Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Master in Digital Innovation

Trabajo Fin de Máster
Master Thesis

Time Series Forecasting using Transformers with Sentiment Analysis on Financial Data

Author: Luca Petracca

Madrid, July 2024

This Master Thesis has been deposited in ETSI Informáticos de la Universidad Politécnica de Madrid.

Master Thesis

Master in Digital Innovation

Title: Time Series Forecasting using Transformers with Sentiment Analysis on Financial Data

July 2024

Author: Luca Petracca

Supervisor:
Aurora Peréz Peréz
Computer Science PhD
Universidad Politécnica de Madrid

E.T.S. de Ingenieros Informáticos
Dpto. Lenguajes y Sistemas Informáticos e Ingeniería de Software
Universidad Politécnica de Madrid

Resumen

El objetivo de esta tesis es explorar la aplicación de modelos Transformer en el dominio del pronóstico de series temporales financieras, enfocándose específicamente en los mercados de criptomonedas. El trabajo evalúa la efectividad de los Transformers para capturar patrones complejos en datos financieros con el fin de predecir los precios de las criptomonedas, que son altamente volátiles y difíciles de predecir. La idea principal es la siguiente: se comparan y evalúan los modelos Transformer frente a modelos tradicionales de pronóstico de series temporales como las redes de Long Short-Term Memory (LSTM) y los híbridos LSTM-CNN para ver las capacidades de los modelos de última generación en el manejo de datos de criptomonedas.

Una característica clave de esta tesis es la combinación de valores de análisis de sentimiento en los modelos de pronóstico. La tesis investiga el potencial de las métricas de análisis de sentimiento (los "likes" de las redes sociales y el volumen de tweets) para mejorar la precisión de la predicción de precios de criptomonedas. Se hipotetiza que esta integración podría sacar a la luz relaciones que no son captadas por los indicadores financieros tradicionales.

El marco metodológico de esta investigación comienza con la recolección de datos, el preprocesamiento y los procesos de ingeniería de características o feature engineering. La estrategia de preprocesamiento aborda problemas comunes en los datos de series temporales financieras, como los valores faltantes y la estacionariedad. Para tratar los valores faltantes y las características desbalanceadas, se emplean técnicas como la imputación de la mediana, el ajuste estacional mediante indicadores alcistas o bajistas y técnicas de medias móviles para refinar el conjunto de datos antes de la fase de definición del modelo.

Para el desarrollo del modelo, se comparan varios modelos de pronóstico basados en la arquitectura de LSTM, híbridos LSTM-CNN y Transformers. Se realizan dos experimentos paralelos: en el primero, sólo se utilizan datos financieros (precios de apertura, cierre, máximos, mínimos y capitalización de mercado) como características. En el segundo experimento, en cambio, se incorporan también características de análisis de sentimiento. Este enfoque permite una comparación directa de cómo cada tipo de datos contribuye a la precisión del pronóstico. Los modelos se validan inicialmente con datos de Bitcoin (BTC) antes de aplicarse a otras criptomonedas como Aave (AAVE), Cardano (ADA), Dogecoin (DOGE), Ethereum (ETH), Monero (XMR) y Ripple (XRP) para explorar la variabilidad en el rendimiento a través de diferentes series temporales y condiciones del mercado con criptomonedas altamente volátiles. También se aborda la implementación de aprendizaje por transferencia en este dominio, entrenando modelos en una criptomoneda (BTC) y probándolos en otra (ETH), para evaluar la generalización de los modelos aprendidos en diferentes pero relacionados instrumentos financieros.

Finalmente, se resumen los hallazgos más importantes de este trabajo: los modelos Transformer incorporando el análisis de sentimiento tienden a superar a los modelos construidos únicamente con datos financieros, lo que significa que modelos de este tipo parecen ser capaces de captar patrones entre datos de diferente naturaleza. Por otro lado, aumentar el conjunto de datos para incluir características de sentimiento no ha resultado particularmente útil para mejorar la eficacia de los modelos LSTM o LSTM-CNN. Estos hallazgos son importantes para el conocimiento académico, pero tienen también una aplicación práctica directa para desarrollar herramientas de pronóstico más sofisticadas en mercados financieros donde la especulación es un factor impulsor, como en el caso de las criptomonedas.

Abstract

The goal of this thesis is to explore the application of Transformer models in the domain of financial time series forecasting, specifically focusing on cryptocurrency markets. The work assesses the effectiveness of Transformers in capturing complex patterns in financial data in order to predict the highly volatile and unpredictable cryptocurrency prices. The main idea is the following: Transformer models are compared and evaluated against traditional time series forecasting models like Long Short-Term Memory (LSTM) networks and LSTM-Convolutional Neural Network (CNN) hybrids to see the capabilities of current state-of-the-art models in handling cryptocurrency data.

A key feature of this thesis is the combination of sentiment analysis values into the forecasting models. The thesis investigates the potential of sentiment analysis metrics—the likes of social media contributions and tweet volumes—to enhance the predictive accuracy of cryptocurrency price predictions. This integration is hypothesized to potentially uncover latent signals that are not captured by traditional financial indicators alone.

The methodological framework of this research starts with the data collection, pre-processing, and feature engineering processes. The pre-processing strategy addresses common issues in financial time series data, such as missing values and non-stationarity. To account for missing values and unbalanced features, median imputation, seasonal adjustment via bullish/bearish indicators, and moving averages techniques are employed to refine the dataset prior to the model definition phase.

For model development, several forecasting models based on the architecture of LSTMs, LSTM-CNN hybrids and Transformers are compared. Two parallel experiments are conducted: in the first, only financial data (including features like open, close, high, low prices, and market capitalization) is used as features. In the second experiment, instead, sentiment analysis features are used. This approach allows for a direct comparison of how each type of data contributes to forecasting accuracy. The models are initially validated on Bitcoin (BTC) data before being applied to other cryptocurrencies such as Aave (AAVE), Cardano (ADA), Dogecoin (DOGE), Ethereum (ETH), Monero (XMR), and Ripple (XRP) to explore variability in performance across different time series data and market conditions with highly volatile cryptocurrencies. The implementation of transfer learning within this domain is also addressed by raining models on one cryptocurrency (BTC) and testing them on another (ETH), to evaluate the generalizability of the learned models across different but related financial instruments.

The evaluation of model performance is discussed in the results section in detail, comparing the performance of the Transformers both with and without sentiment data against the baseline models established by the LSTM and LSTM-CNN configurations plus a simple model predicting for tomorrow the value of today.

Finally, the most important findings from this work are summarized: Transformer models with sentiment analysis tend to outperform the models constructed solely based on the financial data features, which means that models of this type seem to be capable of grasping patterns between data of a different nature. On the other hand, augmentation with sentiment data is not particularly helpful in enlarging the originative efficacy for LSTM or LSTM-CNN models. These insights are of importance to academic knowledge but have a direct practical implementation to further develop more sophisticated forecasting tools in financial markets where speculation is a driving factor, as in the case of cryptocurrencies.

Table of Contents

Table of Figures ..... **v**
1 Introduction **1**
1.1 Financial Markets and Time Series Forecasting 1
1.2 Cryptocurrencies as Financial Instruments 2
2 State of the Art **3**
2.1 Evolution of Time Series Forecasting Models 3
2.1.1 Autoregressive Integrated Moving Average (ARIMA) 3
2.1.2 Convolutional Neural Networks (CNN) 4
2.1.3 Long Short-Term Memory (LSTM) 6
2.1.4 Transformer Model 8
2.2 Transformers in Time Series Forecasting 10
2.3 Sentiment Analysis in Financial Forecasting 11
3 Methodology **12**
3.1 Domain 12
3.1.1 Open Research Queries and Hypotheses 12
3.1.2 Dataset Overview 14
3.2 Data Analysis and Preprocessing 15
3.2.1 Data Cleaning 15
3.2.2 Data Splitting 16
3.2.3 Data Visualization 16
3.3 Stationarity Analysis 19
3.3.1 Visual Stationarity Analysis 19
3.3.2 Statistical Stationarity Analysis 20
3.4 Feature Engineering 21
3.4.1 Identifying Seasonality Patterns 21
3.4.2 Encoding Seasonality 22
3.4.3 Lag Features 23
3.4.4 Fourier Transformation 23
3.4.5 Autocorrelation 25
3.4.6 Partial Autocorrelation 26
3.4.7 Moving Average 27
3.4.8 SMA and EWMA Application 28
4 Model Training and Evaluation **31**
4.1 Long Short-Term Memory (LSTM) 31
4.1.1 LSTM Model Architecture 31
4.1.2 Training and Evaluation 32
4.2 LSTM-CNN Hybrid Model 33
4.2.1 LSTM-CNN Hybrid Model Architecture 33

4.2.2 Training and Evaluation 34
4.3 Transformers 34
4.3.1 Transformer Model Architecture 34
4.3.2 Training and Evaluation 36
5 Results and Findings 37
5.1 First Analysis only using Financial Data 37
5.1.1 LSTM 37
5.1.2 LSTM-CNN Hybrid 39
5.1.3 Transformer Model 42
5.2 Second Analysis using Sentiment Analysis Data 45
5.2.1 LSTM + Sentiment 45
5.2.2 LSTM-CNN + Sentiment Hybrid 47
5.2.3 Transformer + Sentiment Model 50
5.3 Third Analysis: Transfer Learning on BTC-ETH Data 53
5.4 Comparison of Final Results 54
6 Conclusions 55
6.1 Findings 55
6.2 Results Framework 56
6.3 Final Thoughts 57
7 Bibliography 58

Table of Figures

Figure 1: An example of CNN architecture for financial time series forecasting....5
Figure 2: Long Short-Term Memory network....6
Figure 3: The Transformer Model Architecture....8
Figure 4: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention....9
Figure 5: Close Price and Social Volume of BTC....17
Figure 6: Tweets and Social Score of BTC....17
Figure 7: Social Score and Close Price of BTC....18
Figure 8: Close Price per Cryptocurrency (normalized)....18
Figure 9: Rolling Statistics: mean and std_dev of BTC....20
Figure 10: Monthly Mean Close Price of BTC....22
Figure 11: Fourier Transform in the Frequency Domain (BTC)....24
Figure 12: Significant Frequencies in the Fourier Transform (BTC)....24
Figure 13: Close Price and Fourier Transforms of BTC....25
Figure 14: Autocorrelation Long-Short Term of BTC....26
Figure 15: Partial Autocorrelation Long-Short Term of BTC....27
Figure 16: Different windows for MA....29
Figure 17: Different windows for EMA....30
Figure 18: Training and Validation Loss per Epoch....37
Figure 19: Comparison of Actual and Predicted Values on Test Set....38
Figure 20: Training and Validation Loss per Epoch....40
Figure 21: Comparison of Actual and Predicted Values on Test Set....40
Figure 22: Training and Validation Loss per Epoch....42
Figure 23: Comparison of Actual and Predicted Values on Test Set....43
Figure 24: Training and Validation Loss per Epoch (LSTM_Sentiment)....45
Figure 25: Comparison of Actual and Predicted Values on Test Set for LSTM-Sentiment....46
Figure 26: Training and Validation Loss per Epoch....48
Figure 27: Comparison of Actual and Predicted Values on Test Set for LSTM-CNN-Sentiment....48
Figure 28: Training and Validation Loss per Epoch....50
Figure 29: Comparison of Actual and Predicted Values on Test Set for Transformer-Sentiment....51
Figure 30: Comparison of Actual and Predicted Values on Test Set with Transfer Learning....54

1 Introduction

1.1 Financial Markets and Time Series Forecasting

The thesis explores the potential use and impact of Transformer models on financial time series analysis. In recent years, deep learning has contributed significantly to the advancements in machine learning, with Transformer models standing out due to their exceptional ability to handle sequential data. Originally introduced by Vaswani et al. [1], Transformers have revolutionized natural language processing (NLP) tasks, demonstrating high performance in machine translation, among other applications. This thesis extends the application of Transformer models to the domain of financial time series forecasting, an area of study that includes the prediction of market movements, asset prices, and economic indicators.

Financial markets and time series forecasting are inherently complex due to the dynamic and nonlinear nature of financial data and markets. High volatility, noise, and the influence of external factors pose significant challenges for traditional models in effectively handling financial time series data. Accurate financial time series forecasting can be useful for various stakeholders, including investors, policy makers, and financial analysts, as it impacts in decision-making processes. Traditional statistical methods have been extensively used in the past; however, the advent of deep learning has revolutionized the approach towards forecasting in financial markets using machine learning techniques.

The Transformer's architecture, characterized by self-attention mechanisms, enables the model to weigh the importance of different parts of the input data without the constraints of sequential data processing [1] [2]. This is particularly beneficial for financial time series analysis, where long-term dependencies and sudden market shifts can significantly impact predictions. The relevance of Transformer models in this context is supported by their ability to outperform existing models in tasks requiring the understanding of sequential data. For instance, the Transformer model achieved state-of-the-art results in the WMT 2014 English-to-German and English-to-French translation tasks, surpassing previous models by a significant margin [1]. This shows how well the model is able to capture complex patterns and dependencies, a quality that is directly applicable to financial time series analysis.

Moreover, the introduction of multi-head attention mechanisms within Transformer models allows for the parallel processing of data, enabling the model to capture information from different representation subspaces at different positions [1] [3]. This feature is particularly advantageous for financial time series analysis, where multiple factors and temporal scales must be considered simultaneously. The ability to process and integrate diverse information streams in parallel significantly enhances the model's predictive accuracy and efficiency, unlike traditional recurrent neural networks (RNNs) that process data sequentially. This parallel processing capability significantly reduces training times, which is beneficial given the vast amounts of financial data. Moreover, the incorporation of positional encoding in Transformers introduces the notion of time, which is absent in the self-attention mechanism, thereby ensuring that the model is aware of the order of transactions or events within the financial time series.

Another aspect of applying Transformers for financial time series forecasting is how overfitting is addressed by incorporating techniques like dropout, which helps the models generalize better to new data. The Transformer architecture also allows for customizations such as model size adjustments and dropout settings to optimize performance for specific forecasting tasks [1].

1.2 Cryptocurrencies as Financial Instruments

Cryptocurrencies have gained significant attention for their potential to revolutionize the financial sector, promising a decentralized and transparent alternative to traditional fiat currencies and financial systems. The strong appeal of cryptocurrencies lies in their ability to facilitate secure and efficient transactions without the need for intermediaries such as banks, leveraging blockchain technology. This section explores the role of cryptocurrencies as financial instruments, focusing on their implications for financial markets, investment strategies, and the broader economic landscape.

The launch of Bitcoin in 2009 marked the beginning of the cryptocurrency era, introducing a peer-to-peer electronic cash system that operates independently of central authority [4]. Since then, the cryptocurrency market has expanded to include thousands of different digital currencies, each with unique features and underlying technologies. The decentralized nature of cryptocurrencies is achieved through blockchain technology, a distributed ledger that records all transactions across a network of computers. This technology ensures the integrity and security of transaction data, making cryptocurrencies resistant to fraud and censorship [5].

Cryptocurrencies offer several advantages as financial instruments. Transactions recorded on the blockchain are accessible to all participants, ensuring transparency and trust in the system [6]. Furthermore, they offer reduced transaction costs and increased efficiency. With no intermediaries, transactions can be processed more quickly at a lower cost compared to traditional financial systems. Additionally, cryptocurrencies facilitate global transactions, enabling seamless cross-border payments without the need for currency conversion or reliance on the banking infrastructure. However, the use of cryptocurrencies as financial instruments also presents challenges. The high volatility of cryptocurrency prices poses risks for investors and can lead to significant financial losses.

Moreover, the regulatory environment for cryptocurrencies is still evolving, with varying approaches and levels of acceptance across different jurisdictions. This regulatory uncertainty can impact the adoption and integration of cryptocurrencies into the traditional financial system. The application of blockchain technology and cryptocurrencies extends beyond mere transactions to include smart contracts, decentralized finance (DeFi), and tokenization of assets. Smart contracts, self-executing contracts with the terms of the agreement directly written into code, enable automated and conditional transactions without the need for intermediaries. DeFi platforms leverage blockchain technology to offer financial services such as lending, borrowing, and trading without traditional financial institutions [7]. Tokenization allows for the digital representation of real-world assets on the blockchain, facilitating their fractional ownership and trading.

