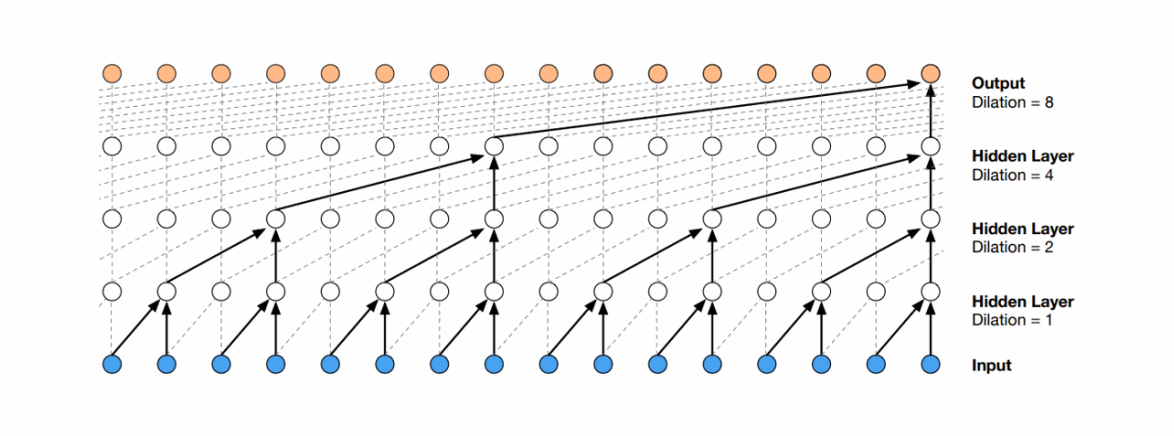

Figure 13: Dilated causal convolutional structure of TCN.

Source: (Oord et al., 2016)

A feature of TCNs is that they have a receptive field dependent on the specified model configuration which determines the number of input frames that can be observed to produce an individual output frame (Ravenscroft et al., 2022).

By consulting Figure 13, the dilation factor is doubled at each layer. For instance, unit two in the first hidden layer processes the information from steps one and two, and unit four in the second layer processes steps one, two, three, and four through the processing of units two and four from the first hidden layer

- ## Transformer

A state-of-the-art deep learning architecture introduced in 2017 (Vaswani et al., 2023), has excelled in a wide range of tasks involving natural language processing and computer vision. The basic building block of the Transformer combines a Feed forward network with a multi-head self-attention layer.

The transformer model employs an encoder-decoder architecture. The encoder consists of encoding layers that process the input tokens iteratively one layer after another, while the decoder consists of decoding layers that iteratively process the encoder's output as well as the decoder output's tokens so far.

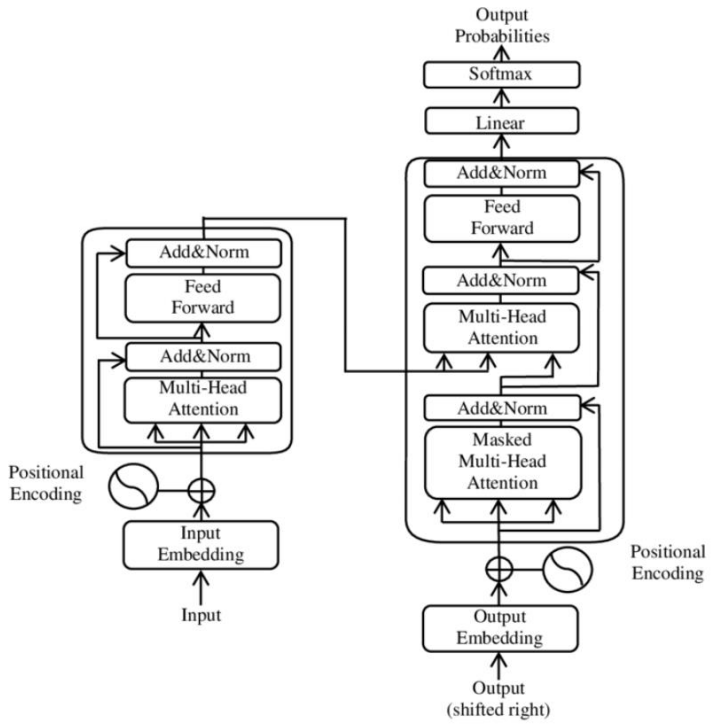

Figure 14: Transformer architecture

Source: (Vaswani et al., 2023)

In this research we investigate the Transformer architecture to address an important sequence learning problem: time series forecasting. The application of a transformer model in the context of time series data involves a sophisticated process of embedding, encoding, and decoding sequential numerical values. Unlike recurrent networks such as LSTMs or GRUs, the Transformer model offers notable advantages, particularly in terms of parallelization. Its design allows for efficient parallel computation of sequences, enabling faster training and inference.

After providing the transformer a set of numerical values representing a time series, the data is embedded into a format suitable for the transformer's operations. This embedding process transforms the input into a more structured representation. The encoder sequentially processes the embedded time series data, aiming to capture temporal patterns and dependencies present in the input time series data. Often comprises multiple layers to provide a deeper understanding and extraction of patterns. A key component in the encoder is the self-attention mechanism, enabling the model to focus on different aspects of the time series data at each step. The self-attention mechanism is different from the attention mechanism, as attention is used to assign different weights to different time steps within the input sequence, indicating their relevance, whereas self-attention allows each time step in the sequence to calculate its attention scores concerning all other time

steps in the same sequence, including itself. The decoder component primarily focuses on making predictions by generating output sequences based on the information learned from the encoder.

- ## Stacking Ensemble

Stacking or stacked generalization considers heterogeneous base learners, learns them in parallel, and combines them by training a meta-learner to output a prediction based on the different base learner's predictions. A meta learner inputs the predictions as the features and the target being the ground truth values in data. It attempts to learn how to best combine the input predictions to make a better output prediction.

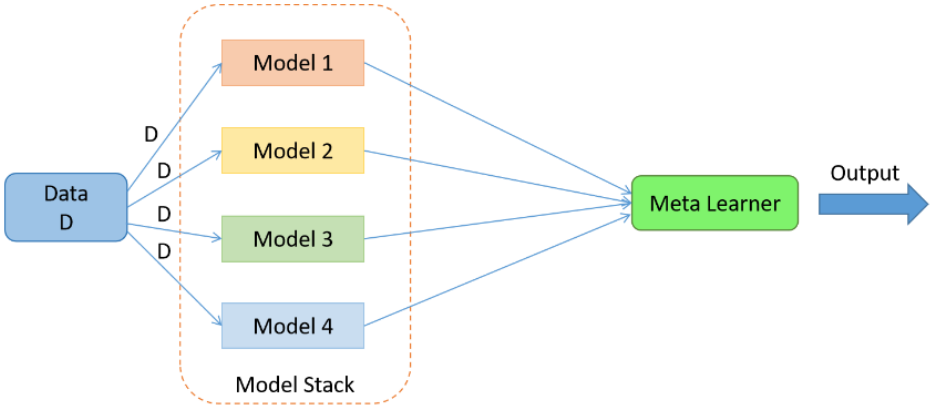

Figure 15: Stacking ensemble high level architecture.

- ## Bias-Variance Trade-Off

In supervised learning, bias and variance are key factors affecting a model's performance. Bias represents the error from overly simplistic assumptions in the learning algorithm, leading to underfitting and the failure to capture the true relationships in the data. On the other hand, variance reflects a model's sensitivity to fluctuations in the training data, causing it to fit noise rather than the actual signal, resulting in overfitting.

High bias usually results in a model that oversimplifies the underlying patterns, leading to poor performance on both training and new data. Conversely, high variance can lead to excellent performance on training data but poor generalization to new data, as it overfits and captures noise instead of true patterns.

The bias-variance trade-off involves analyzing the right balance between these two elements. It is critical to achieving a model that accurately captures the underlying patterns in the data, while also generalizing well to new, unseen data.

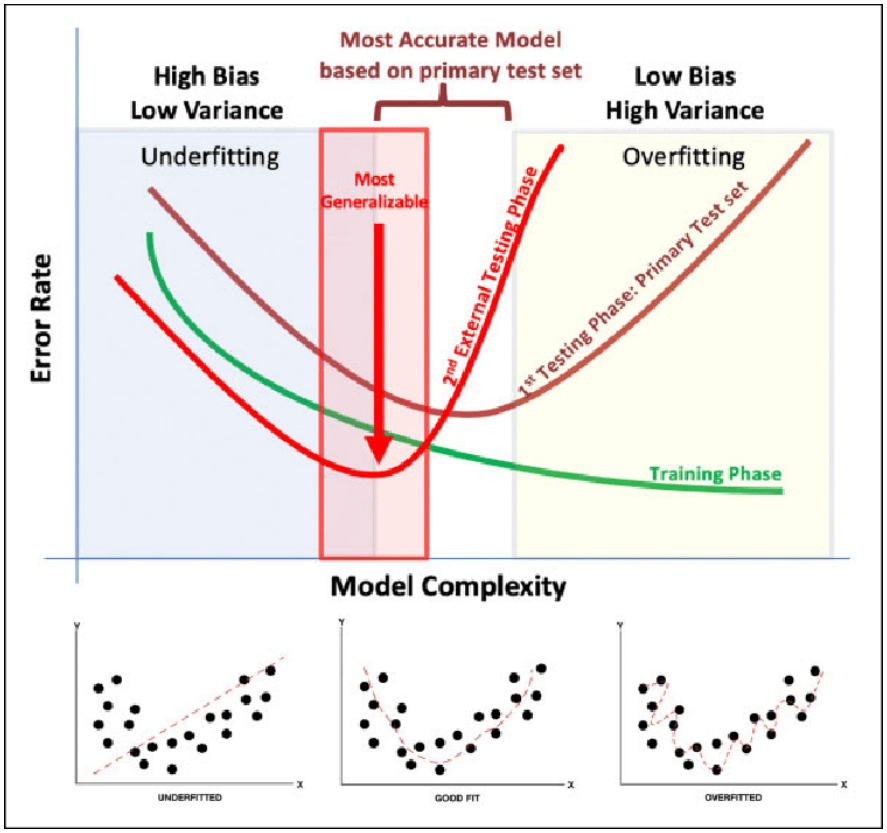

Figure 16: Bias-variance trade-off in supervised learning.

Source: (Rashidi et al., 2019)

- ## Overfitting Mitigation

Overfitting is a common issue encountered in both machine learning and deep learning. It arises when a model becomes overly intricate and begins fitting the noise present in the training data, resulting in poor generalization to new data. To combat this challenge, a range of techniques for addressing overfitting were developed.

Some notable approaches include regularization methods such as L1 (Lasso) and L2 (Ridge), introducing penalty terms into the model's loss function. These penalties discourage the development of excessively large model weights, thus curbing overfitting.

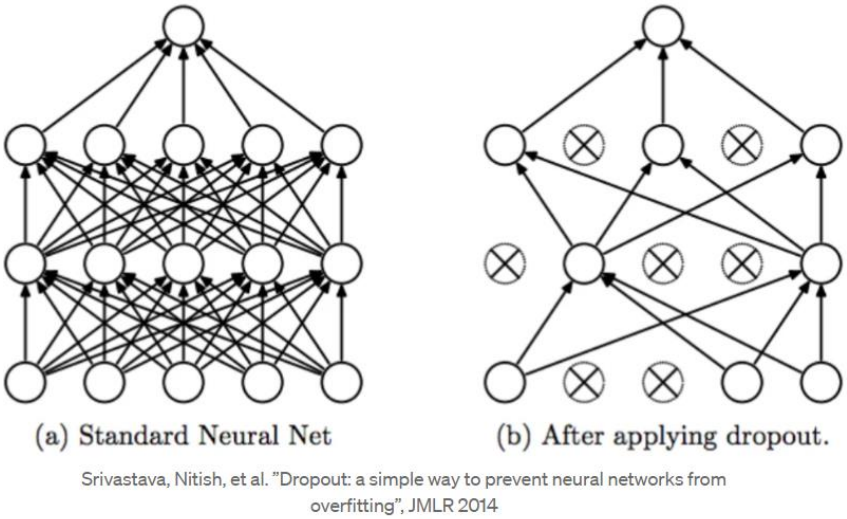

Figure 17: Dropout technique dropping units along with respective connections.

Source: (Srivastava et al., 2014)

Dropout, introduced in (Srivastava et al., 2014), is particularly relevant to neural networks, by randomly deactivating a subset of neurons during training. This randomness prevents certain neurons from influencing both forward and backward passes, promoting more robust feature learning.

Other techniques include early stopping, as studied in (Langer, 1997), a regularization technique for deep neural networks with the primary goal of preventing overfitting.