In the context of financial time series analysis, cryptocurrencies present both opportunities and challenges. The predictive modeling of cryptocurrency prices requires sophisticated techniques capable of capturing their complex dynamics, which are not only finance-related, but are also dependent on the *sentiment* of the market. In this context, Transformers models have shown promising results in handling sequential data with long-term dependencies.

The adoption of cryptocurrencies as financial instruments is a testament to the ongoing evolution of the financial sector. Despite the challenges, the potential of cryptocurrencies to enhance the efficiency, transparency, and accessibility of financial services cannot be understated. As the technology matures and regulatory frameworks evolve, cryptocurrencies are likely to play an increasingly significant role in the global financial landscape.

2 State of the Art

2.1 Evolution of Time Series Forecasting Models

Time series forecasting models have recently evolved with the introduction of deep learning methods. Initially, traditional statistical models like ARIMA (Autoregressive Integrated Moving Average) dominated the field offering a robust methodology to model and predict linear time series data.

However, the financial markets' complexity and variability demanded more sophisticated approaches to capture non-linear patterns and relationships within the data. The rise of neural networks has brought new state-of-the-art techniques for time series forecasting, taking advantage of their ability to capture complex, non-linear relationships from data.

This section introduces several classic deep learning models for stock market prediction, including CNN (Convolutional Neural Networks) and LSTM (Long Short-Term Memory), which are used in the following model comparisons.

2.1.1 Autoregressive Integrated Moving Average (ARIMA)

ARIMA (Autoregressive Integrated Moving Average) models are a pillar of time series forecasting due to their robust methodology for capturing linear patterns in data [8]. The ARIMA model is characterized by three key components: autoregression (AR), differencing (I), and moving average (MA). The AR part of the model involves regressing the time series on its own lagged values, essentially leveraging past observations to predict future ones. The formula for the AR component is given by:

$$X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t$$

where X_t is the time series at time t, ϕ_i are the coefficients, p is the number of lagged terms, and ϵ_t is the error term. Differencing, represented by the "I" in ARIMA, involves subtracting the previous observation from the current observation to make the time series stationary, effectively removing trends and seasonality. The MA component models the relationship between an observation and a residual error from a moving average model applied to lagged observations. The formula for the MA part is:

$$X_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

where μ is the mean of the series, θ_i are the coefficients, and q is the number of lagged forecast errors.

Building on ARIMA, the SARIMA (Seasonal ARIMA) model incorporates seasonal components to handle time series data with regular seasonal patterns. SARIMA extends ARIMA by adding seasonal autoregressive (SAR), seasonal differencing (SI), and seasonal moving average (SMA) components. These components follow the same principles as their non-seasonal counterparts but are applied to the data at seasonal lags. The SARIMA model can be represented as ARIMA$(p, d, q)(P, D, Q)m$,

where p, d, q are the non-seasonal orders, P, D, Q are the seasonal orders, and m is the number of observations per season.

The inclusion of these seasonal terms allows SARIMA to capture periodic fluctuations more accurately. For instance, if a dataset exhibits a strong seasonal pattern every 12 months, the seasonal components of SARIMA would be configured to account for this periodicity. By combining non-seasonal and seasonal components, SARIMA models provide a comprehensive framework for analyzing time series data that exhibit both short-term and seasonal variations. This makes them particularly valuable in fields such as finance and economics, where such patterns are prevalent. By modeling both the trend and seasonality, SARIMA offers a robust tool for forecasting complex time series data.

2.1.2 Convolutional Neural Networks (CNN)

The introduction of Convolutional Neural Networks (CNNs) to time series forecasting further expanded the toolbox available to researchers and practitioners. CNNs, primarily known for their success in image processing, have been adapted to handle time series data. By applying convolutional layers to time series, CNNs can identify local patterns or features within the data, offering a complementary approach to the sequence modeling capabilities of RNNs and LSTMs [9].

CNN Architecture Overview

The concept of Convolutional Neural Networks was first introduced in the seminal paper "Gradient-Based Learning Applied to Document Recognition" by Yann LeCun et al. in 1998 [10]. This paper laid the foundation for CNNs and their application to image recognition tasks.

A CNN consists of several key layers: convolutional layers, pooling layers, and fully connected layers. Each plays a critical role in the model's ability to learn and generalize from the data.

Convolutional Layers: These layers are the core of CNNs. They apply a series of filters (or kernels) to the input data. Each filter convolves across the input time series, performing element-wise multiplications and summing the results. This process extracts local features from the input data. The depth of the convolutional layer, defined by the number of filters, determines the number of features learned. For example, in financial time series forecasting, filters can capture different patterns such as short-term price movements or sudden spikes and drops. Mathematically, if $x(t)$ is the input time series and w is the filter, the convolution operation is defined as:

$$(x * w)(t) = \sum_{i=0}^{k-1} x(t+i) \cdot w(i)$$

where k is the filter size.

Activation Function: After convolution, an activation function such as ReLU (Rectified Linear Unit) is applied. ReLU introduces non-linearity into the model, allowing it to learn more complex patterns. The ReLU function is defined as:

$$\text{ReLU}(x) = \max(0, x)$$

Pooling Layers: Following the convolutional layers, pooling layers are used to reduce the spatial dimension of the data, which helps in decreasing the computational complexity and preventing overfitting. The most common type is max pooling, which takes the maximum value from a segment of the input. For example, in a max pooling layer with a pool size of 2, the output is:

$$y(t) = \max\big(x(2t), x(2t+1)\big)$$

Fully Connected Layers: After several convolutional and pooling layers, the resulting feature maps are flattened into a one-dimensional vector. This vector is fed into fully connected (dense) layers, where each neuron is connected to every neuron in the previous layer. These layers combine the features extracted by the convolutional layers to make the final prediction. The fully connected layers can be mathematically represented as:

$$y = \sigma(Wx + b)$$

where W is the weight matrix, x is the input vector, b is the bias vector, and σ is an activation function like ReLU or sigmoid.

Application to Financial Time Series Forecasting

In the domain of financial time series forecasting, CNNs have demonstrated significant potential [11]. Financial time series data, such as stock prices or cryptocurrency values, exhibit complex patterns with both short-term fluctuations and long-term trends. The convolutional layers in CNNs are particularly adept at capturing these intricate patterns.

Through the one-dimensional convolutional operator, the original sequence is transformed into a new sequence with extracted features. Then the sequence may be subsampled through the pooling layer and output the result through a fully connected layer. The result should be a real value as the predicted price at time T+1. Fig. 1 presents an example of the CNN architecture for stock market prediction,

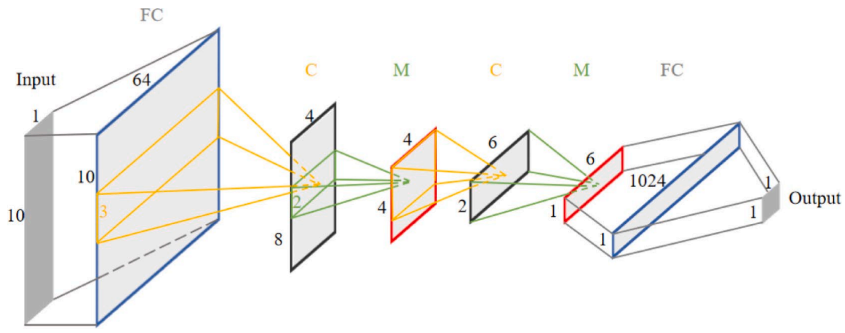

Figure 1: An example of CNN architecture for financial time series forecasting

where FC represents the fully connected layer, C represents the one-dimensional convolutional layer (with kernel size = 3 and stride = 1), and M represents the max-pooling layer (with kernel size = 2 and stride = 2)

By applying a set of filters, a CNN can learn to identify important local features like sudden spikes or drops in prices, periodic patterns, or trend reversals. These features are then combined through pooling and fully connected layers to provide robust predictions.

One of the strengths of CNNs in time series forecasting is their ability to handle the hierarchical structure of time series data. Initially, lower layers may capture basic patterns such as local trends and seasonality. As the data propagates through deeper layers, the model can learn more abstract features and longer-term dependencies, improving the overall forecasting performance.

Moreover, CNNs can be combined with other neural network architectures to enhance forecasting performance. A common approach is to use CNNs to extract features from the time series data, which are then fed into an LSTM or another type of recurrent network. This hybrid model leverages the strength of CNNs in capturing local patterns and the capability of LSTMs in modeling long-term dependencies, resulting in improved accuracy for financial forecasts.

However, these models still present some crucial limitations. For example, the pooling layers in CNN cause the loss of valuable information by ignoring the part-whole relationships [12].

2.1.3 Long Short-Term Memory (LSTM)

LSTM Architecture

Long Short-Term Memory (LSTM) networks are a sophisticated form of recurrent neural networks (RNNs) specifically designed to address the issues of vanishing and exploding gradients, which are common in traditional RNNs. This characteristic makes LSTMs particularly effective for tasks requiring the capture of long-range dependencies within sequential data, such as financial time series forecasting [13]. Fig. 2 presents the details of the LSTM architecture.

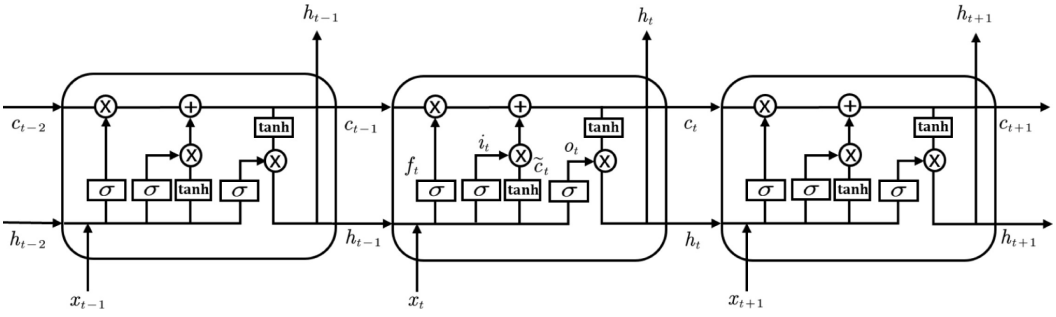

Figure 2: Long Short-Term Memory network

Components of LSTM

LSTM networks comprise a series of interconnected modules, each containing four main interacting layers known as gates: the forget gate, the input gate, the cell state, and the output gate. These gates control the flow of information and ensure that the model retains relevant data while discarding unnecessary information.

- Forget Gate: The forget gate decides which part of the long-term memory should be retained or discarded. This gate uses a sigmoid function to output values between 0 and 1, where values closer to 0 mean more information will be forgotten, and values closer to 1 mean more information will be retained.
- Input Gate: The input gate determines which new information should be added to the cell state. It combines a sigmoid function to decide which values to update and a tanh function to create a vector of potential values to add to the state.

- Cell State: The cell state is a key component of LSTMs that carries long-term information across the entire network. It can be modified indirectly via the forget and input gates but remains largely unaffected by direct weights, thus preserving the gradient and preventing issues of vanishing or exploding gradients.
- Output Gate: The output gate controls the final output based on the cell state and determines what the next hidden state should be. This hidden state is then passed to the next time step, allowing the LSTM to make predictions based on both long-term and short-term dependencies.

How LSTM models work

The operations of an LSTM can be divided into several steps, involving the interaction of the cell state and hidden state through the gates:

- Forget Gate Operation: The previous hidden state and current input are combined and passed through a sigmoid function, which determines the fraction of the cell state to retain or forget.
- Input Gate Operation: A combination of the previous hidden state and current input, after passing through a sigmoid function and a $tanh$ function, generates new candidate values. The sigmoid function decides which parts of these candidate values will be added to the cell state.
- Cell State Update: The cell state is updated by combining the retained portion of the old cell state with the new candidate values weighted by the input gate's output.
- Output Gate Operation: The new cell state, after passing through a $tanh$ function, is multiplied by the output of a sigmoid function applied to the combination of the previous hidden state and current input, generating the new hidden state.

Limitations of LSTM

However, these models face several significant limitations due to the sequential processing of input data and the challenges associated with back-propagation through time (BPTT), particularly when processing datasets with long dependencies [2]. The training process of Long Short-Term Memory (LSTM) models is particularly susceptible to vanishing and exploding gradient problems [14]. When dealing with long sequences, the gradient descent algorithm (utilizing BPTT) may fail to update the model parameters effectively, as the gradient information can either diminish to zero or grow to infinity. Moreover, these models generally do not fully benefit from the parallel computation capabilities provided by graphical processing units (GPUs), tensor processing units (TPUs), and other hardware accelerators [15].

Although certain architectural modifications and training techniques can help LSTM models mitigate gradient-related issues to some extent, these solutions are often not sufficient. The inherent challenges in learning from long data sequences, combined with the limited capacity for parallelization afforded by modern hardware, significantly affect the performance and efficiency of recurrent neural network (RNN)-based models [16]. Additionally, the sequential nature of these models means that they process one data point at a time, which further limits their speed and scalability in handling large datasets.

This limitation becomes a bottleneck in applications requiring real-time processing or those involving extensive computational resources. As a result, while LSTMs and other RNN-based models have been revolutionary in many areas, their practical application is still affected by relevant constraints.

2.1.4 Transformer Model

The *transformative* moment in the evolution of time series forecasting models came with the development of the Transformer model. The Transformer model revolutionized natural language processing (NLP) and subsequently found applications in various domains, including time series forecasting [11]. While LSTMs, with their gating mechanisms, allow the model to pay attention to or forget different parts of the input data, the Transformer model introduced a more explicit and flexible approach. At the core of the Transformer architecture is the self-attention mechanism, which allows the model to weigh the importance of different parts of the input data differently, enabling it to capture complex, long-range dependencies in the data without being constrained by the sequential processing of RNNs and LSTMs.

The Transformer model's ability to process input data in parallel significantly reduces training times compared to RNNs and LSTMs, making it highly efficient for large datasets, a common characteristic of financial time series data. Furthermore, the introduction of multi-head attention within the Transformer architecture allows the model to attend to information from different representation subspaces simultaneously, providing a more nuanced understanding of the data.

This section delves into the architecture of Transformer models and their application in the financial domain.

Transformer Architecture

Transformer models consist of an encoder-decoder structure, where both parts are built from layers of self-attention mechanisms and feed-forward neural networks. This architecture enables the model to process entire sequences of data simultaneously, rather than sequentially, as in RNNs. The Transformer follows this overall architecture using stacked self-attention and pointwise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Fig.3, respectively [1].

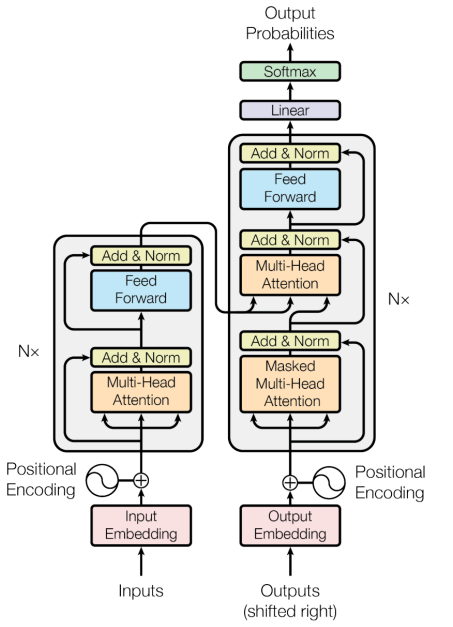

Figure 3: The Transformer Model Architecture

Self-Attention Mechanism: At the heart of the Transformer is the self-attention mechanism (scaled dot-product attention), which allows the model to weigh the importance of different elements in the input sequence. Given an input sequence $X = [x_1, x_2, \ldots, x_n]$, the self-attention mechanism computes a weighted sum of the values V, where the weights are determined by the query Q and key K matrices. The attention score is calculated as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Here, d_k is the dimensionality of the key vectors, and the softmax function ensures that the weights sum to one [1] [2].