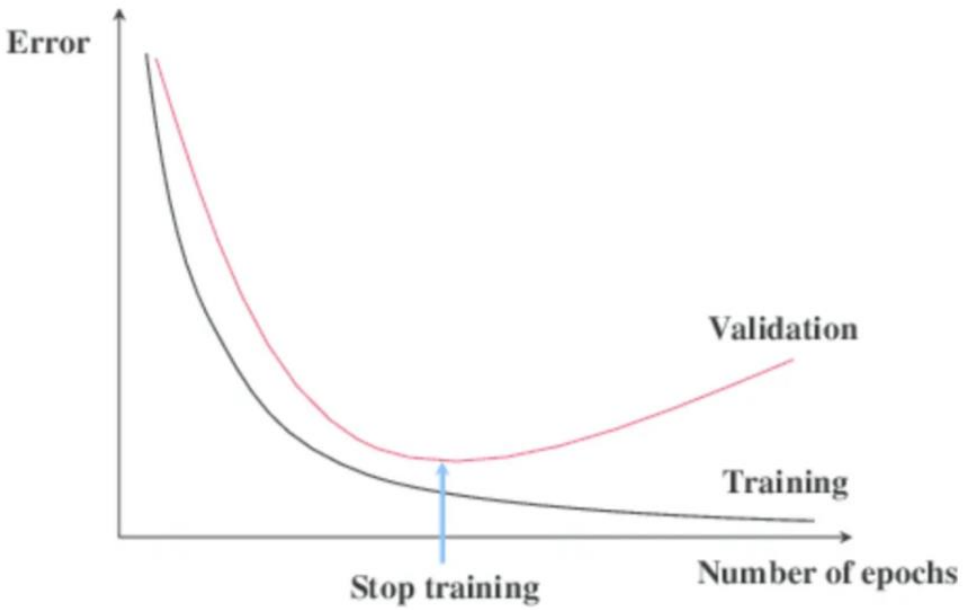

Figure 18: Early stopping regularization technique to prevent overfitting.

Source: (Shin et al., 2016)

Ensemble methods, such as bagging and stacking, combine predictions from multiple models to enhance generalization. By doing so, they can mitigate overfitting by reducing the model's reliance on any single model's predictions.

Feature selection by identifying and removing irrelevant or redundant features from the input data can decrease model complexity and mitigate the risk of overfitting. Methods like Random Forest - Recursive Feature Elimination (RF-RFE) detailed in (Granitto et al., 2006) aid in this process.

Implementing cross-validation techniques, such as forward chaining, suitable for time series problems as it preserves the data's temporal order, may help in identifying a case of overfitting.

By exploring these strategies, researchers and practitioners aim to strike a balance between model complexity and generalization, addressing a fundamental challenge in the field of machine learning.

2.8 Evaluation Metrics

Within the domain of classification tasks, where the objective is to predict discrete outcomes, we rely on a suite of critical evaluation metrics to assess the performance of predictive models. These metrics serve as fundamental tools in quantifying the models' accuracy and reliability.

Accuracy: this metric calculates the ratio of correctly predicted instances to the total number of instances, providing a holistic view of the model's performance.

$$Acuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision: assesses the accuracy of positive predictions, measuring the proportion of true positive predictions among all positive predictions. A high precision score indicates a low rate of false positives.

$$Precision = \frac{TP}{TP + FP}$$

Recall: also known as true positive rate, evaluates the model's ability to correctly identify all positive instances. It calculates the ratio of true positives to the total number of actual positives.

$$Recall = \frac{TP}{TP + FN}$$

A classification model with high accuracy, precision and recall is desirable.

In the context of regression tasks, where the aim is to predict continuous numerical values, we employ a distinct set of evaluation metrics to assess the quality of predictive models.

Mean absolute error: MAE computes the average absolute difference between the model's predictions and the true values. It offers insights into the magnitude of errors in our forecasts.

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

Root mean squared error: It indicates the spread of the forecast errors. A model that predicts occasionally erratic values will have higher RMSE value, although it may still have lower MAE or MAPE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{I=1}^{N}|y_i - \widehat{y_i}|^2}$$

R^2: the coefficient of determination, assesses the goodness of fit of the regression model. It measures the proportion of the variance in the target variable that is explained by our model. Higher R^2values indicate a better fit.

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

Mean absolute percentage error: evaluates the percentage difference between the model's predictions and the actual values. It not only provides a relative measure of forecasting accuracy but is also highly interpretable. MAPE's interpretability allows stakeholders to easily understand the percentage by which our forecasts deviate from actual prices, aiding in informed decision-making.

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y_i}|}{y_i}$$

A model with low MAE, RMSE, MAPE, and high R^2 is desirable.

By using evaluation metrics such as those described, it is possible to ensure a comprehensive assessment of a model's performance.

3 Methodology

The methodology used here is inspired by the CRISP-DM methodology (Costa & Aparicio, 2020, 2021), which is described in detail in Figure 19. This methodology can even be incorporated into the context of a design science approach (Aparicio et al., 2023).

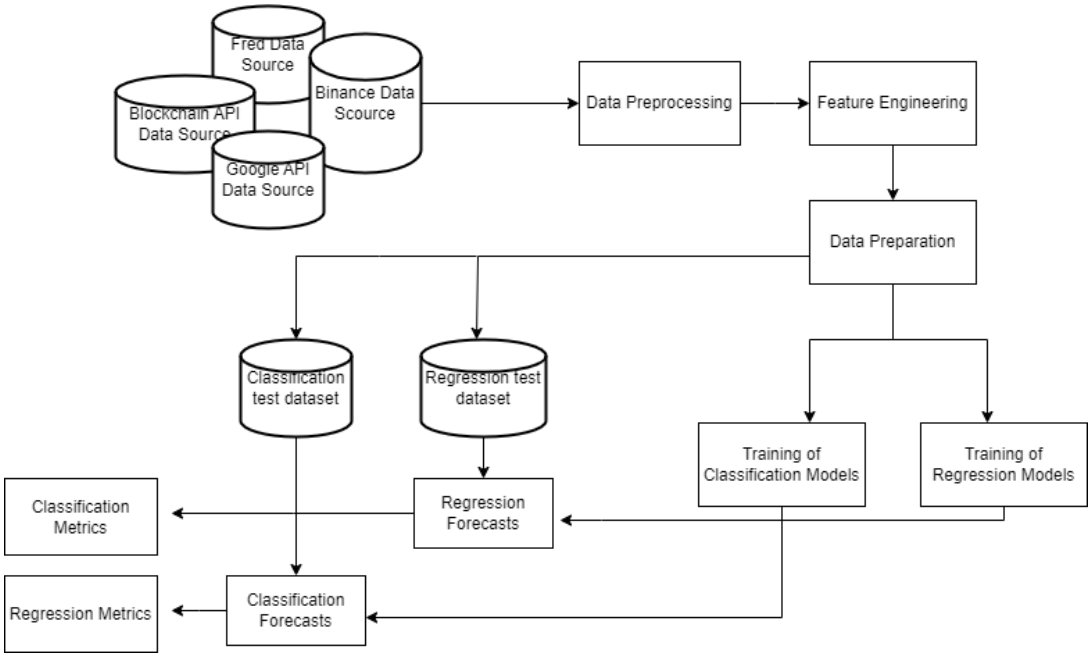

Figure 19: High-level solution architecture.

3.1 Data Collection

We collected five years of historical daily data on Bitcoin price from January 1, 2017, until December 31, 2022, from Binance API, along with a set of 50 additional features selected based on our domain knowledge and their significance in previous research. The feature dataset consisted of Bitcoin's blockchain network data, market trends, social popularity, relevant macroeconomic data, and technical indicators. The volume of Bitcoin daily transactions, its daily price, the three highest alternative cryptocurrencies by market capitalization daily price and their respective daily volume, VIX daily price data, and both SP500 and NASDAQ 100 indices were selected to reflect market dynamics. Average block size, mining difficulty, miners' revenue, n confirmed transactions per day, n-unique addresses, transaction-fees (USD) and average confirmation time, as a proxy for network congestion, were selected to represent the network status. Google trends was used to highlight Bitcoin's daily popularity. Other relevant macroeconomic data, retrieved from the FRED API, were also integrated into the feature dataset: daily oil price, 10-year treasury yields, copper and Iron price, median CPI data, unemployment data, savings rate in the US, US GDP and both five- and ten-year breakeven inflation data, totaling to 35 features. For better understanding of the collected variables, a description is available in Appendix (Table 14).

3.2 Data Preprocessing

In this section, we detail the essential data preprocessing steps undertaken to prepare the dataset for modeling. Our goal was to enhance the quality and integrity of the data, creating a robust foundation for subsequent feature engineering and modeling.

Missing data was addressed using a rolling forward technique, ensuring that time series gaps were filled appropriately. This process was vital in ensuring that each variable within the dataset was uniformly represented daily. By employing the rolling forward technique to fill missing data, temporal continuity was preserved across the dataset despite variations in the frequencies of the collected variables.

To standardize the data and prevent any single feature from dominating the learning algorithms due to its scale, we used the standard scaler. This scaler was chosen, among other scaling techniques, for its robustness in dealing with outliers.

$$z = \frac{x - \frac{1}{N}\sum_{i=1}^{N} x_i}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - u)^2}}$$

For the classification task aimed at predicting price movement, a binary encoding was implemented to attribute a value of 0 if the next day's price decreased and 1 if it increased. This encoding method effectively captures the directional change in price, enabling the model to learn and predict based on this binary classification.

Furthermore, it's noteworthy that the distribution of this encoded data is balanced, exhibiting approximately equal proportions for both classes. The dataset contains a similar percentage of instances where the price decreased as those where it increased. This balanced distribution alleviates the need for employing techniques like synthetic minority oversampling technique (Chawla et al., 2002) or other sampling methodologies specifically designed to address imbalanced data. Consequently, the classification models can learn from a dataset that inherently represents both potential outcomes in a comparable manner, fostering a more accurate and unbiased training process.

Following data cleaning and scaling, a detailed process of feature engineering was implemented, including creating new features based on the original ones, feature selection

to compute a subset of the original universe of features and feature extraction, where new features are generated from the previous feature set, resulting in an increased total number of features to 51. To capture key insights from the dataset, a set of additional variables was generated:

- Exponential moving average with a period of nine days.
- 14-day annualized volatility, to provide insight into price fluctuations.
- On-balance volume.
- 14-day period average true range, aiming to measure market volatility.
- Volume weighted average price, offering a weighted average of prices by volume of transactions.
- Daily return of the four digital assets contained in the dataset and weekly BTC return.
- Rolling correlation between these newly created variables and the target variable, aiming to capture their relationship over time.

Feature selection was an important step to refine the feature set for robust forecasting. In that sense, we resorted to the Granger causality test to identify causal relationships between time series variables based on temporal sequences.

As a prerequisite, we ensured that our time series data met the stationarity requirement. Stationarity is a crucial condition for reliable Granger causality testing, as it stabilizes the statistical properties of the time series. This step involved differencing the data as needed to achieve stationarity.

With stationarity established and ensured by the ADF test, we applied the Granger causality statistical test to assess whether past values of one variable could provide statistically significant predictive information about the target variable, Bitcoin price. This analysis was performed iteratively for the entire set of candidate features. Granger causality test yielded valuable insights into the temporal dependencies between various features and Bitcoin price. As a result, we were able to identify 26 features that exhibited statistically significant causal relationships with the target variable. These features were deemed instrumental in capturing critical information for our short-term forecasting models.

Feature extraction was also implemented in this work. Two main techniques were employed to extract new features from the original dataset. Principal Component Analysis (PCA) was implemented, reducing the dimensionality of the dataset while preserving its essential information. We retained five principal components, explaining approximately 85% of the variance of the target variable, which contribute significantly to capturing underlying patterns.

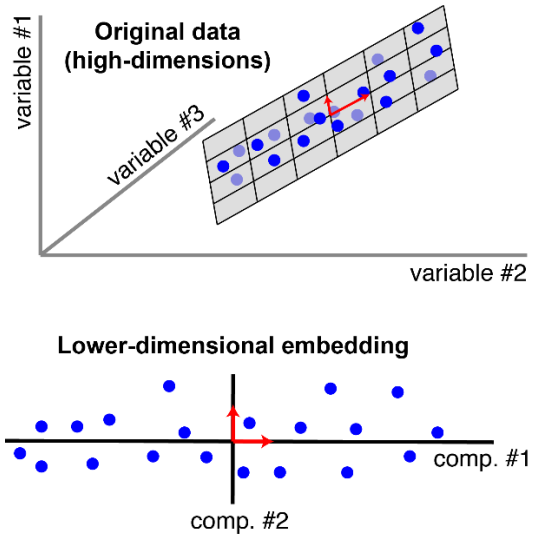

Figure 20: Example illustration of principal component analysis

Source: https://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/

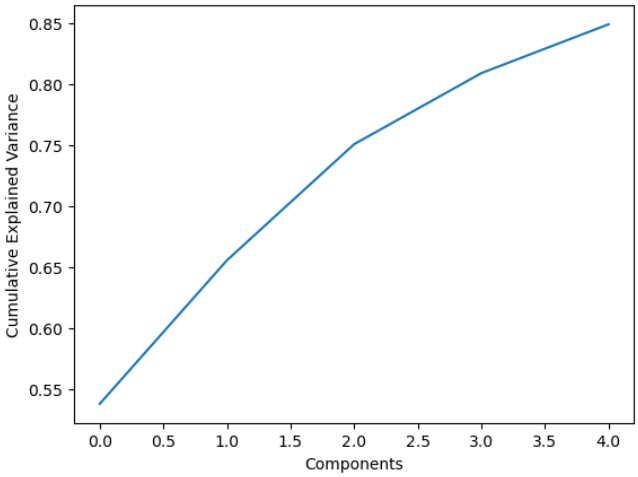

Figure 21: Cumulative explained variance by the number of PCs retained.

Additionally, we employed the cross-correlation function from "statsmodels" package to systematically quantify the cross-correlation between each principal component and the target variable. This step facilitated a deeper understanding of how these components related to the target variable across different time lags.

Also, for feature extraction, an autoencoder was implemented. An unsupervised learning method, consisting of a neural network model that seeks to learn a compressed representation of an input. Once fit, the encoder of the model was used to compress sequence data, with the bottleneck layer, that in turn was used as a feature vector input to our supervised learning models. Using this model for feature extraction 17 new features were created from the original 51 features.

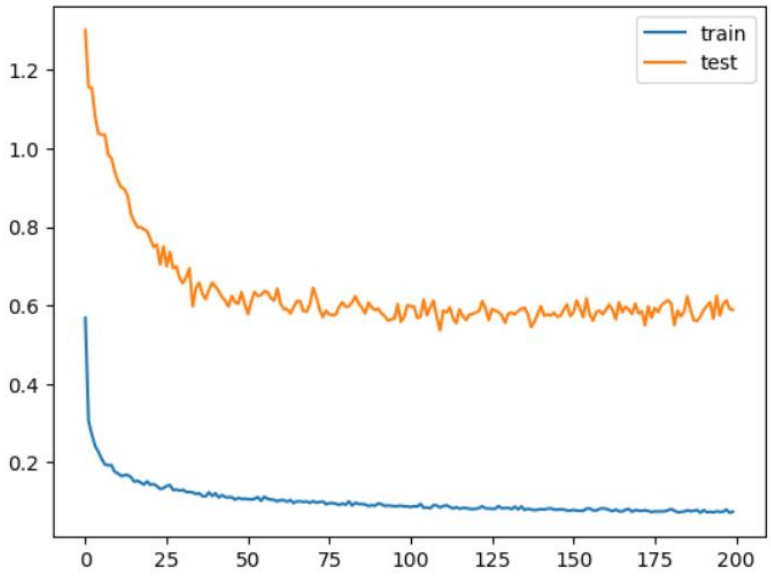

Figure 22: Learning curve of autoencoder

All these transformed features, totaling 27 in number, are then concatenated. Among these features, 17 are derived from the autoencoder, while the remaining 10 features are the result of Granger selection, Principal Component Analysis (PCA) and cross correlation preprocessing pipeline.

Due to market regime switch, imposed by the continuous market evolving, training and validation datasets distributions can be very different from the test dataset distribution, increasing the difficulty of implementing a robust deep learning model. To this end, a posterior transformation involves using the Yeo-Johnson power transformation to make the data distributions more similar to a Gaussian distribution. This transformation is used to stabilize the variance and reduce the skewness of the data distributions in order to improve the predictive ability of the models, as a skewed predictor distribution can have a detrimental effect on the prediction models as the tails of the distribution can dominate the underlying calculations.

Following the power transformation, the feature dataset was transformed to achieve stationarity through differencing, by subtracting the current value of the series from the previous one. ADF statistical test was employed to compare the null hypothesis of non-stationarity with the alternative hypothesis of stationarity.

Differencing was not applied to the target variable to ensure that the model's output directly represents what is expected to be predicted.

Although this technique has some disadvantages for forecasting, such as the possibility of introducing noise and randomness into the data, as it eliminates some of the information and structure of the original series, differencing has proved its importance by making the data easy to model and improving the performance of the forecasting models studied.

The resulting features, after the detailed preprocessing pipeline, are then used as inputs to the predictive models.

3.3 Impact of Feature Engineering in LSTM Predictive Performance

Figure 23 represents four different confusion matrices with respect to the prediction of an LSTM classifier on the same test dataset. This LSTM model shared the same architecture in all four experiments. However, different features were used to train the model.

In the first quadrant confusion matrix, the LSTM was trained with features resulting from the Granger causality test and PCA preprocessing pipeline. The second quadrant confusion matrix represents an LSTM trained with multivariate data without any transformations other than data cleaning and scaling, and in the third quadrant confusion matrix, features extracted by the autoencoder were used for training. Lastly, in the fourth quadrant, both feature sets resulting from the Granger and PCA pipeline and from the autoencoder were concatenated and used for training the model.

These predictive experiments, using an LSTM classifier, demonstrate the influence and importance of a robust feature engineering process aiming to extract the maximum possible information from raw data.

By observing Figure 23, using multivariate data without any transformation other than cleaning and scaling as a baseline, it is possible to observe that applying a feature selection method, namely the Granger causality test along with PCA yielded better predictive results. It can also be observed that training on data extracted from the autoencoder produced results similar to those of the transformed data used to construct the first quadrant confusion matrix. However, a steep increase in prediction accuracy is noticeable in the fourth quadrant, when both feature sets are concatenated and serve as input for the LSTM training.

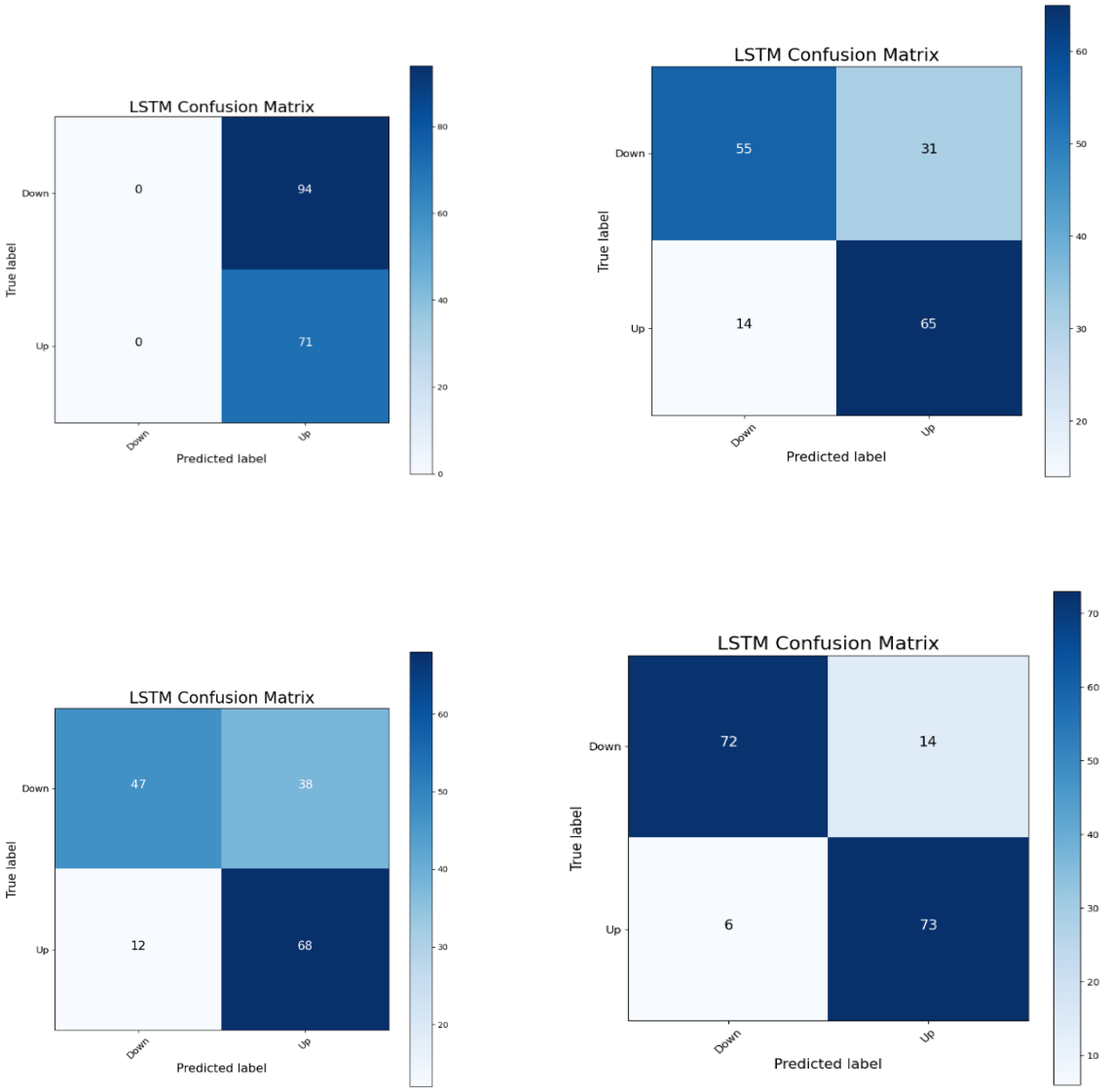

Figure 23: Distinct LSTM predictive performance by training with different feature transformations.

3.4 Data Preparation

The dataset was partitioned into training, validation, and test subsets following an 80/10/10 split as part of the data preparation process.

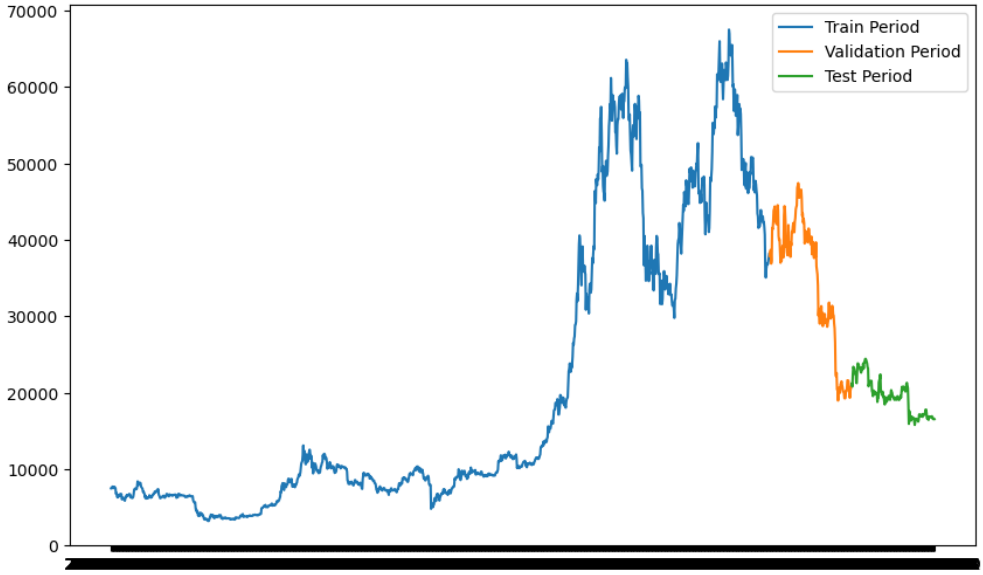

Figure 24: Splitting BTC price series into train, validation, and test periods.

In this phase of the data preprocessing step, the focus was on structuring of the dataset to align with the requirements of the explored deep learning models. To achieve this, a custom function was implemented to transform the data into a suitable format to effective modeling.

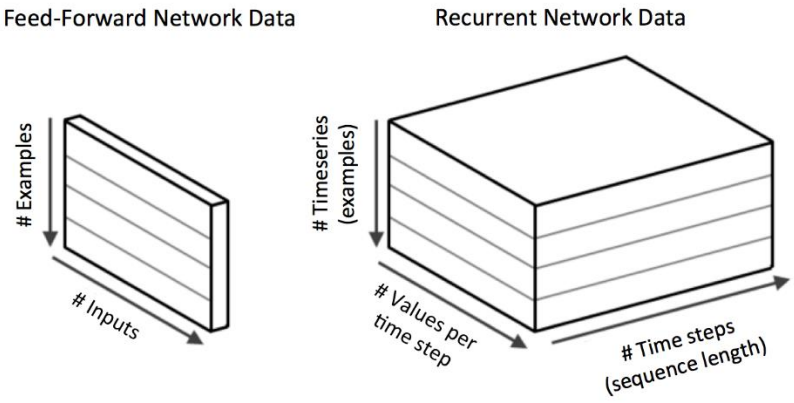

Figure 25: Normal input vectors compared to recurrent neural networks input.