Multi-Head Attention: To capture different aspects of the input data, Transformers employ multi-head attention, which performs multiple self-attention operations in parallel. Each attention head has its own set of query, key, and value matrices, enabling the model to focus on different parts of the sequence simultaneously. The outputs of these heads are then concatenated and linearly transformed:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O$$

where h is the number of heads and W_O is the output projection matrix [1] [2].

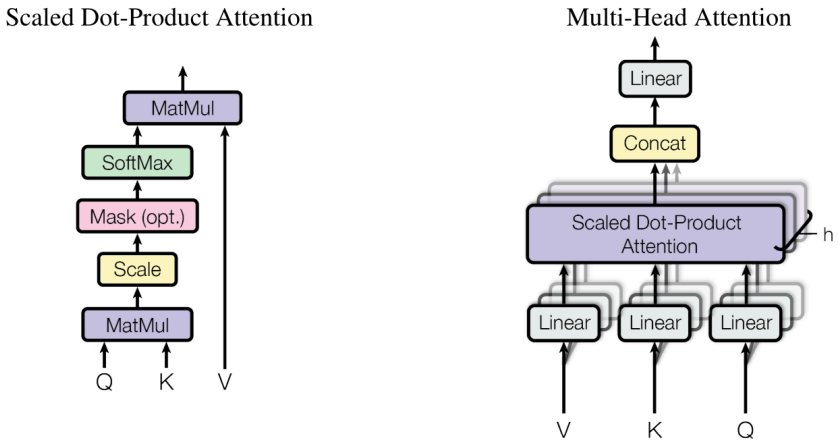

Figure 4: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention

Positional Encoding: Since Transformers do not inherently capture the order of the input sequence, positional encodings are added to the input embeddings to provide this information [1]. These encodings use sine and cosine functions of different frequencies:

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dmodel}}\right)$$

$$\text{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dmodel}}\right)$$

where pos is the position and i is the dimension index.

Feed-Forward Neural Networks: Each attention layer is followed by a fully connected feed-forward network, applied identically to each position in the sequence [1]. This consists of two linear transformations with a ReLU activation in between:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$$

Residual Connections and Layer Normalization: To facilitate gradient flow and improve convergence, the Transformer employs residual connections around each sub-layer, followed by layer normalization:

$$\mathrm{LayerNorm}\big(x + \mathrm{Sublayer}(x)\big)$$

2.2 Transformers in Time Series Forecasting

As discussed, the architecture of Transformers is designed to handle sequential data, making it suitable for time series forecasting. The model comprises an encoder and decoder, each consisting of multiple layers of multi-head self-attention and position-wise fully connected feed-forward networks. The multihead self-attention mechanism enables the model to capture various aspects of the data by projecting the queries, keys, and values multiple times with different, learned linear projections [1] [2]. This allows the model to focus on different positions of the input sequence, enhancing its ability to learn from complex data patterns.

Traditional models like RNNs and LSTMs struggle with long-term dependencies due to issues like vanishing or exploding gradients, which are dealt with the self-attention mechanism in Transformers by directly computing the output of each position in the sequence, allowing for more extended dependencies to be captured without the need for recurrent connections. This is particularly advantageous for financial markets, where the significance of past events can extend over varying time periods.

Another key aspect is how volatility is managed with transformers compared to traditional models. Volatility refers to the rate at which the price of a security increases or decreases for a given set of returns. Handling volatility in time series data is crucial as it affects the predictability and stability of models. Traditional methods like GARCH and ARIMA models have been prevalent, but they struggle with the complex patterns and non-linear relationships present in financial markets [17].

Deep learning models, particularly those using architectures like Long Short-Term Memory (LSTM) networks, have shown great promise in capturing temporal dependencies and nonlinear patterns in volatile data streams [17]. The adaptability of these models to learn from large volumes of data allows them to outperform conventional statistical models, especially in handling the complexities of financial time series.

Transformer models have been demonstrated to be every effective in handling volatility in financial time series by enhancing prediction accuracy and handling long-range dependencies in volatile markets through a linear Transformer structure which reduces complexity and increases efficiency in training and inference times [17] [18].

Transformers enhance financial time series forecasting with their ability to parallelize processing through the self-attention mechanism, efficiently handling large datasets. Their adaptability, proven in various applications, allows for effective tuning to specific financial tasks like price prediction and sentiment analysis. Integrating domain-specific adjustments and employing advanced training strategies, such as hyperparameter tuning, further boosts their forecasting accuracy.

2.3 Sentiment Analysis in Financial Forecasting

Financial markets are influenced by several factors, including economic indicators, political events, and public sentiment. Traditional financial forecasting models often struggle to incorporate these qualitative factors effectively. However, with the advancements in research on deep learning models, especially Transformers, new ways of incorporating sentiment analysis into financial forecasting are being investigated. By analyzing textual data from news articles, financial reports, and social media, sentiment analysis models can grasp the market sentiment, which is a powerful predictor of market movements, especially in volatile markets.

The self-attention mechanism of Transformers allows the model to weigh the importance of different words in a sentence, enabling it to capture the context and sentiment more accurately than conventional models. This feature is particularly beneficial for financial sentiment analysis, where the context in which a word is used can significantly alter its sentiment. For instance, the word "loss" in a financial report may have a negative sentiment, but in the context of "loss reduction," the sentiment is positive. Transformers can discern these differences, thereby enhancing the accuracy of sentiment analysis.

Moreover, the scalability of Transformer models, facilitated by stacking multiple encoder and decoder layers, allows for processing large datasets of financial news and social media posts. This is especially relevant in the field of cryptocurrency financial forecasting, where the volume of textual data is vast and continuously growing. The ability to process and analyze large datasets ensures that the models are trained on comprehensive data, encompassing a wide range of sentiments and opinions, which enhances the model's predictive performance.

The optimization techniques used in training Transformer models, such as the Adam optimizer and learning rate adjustments play a significant role in improving the model's convergence and overall performance [1] [2] [13]. These techniques ensure that the model efficiently learns from the training data, including the sentiment-laden textual data, and generalizes well to unseen data.

Furthermore, the application of pre-trained models and transfer learning has significantly reduced the barriers to implementing sentiment analysis in financial forecasting. Libraries compatible with deep learning frameworks, such as PyTorch and TensorFlow, offer a vast collection of pre-trained Transformer models that can be fine-tuned for specific financial sentiment analysis tasks [2]. This approach leverages the vast amount of knowledge these models have learned from diverse datasets, enhancing their predictive accuracy.

These models can provide a better understanding of these highly volatile financial markets by effectively analyzing the sentiment of financial news and social media. This can also lead to more accurate predictions of prices and trends.

3 Methodology

3.1 Domain

The domain of this project is centered on the application of advanced machine learning techniques to the field of financial time series forecasting, with a specific focus on cryptocurrencies.

Cryptocurrencies represent a rapidly evolving and highly volatile segment of the financial markets, characterized by significant price fluctuations and trading volumes. The time series data utilized in this study comprises several prominent cryptocurrencies, including Bitcoin (BTC), Ethereum (ETH), Cardano (ADA), Dogecoin (DOGE), Monero (XMR), Ripple (XRP), and Aave (AAVE).

Each of these digital assets exhibits unique trading behaviors and market dynamics, offering a diverse dataset for evaluating the forecasting models. The data includes traditional financial metrics such as 'open', 'close', 'high', 'low' prices, and 'market_cap', providing a comprehensive view of market trends and performance.

3.1.1 Open Research Queries and Hypotheses

This chapter outlines the open research queries and hypotheses formulated in this thesis. The primary goal is to explore the application of Transformer models within the volatile domain of cryptocurrency markets and assess their ability to be implemented for financial time series forecasting. This exploration is grounded in comparing the performance of Transformer models with traditional models like LSTMs and LSTM-CNN hybrids, particularly focusing on the integration of sentiment analysis into these models.

Research Queries

- Effectiveness of Transformer Models in Cryptocurrency Forecasting:
  1. How effective are Transformer models compared to traditional time series forecasting models such as LSTM and LSTM-CNN hybrids in predicting cryptocurrency prices?
- Handling of Volatility and Non-Stationarity:
  1. Can Transformer models handle the inherent volatility and non-stationarity in cryptocurrency markets compared to traditional models?
  2. Are specific pre-processing strategies like median imputation, seasonal adjustment, and moving averages more effective in preparing data for Transformers than for traditional models?
- Impact of Sentiment Analysis on Forecasting Accuracy:
  1. Do sentiment analysis features, such as social media contributions and tweet volumes, provide significant improvements on the predictive accuracy of cryptocurrency prices?
- Generalizability Across Different Cryptocurrencies (Transfer Learning):
  1. How well do models trained on data from one cryptocurrency (e.g., Bitcoin) generalize to other cryptocurrencies (e.g., Ethereum)?
  2. What role does transfer learning play in enhancing model performance across different but related financial instruments?

Hypotheses

Based on the research queries, the following hypotheses are defined:

- Hypothesis 1: Transformer models will outperform traditional time series forecasting models in predicting cryptocurrency prices due to their ability to efficiently process sequences with self-attention mechanisms, allowing them to manage volatility and non-stationarity more effectively than traditional models and thus leading to improved forecasting accuracy.
- Hypothesis 2: The integration of sentiment analysis into forecasting models will enhance predictive accuracy, uncovering latent signals that are not captured by traditional financial indicators alone, thereby providing a more comprehensive understanding of market dynamics.
- Hypothesis 3: Transfer learning will demonstrate generalizability across different cryptocurrencies, leveraging learned patterns from one market to predict another with reduced model training requirements and similar performance.

The following framework summarizes the relationship between the open research queries and the correspondent hypotheses and will be used to assess the results of the thesis work.

<table>
<tr><th>Research Query</th><th>Hypothesis</th></tr>
<tr><td>Q1: How effective are Transformer models compared to traditional time series forecasting models such as LSTM and LSTM-CNN hybrids in predicting cryptocurrency prices?</td><td rowspan="3">H1: Transformer models will outperform traditional time series forecasting models in predicting cryptocurrency prices due to their ability to efficiently process sequences with self-attention mechanisms, allowing them to manage volatility and non-stationarity more effectively than traditional models and thus leading to improved forecasting accuracy.</td></tr>
<tr><td>Q2: Can Transformer models handle the inherent volatility and non-stationarity in cryptocurrency markets compared to traditional models?</td></tr>
<tr><td>Q3: Are specific pre-processing strategies like median imputation, seasonal adjustment, and moving averages more effective in preparing data for Transformers than for traditional models?</td></tr>
<tr><td>Q4: Do sentiment analysis features, such as social media contributions and tweet volumes, provide significant improvements on the predictive accuracy of cryptocurrency prices?</td><td>H2: The integration of sentiment analysis into forecasting models will enhance predictive accuracy, uncovering latent signals that are not captured by traditional financial indicators alone, thereby providing a more comprehensive understanding of market dynamics.</td></tr>
<tr><td>Q5: How well do models trained on data from one cryptocurrency (e.g., Bitcoin) generalize to other cryptocurrencies (e.g., Ethereum)?</td><td rowspan="2">H3: Transfer learning will demonstrate generalizability across different cryptocurrencies, leveraging learned patterns from one market to predict another with reduced model training requirements and similar performance.</td></tr>
<tr><td>Q6: What role does transfer learning play in enhancing model performance across different but related financial instruments?</td></tr>
</table>

3.1.2 Dataset Overview

The raw dataset for this project encompasses a wide range of financial and sentiment-related variables, providing a comprehensive view of the cryptocurrency market dynamics. Below is a schematic overview of the key components:

Financial Metrics:

- Prices:
  - `open`: Opening price
  - `close`: Closing price
  - `high`: Highest price of the period
  - `low`: Lowest price of the period
- Volume:
  - `volume`: Trading volume
  - `volume_24h`: 24-hour trading volume
- Market Data:
  - `market_cap`: Market capitalization
  - `circulating_supply`: Number of tokens in circulation

Social Media and Sentiment Metrics:

- Reddit Metrics:
  - `reddit_posts`: Number of Reddit posts about the asset
  - `reddit_posts_score`: Score of Reddit posts
  - `reddit_comments`: Number of comments on Reddit posts
  - `reddit_comments_score`: Score of comments on Reddit posts
- Twitter Metrics:
  - `tweets`: Number of tweets
  - `tweet_spam`: Number of tweets considered spam
  - `tweet_followers`: Number of followers engaging with the tweets
  - `tweet_retweets`: Number of retweets
  - `tweet_replies`: Number of replies to tweets
  - `tweet_favorites`: Number of favorites on tweets
  - `tweet_score`: Aggregated relevance score of tweets
  - `tweet_sentiment1` to `tweet_sentiment5`: Sentiment scores from different algorithms
  - `tweet_sentiment_impact1` to `tweet_sentiment_impact5`: Impact measurements of sentiment scores
- URL Metrics:
  - `url_shares`: Total number of URLs shared on Reddit
  - `unique_url_shares`: Number of unique URLs shared
- Social Media Engagement:
  - `social_contributors`: Likely the number of social interactions
  - `social_volume`: Volume of social media mentions
  - `social_score`: Score indicating overall sentiment (bullish/bearish)
  - `average_sentiment`: Average sentiment score (1-5 scale)
  - `sentiment_absolute`: Absolute sentiment score (1-5 scale)
  - `sentiment_relative`: Ratio of bullish to bearish sentiment
  - `social_dominance`: Share of voice in social media
  - `social_impact_score`: Volume/interaction impact score

- `news`: Number of news articles about the asset

Market Analysis Metrics:

- Moving Averages:
  - `sma_50`: 50-period simple moving average
  - `sma_100`: 100-period simple moving average
  - `sma_200`: 200-period simple moving average
- Rankings:
  - `market_cap_rank`: Market capitalization ranking
  - `percent_change_24h_rank`: 24-hour change ranking
  - `volume_24h_rank`: 24-hour volume ranking
  - `social_volume_24h_rank`: 24-hour social volume ranking
  - `social_score_24h_rank`: 24-hour social score ranking
- Additional Metrics:
  - `price_score`: Moving average-based trend indicator
  - `volatility`: Volatility of the asset
  - `correlation_rank`: Correlation of social data to price/volume
  - `galaxy_score`: Overall score measuring the asset against community metrics

This rich dataset allows for a detailed exploration of how financial and sentiment-related features can be leveraged to improve cryptocurrency price predictions using advanced machine learning models like Transformers, compared to traditional models like LSTM and LSTM-CNN hybrids.

3.2 Data Analysis and Preprocessing

In this chapter, we dive deep into the exploration and understanding of the dataset. Data analysis and preprocessing are crucial steps to transform the raw cryptocurrency data into a suitable format for modeling. This process ensures data cleanliness, consistency, and enhances the model's performance.

Data visualization and exploratory analysis are the focus of this phase, as they are essential tools for identifying underlying patterns, trends, and characteristics in the data. Using a variety of visualization techniques, we want to gain a thorough understanding of the structure and behavior of the data, which will help to direct our preprocessing steps going forward.

The following sections provide a detailed overview of the steps taken for data analysis and preprocessing, focusing on Bitcoin (BTC) data.

3.2.1 Data Cleaning

In the data cleaning phase of the project, the initial step involved eliminating columns with over 25% missing values. This threshold was set following a detailed analysis of the structure and content of the data, including the number of non-null entries per column, insights into the distribution, central tendency, and variability of the dataset, a common practice in time series forecasting to ensure robustness and accuracy. Columns with excessive missing values can introduce significant bias and uncertainty, thus their removal is essential to maintain the integrity of the dataset.

For the remaining columns, several strategies were employed to address missing values. The rolling median method was used for imputation, calculated over a specified window size. This technique is particularly beneficial in financial time series data [19], as it preserves the temporal integrity and accounts for local trends without introducing artificial fluctuations. Before applying this imputation method, the dataset was sorted chronologically to maintain the sequence of data points, ensuring that the temporal order critical to financial forecasting was respected.

Moreover, the authenticity of missing values was validated against historical market data from reliable online sources [20]. This validation step is crucial in financial markets, where gaps in data can reflect actual market closures or anomalies. By avoiding interpolation of potentially non-existent values and opting for removal instead, the dataset's authenticity and alignment with real market conditions were preserved. Such a conservative approach is vital in maintaining the credibility of the analysis and ensuring that the forecasting model is built on accurate and reliable data.

3.2.2 Data Splitting

The process of splitting the dataset into training, validation, and test sets is a critical practice in time series forecasting to ensure the effectiveness and generalizability of the model. Given the extensive size of the dataset, a strategic approach was adopted. Initially, the dataset was divided into two primary segments: 90% for the combined training and validation set, and 10% for the test set. This allocation is designed to provide a substantial portion of data for model training, facilitating the model's ability to learn and adapt to the complexities of financial time series data.

Further refinement led to an 80-10-10 split for training, validation, and testing respectively. This final split enhances model performance by ensuring that each phase of model development—training, validation, and testing—receives a representative portion of the data. This method is essential in financial time series forecasting to capture the underlying patterns and validate the model's predictive capabilities effectively.

To prevent data leakage, the dataset was sorted chronologically before splitting. This step is crucial as it preserves the temporal sequence of events, preventing the use of future information in the training phase which could artificially inflate model performance. Ensuring that the chronological order is maintained guarantees that the model is tested in conditions that closely resemble real-world scenarios, where future data is unknown at the time of prediction. This rigorous data splitting strategy is fundamental in developing a model that generalizes well to unseen data, thereby providing reliable and accurate forecasts in the volatile and dynamic domain of financial markets.

3.2.3 Data Visualization

Correlation Matrix

The correlation matrix is a powerful tool for understanding how different aspects of a cryptocurrency price behavior are interrelated. By plotting the matrix, features like 'Close', 'High', 'Low', 'Open', and 'Volume' are observed to be highly connected. This high correlation typically arises because these prices within a single trading day are inherently interrelated. For instance, the 'High' and 'Low' prices are within the range set by the 'Open' and 'Close' prices. Market dynamics, such as investor reactions to news events or changes in market sentiment, tend to impact these prices simultaneously, leading

to their closely aligned movements. This phenomenon reflects the integrated nature of market prices and is a crucial aspect to consider in financial time series analysis.

This analysis was also very useful in order to discard some of the features which showed very low or negative correlation to the 'Close' price. This step will be discussed more in depth in the Feature Engineering section.

Price and Social Volume Trends

The relationship between the cryptocurrency's close price and social volume over time is visualized using dual-axis plots, where the close price and social volume are plotted against time, allowing for an intuitive understanding of how social volume correlates with price movements.

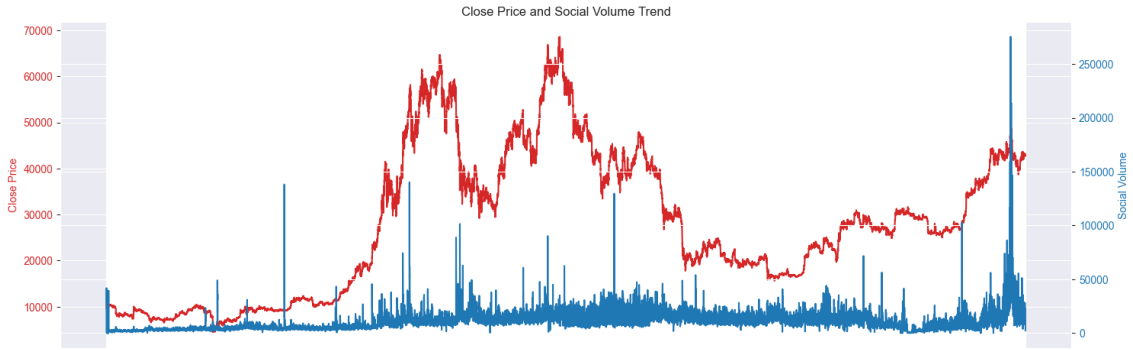

Figure 5: Close Price and Social Volume of BTC

Social Media Sentiment Trends

Social media sentiment metrics, such as the number of tweets and social scores, are plotted against time to visualize their trends, providing insights into how social media activity and sentiment evolve and their potential impact on cryptocurrency prices.

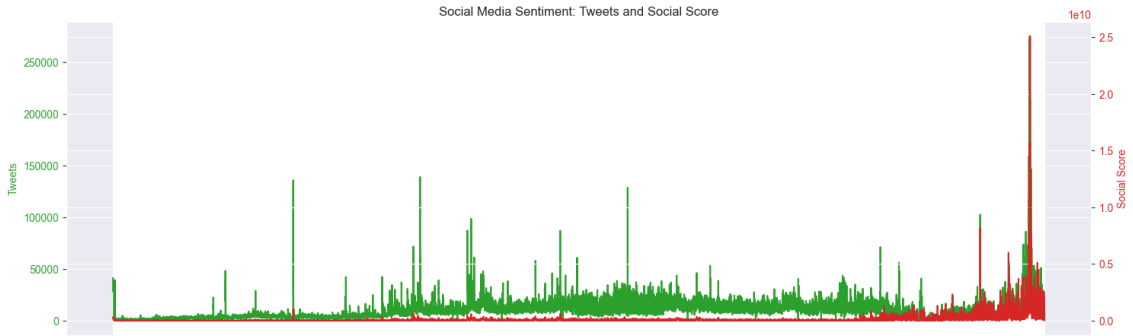

Figure 6: Tweets and Social Score of BTC

Sentiment vs. Price Time Series

Time series plots overlay sentiment scores with price changes to highlight potential correlations between market sentiment and price movements, offering a visual representation of the interaction between sentiment metrics and price trends.

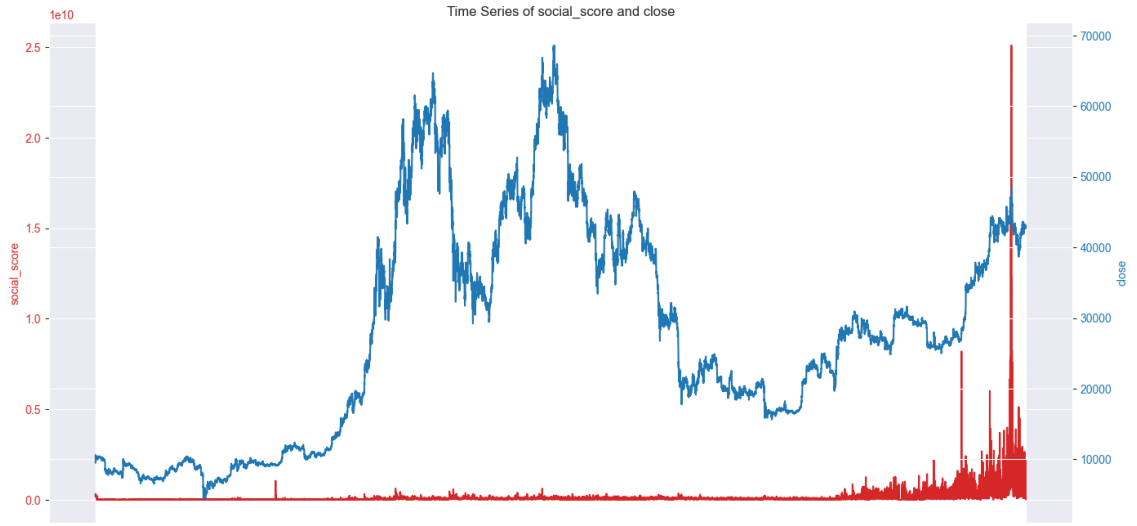

Figure 7: Social Score and Close Price of BTC

Cryptocurrencies Comparison

Data for multiple cryptocurrencies (e.g., ADA, BTC, DOGE) is preprocessed using the same methodology, ensuring consistency across datasets. Selected columns for analysis include 'timestamp', 'time', 'open', 'close', 'high', 'low', 'volume_24h', 'market_cap', 'social_contributors', and 'social_volume'. These columns provide a comprehensive view of each cryptocurrency's market activity and social engagement.

The closing prices of different cryptocurrencies are normalized to allow for a fair comparison of their price evolution over time. The normalized close prices are plotted using a visualization function, showing how different cryptocurrencies' prices change relative to each other.

Figure 8: Close Price per Cryptocurrency (normalized)

The plot helps in understanding the relative performance and volatility of various cryptocurrencies, which is crucial for the subsequent modeling and analysis. As the data suggests, the price of the different cryptocurrencies is highly correlated. Specifically, the plot shows how in bullish markets, where the biggest cryptocurrencies like Bitcoin (BTC) exhibit great periods of growth, there is always

a subsequent or concurrent growth of the other smaller cryptocurrencies. This also holds for bearish markets.

This detailed data analysis and preprocessing workflow ensures that the cryptocurrency data is clean, feature-rich, and ready for model training and evaluation. The visualizations provide initial insights into the relationships between financial metrics, social sentiment, and cryptocurrency prices, setting the stage for a comprehensive analysis using advanced forecasting models.

3.3 Stationarity Analysis

Stationarity analysis is a crucial step in time series forecasting as it ensures the data meets the assumptions required by many forecasting models. A stationary time series has a constant mean, variance, and autocorrelation structure over time, which is essential for the stability and reliability of the models. This section provides a detailed account of the stationarity analysis conducted on the cryptocurrency data, highlighting both visual and statistical approaches, as well as the transformations applied to achieve stationarity.

3.3.1 Visual Stationarity Analysis

Visual methods provide an intuitive understanding of the stationarity of the time series data. The following steps were undertaken to analyze the visual stationarity:

- Splitting the Data: The dataset was divided into two equal halves to compare rolling statistics over different periods. This approach helps in identifying any significant changes in the statistical properties of the series over time.
- First Half and Second Half: The first half consists of the initial 50% of the data points. The second half consists of the remaining 50% of the data points.
- Rolling Statistics Calculation: A window size of 12 was chosen to calculate the rolling mean and rolling standard deviation. This window size was selected to smooth out short-term fluctuations and highlight longer-term trends.
- Rolling Mean and Standard Deviation: Rolling mean and standard deviation were computed for both the first and second halves of the data. These statistics provide insight into the stability of the mean and variance over time.
- Rolling Mean and Standard Deviation Plots: Two plots were generated: one for the rolling mean and one for the rolling standard deviation. The rolling mean and standard deviation for the first half were plotted in blue, while those for the second half were plotted in red. This color differentiation aids in visual comparison.

The visual inspection of these plots helps to identify trends or shifts in the mean and variance, indicating non-stationarity if present.

Figure 9: Rolling Statistics: mean and std_dev of BTC

3.3.2 Statistical Stationarity Analysis

Statistical tests provide a more rigorous and objective measure of stationarity. Two tests were employed in this analysis: the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test.

- Augmented Dickey-Fuller (ADF) Test: The ADF test is used to test the null hypothesis that a time series is non-stationary. A significant p-value (typically less than 0.05) indicates rejection of the null hypothesis, suggesting that the series is stationary.

The hypotheses for the Augmented Dickey-Fuller (ADF) test are:

  - Null hypothesis (H0): The time series is not stationary because there is a unit root (if p-value > 0.05)
  - Alternative hypothesis (H1): The time series is stationary because there is no unit root (if p-value <= 0.05) The time series is stationary if we can reject the null hypothesis of the ADF test.

In this analysis, the initial ADF test was performed on the raw 'close' price series, yielding an ADF statistic of -2.219344 with a p-value of 0.199300. This result, compared against the critical values (1%: -3.431, 5%: -2.862, 10%: -2.567), fails to reject the null hypothesis, indicating that the series is not stationary.

- Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test

The hypotheses for the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test are:

  - Null hypothesis (H0): The time series is stationary because there is no unit root (if p-value > 0.05)
  - Alternative hypothesis (H1): The time series is not stationary because there is a unit root (if p-value <= 0.05) The time series is stationary if we fail to reject the null hypothesis of the KPSS.

The initial KPSS test on the 'close' price series produced a KPSS statistic of 7.91279954990115 with a p-value of 0.01. Given the critical values (10%: 0.347, 5%: 0.463, 2.5%: 0.574, 1%: 0.739), this result leads to the rejection of the null hypothesis, again indicating that the series is not stationary.

Given the non-stationarity indicated by both tests, various transformations were applied to the data to induce stationarity:

- Differencing: This transformation involves subtracting the previous observation from the current observation. Applying differencing to the 'close' series, the ADF test yielded a statistic of -27.777032 with a p-value of 0.000000, significantly below the 0.05 threshold, thus rejecting the null hypothesis and confirming stationarity.
- Log Transformation: This transformation helps stabilize the variance of a time series. However, the ADF test on the log-transformed 'close' series resulted in a statistic of -1.196466 with a p-value of 0.675114, indicating non-stationarity.
- Detrending by Model Fitting: This involves removing trends from the series. The ADF test on the detrended 'close' series produced a statistic of -1.500177 with a p-value of 0.533424, which also indicated non-stationarity.

Based on these tests, differencing was the most effective transformation for achieving stationarity in the 'close' price series. This transformation was crucial for ensuring the suitability of the time series data for modeling and forecasting.

We initially considered using ARIMA models, but ultimately decided against it due to our project's objectives. Our approach involved using 5-day batches to predict the next day, whereas ARIMA models are more effective without using data batches.

3.4 Feature Engineering

In this chapter, we explore the feature engineering process, which is crucial in predictive modeling, especially within the volatile sphere of cryptocurrency data. By creating informative and unique features, we can significantly improve the predictive power of our algorithms, providing more precise and robust forecasts.

3.4.1 Identifying Seasonality Patterns

Seasonal patterns reflect the markets' cyclical nature, where certain times of the year are more bullish, causing stocks to rise, while others are more bearish, leading to a decline in stock values. Identifying these patterns is crucial in gaining a deeper knowledge about the context and for better feature engineering.

The initial step in the analysis involved computing the monthly mean close prices to identify significant trends and patterns in the cryptocurrency market over time. By grouping the data by year and month, the average closing prices for each month were calculated. This aggregation smooths out daily volatility and highlights broader market movements.

The resulting plot (Figure 6) displays the temporal fluctuations in mean close prices, capturing dramatic rises and declines in prices, as well as periods of relative stability. This visualization aids in understanding the cyclical nature of cryptocurrency prices, which is crucial for developing robust forecasting models.

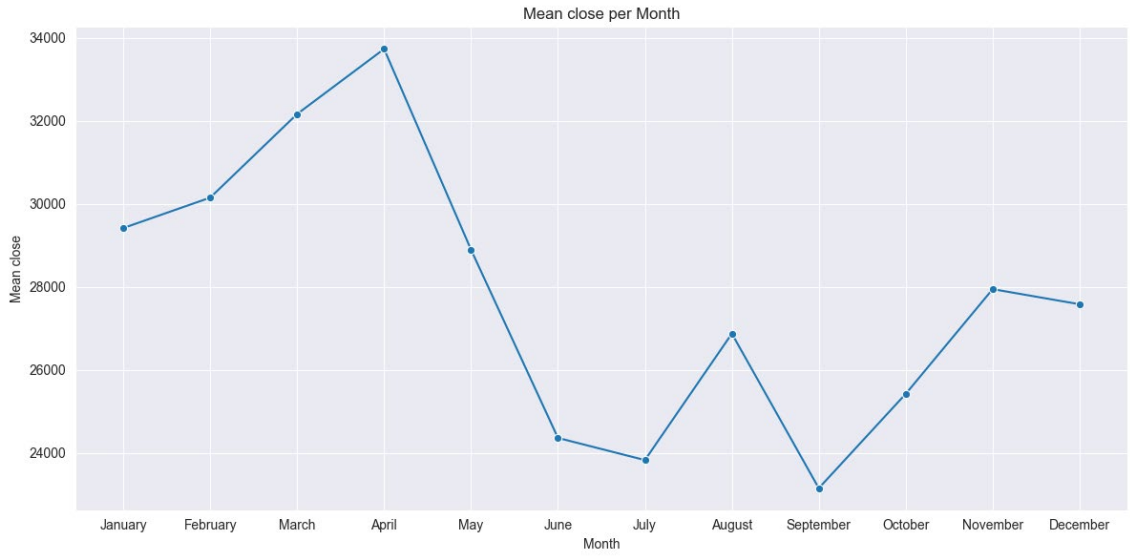

Figure 10: Monthly Mean Close Price of BTC

3.4.2 Encoding Seasonality

In the feature engineering process, a trigonometric transformation was applied to the date information in the dataset. Specifically, the day of the week was converted into two separate features using sine and cosine functions. This cyclical encoding captures the repetitive nature of weekly patterns, providing the model with structured temporal information that is crucial for forecasting. Weekly cycles are common in financial markets due to regular human activities and institutional operations. By including these sinusoidal time features, the machine learning model can better understand and leverage the inherent periodicity in cryptocurrency data, potentially improving predictive performance for assets like Bitcoin, Ethereum, and other cryptocurrencies.