Source: (Singh et al., 2022)

This implemented function's primary objective is to organize our dataset into a three-dimensional structure that captures the temporal relationships necessary for the deep learning models implemented such as LSTM, GRU, TCN.

It segments the dataset into sequential windows of a defined length, carefully aligned with the corresponding target variables. Doing so imbues our data with temporal context, enabling our models to discern and leverage historical patterns and dependencies.

It is also important to notice this function accommodates both classification and regression tasks. For classification, it annotates each data window with binary labels, discerning whether the subsequent data point represents an increase or decrease in the target variable. In regression tasks, it retains the temporal sequence without labeling, allowing the models to predict future values directly.

```python
def sequential_window_dataset(series, window_size, classification=True):
    ds = tf.data.Dataset.from_tensor_slices(
        series
    )  # returns a dataset where each item corresponds to one time step (a row in the series)
    ds = ds.window(window_size + 1, shift=window_size, drop_remainder=True)
    ds = ds.flat_map(lambda window: window.batch(window_size + 1))
    if classification:
        ds = ds.map(
            lambda window: (
                window[:-1],
                tf.expand_dims(
                    tf.where(window[1:, -1] > window[:-1, -1], 1.0, 0.0), axis=-1
                ),
            )
        )
    else:
        ds = ds.map(
            lambda window: (window[:-1], tf.expand_dims(window[1:, -1], axis=-1))
        )
    return ds.batch(1).prefetch(1)
```

Figure 26: Data preparation function "sequential_window_dataset".

This comprehensive data preprocessing pipeline was a vital component in identifying hidden patterns in the data, also ensuring the data was well-suited for deep learning models training and evaluation. All the applied transformations laid a solid foundation for the subsequent stages of model development.

3.5 Deep Learning Models

In this section, the ensemble of deep learning models harnessed to tackle the complex task of Bitcoin price forecasting is presented. These sequence-to-vector models were designed and configured to capture the temporal patterns within the data to predict one day-ahead.

A comprehensive exploration of deep learning algorithms for both classification and regression tasks were employed, with the implementation of different architectures as it will be detailed below.

Zero Rule Classifier:

ZeroR is a naive classification method that relies on the target and ignores all predictors. The ZeroR classifier simply predicts the majority class.

Although there is no predictive power in ZeroR, it is useful for establishing a baseline performance as a benchmark for the other classification methods included in this research.

Persistence Model:

A persistence model assumes the future value of a time series is calculated under the assumption that nothing changes between the current time and the forecast time (Paulescu et al., 2021), using the value at the current time step (t) to predict the expected outcome at the next time step ($t + 1$).

This model acts as a baseline for our regression models.

LSTM Network:

After reviewing related literature, LSTM model is found to be a good candidate for predictive modeling due to its memory-keeping ability.

With respect to its architecture and design, tanh activation functions were applied in all LSTM layers, and stateful architecture was preferred to maintain temporal dependencies: hidden state is preserved at each training iteration, allowing the model to detect patterns concealed in larger data than the defined input sequence. At the end of each epoch, states were reset, as they could grow too large and become unstable.

Model	Description
LSTM	LSTM layer with 100 units LSTM layer with 100 units LSTM layer with 100 units Dropout layer with rate = 0.2 Output dense layer with 1 unit

Table 1: Parameter specification for LSTM

GRU Network:

Configured with similar layer dimensions as the LSTM model for consistency. Tanh activation functions were set in all GRU layers.

Model	Description
GRU	GRU layer with 100 units GRU layer with 100 units GRU layer with 100 units Dropout layer with rate = 0.2 Output dense layer with 1 unit

Table 2: Parameter specification for GRU.

Bidirectional LSTM Network:

Model	Description
BiLSTM	BiLSTM layer with 2 x 100 units BiLSTM layer with 2 x 100 units BiLSTM layer with 2 x 100 units Dropout with rate = 0.2 Output dense layer with 1 unit

Table 3: Parameter specification for BiLSTM

Hybrid Architectures:

In time series analysis, it is common to apply a smoothing technique prior to analysis, such as a moving average or a weighted moving average based on domain knowledge, to reduce noise in the data. Convolutional neural networks are explored in this research since

their architecture allows to learn smoothing parameters. In this sense, three hybrid models have been implemented by combining a one- dimensional convolutional layer for pre-processing and noise reduction with LSTM, GRU and BiLSTM layers for sequential modeling. Since the 1D convolutional preprocessing layers are part of the same network that outputs the predictions, by optimizing the neural network loss, one optimizes the smoothing parameters directly to perform well in the prediction tasks.

The 1D convolutional layer, present in all three hybrid models implemented for each task, shifts 20 filters, also known as kernels, across the time axis in all input sequences. Since each filter has a size of four, the output of each filter is computed based only on the last four time steps. Specifically, the output of each filter is computed by determining the weighted sum of the values of these four-time steps and a bias. Then the resulting smoothed vector is used as input to a ReLu activation function, which applies a non-linearity to it to obtain the final filter output. Two other different hyperparameters were configured for these convolutional layers, namely the type of padding and the strides. The type of padding was set to "causal", so that the input sequence is the same size as the output sequence, by padding the left side of the output sequence with zeros. The stride was set to one, which means that the filters slide one time step at a time. It is also important to note that these convolutional layers do not retain any memory like RNNs, and they can handle input sequences of any size. The number of parameters just depends only on the kernel size and the number of kernels, not on the length of the input sequences.

Consult Figure 38 in Appendix for an example implementation of hybrid CNN-LSTM model.

Model	Description
CNN-LSTM	Convolutional layer with 20 filters of size (4,0) and padding = “causal” LSTM layer with 100 units LSTM layer with 100 units Dropout layer with rate = 0.2 Output dense layer with 1 unit
CNN-GRU	Convolutional layer with 20 filters of size (4,0) and padding = “causal” LSTM layer with 100 units LSTM layer with 100 units Dropout layer with rate = 0.2 Output dense layer with 1 unit

CNN-BiLSTM	Convolutional layer with 20 filters of size (4,0) and padding = "causal" BiLSTM layer with 2 x 100 units BiLSTM layer with 2 x 100 units Dropout layer with rate = 0.2 Output dense layer with 1 unit

Table 4: Parameter specification for hybrid architectures

Attention-Based Architecture:

After reviewing recent literature (Zhang et al., 2023), CNN-BiLSTM-Attention demonstrated superior predictive accuracy compared to other variants like LSTM, CNN-LSTM, and CNN-LSTM-Attention models in forecasting the Chinese stock index – CSI300. Given its notable performance and the potential advantage it offers in dissecting the contributions of individual hidden states for making informed decisions, we have included the CNN-BiLSTM-Attention architecture as part of our array of deep learning models for the tasks proposed in this study.

The implemented Bahdanau attention mechanism is built with a custom attention layer that outputs a "context vector" and "attention weights". Consult Figure 39 in the Appendix for the implementation of the CNN-BiLSTM-Attention model.

Model	Description
CNN-BiLSTM-Attention	Convolutional layer with 20 filters of size (4,) and padding = "causal" BiLSTM layer with 2 x 100 units Dropout layer with rate = 0.2 BiLSTM layer with 2 x 100 units Attention layer with 100 units. Output dense layer with 1 unit

Table 5: Parameter specification for CNN-BiLSTM-Attention

Temporal Convolutional Network

The implemented TCN employs a single layer to process temporal sequences. It utilizes dilations as a key mechanism to control the network's exploration of past information within the input sequence.

Dilations, represented as factors (1,2,4,8,16,32), control how the convolutional kernel scans the input sequence. Each dilation factor dictates the gap between observed elements, influencing the receptive field (the span of past context examined at each step). With a kernel size of three, the convolutional kernel transverses the input sequences based on the specified dilation factors. As dilation increases, the kernel incorporates information from temporally distant elements, effectively broadening the network's receptive field without the need for additional layers. The receptive field of this single-layer TCN results from the interplay between the kernel size and the chosen dilations factors.

In summary, this TCN's architecture harnesses dilations to sample temporal information selectively across different spans, culminating in a receptive field that facilitates learning of diverse temporal patterns within the input sequence.

Model	Description
TCN	TCN layer with 64 filters of size (3,), padding = "causal" and dilations = (1,2,4,8,16,32) Output dense layer with 1 unit

Table 6: Parameter specification for TCN.

Transformer Encoder:

Since the decoder of the Transformer model is tailored for sequence-to-sequence tasks, translating input sequences into corresponding output sequences, this research focuses solely on the encoder component, connecting the encoder output directly to the final layer. This design allows one-step-ahead forecasts, aligning with the sequence-to-vector modeling approach.

This encoder should outperform the recursive models because it allows for more flexible parallelization, more efficient long-term memory retention, and fewer vanishing or exploding gradient problems. The activation function in the last layer of the model was modified to handle the different tasks: classification and regression.

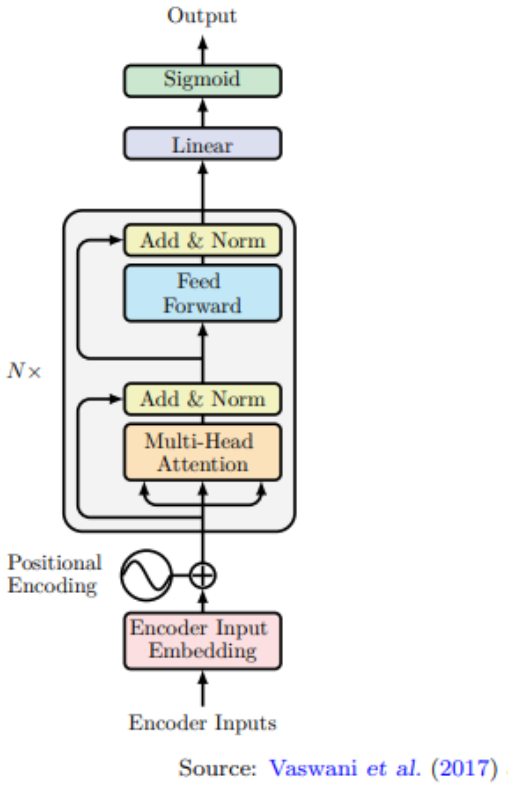

Figure 27: Transformer encoder architecture

Model	Description
Transformer Encoder	Layer normalization Multi-head attention layer with head_size=256, num_head=4, dropout=0.2 Dropout layer with rate = 0.2 Layer normalization Convolutional layer with 4 filters of size (2,) Dropout layer with rate = 0.2 Convolutional layer with “features_dim” filters of size (1,) Global average pooling layer Dense layer with 128 units Dropout layer with rate = 0.2 Output dense layer with 1 unit

Table 7: Parameter specification for Transformer Encoder

Stacking Ensembles:

Stacking ensemble technique was selected based on literature. The authors in (Livieris et al., 2020) after comparing multiple ensemble strategies for cryptocurrency price prediction, elected stacking ensemble with KNN as meta-learner, as the best ensemble. A stacking ensemble utilizes a meta-learner to learn the prediction behavior of the base learners, with respect to the final output. KNN is a non-parametric supervised learning method, where its input consists of the *k* closest training examples in a dataset.

For the regression stacking ensemble, base learners were selected based on the lowest value of root mean squared error metric, namely the GRU, BiLSTM and CNN-BiLSTM from the univariate approach. The meta-learner chosen to train with the predictions made by these base learners was linear regression, as it gave better results than the KNN regressor.