Further, the time series analysis involved encoding months into three distinct categories based on the results of the seasonality analysis: 'Bullish', 'Bearish', and 'Normal'. This categorization was derived from the results of previous analyses and reflects the typical behavior of the market in different months.

- Bullish Months: These are months where the mean close price is significantly higher than in other months. For example, the plot indicates peak prices around certain months, suggesting a general bullish trend during these periods. The plot shows higher values for January, February, March and April, which were encoded as bullish months.
- Bearish Months: Characterized by a comparatively lower mean close price, indicating a bearish trend. The transition from one month to another may show a dip in the mean close price, marking the start of a potentially bearish period. The plot shows lower values for June, July, September and October, which were encoded as bullish months.
- Normal Months: Months in which the mean close price does not show significant peaks or troughs and remains relatively stable. These months exhibit a normal market trend without notable highs or lows. The remaining four months, May, August, November and December were encoded as normal.

To integrate this information effectively into the predictive models, one-hot encoding was utilized. This standard technique in machine learning converts categorical data into a numerical format by explicitly adding three columns to the dataset, representing 'Bullish', 'Bearish', and 'Normal'.

Capturing these seasonal trends enhances the model's capability to recognize and adapt to recurring market patterns, which is crucial for accurate time series forecasting in the financial domain.

3.4.3 Lag Features

The introduction of lag features was explored as an option. Lag features provide historical context, which can be important in financial markets where past trends and movements influence future prices. The primary focus was on the 'Close' price, for which three lag features representing the 'Close' prices from the previous three days were considered.

To handle the initial missing values, the first available 'Close' value for each cryptocurrency was used to fill the gaps. This approach ensured that the model had access to a complete dataset, avoiding the loss of potentially valuable information in the initial days. By incorporating these lag features, the model is better equipped to capture the temporal dependencies and patterns within the data, thereby enhancing its predictive performance.

However, this approach introduced noise and the potential for data leakage. As a result, lag features were not used in the final models. This decision was made to ensure the integrity and reliability of the predictive models, avoiding the pitfalls associated with noisy data and inadvertent inclusion of future information.

3.4.4 Fourier Transformation

In this section, the analysis involved transforming the closing price data from the time domain into the frequency domain using the Fast Fourier Transform (FFT) method. This mathematical transformation decomposes the time series into its constituent frequencies, revealing periodic patterns and underlying trends in price movements. Understanding these patterns is crucial for time series forecasting in the volatile and dynamic domain of cryptocurrency markets.

By plotting the frequencies against their amplitudes on a logarithmic scale, it became possible to discern the most significant cycles that characterize price behavior. The resulting visualization, shown in Figure 7, served as a powerful tool for identifying dominant frequencies. These dominant frequencies correspond to regular fluctuations in the cryptocurrency's closing price and can provide valuable insights into the cyclical nature of the market. Identifying such cycles helps in understanding recurring events that might be driven by market sentiment, investor behavior, or macroeconomic factors.

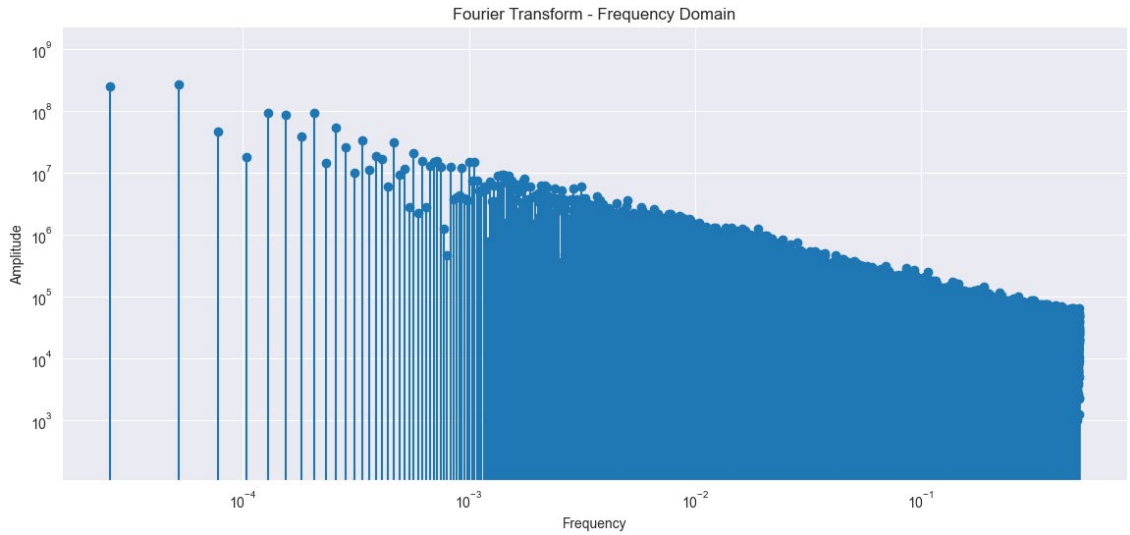

Figure 11: Fourier Transform in the Frequency Domain (BTC)

In the cryptocurrency market, where prices are highly volatile and often influenced by speculative trading, uncovering these periodic patterns can be particularly beneficial. For instance, regular cycles might reflect the impact of recurring news events, regulatory announcements, or even predictable trading behaviors linked to market participants' strategies. By recognizing these cycles, forecasting models can be better tailored to account for predictable periodic fluctuations, thus enhancing their accuracy.

The analysis then filtered these frequencies to focus on those corresponding to time periods ranging from five days to approximately a trading year (252 days), as shown in Figure 8. This filtering helped isolate the most relevant cycles impacting cryptocurrency movements. Focusing on these specific ranges allows the model to capture both short-term trading patterns and longer-term investment cycles, providing a more comprehensive understanding of market dynamics.

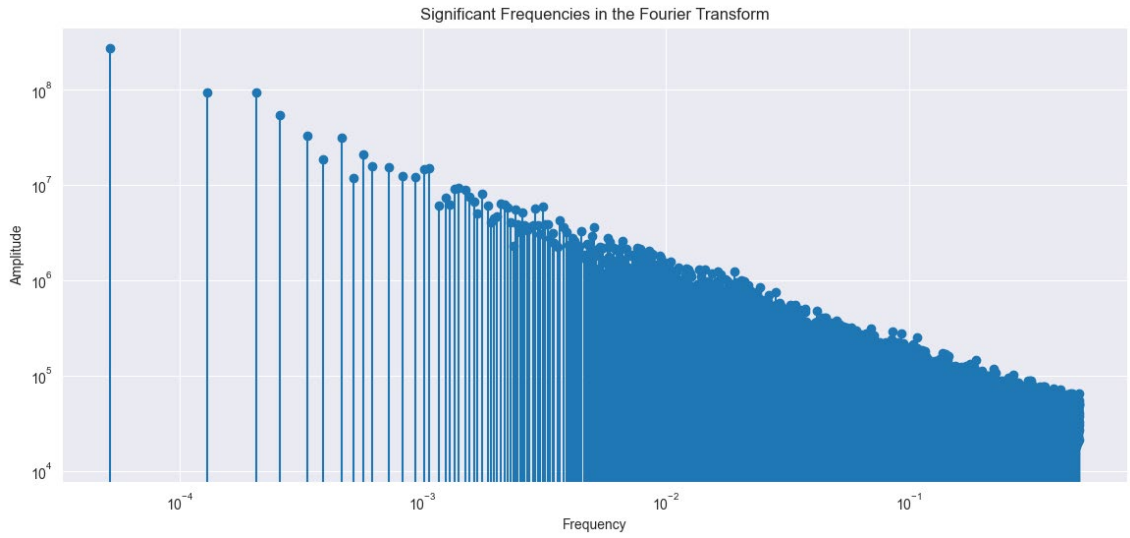

Figure 12: Significant Frequencies in the Fourier Transform (BTC)

The identified significant frequencies were then converted back into time periods, providing a clear view of the cyclic patterns that could be critical for predicting future price movements. To approximate the cryptocurrency's price, a limited number of frequency components were used to

reconstruct the time series. Figure 9 shows how the reconstructed signals with 5, 10, and 25 frequency components were compared with the actual closing prices, revealing the underlying patterns and trends in the data.

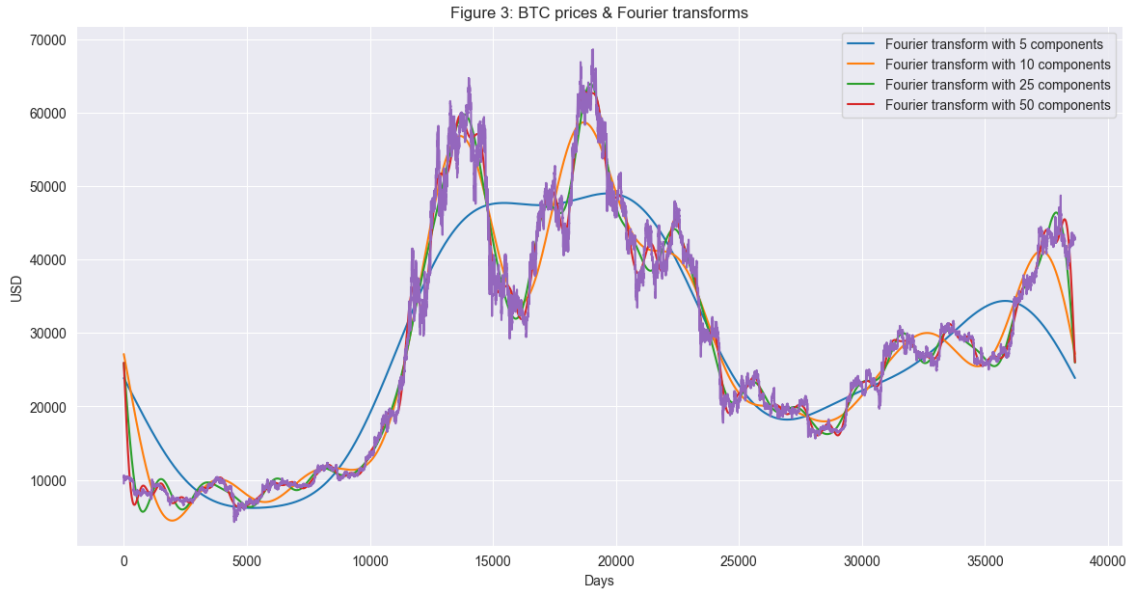

Figure 13: Close Price and Fourier Transforms of BTC

This reconstruction process highlights how key frequencies contribute to the overall price movement, helping to distinguish between noise and meaningful signals. In time series forecasting, particularly for cryptocurrencies, this capability is vital. It allows for the creation of more robust models that can adapt to the inherent noise and volatility of the market by focusing on the significant patterns that genuinely influence price changes.

By employing FFT, the analysis gained insights into the cyclical nature of cryptocurrency price movements, enhancing the understanding of periodic influences. This deeper understanding aids in improving the predictive capabilities of the forecasting models. It enables the models to leverage the identified cycles to anticipate future price movements more accurately, thus offering better decision-making tools for traders and investors operating in the highly unpredictable cryptocurrency markets.

3.4.5 Autocorrelation

Autocorrelation measures the linear relationship between lagged values of a time series. If a series is significantly autocorrelated, it means the previous values of the series (lags) may be helpful in predicting the current value. In this analysis, 'Date' was set as the index and resampled to monthly averages, which is essential for meaningful autocorrelation analysis. This approach helps in understanding whether previous monthly 'Close' values are correlated with future ones.

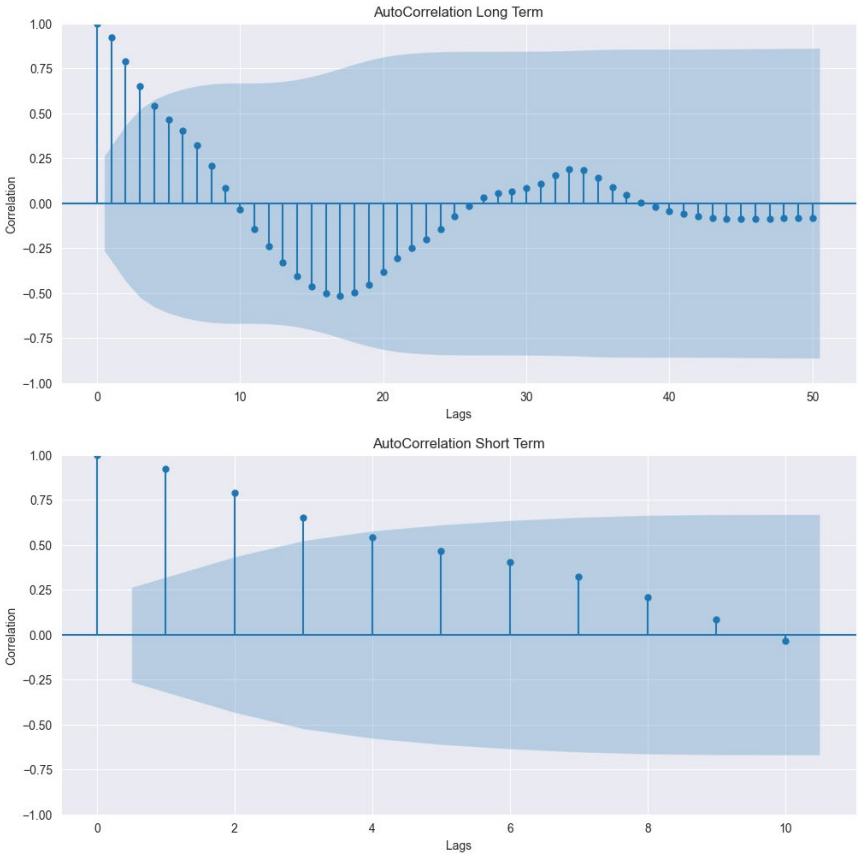

Figure 14: Autocorrelation Long-Short Term of BTC

The autocorrelation plots generated for the monthly average data are shown in the figure. The first plot illustrates the long-term autocorrelation up to 50 lags, while the second focuses on the short-term autocorrelation up to 10 lags.

The long-term autocorrelation plot demonstrates a gradual decay in correlation as the number of lags increases, indicating a slow decline in the relationship over time. The initial lags show high positive autocorrelation, which gradually decreases, becomes negative, and then fluctuates around zero. This pattern suggests that past values have a persisting influence on future values, characteristic of a non-stationary time series. The persistent influence of past values implies that the series is not random and is suitable for time series modeling. This is particularly relevant for cryptocurrency markets, where historical price movements can significantly impact future prices due to the market's inherent volatility and speculative nature.

The short-term autocorrelation plot shows that the first few lags have significant positive autocorrelation, which gradually diminishes. This suggests that recent past values have a strong influence on the current value, which is a common characteristic in financial time series data.

3.4.6 Partial Autocorrelation

Partial autocorrelations measure the linear dependence of one variable after removing the effect of other variables that affect both variables. The key difference between the autocorrelation (ACF) and partial autocorrelation (PACF) plots lies in the relationships they measure. The ACF plot shows the total correlation between a time series and its lagged values, including indirect effects. In contrast,

the PACF plot isolates the direct correlation at each lag, removing the influence of correlations at shorter lags.

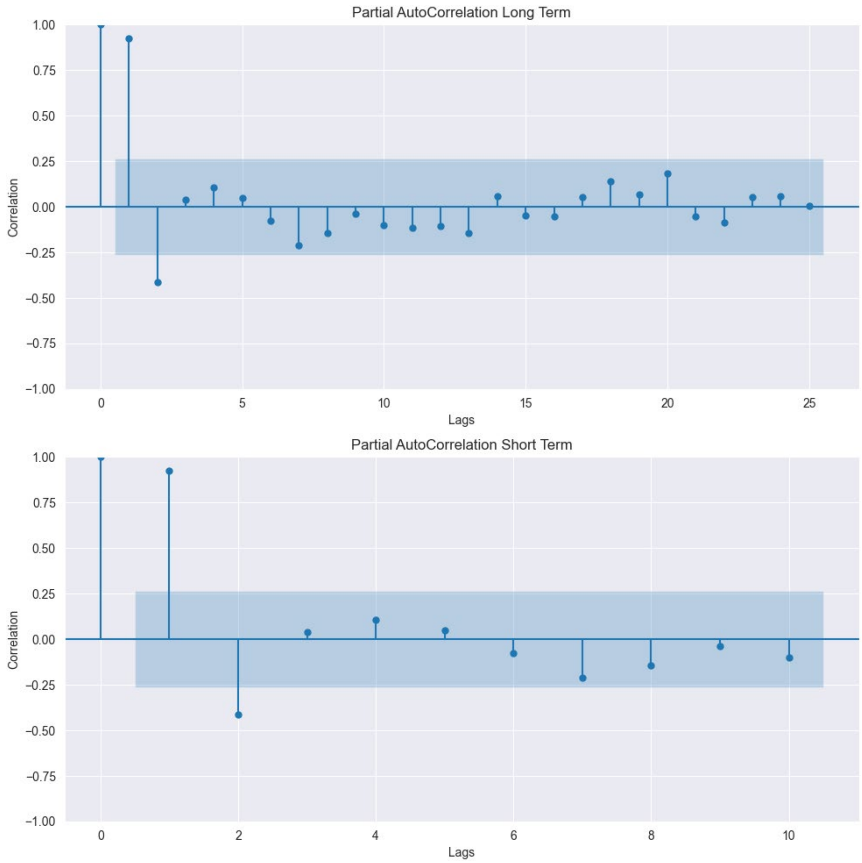

Figure 15: Partial Autocorrelation Long-Short Term of BTC

The partial autocorrelation plots show significant direct correlation at the first lag for both long-term and short-term views, which quickly diminishes and becomes statistically insignificant for subsequent lags. This pattern suggests a potential AR(1) process, where only the immediate past value has a direct impact on the current value. Understanding these direct correlations is crucial for developing models that can accurately predict future values based on past data.

3.4.7 Moving Average

In the field of financial forecasting, two predominant types of moving averages stand out: the Simple Moving Average (SMA) and the Exponentially Weighted Moving Average (EWMA). Both serve the purpose of smoothing out price data to identify trends, yet they differ fundamentally in their approach to weighting data points, which leads to distinct implications in their application in financial markets.

- Simple Moving Average (SMA): SMA is the arithmetic mean of a specified number of past prices. It assigns equal weight to each data point within its window, treating older and newer data with identical importance. This equal weighting results in a smoothed line that is easy to calculate and interpret, making it a popular choice for identifying long-term trends and support/resistance levels in financial markets.
- Exponentially Weighted Moving Average (EWMA): In contrast, EWMA prioritizes recent data points by assigning them exponentially decreasing weights. The most recent data has the most influence on the moving average, with the impact fading for older data. This

weighting approach makes EWMA more sensitive to recent market movements, allowing it to adjust more quickly to new trends and reversals.

In financial forecasting, choosing between Simple Moving Average (SMA) and Exponentially Weighted Moving Average (EWMA) depends on the analysis goals. SMA, with its equal weighting across data points, is ideal for long-term trend analysis due to its stability and simplicity. It effectively filters out short-term fluctuations, providing a clear view of longer-term market trends. Conversely, EWMA is preferred for short-term strategies as it's more responsive to recent market changes. Its exponentially decreasing weights for older data make it sensitive to immediate market movements, crucial for rapid response in volatile markets.

3.4.8 SMA and EWMA Application

Both SMA and EWMA were applied to the cryptocurrency closing price data to analyze their effectiveness in capturing market trends.

Various SMA windows (10, 30, 50, 100 days) were applied to the closing prices to observe how the moving averages smooth the data and capture trends. The different window lengths were chosen to understand how shorter and longer-term trends are reflected in the data. Shorter windows like 10 days provide a more granular view of recent price movements, while longer windows such as 100 days smooth out short-term volatility, offering a clearer picture of long-term trends.

The resulting visualizations demonstrated that longer SMA windows provide a smoother curve, effectively filtering out short-term fluctuations. This makes SMA particularly useful for identifying long-term market trends and support/resistance levels, which are essential for strategic decision-making in cryptocurrency trading.

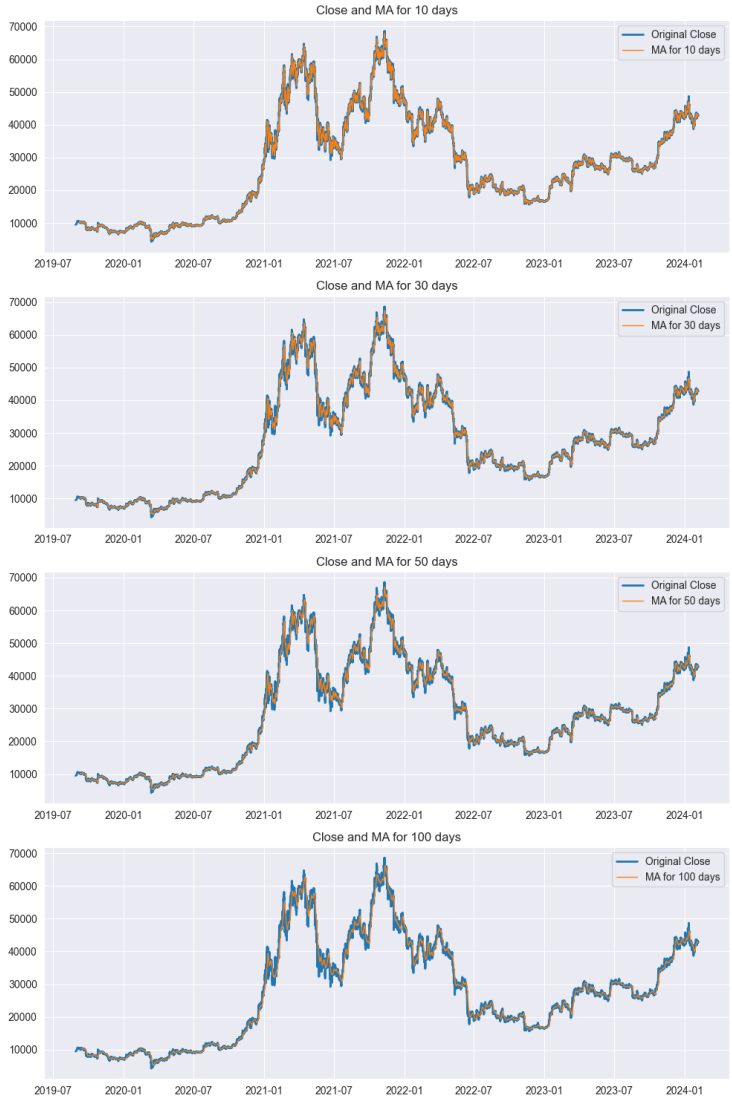

Figure 16: Different windows for MA

Similarly, EWMA windows (10, 30, 50, 100 days) were applied to the closing prices to observe their responsiveness to recent price changes. The exponential weighting of EWMA allows it to respond more quickly to recent market movements compared to SMA. This makes EWMA particularly useful for short-term trend analysis in volatile markets like cryptocurrencies, where quick adaptation to market changes is crucial.

The visualizations revealed that EWMA captures trends more dynamically, reflecting the most recent price changes more accurately. This responsiveness is advantageous for traders and analysts who need to make timely decisions based on the latest market data.

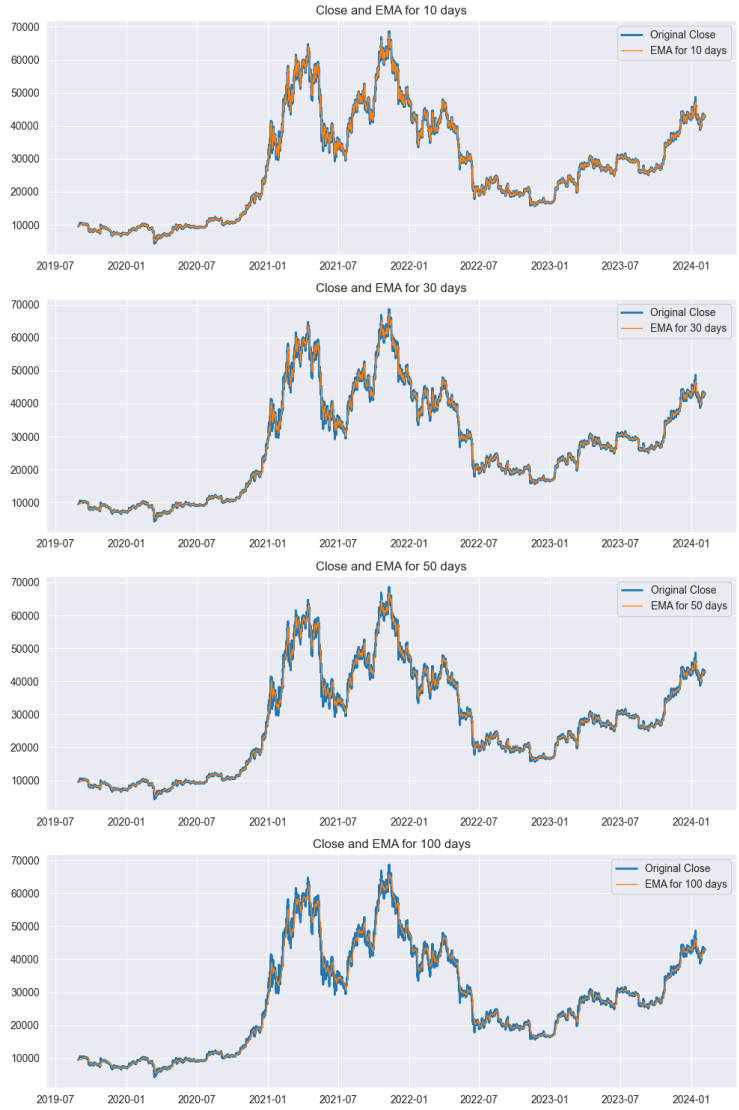

Figure 17: Different windows for EMA

Rate of Change (ROC)

To further enhance the feature set, the Rate of Change (ROC), a momentum indicator, was analyzed based on a 50-day moving average. The ROC measures the percentage change in the moving average over a designated period, providing insights into the speed and direction of price movements. This indicator was then categorized into five levels, ranging from 'very high positive' to 'high negative', offering a detailed view of market trends. Categorizing ROC in this way helps in identifying strong or weak upward or downward movements, as well as periods of stability, which are critical for making informed trading decisions.

Feature Generation and Model Integration

The decision to utilize EWMA over SMA was driven by the focus on short-term trends. EWMA's greater responsiveness to recent market movements made it more suitable for the needs of this analysis. This choice was particularly beneficial given the emphasis on agility and adaptability in response to rapidly changing market conditions.

Finally, to integrate this feature effectively into the predictive model, categorical encoding was employed. This process transformed the ROC categories into a format conducive to quantitative analysis, ensuring seamless incorporation into the broader forecasting framework. The encoded EWMA-based ROC feature thus became a pivotal element of the cryptocurrency forecasting model.

The application of both SMA and EWMA to cryptocurrency data provided valuable insights into their respective strengths. SMA, with its equal weighting, is ideal for long-term trend analysis, while EWMA, with its sensitivity to recent data, is better suited for short-term forecasting. The ROC analysis and categorical encoding further enriched the feature set, enhancing the predictive capabilities of the model in capturing the complex and volatile nature of cryptocurrency markets.

4 Model Training and Evaluation

In this section, the model training and evaluation process is shown. Three models were constructed and tested against different cryptocurrency time series data. At first, the models were tested on Bitcoin (BTC) data, to assess the general predictive capabilities on the most stable and long-lasting time series. Then, the analysis is extended on the other cryptocurrencies and results are compared. A baseline model, predicting prices of tomorrow using the price of today is used for benchmarking purposes.

4.1 Long Short-Term Memory (LSTM)

In the context of financial time series forecasting, LSTMs are highly valued for their ability to model complex temporal dependencies. This ability is crucial in understanding and predicting the volatile and often unpredictable nature of financial markets. LSTMs can effectively capture patterns and trends over long periods, making them suitable for forecasting tasks in the financial domain, including cryptocurrency price prediction.

4.1.1 LSTM Model Architecture

The LSTM model architecture was designed to predict future cryptocurrencies prices based on historical data. The architecture consists of the following layers:

- Input Layer: Takes in the preprocessed time-series data.
- LSTM Layers: Two LSTM layers, each with 16 units and 'leaky_relu' activation, were used. The first LSTM layer returns sequences to feed into the second LSTM layer.
- Output Layer: A dense layer with a single unit, suitable for regression tasks, outputs the predicted price.

This architecture allows the model to capture both short-term fluctuations and long-term trends in the cryptocurrency market. The specific details of the LSTM model are as follows:

Layer (type)	Output Shape	Param #
Input (InputLayer)	[(None, 5, 12)]	0
lstm_20 (LSTM)	(None, 5, 16)	1856
lstm_21 (LSTM)	(None, 16)	2112
dense_10 (Dense)	(None, 1)	17

- Total params: 3985
- Input Shape: The input shape is defined by the number of timesteps and features in the training data.
- Activation Function: 'Leaky ReLU' was chosen for the LSTM layers to allow a small gradient when the unit is not active, preventing dead neurons.
- Output Units: The final dense layer has one unit to produce a single price prediction.
- Optimizer and Loss Function: The model was compiled with the 'Adam' optimizer and 'mean squared error' (MSE) loss function, with additional metrics such as 'mean absolute error' (MAE) and 'mean absolute percentage error' (MAPE).

4.1.2 Training and Evaluation

The LSTM model was trained on the preprocessed cryptocurrencies price data using the following configuration:

- Batch Size: 64
- Epochs: 100
- Validation Split: The validation set was used to monitor the model's performance and adjust training dynamically.
- Early Stopping: Implemented to halt training when the validation loss did not improve for 10 consecutive epochs.
- Learning Rate Reduction: The learning rate was reduced by a factor of 0.5 if the validation loss plateaued for 5 epochs, with a minimum learning rate of 1e-5.

The training process aimed to optimize the model's parameters to minimize the loss on the validation set. Model performance was evaluated using the test set, focusing on the following metrics:

- Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values.
- Mean Absolute Error (MAE): Measures the average absolute difference between predicted and actual values.
- Mean Absolute Percentage Error (MAPE): Measures the average percentage error between predicted and actual values.

4.2 LSTM-CNN Hybrid Model

This section describes the implementation of a hybrid Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) model for predicting cryptocurrency prices, following the same process of the previous LSTM model, by first testing it on Bitcoin (BTC) data before extending it to other highly volatile cryptocurrencies. The approach leverages the strengths of both LSTM and CNN architectures to capture both temporal dependencies and local features in the time series data. The following discussion outlines the architecture of the hybrid model, and the evaluation of its performance.