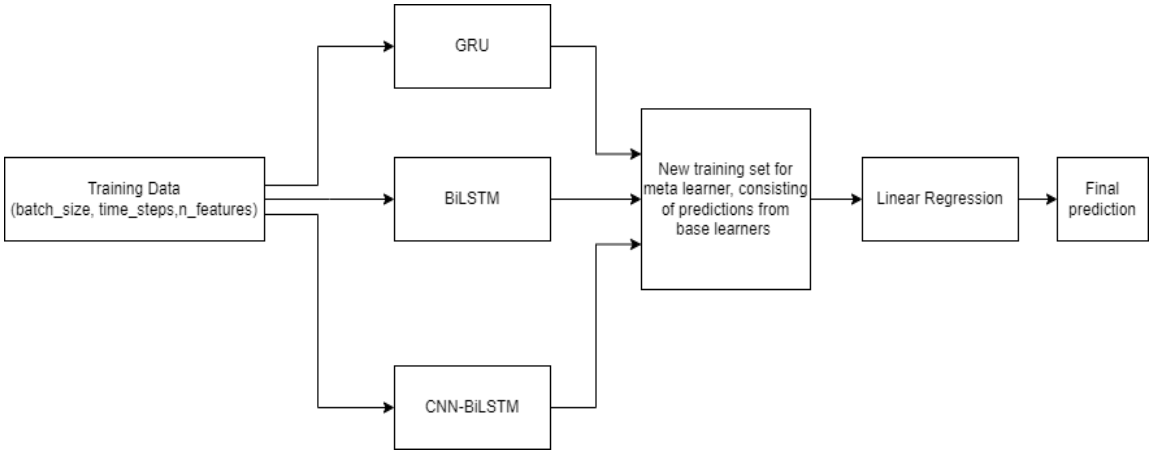

Figure 28: High level stacking ensemble architecture for regression predictive modeling to predicting the price on test dataset.

For the stacking ensemble for classification, three base learners were selected based on the highest accuracy. The selected base learners were TCN, LSTM and GRU from the multivariate approach, and includes a KNN classifier with 10 neighbors as the meta-learner.

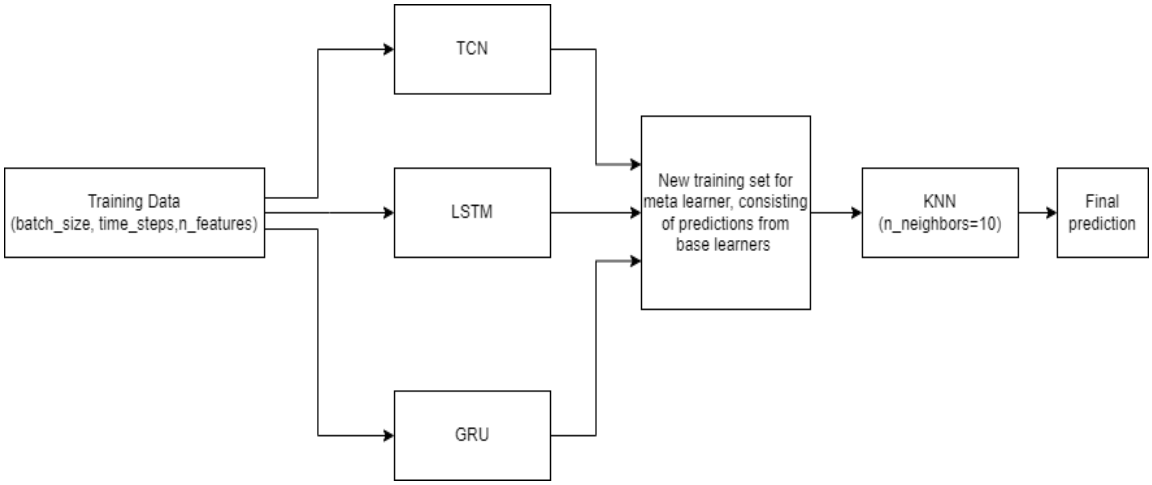

Figure 29: High level stacking ensemble architecture for classification predictive modeling to predict the trend on test dataset.

For the stacking ensemble responsible for modelling and predicting next day returns, the selection of the three base learners was based on the lowest RMSE. Using this criterion, BiLSTM, CNN-BiLSTM-Attention, and LSTM were selected as the base learners, and KNN was selected as the meta-model.

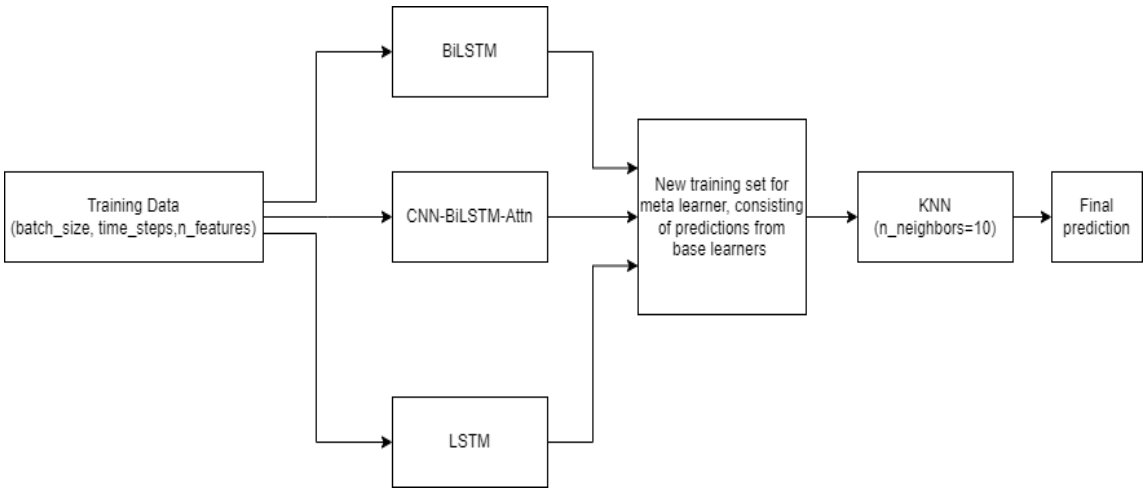

Figure 30: High level stacking ensemble architecture for regression predictive modeling to predict the return on test dataset.

It is important to note that a natural implementation design common to all deep learning regressors is that the output-dense layer has a linear activation function, whereas, in all deep learning classifiers, the output-dense layer has a sigmoid activation function.

3.6 10.6 Model Training

This phase of the research is related to the training of the deep learning models. This section encapsulates the rigorous process of model optimization and fine-tuning, which includes several key elements.

All models were trained using the ADAM optimizer, a variation of the gradient descent optimization algorithm tasked with updating the weights of the neural networks during training.

The Many-To-One relationship is implemented across all models, defining 15 as the time window size, which results in the models processing 15 lags of each feature to predict the next value of the target variable.

It is also important to mention there are two different cost functions for each type of task being researched. For regression, the configured cost function is the mean squared error, and binary cross-entropy is used as the cost function for the models solving for classification.

The model training pipeline featured a range of callbacks playing a key role in their performance results, including learning rate scheduling, a critical element in training deep learning models by dynamically adjusting the learning rates. By leveraging a learning rate scheduler, it is possible to systematically adapt the learning rate throughout the training process. This dynamic adjustment was instrumental in finding the optimal convergence path, improving the models' ability to navigate complex optimization problems efficiently.

Early stopping callback was implemented to act as a monitor and keep track of the validation loss during training. It intervened when it detected signs of overfitting, halting training to prevent the models from memorizing noise in the data. This approach not only preserved model integrity but also improved generalization.

Resetting states was also featured in models with recurrent layers preserving state information across training batches. The reset states callback ensured that stateful models started each epoch with a clean slate, maintaining the temporal dependencies while preventing excessive reliance on past data.

To also monitor the validation loss, model checkpoints were employed to save the best-performing model during training. This safeguarded against potential disruptions and allowed us to retain the model with the highest predictive performance.

Throughout the training process, we evaluated model performance using dedicated validation datasets. This enabled tracking progress, identifying potential issues, and making informed decisions about model adjustments.

4 Experimental Results

4.1 Model Performance

This section will present the evaluation of the predictive models implemented for solving the three proposed tasks: forecasting next-day trends, price, and return.

First, a comprehensive evaluation of Bitcoin’s directional state is presented.

	Univariate			Multivariate		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Zero Rule Classifier	45.5	45.5	**100**	45.5	45.5	**100**
LSTM	43	43	100	87.9	83.9	92.4
GRU	43	43	100	86.1	84.1	87.3
BiLSTM	81.8	**75.9**	84.5	83.6	82.5	83.5
CNN-LSTM	43	43	100	83	83.1	81.1
CNN-GRU	43	43	100	80	81.1	75.6
CNN-BiLSTM	74.5	64.9	88.7	73.9	64	90.1
CNN-BiLSTM-Attn	**82.4**	73.3	93	81.8	81.8	79.7
TCN	43	43	100	**90.9**	**89**	92.4
Transformer Encoder	45.4	44.1	94	81	73.2	89.6

Table 8: Evaluation results of BTC directional status classification on test dataset

Meta learner	Base Learners	Accuracy	Precision	Recall
	TCN	74.5	72.3	66.2
KNN	LSTM			
	GRU			

Table 9: Evaluation of stacking ensemble classifier for trend forecasting on test dataset.

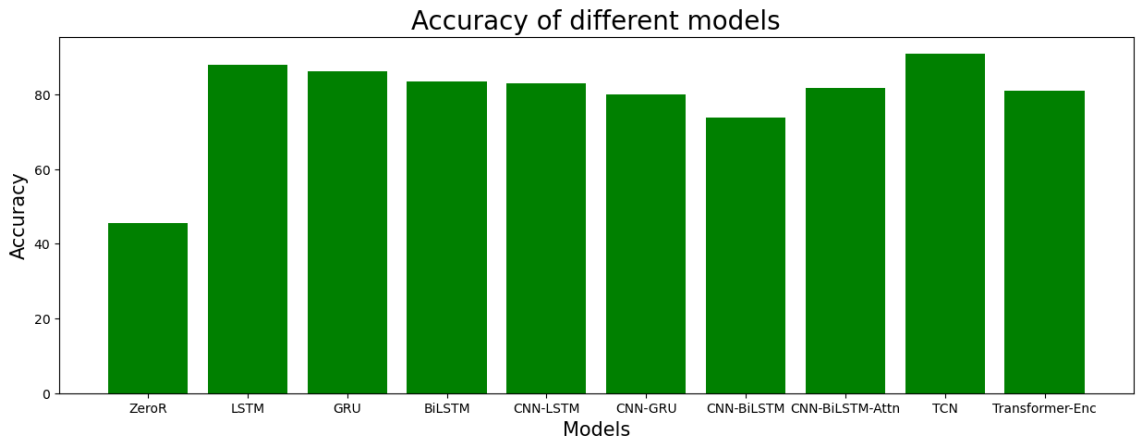

Figure 31: Accuracy scores of all models on test dataset

By observing Figure 31, TCN is the model that achieves the highest accuracy among the other deep learning models.

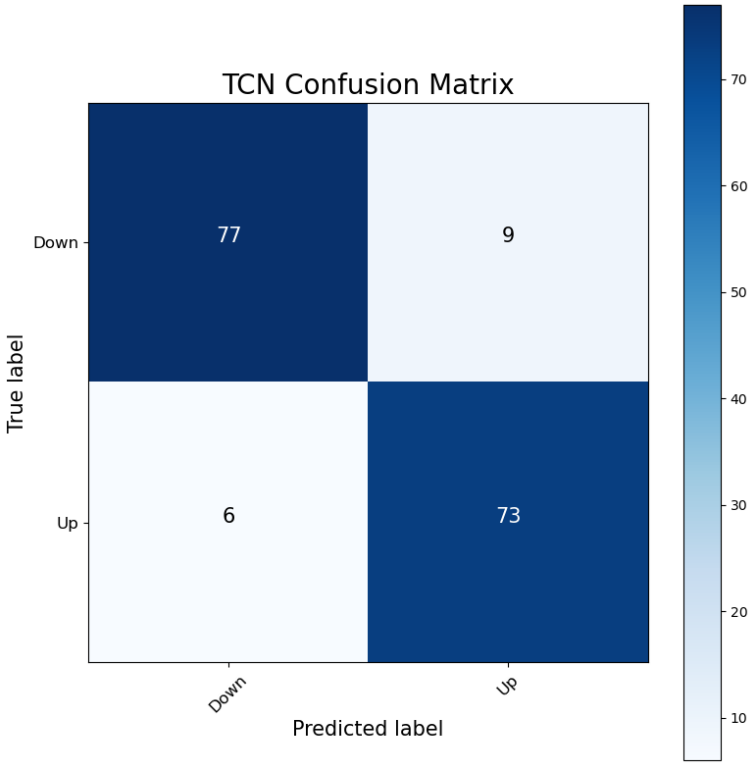

Figure 32: TCN confusion matrix on test dataset.

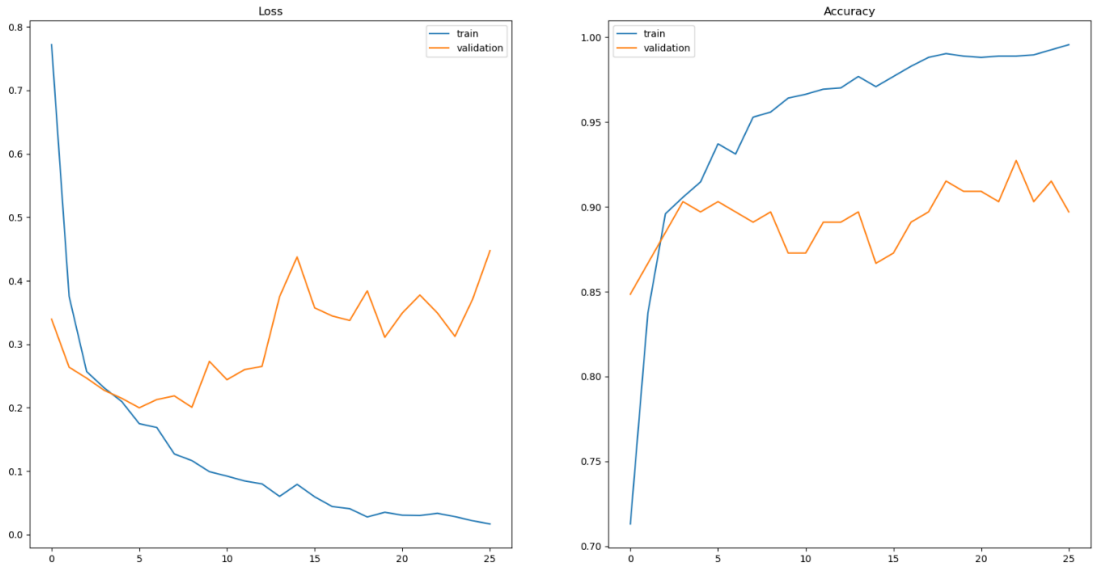

Figure 33: Learning curve of TCN classifier

After analyzing the classification task, price forecasting models were also evaluated.

	Univariate				**Multivariate**			
	MAE	RMSE	MAPE	R^2	MAE	RMSE	MAPE	R^2
Persistence Model	646.5	1002471.	2.31	**0.99**	646.5	10024	2.31	0.99
LSTM	622.2	921.2	3.1	0.846	909.7	1075.8	4.6	0.79
GRU	383	619.6	1.94	0.93	**662.2**	**812.5**	**3.4**	**0.88**
BiLSTM	404	589.7	2.06	0.94	1186.2	1367.2	6.1	0.66
CNN-LSTM	601.6	915	3.01	0.85	1677.3	2097.2	8.5	0.2
CNN-GRU	528.8	780.9	2.6	0.89	1282.5	1678.9	6.6	0.49
CNN-BiLSTM	**363.1**	**554**	**1.82**	0.94	1257.8	1560.9	6.37	0.55
CNN-BiLSTM-Attn	446.5	663.2	2.28	0.92	1590	1972	8.33	0.3

TCN	792.1	925.7	4.2	0.84	915.4	1191	4.70	0.74
Transformer Encoder	440	686.7	2.25	0.91	1094	1433	5.81	0.59

Meta learner	**Base learners**	**MAE**	**RMSE**	**MAPE**	R^2
	GRU	336	454.8	1.74	0.96
Linear Regression	BiLSTM				
	CNN-BiLSTM				

Table 10:

Table 11: Evaluation of stacking ensemble for BTC price forecasting

By observing Table 10 and Table 11, stacking ensemble for BTC price forecasting surpasses all base learner's predictive performance.

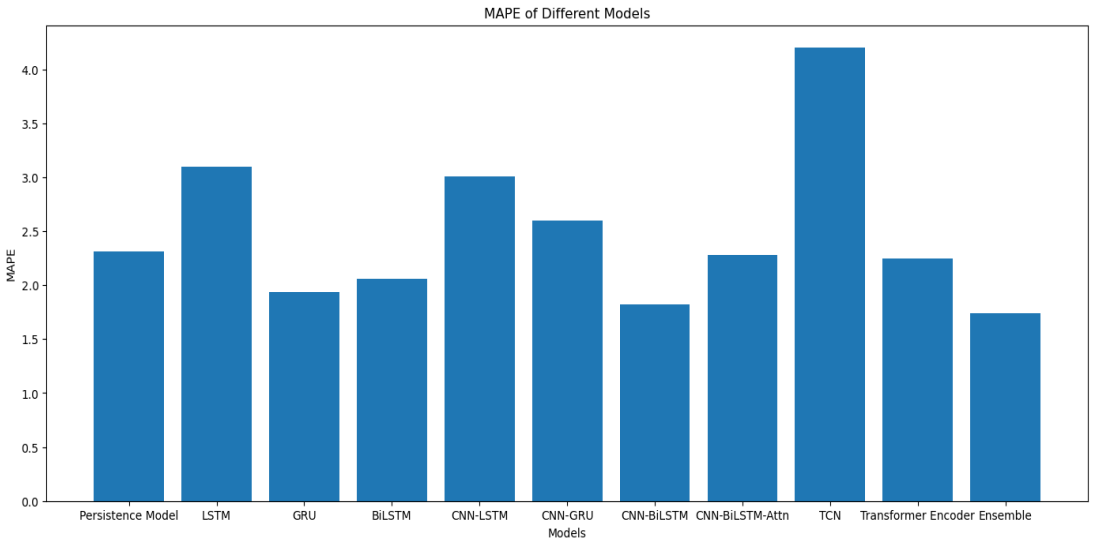

Table 12: MAPE score across all models on test dataset.

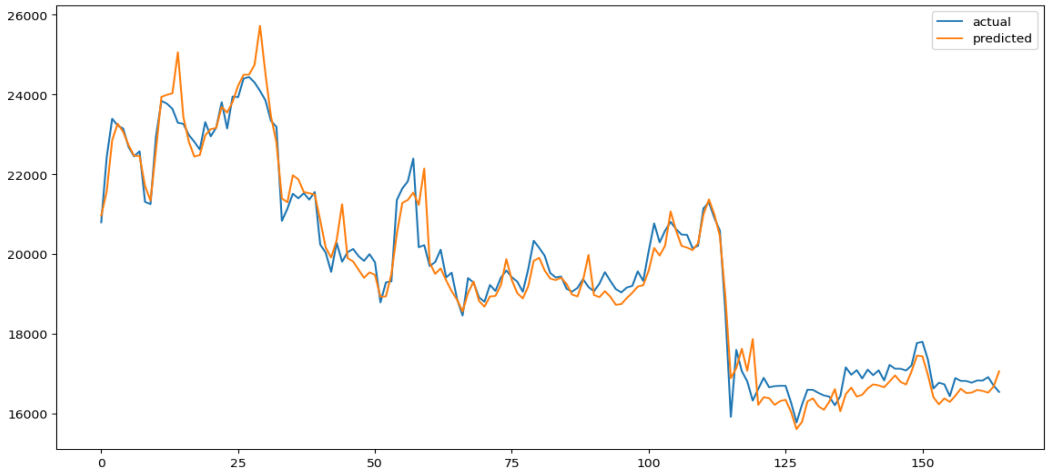

Figure 34: Predicted vs ground truth with stacking ensemble on test dataset.

Finally, the last regression problem, concerning to BTC return forecast is evaluated.

	MAE	**RMSE**	R^2
Persistence Model	0.93	1.3	-1.08
LSTM	0.25	0.35	0.8
GRU	0.28	0.41	0.73
BiLSTM	**0.11**	**0.16**	**0.96**
CNN-LSTM	0.31	0.42	0.71
CNN-GRU	0.34	0.46	0.65
CNN-BiLSTM	0.34	0.46	0.65
CNN-BiLSTM-Attn	0.21	0.28	0.87
TCN	0.26	0.37	0.77
Transformer Encoder	0.32	0.45	0.63

Table 13: Evaluation of BTC returns forecasting.

Meta learner	Base learners	MAE	RMSE	R^2
	BiLSTM	0.53	0.8	0
KNN	CNN-BiLSTM-Attn			
	LSTM			

Table 14:Evaluation of stacking ensemble for BTC return forecasting.

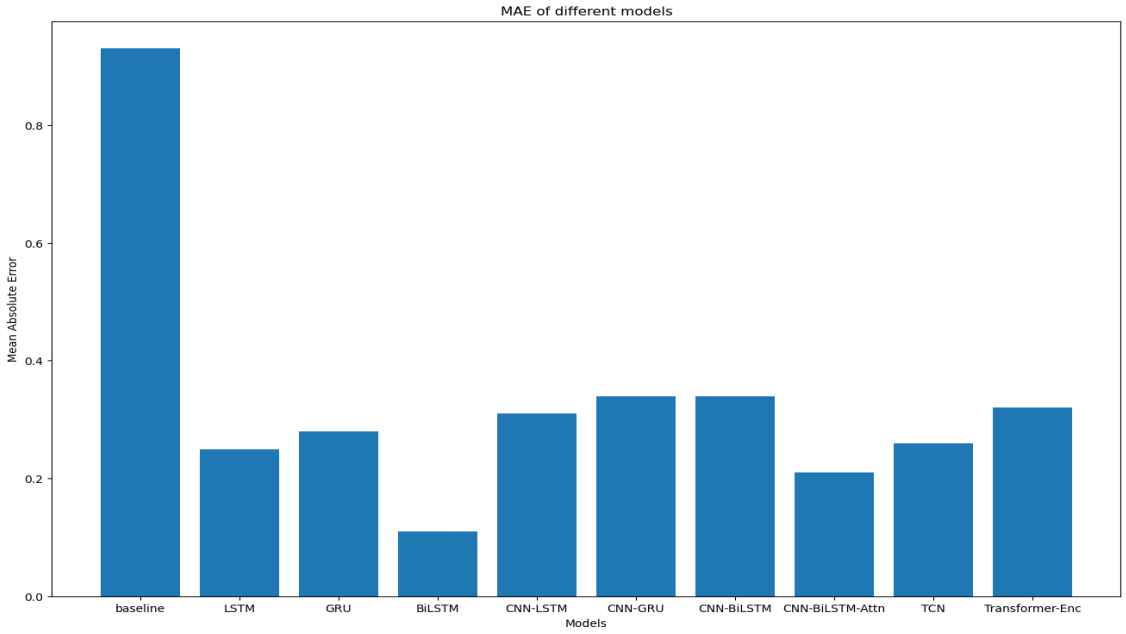

Figure 35: MAE score across all models on test dataset.

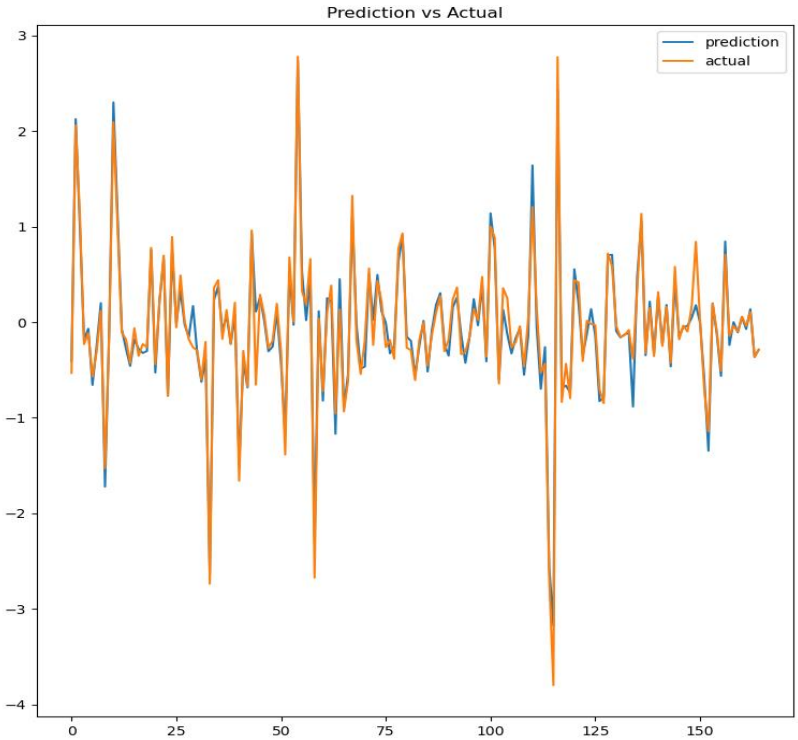

Figure 36: Predicted vs ground truth with BiLSTM on test dataset.