4.2.1 LSTM-CNN Hybrid Model Architecture

The hybrid model combines the convolutional layers of CNNs, which excel at capturing local patterns, with the recurrent layers of LSTMs, which are adept at learning long-term dependencies. The detailed architecture is as follows:

Layer (type)	**Output Shape**	**Param #**
Input (InputLayer)	(None, 5, 18)	0
conv1d (Conv1D)	(None, 5, 256)	14,080
leaky_re_lu (LeakyReLU)	(None, 5, 256)	0
max_pooling1d (MaxPooling1D)	(None, 2, 256)	0
conv1d_1 (Conv1D)	(None, 2, 128)	98,432
leaky_re_lu_1 (LeakyReLU)	(None, 2, 128)	0
max_pooling1d_1 (MaxPooling1D)	(None, 1, 128)	0
lstm (LSTM)	(None, 1, 128)	131,584
dropout (Dropout)	(None, 1, 128)	0
lstm_1 (LSTM)	(None, 128)	131,584
dropout_1 (Dropout)	(None, 128)	0
dense (Dense)	(None, 64)	8,256
leaky_re_lu_2 (LeakyReLU)	(None, 64)	0
dropout_2 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 1)	65

- **Input Layer**: The input layer takes sequences of historical price data with multiple features.
- **Convolutional Layers:**
  - **CNN Layer 1**: The first convolutional layer applies 256 filters of size 3, followed by a Leaky ReLU activation and max pooling. This layer captures local trends and reduces dimensionality.
  - **CNN Layer 2**: The second convolutional layer uses 128 filters of size 3, again followed by a Leaky ReLU activation and max pooling. This layer further refines feature extraction.
- **LSTM Layers:**
  - **LSTM Layer 1**: This layer consists of 128 units and returns sequences to allow stacking. Dropout is applied for regularization to prevent overfitting.
  - **LSTM Layer 2:** Another LSTM layer with 128 units processes the output from the previous layer, followed by dropout.

- **Dense Layer**: A dense layer with 64 units and Leaky ReLU activation is added for feature extraction and further regularization through dropout.
- **Output Layer**: The final dense layer with a single unit outputs the predicted price, suitable for regression tasks.
- **Total params**: 384001

The model is compiled using the 'adam' optimizer and 'mean squared error' (MSE) loss function, with 'mean absolute error' (MAE) as an additional metric.

4.2.2 Training and Evaluation

The model is trained on the training set with early stopping and learning rate reduction callbacks to optimize training duration and performance:

- **Early Stopping**: Monitors the validation loss and stops training if there is no improvement for 100 epochs, restoring the best weights.
- **ReduceLROnPlateau:** Reduces the learning rate by a factor of 0.5 if the validation loss plateaus for 5 epochs, with a minimum learning rate of 1e-5.

The model's performance is evaluated on the test set using mean squared error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). These metrics provide a comprehensive assessment of the model's accuracy in predicting future prices.

4.3 Transformers

This section outlines the implementation and evaluation of a Transformer model for predicting cryptocurrency prices, following the same process of the previous LSTM model, by first testing it on Bitcoin (BTC) data before extending it to other highly volatile cryptocurrencies. The approach leverages the Transformer architecture's ability to capture complex temporal patterns and dependencies in financial time series data. The following discussion covers the model architecture and the evaluation of the model's performance.

4.3.1 Transformer Model Architecture

The Transformer model is designed to handle sequential data and capture long-range dependencies, making it highly suitable for financial time series forecasting.

The architecture of the Transformer model included:

- **Head Size:** 128
- **Number of Attention Heads:** 4
- **Feed Forward Dimension:** 2
- **Number of Transformer Blocks:** 4
- **MLP Units:** [256]
- **MLP Dropout:** 0.10
- **Attention Dropout:** 0.10
- **Attention Axes:** 1

The detailed layers' architecture is as follows:

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 5, 18)	0	[]
layer_normalization	(None, 5, 18)	36	['input_1[0][0]']
multi_head_attention	(None, 5, 18)	38,418	['layer_normalization[0][0]', 'layer_normalization[0][0]']
dropout	(None, 5, 18)	0	['multi_head_attention[0][0]']
tf.**operators**.add	(None, 5, 18)	0	['dropout[0][0]', 'input_1[0][0]']
layer_normalization_1	(None, 5, 18)	36	['tf.**operators**.add[0][0]']
conv1d	(None, 5, 2)	38	['layer_normalization_1[0][0]']
dropout_1	(None, 5, 2)	0	['conv1d[0][0]']
conv1d_1	(None, 5, 18)	54	['dropout_1[0][0]']
tf.**operators**.add_1	(None, 5, 18)	0	['conv1d_1[0][0]', 'tf.**operators**.add[0][0]']
layer_normalization_2	(None, 5, 18)	36	['tf.**operators**.add_1[0][0]']
multi_head_attention_1	(None, 5, 18)	38,418	['layer_normalization_2[0][0]', 'layer_normalization_2[0][0]']
dropout_2	(None, 5, 18)	0	['multi_head_attention_1[0][0]']
tf.**operators**.add_2	(None, 5, 18)	0	['dropout_2[0][0]', 'tf.**operators**.add_1[0][0]']
layer_normalization_3	(None, 5, 18)	36	['tf.**operators**.add_2[0][0]']
conv1d_2	(None, 5, 2)	38	['layer_normalization_3[0][0]']
dropout_3	(None, 5, 2)	0	['conv1d_2[0][0]']
conv1d_3	(None, 5, 18)	54	['dropout_3[0][0]']
tf.**operators**.add_3	(None, 5, 18)	0	['conv1d_3[0][0]', 'tf.**operators**.add_2[0][0]']
layer_normalization_4	(None, 5, 18)	36	['tf.**operators**.add_3[0][0]']
multi_head_attention_2	(None, 5, 18)	38,418	['layer_normalization_4[0][0]', 'layer_normalization_4[0][0]']
dropout_4	(None, 5, 18)	0	['multi_head_attention_2[0][0]']
tf.**operators**.add_4	(None, 5, 18)	0	['dropout_4[0][0]', 'tf.**operators**.add_3[0][0]']
layer_normalization_5	(None, 5, 18)	36	['tf.**operators**.add_4[0][0]']
conv1d_4	(None, 5, 2)	38	['layer_normalization_5[0][0]']
dropout_5	(None, 5, 2)	0	['conv1d_4[0][0]']
conv1d_5	(None, 5, 18)	54	['dropout_5[0][0]']
tf.**operators**.add_5	(None, 5, 18)	0	['conv1d_5[0][0]', 'tf.**operators**.add_4[0][0]']
layer_normalization_6	(None, 5, 18)	36	['tf.**operators**.add_5[0][0]']
multi_head_attention_3	(None, 5, 18)	38,418	['layer_normalization_6[0][0]', 'layer_normalization_6[0][0]']
dropout_6	(None, 5, 18)	0	['multi_head_attention_3[0][0]']
tf.**operators**.add_6	(None, 5, 18)	0	['dropout_6[0][0]', 'tf.**operators**.add_5[0][0]']
layer_normalization_7	(None, 5, 18)	36	['tf.**operators**.add_6[0][0]']
conv1d_6	(None, 5, 2)	38	['layer_normalization_7[0][0]']
dropout_7	(None, 5, 2)	0	['conv1d_6[0][0]']
conv1d_7	(None, 5, 18)	54	['dropout_7[0][0]']

tf.**operators**.add_7	(None, 5, 18)	0	['conv1d_7[0][0]', 'tf.**operators**.add_6[0][0]']
global_average_pooling1d	(None, 5)	0	['tf.**operators**.add_7[0][0]']
dense	(None, 256)	1,536	['global_average_pooling1d[0][0]']
dropout_8	(None, 256)	0	['dense[0][0]']
dense_1	(None, 1)	257	['dropout_8[0][0]']

- **Input Layer:** The input layer takes sequences of historical price data with multiple features.
- **Transformer Encoder:**
    - **Layer Normalization:** Normalizes the input to stabilize and accelerate the training process.
    - **Multi-Head Attention:** This mechanism allows the model to focus on different parts of the input sequence simultaneously, capturing various temporal patterns.
    - **Residual Connection and Dropout:** Residual connections help in preserving the learned features and enable better gradient flow, while dropout is used for regularization to prevent overfitting.
    - **Feed-Forward Neural Network:** A series of convolutional layers followed by ReLU activation to process the outputs from the attention mechanism and capture higher-level features.
- **Stacking Transformer Blocks:** Multiple transformer blocks are stacked to deepen the model, allowing it to learn more complex patterns in the data.
- **Global Average Pooling:** Reduces the dimensionality of the data by averaging the features, which helps in capturing the overall trend.
- **Dense Layers:** Fully connected layers with ReLU activation for further feature extraction and regularization through dropout.
- **Output Layer:** A dense layer with a single unit outputs the predicted price, suitable for regression tasks.

4.3.2 Training and Evaluation

Hyperparameters Configuration:

- **Loss Function:** Mean Squared Error (MSE)
- **Optimizer**: Adam with a learning rate of 0.001
- **Metrics:** MAE, MAPE
- **Batch Size:** 64
- **Epochs**: 100

The model is trained on the training set with the following callbacks strategies to optimize performance:

- **Early Stopping:** Monitors the validation loss and stops training if there is no improvement for 10 epochs, restoring the best weights.
- **ReduceLROnPlateau**: Reduces the learning rate by a factor of 0.5 if the validation loss plateaus for 5 epochs, with a minimum learning rate of 1e-5.

The model's performance is evaluated on the test set using mean squared error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). These metrics provide a comprehensive assessment of the model's accuracy in predicting future prices.

5 Results and Findings

5.1 First Analysis only using Financial Data

The initial analysis focuses on utilizing only financial data to forecast cryptocurrency prices, leveraging LSTM, LSTM-CNN hybrids, and Transformer models. This approach establishes an efficient benchmark by evaluating the predictive power of traditional and advanced neural network models based solely on intrinsic market variables such as 'open', 'close', 'high', 'low', and 'market_cap'.

By isolating financial features, this analysis aims to determine the extent to which these models can capture underlying market dynamics and temporal dependencies inherent in the financial data. It provides a clear baseline performance against which the impact of integrating additional data sources, such as sentiment analysis features, can be measured. This step is essential for understanding the strengths and limitations of each model in predicting the highly volatile and unpredictable nature of cryptocurrency prices based purely on market data.

5.1.1 LSTM

The results are visualized by plotting the actual versus predicted values for the training, validation, and test sets. This visual comparison helps in understanding the model's performance and identifying any patterns or discrepancies. The training and validation loss per epoch are plotted to analyze the convergence and stability of the training process. The early stopping mechanism ensures that the model does not overfit, as indicated by the minimal gap between training and validation losses.

The LSTM model's performance on Bitcoin (BTC) data over the training epochs is visualized in Fig 14. The training and validation loss curves exhibit the typical behavior of a learning model. Initially, both losses decrease rapidly, with training loss nearing zero by epoch 10, while validation loss stabilizes at a low value. The close tracking of both curves indicates good generalization without overfitting.

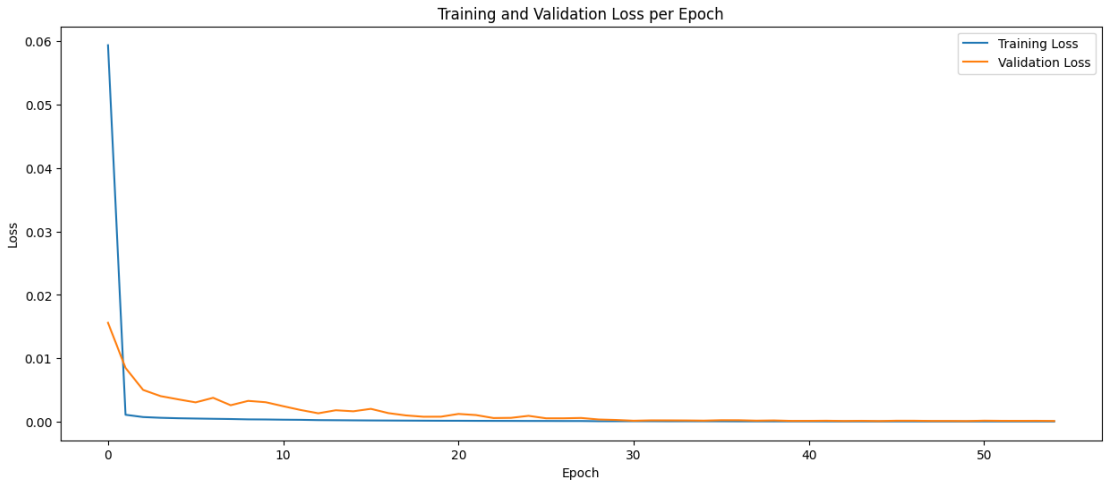

Figure 18: Training and Validation Loss per Epoch

Test Set Results

The results on Bitcoin (BTC) data were visualized by plotting actual versus predicted values to provide a clear comparison and understanding of the model's accuracy. Fig. 15 shows the comparison on the test set.

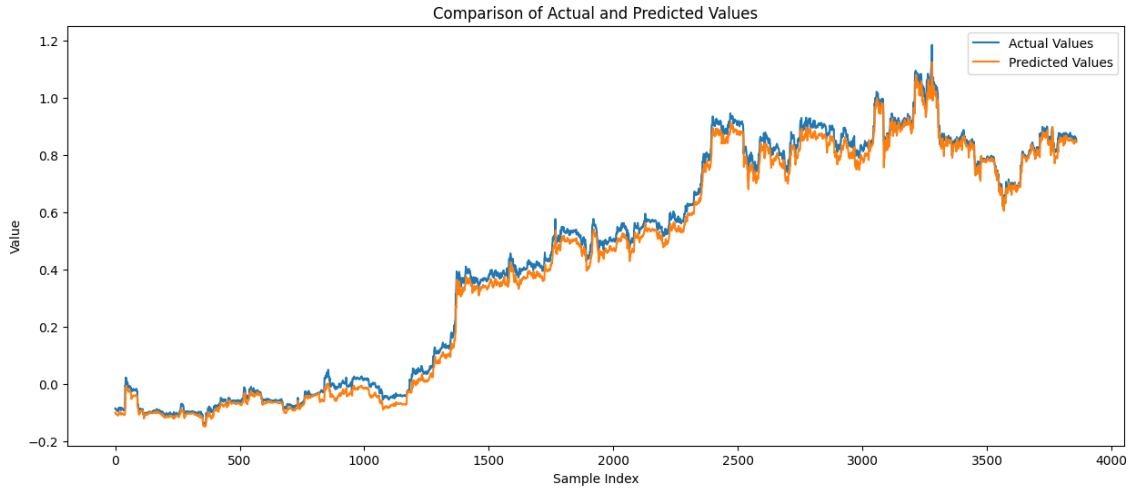

Figure 19: Comparison of Actual and Predicted Values on Test Set

The close correspondence between the two curves in the test set indicates that the model can generalize well to new data. This is a promising result for the application of the LSTM model in predicting prices, showing its potential as a tool for financial analysis and decision-making.

The following table summarizes the final results for the LSTM model on Bitcoin (BTC) data.

Data	**Model**	**MSE**	**MAE**	**MAPE**
BTC	LSTM	0.004930	0.058811	0.382770

LSTM on multiple Cryptocurrencies

This section describes the implementation of the described LSTM model for predicting different cryptocurrency prices, including ADA, BTC, DOGE, ETH, XMR, XRP, and AAVE, leveraging the strengths of LSTM architectures to capture temporal dependencies in diverse financial time series data.

Evaluating the LSTM model across multiple cryptocurrencies with varying volatility and time series characteristics is valuable for many reasons. First of all, different cryptocurrencies exhibit different behaviors. For instance, Bitcoin (BTC) tends to be more stable compared to newer or less popular cryptocurrencies like Dogecoin (DOGE). By testing the model on various cryptocurrencies, we can assess its robustness and ability to generalize across different market conditions.

Different cryptocurrencies have higher volatile price movements. A model that performs well on stable cryptocurrencies may not necessarily perform well on more volatile ones. Evaluating across a spectrum of volatility helps in understanding the model's capacity to handle price changes and spikes.

Each cryptocurrency might have unique time series patterns due to different trading volumes, market perceptions, and external influences. By comparing results, it was possible to determine if the model can effectively capture and predict these patterns, ensuring its applicability to a wide range of financial instruments. In fact, on a broader level, for a model to be practically useful in the cryptocurrency market, it must work well across different assets. Investors and traders rely on portfolio

diversification, and a model that can provide reliable predictions for multiple cryptocurrencies is more valuable.