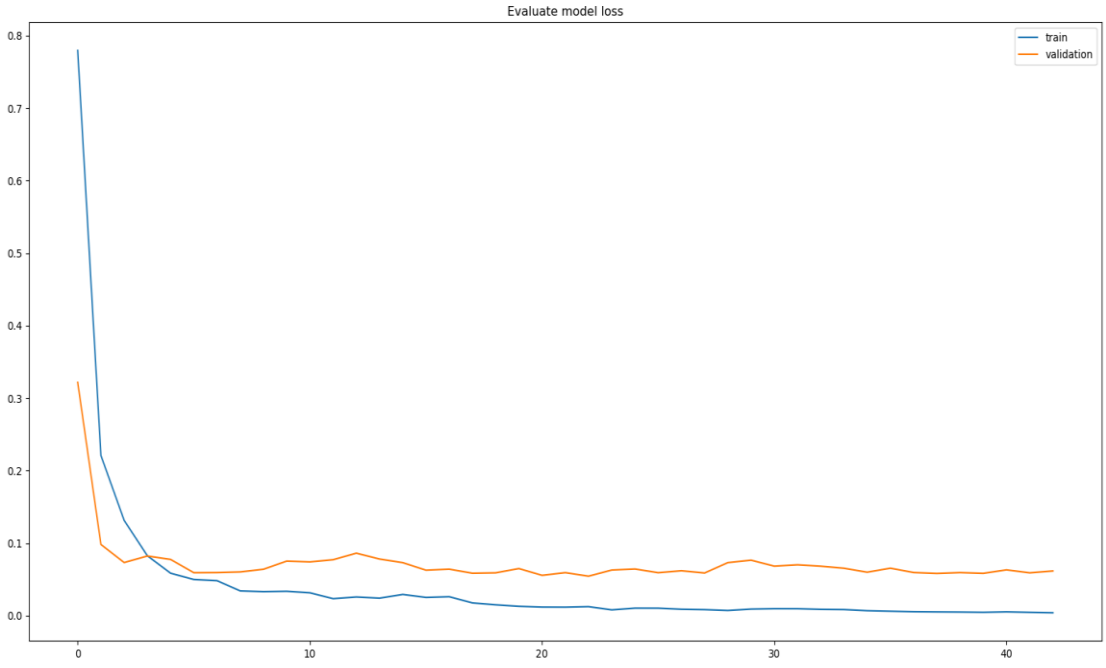

Figure 37: Learning curve of BiLSTM.

5 Discussion

In this section, we discuss and compare the results of our proposed model, other approaches, and the most related works.

As stated by several researchers, like McNally et al. (2018), Adcock & Gradojevic (2019) or Aparicio et al.(2022), classical statistical techniques do not perform as well as deep learning in the realm of cryptocurrency price forecasting. Considering that most of the related work relates to research and implementation of advanced machine learning and deep learning algorithms aimed to tackle this problem.

This research shows superior performance when compared to most of the related work, surpassing the predictive accuracy of all related work, except for (J. Shen & Shafiq, 2020), where the authors achieved 93.3% accuracy. Our study shares some similarities with the authors regarding the significance of feature engineering and deep learning models. However, a critical divergence lies in the different datasets used for modeling and forecasting, as the aim of the research was to predict bi-weekly equity price trend. When comparing to (Greaves & Au, 2015), the author only considered blockchain-related information, resulting in a movement classification accuracy of 55%, which relates to the importance of a diversified data collection process. Collecting data from multiple sources was an important process of our research, enabling the predictive models to capture patterns more effectively and attain superior performance.

Other research, such as (Mudassir et al., 2020) and (Dimitriadou & Gregoriou, 2023), although similar to ours with respect to the collection of data from multiple sources and the implementation of a careful feature preprocessing, fails to implement more advanced architectures, which could have led to an improved result from the 65% and 66% accuracy obtained, respectively.

It is also important to mention that Ortu et al., (2022) explored deep learning models and a wide set of features, including technical, trading, and social data. Our research complements their findings by also encompassing similar types of data and improving the performance of daily classification of the price trend.

This demonstrates the impact of data preprocessing techniques and advanced architecture exploration in enhancing the accuracy of cryptocurrency price forecasting models.

6 Conclusion

6.1 Findings

Non-stationary time series exhibit evolving statistical distributions over time, which results in a changing dependency between the input and output variables. Nevertheless, almost all implemented deep learning classifiers were able to take advantage of the explanatory features to increase their predictive accuracy largely, ranging from 73.9% to 90.9% in the multivariate approach. It is important to note that feature engineering played a significant role in achieving these lower forecast errors when compared to the univariate approach.

In the regression task, multivariate data did not have a positive effect on predictive performance, with all regressors performing worse on multivariate data than on univariate data. However, five regressors were able to outperform the persistence model baseline in the univariate approach, with a MAPE below 2.31%, with the hybrid architecture CNN-BiLSTM showing the lowest MAPE of 1.82% when considering stand-alone models only.

In the regression problem of predicting bitcoin daily returns, all deep learning models outperform the baseline in all evaluation metrics, with BiLSTM being the model with the lowest prediction errors. In fact, BiLSTM shows good performance in each of the forecasting problems. It is always able to outperform the baseline, and in addition to being the best model for forecasting bitcoin daily returns, it also presents the highest accuracy in the univariate approach of the classification task with 81.8% accuracy, which is surpassed by the TCN with 90.9% accuracy in the multivariate approach.

Looking at the evaluation metrics of the hybrid architectures, adding a CNN preprocessing layer generally reduces the predictive performance of the model, as the evaluation metrics of the hybrid models worsen across the prediction tasks. The attention mechanism in the hybrid CNN-BiLSTM resulted in better performance in two tasks out of three, by improving the predictive performance of the hybrid model in forecasting the daily return and trend.

It is also important to note that the price prediction stacking ensemble was able to outperform the base learners, achieving a MAPE of 1.74%, making it the best performing model in the price prediction modelling task. The classifier stacking ensemble was not able to outperform the base learners, achieving an accuracy of 75%. Similarly, the ensemble model tasked with predicting yields produced negative results, achieving an RMSE of 0.8, which was only better than the baseline RMSE.

6.2 Implications

This research has a strong focus on feature engineering and its impact on prediction performance. Several techniques have been tested, such as RFE or Random Forest feature importance. However, the integration of methods that extract features through linear relationships, such as PCA, and methods capable of extracting new features through non-linear relationships, such as AE, proved to be critical to achieving high quality. Furthermore, the prediction of the daily trend of the BTC showed a significantly high accuracy, thanks to preprocessing techniques such as stationarity and power transformations. These transformations facilitated pattern learning, as evidenced by the faster convergence of the models when applied.

The data preprocessing pipeline developed here can serve as a guideline for future research in the field of financial time series forecasting.

Furthermore, it is worth noting that our research included an extensive exploration of architectural designs. These explorations encompassed recurrent models, hybrid models and an attention mechanism, evaluated both from a univariate and multivariate approach, enabling a comprehensive analysis of which model and approach are best suited for different tasks.

6.3 Limitations

In the development of robust forecasting models to predict financial time series, several limitations and challenges have surfaced. These limitations are essential to

acknowledge, as they may impact the scope and generalizability of our findings. These limitations are also detailed in (Israel et al., 2020).

One significant limitation pertains to data availability. While we collected a diverse set of features, encompassing market data, macroeconomic indicators, network data, and social popularity, some essential variables remained beyond our reach. Notably, we lacked access to options data such as open interest data, and certain other proprietary information that could potentially enhance our predictive capabilities. Also, regarding data restrictions, it is important to note the limited size of important financial datasets, as some datasets have few data points, also distributed among different market regimes, and their nature is usually stochastic and non-stationary, making it difficult to generate synthetic data that tracks the temporal dynamics of the financial markets.

Low signal-to-noise ratio was also a limitation in our research. Unlike traditional use cases of machine learning techniques such as natural language processing, generally the signal-to-noise ratio in financial data is low.

Another difficulty is the market-changing dynamics and regime switches, presenting a challenge for a machine learning model to tackle a forecasting problem, as the training dataset can have a very different distribution from the test dataset.

6.4 Future Work

As we conclude our research in Bitcoin time series forecasting with deep learning models, we recognize several promising avenues for future research and areas where further exploration could yield significant advancements. This section outlines some of these potential directions:

Integration of sentiment analysis with natural language processing (NLP). NLP-based sentiment analysis techniques, such as BERT (Bidirectional Encoder Representations from Transformers), to extract sentiment from social media and news feeds. The results from (D. Shen et al., 2019) showed that the number of tweets on Twitter can influence the trading volume of BTC for the next day. In this sense, analyzing sentiment data from different sources can serve as a valuable predictor and lead to promising results.

Exploring advanced transformer models, including the Temporal Fusion Transformer (Lim et al., 2021) and other transformer-based neural network architectures tailored for time series data, to investigate the application of transformers to capture long-term dependencies and temporal patterns in cryptocurrency price data.

Also, expanding the dataset to include options data such as open interest and miners' outflow sources will enhance the feature set and deepen the analysis.

It is also important to mention further incorporation of regularization techniques to contribute to more stable deep learning models such as L1 and L2 regularization.

Furthermore, to enhance the robustness and reliability of the predictive models, incorporating residual analysis can provide valuable insights into unexplained variations in predictions and can be integrated into model architectures, allowing learning from past prediction errors and enhancing predictive performance. There are already models that leverage residual components, such as residual neural networks, that are worth exploring (He et al., 2015).

Fractional differencing for time series modeling discussed in (Lopez de Prado, 2018), is also a promising avenue for time series modeling because it allows the series to retain more memory while still being stationary. Since differencing extracts mathematical memory from the original time series, instead of relying on integer differencing to make the series stationary, the differencing coefficient is fractional, allowing the series to retain its predictive power.

To finalize, it is also important to note the exploration of Time2Vec as a potential direction, a model-agnostic vector representation of time (Kazemi et al., 2019), an embedding technique used in time series that can lead to improved forecasting and predictions in various time-dependent applications such as price prediction.

References

Adcock, R., & Gradojevic, N. (2019). Non-fundamental, non-parametric Bitcoin forecasting. *Physica A: Statistical Mechanics and Its Applications*, *531*, 121727. https://doi.org/10.1016/j.physa.2019.121727

Ahmad, I., Wan, Z., & Ahmad, A. (2023). A big data analytics for DDOS attack detection using optimized ensemble framework in Internet of Things. *Internet of Things*, *23*, 100825. https://doi.org/10.1016/j.iot.2023.100825

Aparicio, J. T., Aparicio, M., & Costa, C. J. (2023). Design Science in Information Systems and Computing. In Proceedings of International Conference on Information Technology and Applications: ICITA 2022 (pp. 409-419). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-19-9331-2_35

Aparicio, J. T., Romao, M., & Costa, C. J. (2022). Predicting Bitcoin prices: The effect of interest rate, search on the internet, and energy prices. In 2022 17th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-5). IEEE. https://doi.org/10.23919/CISTI54924.2022.9820085

Bahdanau, D., Cho, K., & Bengio, Y. (2016). *Neural Machine Translation by Jointly Learning to Align and Translate* (arXiv:1409.0473). arXiv. https://doi.org/10.48550/arXiv.1409.0473

Bai, S., Kolter, J. Z., & Koltun, V. (2018). *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling* (arXiv:1803.01271). arXiv. https://doi.org/10.48550/arXiv.1803.01271

Biswas, K., Kumar, S., & Pandey, A. K. (2021). *Intensity Prediction of Tropical Cyclones using Long Short-Term Memory Network* (arXiv:2107.03187). arXiv. http://arxiv.org/abs/2107.03187

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches* (arXiv:1409.1259). arXiv. https://doi.org/10.48550/arXiv.1409.1259

Costa, C. J., & Aparicio, J. T. (2020). POST-DS: A methodology to boost data science. In 2020 15th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-6). IEEE. https://doi.org/10.23919/CISTI49556.2020.9140932

Costa, C. J., & Aparicio, J. T. (2021). A Methodology to Boost Data Science in the Context of COVID-19. In Advances in Parallel & Distributed Processing, and Applications: Proceedings from PDPTA'20, CSC'20, MSV'20, and GCC'20 (pp. 65-75). Springer International Publishing.https://doi.org/10.1007/978-3-030-69984-0_7

Dimitriadou, A., & Gregoriou, A. (2023). Predicting Bitcoin Prices Using Machine Learning. *Entropy*, *25*(5), 777. https://doi.org/10.3390/e25050777

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193–202. https://doi.org/10.1007/BF00344251

Granger, C. W. J. (1969). Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica*, *37*(3), 424–438. https://doi.org/10.2307/1912791

Granitto, P., Furlanello, C., Biasioli, F., & Gasperi, F. (2006). Recursive Feature Elimination with Random Forest for PTR-MS analysis of agroindustrial products. *Chemometrics and Intelligent Laboratory Systems - CHEMOMETR INTELL LAB SYST*, *83*. https://doi.org/10.1016/j.chemolab.2006.01.007

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5), 602–610. https://doi.org/10.1016/j.neunet.2005.06.042

Greaves, A., & Au, B. (2015). *Using the Bitcoin Transaction Graph to Predict the Price of Bitcoin*. https://www.semanticscholar.org/paper/Using-the-Bitcoin-Transaction-Graph-to-Predict-the-Greaves-Au/a0ce864663c100582805ffa88918910da89add47

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. https://doi.org/10.1109/TNNLS.2016.2582924

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, *46*(1), 389–422. https://doi.org/10.1023/A:1012487302797

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition* (arXiv:1512.03385; Version 1). arXiv. https://doi.org/10.48550/arXiv.1512.03385

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

Israel, R., Kelly, B. T., & Moskowitz, T. J. (2020). *Can Machines "Learn" Finance?* (SSRN Scholarly Paper 3624052). https://doi.org/10.2139/ssrn.3624052

Jiao, Y., Tu, M., Berisha, V., & Liss, J. (2016). *Accent Identification by Combining Deep Neural Networks and Recurrent Neural Networks Trained on Long and Short Term Features*. 2388–2392. https://doi.org/10.21437/Interspeech.2016-1148

Kazemi, S. M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., & Brubaker, M. (2019). *Time2Vec: Learning a Vector Representation of Time* (arXiv:1907.05321). arXiv. https://doi.org/10.48550/arXiv.1907.05321

Langer, A. (1997). Early stopping of trials. *The Lancet*, *350*(9081), 890–891. https://doi.org/10.1016/S0140-6736(05)62078-8

Li, Y., Harfiya, L. N., Purwandari, K., & Lin, Y.-D. (2020). Real-Time Cuffless Continuous Blood Pressure Estimation Using Deep Learning Model. *Sensors, 20*. https://doi.org/10.3390/s20195606

Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, *37*(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Livieris, I. E., Pintelas, E., Stavroyiannis, S., & Pintelas, P. (2020). Ensemble Deep Learning Models for Forecasting Cryptocurrency Time-Series. *Algorithms, 13*(5), Article 5. https://doi.org/10.3390/a13050121

Lopez de Prado, M. (2018). *Advances in Financial Machine Learning (Chapter 1)* (SSRN Scholarly Paper 3104847). https://papers.ssrn.com/abstract=3104847

McNally, S., Roche, J., & Caton, S. (2018). Predicting the Price of Bitcoin Using Machine Learning. *2018 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 339–343. https://doi.org/10.1109/PDP2018.2018.00060

Mudassir, M., Bennbaia, S., Unal, D., & Hammoudeh, M. (2020). Time-series forecasting of Bitcoin prices using high-dimensional features: A machine learning approach. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-020-05129-6

Mukhanov, S., Uskenbayeva, R., Cho, Y., Kabyl, D., Les, N., & Amangeldi, M. (2023). GESTURE RECOGNITION OF MACHINE LEARNING AND CONVOLUTIONAL NEURAL NETWORK METHODS FOR KAZAKH SIGN LANGUAGE. *Scientific Journal of Astana IT University*, 85–100. https://doi.org/10.37943/15LPCU4095

Nugroho, H. (2020). Fully Convolutional Variational Autoencoder For Feature Extraction Of Fire Detection System. *Jurnal Ilmu Komputer Dan Informasi*, *13*. https://doi.org/10.21609/jiki.v13i1.761

Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016, September 12). *WaveNet: A Generative Model for Raw Audio*. arXiv.Org. https://arxiv.org/abs/1609.03499v2

Ortu, M., Uras, N., Conversano, C., Bartolucci, S., & Destefanis, G. (2022). On technical trading and social media indicators for cryptocurrency price classification through deep learning. *Expert Systems with Applications*, *198*, 116804. https://doi.org/10.1016/j.eswa.2022.116804

Paulescu, M., Paulescu, E., & Badescu, V. (2021). Chapter 9—Nowcasting solar irradiance for effective solar power plants operation and smart grid management. In R. Deo, P. Samui, & S. S. Roy (Eds.), *Predictive Modelling for Energy Management and Power Systems Engineering* (pp. 249–270). Elsevier. https://doi.org/10.1016/B978-0-12-817772-3.00009-4

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. https://doi.org/10.1080/14786440109462720

Pérez-Enciso, & Zingaretti, L. (2019). A Guide for Using Deep Learning for Complex Trait Genomic Prediction. *Genes*, *10*, 553. https://doi.org/10.3390/genes10070553

Politis, A., Doka, K., & Koziris, N. (2021). Ether Price Prediction Using Advanced Deep Learning Models. *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 1–3. https://doi.org/10.1109/ICBC51069.2021.9461061

Rashidi, H., Tran, N., Betts, E., Howell, L., & Green, R. (2019). Artificial Intelligence and Machine Learning in Pathology: The Present Landscape of Supervised Methods. *Academic Pathology*, *6*, 237428951987308. https://doi.org/10.1177/2374289519873088

Ravenscroft, W., Goetze, S., & Hain, T. (2022). *Receptive Field Analysis of Temporal Convolutional Networks for Monaural Speech Dereverberation* (arXiv:2204.06439). arXiv. http://arxiv.org/abs/2204.06439

Shen, D., Urquhart, A., & Wang, P. (2019). Does twitter predict Bitcoin? *Economics Letters*, *174*, 118–122. https://doi.org/10.1016/j.econlet.2018.11.007

Shen, J., & Shafiq, M. O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. *Journal of Big Data*, *7*(1), 66. https://doi.org/10.1186/s40537-020-00333-6

Shin, S., Hoàng, T., Le, T.-H., & Lee, M.-Y. (2016). A New Robust Design Method Using Neural Network. *Journal of Nanoelectronics and Optoelectronics*, *11*, 68–78. https://doi.org/10.1166/jno.2016.1871

Singh, E., Kuzhagaliyeva, N., & Sarathy, S. M. (2022). Chapter 9—Using deep learning to diagnose preignition in turbocharged spark-ignited engines. In J. Badra, P. Pal, Y. Pei, & S. Som (Eds.), *Artificial Intelligence and Data Driven Optimization of Internal Combustion Engines* (pp. 213–237). Elsevier. https://doi.org/10.1016/B978-0-323-88457-0.00005-9

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958.

Tomás, D., Costa, C. J., Gaivão, J. P., & Carvalho, J. P. (2018). "Time series for incidences, orders and invoicing forecast" CAPSI 2018 Proceedings. 36.https://aisel.aisnet.org/capsi2018/36

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention Is All You Need* (arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103. https://doi.org/10.1145/1390156.1390294

Yang, M., Li, X., & Liu, Y. (2021). Sequence to Point Learning Based on an Attention Neural Network for Nonintrusive Load Decomposition. *Electronics*, *10*, 1657. https://doi.org/10.3390/electronics10141657

Yu, D., Xu, Q., Guo, H., Zhao, C., Lin, Y., & Li, D. (2020). An Efficient and Lightweight Convolutional Neural Network for Remote Sensing Image Scene Classification. *Sensors*, *20*(7), Article 7. https://doi.org/10.3390/s20071999

Zhang, J., Ye, L., & Lai, Y. (2023). Stock Price Prediction Using CNN-BiLSTM-Attention Model. *Mathematics*, *11*(9), Article 9. https://doi.org/10.3390/math11091985

Appendix

Variable Name	Description
Oil price	Crude oil daily prices – WTI, serves as one of the main oil benchmarks (dollars per barrel)
Treasury yields	Represent the interest rates paid on U.S. government debt.
Copper price	Market value of one tone of copper.
Iron price	Market value of one tone of iron.
CBOE	Refers to an ETF that tracks the performance of gold prices..
SP500	Stock market index, representing the performance of 500 of the largest companies on the US.
MedianCPI	Variation of the Consumer Price Index (CPI), serving as a proxy for inflation.
Nasdaq100	Stock market index that containing technology and non-financial companies.
UnemploymentUSA	The percentage of the labor force that is not currently employed in USA.
USA_GDP	Gross domestic product of the USA.
Debt_GDP_USA	Metric to compare USA's total outstanding of debt to its GDP.
FiveYrBreak	Estimation of market-based expectations of future inflation.
TenYrBreak	Estimation of market-based expectations of future inflation.
transaction_fees	Costs associated with processing and verifying transactions on the Bitcoin network.
avg_block_size	Average size of a block (contains a group of transactions) within the blockchain.
miners_revenue	Total amount of rewards that cryptocurrency miners receive for their work in processing and validating transactions.
confirmed_transactions	Refers to cryptocurrency transactions that have been successfully processed and added to a blockchain.

unique_addresses	Number of individual addresses within the blockchain network.
hash_rate	Refers to the measurement of computational power or processing capacity within a blockchain network.
Difficulty	Refers to a parameter that regulates the complexity of the mathematical problems miners must solve to validate transactions and create new blocks in the blockchain.
estimated_output_volume	Represents the estimated total value or quantity of BTC transferred in a set of transactions.
estimated_transaction_volume	Represents the estimated total value or quantity of BTC being transacted within a given time frame.
btc_trends	Represents a normalized google search data, respective to interest in BTC.
High	Represents highest trading price of BTC on a given day.
Low	Represents the lowest trading price of BTC on a given day.
Volume	Total amount of transaction of BTC on a given day.
Close_ETHUSDT	Last market price of Ethereum on a given day
Volume_ETHUSDT	Total amount of transactions of Ethereum on a given day
Close_BNBUSDT	Last market price of Binance Coin on a given day
Volume_BNBUSDT	Total amount of transactions of Binance Coin on a given day
Close_XRPUSDT	Last market price of Binance Coin on a given day.
Volume_XRPUSDT	Total amount of transactions of Ripple on a given day.
Close_btc	Last market price of BTC.
Debt_USA	Total national debt USA.

SavingsRate	Portion of a person's or household's disposable income that is saved.

Table 15: Collected features set.

```
model = keras.models.Sequential(
    [
        # 1D-Conv layer will slide filters across one-dimension (time
axis) of the input; kernel:filter
        keras.layers.Conv1D(
            filters=20,
            kernel_size=4,
            strides=1,
            padding="causal",
            activation="relu",
            batch_input_shape=[1,None, n_features],
        ),
        keras.layers.LSTM(100, return_sequences=True, stateful=True),
        keras.layers.LSTM(100, return_sequences=True, stateful=True),
        # dropout layer to prevent overfitting
        keras.layers.Dropout(0.2),
        keras.layers.Dense(1, activation="linear"),
    ]
)
```

Figure 38: Hybrid CNN-LSTM implementation

```
import tensorflow as tf
from tensorflow import keras

n_features = X_train.shape[-1]

# attention layer
class Attention(keras.layers.Layer):
    def __init__(self, units):
        super().__init__()
        self.W1 = keras.layers.Dense(units)
        self.W2 = keras.layers.Dense(units)
```

```
        self.V = keras.layers.Dense(1)

    def call(self, features):
        score = tf.nn.tanh(self.W1(features) + self.W2(features))
        attention_weights = tf.nn.softmax(self.V(score), axis=1)
        context_vector = attention_weights * features
        context_vector = tf.reduce_sum(context_vector, axis=1)
        return context_vector, attention_weights

# Input layer
input_layer = keras.layers.Input(shape=(None, n_features))

# CNN-BiLSTM-Attention model
x = keras.layers.Conv1D(
    filters=20,
    kernel_size=4,
    strides=1,
    padding="causal",
    activation="relu"
)(input_layer)

x = keras.layers.Bidirectional(keras.layers.LSTM(100,
return_sequences=True))(x)
# add dropout to control overfitting
x = keras.layers.Dropout(0.2)(x)
x = keras.layers.Bidirectional(keras.layers.LSTM(100,
return_sequences=True))(x)

context_vector, attention_weights = Attention(100)(x)
output = keras.layers.Dense(1, activation="linear")(x)

model = keras.models.Model(inputs=input_layer, outputs=output)

model.compile(loss="mse" , optimizer="adam", metrics=["mae", "mse"])
early_stopping = keras.callbacks.EarlyStopping(
    monitor="val_loss",
    patience=20,
    mode="min",
    restore_best_weights=True,
)
```

```
lr_schedule = keras.callbacks.ReduceLROnPlateau(
    monitor="val_loss",
    factor=0.5,
    patience=10,
    min_lr=0.0001,
    mode="min",
    verbose=1,
)
checkpoint = keras.callbacks.ModelCheckpoint(

f"artifacts/{features_set}_features_set/saved_deepLmodels/univariate/cnn-bilstm-attn_regressor.h5", save_best_only=True, monitor="val_loss", mode="min"
)
reset_states = ResetStatesCallback()
history = model.fit(
    train_set,
    epochs=200,
    validation_data=valid_set,
    verbose=1,
    shuffle=False,
    callbacks=[early_stopping, lr_schedule, checkpoint, reset_states]
)
```

Figure 39: CNN-BiLSTM-Attention implementation.