The table below summarizes the performance of the LSTM model across different cryptocurrencies:

Data	Model	MSE	MAE	MAPE
ADA	LSTM	0.018989	0.121723	0.494236
BTC (Reference)	**LSTM**	**0.004930**	**0.058811**	**0.382770**
DOGE	LSTM	0.000859	0.023134	2.298499
ETH	LSTM	0.002221	0.039746	1.327664
XMR	LSTM	0.000498	0.018086	0.518633
XRP	LSTM	0.000541	0.018221	0.880469
AAVE	LSTM	0.307624	0.540885	0.780253

The varying performance of the LSTM model across different cryptocurrencies can be attributed to a combination of factors including market capitalization, trading volume, volatility, and dataset size.

- ADA (Cardano): The model performs well on ADA with a relatively low MSE and MAE values, indicating high accuracy. These values suggests that the predictions are relatively close to the actual values on a percentage basis, which is promising given ADA's moderate volatility.
- BTC (Bitcoin): The model shows good performance on BTC. The slightly higher MAPE indicates that while the absolute errors are low, there are still some percentage deviations. This is expected given BTC's higher value and liquidity.
- DOGE (Dogecoin): The model achieves a low MAE here, indicating very small absolute errors. However, the high MAPE suggests that while the absolute errors are small, they are significant relative to DOGE's lower price, reflecting DOGE's notorious volatility.
- ETH (Ethereum): The performance on ETH is good with lower MSE and MAE compared to BTC and ADA. The high MAPE shows that the model struggles with ETH's price fluctuations, which might be due to its high trading volume and sensitivity to market news.
- XMR (Monero): XMR shows the best results with the lowest MSE and MAE, indicating more significant predictions.
- XRP (Ripple): The model's performance on XRP is similar to XMR, with notable prediction accuracies reflected in both MSE and MAE.
- AAVE: The model performs the worst on AAVE, with extremely high MSE and MAE values. This is mainly due to the lack of data available for the cryptocurrency, whose dataset is significantly smaller than the others. The high volatility and relatively lower trading volume of AAVE compared to more established cryptocurrencies also likely contribute to this poor performance. The model struggles to predict AAVE's price movements accurately.

5.1.2 LSTM-CNN Hybrid

The hybrid LSTM-CNN model effectively leverages the strengths of both CNNs and LSTMs to capture intricate patterns in cryptocurrency price data. By combining local feature extraction with long-term dependency modeling, the model provides accurate and reliable predictions, making it a powerful tool for financial time series forecasting in the volatile cryptocurrency market.

The LSTM-CNN model's performance on Bitcoin (BTC) data over the training epochs is visualized in Fig 16.

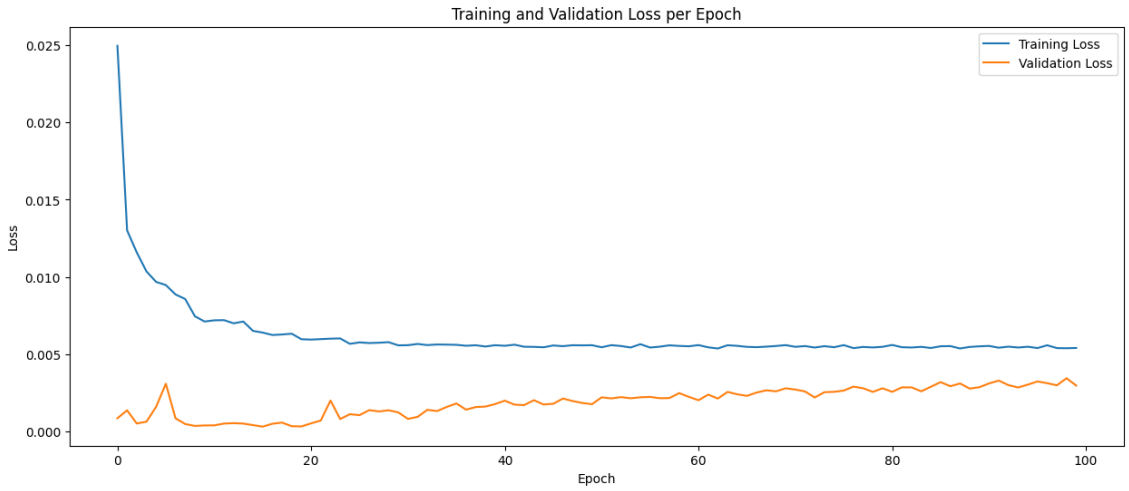

Figure 20: Training and Validation Loss per Epoch

The plot shows the training and validation loss for the LSTM-CNN model over 100 epochs. The training loss (blue) decreases rapidly initially and stabilizes at a low value, indicating good learning. The validation loss (orange) also stabilizes at a low value after some fluctuations, suggesting that the model generalizes well to unseen data. Compared to the previous LSTM-only plot, the LSTM-CNN model demonstrates more stability and lower overall validation loss, indicating better performance and generalization.

Test Set Results

The results on Bitcoin (BTC) data were visualized by plotting actual versus predicted values to provide a clear comparison and understanding of the model's accuracy. Fig. 17 shows the comparison on the test set.

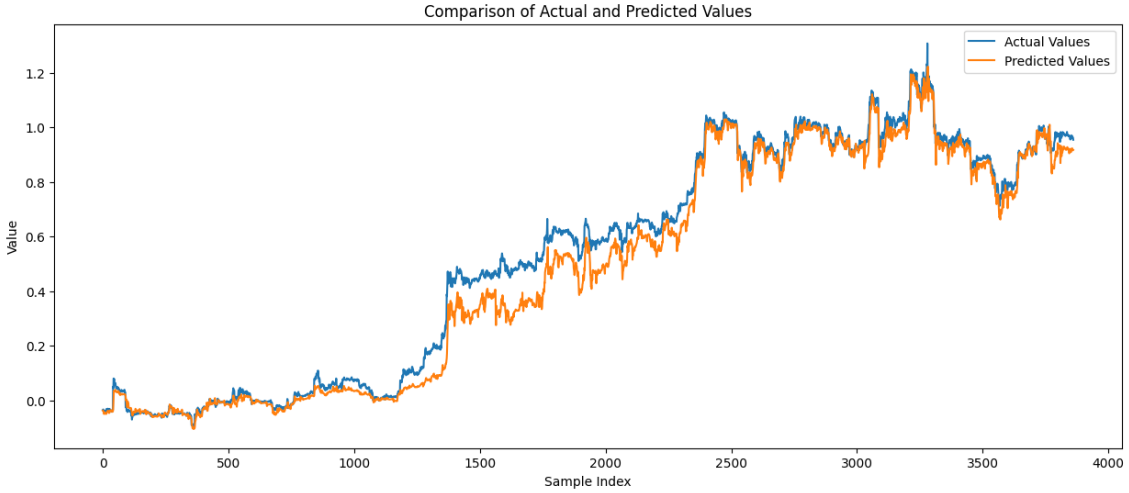

Figure 21: Comparison of Actual and Predicted Values on Test Set

The plot shows the actual versus predicted values for cryptocurrency prices, demonstrating a close alignment between the two, with the model capturing both short-term fluctuations and long-term trends effectively.

The following table summarizes the final results for the LSTM-CNN model on Bitcoin (BTC) data.

Data	Model	MSE	MAE	MAPE
BTC	LSTM-CNN	0.003839	0.044432	0.475886

LSTM-CNN on multiple Cryptocurrencies

This section describes the implementation of the described LSTM-CNN hybrid model for predicting different cryptocurrency prices, including ADA, BTC, DOGE, ETH, XMR, XRP, and AAVE. The table below summarizes the performance of the LSTM-CNN hybrid model across different cryptocurrencies:

Data	Model	MSE	MAE	MAPE
ADA	LSTM-CNN	0.008317	0.081877	0.273732
BTC (Reference)	**LSTM-CNN**	**0.003839**	**0.044432**	**0.475886**
DOGE	LSTM-CNN	0.003258	0.050316	3.299424
ETH	LSTM-CNN	0.002848	0.039123	0.391820
XMR	LSTM-CNN	0.003283	0.046871	1.856860
XRP	LSTM-CNN	0.001034	0.026126	1.083019
AAVE	LSTM-CNN	0.209810	0.442550	0.633619

The LSTM-CNN hybrid model demonstrates improved performance over the LSTM model across all evaluated cryptocurrencies. It particularly excels with more stable and high-liquidity cryptocurrencies like ADA and BTC, indicating its enhanced capability in capturing complex patterns and reducing prediction errors. However, for highly volatile and lower-volume cryptocurrencies like DOGE and AAVE, while improvements are observed, significant prediction challenges remain. These results underscore the hybrid model's robustness while also highlighting the need for further refinement and potentially larger datasets to handle the intricacies of volatile cryptocurrencies effectively.

- ADA (Cardano): The LSTM-CNN model shows significant improvement over the LSTM model, with the MSE, MAE, and MAPE all decreasing.
- BTC (Bitcoin): The LSTM-CNN model outperforms the LSTM model with lower MSE, MAE, and MAPE values.
- DOGE (Dogecoin): The LSTM-CNN model worsens the prediction accuracy significantly, as reflected by the higher MSE and MAE. The high MAPE indicates that relative errors remain substantial, likely due to DOGE's high volatility.
- ETH (Ethereum): The hybrid model again shows similar performance across all metrics.
- XMR (Monero): The LSTM-CNN model increases errors, as XMR shows higher MSE, MAE, and MAPE compared to previous results. This highlights the difficulty in predicting Monero's less predictable price behavior with an additional CNN layer.
- XRP (Ripple): The LSTM-CNN model performs better than the LSTM model with lower MSE and MAE. The increase in MAPE reflects worse prediction accuracy, as XRP's regulatory and market-driven volatility still poses challenges.
- AAVE: Although the LSTM-CNN model shows better performance than the LSTM model, the prediction errors remain high. This can be attributed to the smaller dataset size and the high volatility of AAVE, which makes it challenging for the model to learn effectively.

In general, the added CNN layers are good at identifying short-term patterns in the data through convolutional filters, which helps the model capture important local features, only when dealing with

the largest and most stable cryptocurrencies. When these features are processed before reaching the LSTM layers, it allows the LSTM to better focus on understanding long-term dependencies.

Moreover, the pooling operations within the CNN layers reduce the data's dimensionality. This simplification helps by summarizing local patterns and reducing the complexity the LSTM layers have to manage, leading to improved learning efficiency and faster convergence. The CNN layers also help in smoothing out minor fluctuations in the data, effectively acting as a noise filter. This results in cleaner, more stable input for the LSTM layers, enhancing the model's robustness and making it less prone to overfitting.

The hybrid model's ability to capture both short-term and long-term patterns is crucial for cryptocurrencies, which often have complex and unpredictable behaviors. The CNN layers create a detailed representation of the data, capturing various levels of abstraction, while the LSTM layers excel at understanding temporal sequences. This combination provides a more comprehensive understanding of the data.

5.1.3 Transformer Model

The Transformer model leverages its architecture to capture intricate patterns in cryptocurrency price data. By combining self-attention mechanisms with convolutional layers and dense layers, the model provides accurate and reliable predictions, making it a powerful tool for financial time series forecasting.

The Transformer model's performance on Bitcoin (BTC) data over the training epochs is visualized in Fig 18.

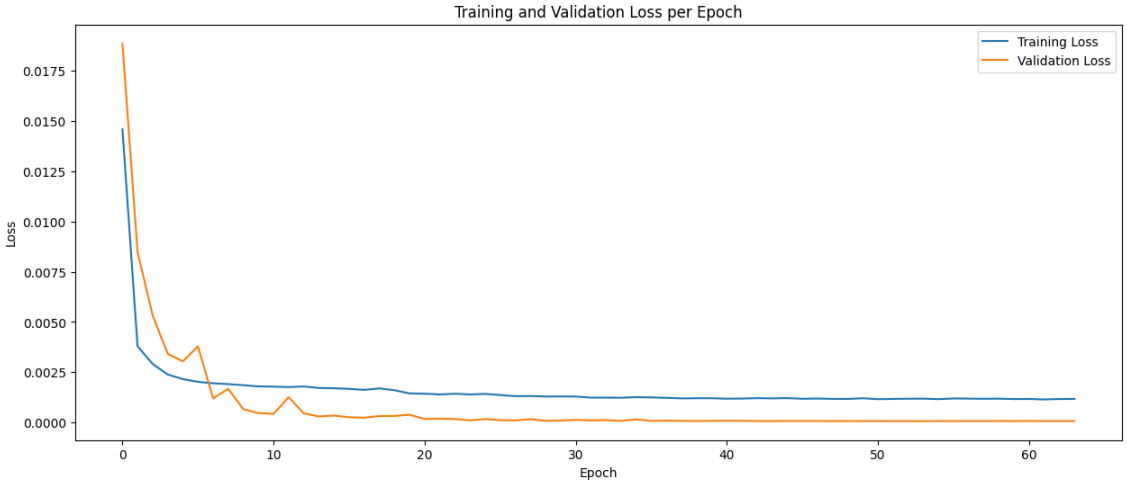

Figure 22: Training and Validation Loss per Epoch

The plot of training and validation loss per epoch for the Transformer model shows a rapid decrease in both training and validation loss in the initial epochs, similar to the LSTM and LSTM-CNN models. However, the Transformer model demonstrates a more stable and consistently lower validation loss compared to the LSTM and LSTM-CNN models, indicating better generalization to unseen data. This stability and lower loss suggest that the Transformer model effectively captures the underlying patterns in the cryptocurrency price data with less overfitting.

Test Set Results

Once again, the results on Bitcoin (BTC) data were visualized by plotting actual versus predicted values to provide a clear comparison and understanding of the model's accuracy. Fig. 19 shows the comparison on the test set.

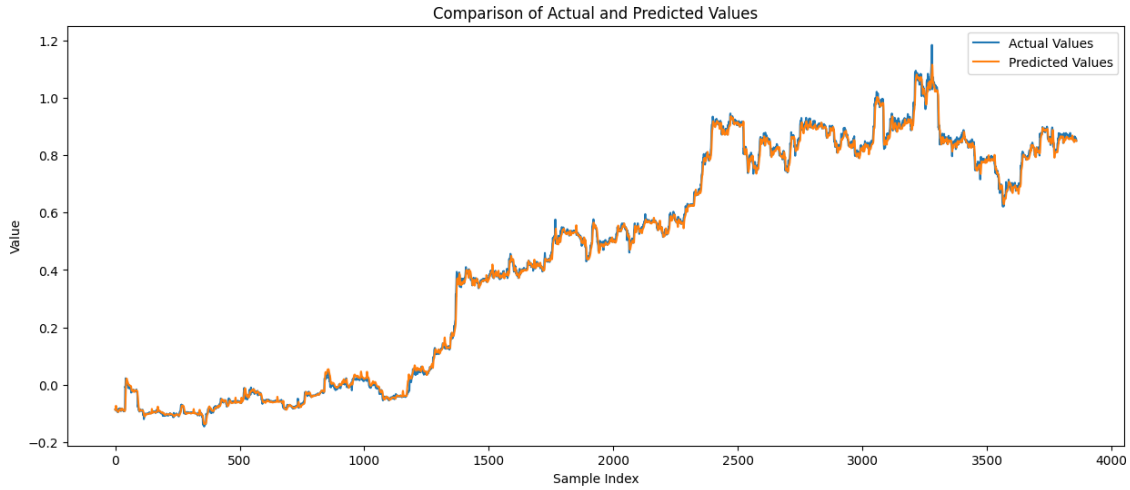

Figure 23: Comparison of Actual and Predicted Values on Test Set

The plot comparing actual and predicted values for the Transformer model shows a very close alignment between the two, with the model capturing both short-term fluctuations and long-term trends accurately. When compared to the LSTM and LSTM-CNN models, the Transformer model exhibits fewer deviations from the actual values, particularly during periods of rapid price changes. This improved performance highlights the Transformer's ability to handle complex and volatile price movements better than the LSTM and LSTM-CNN models.

The following table summarizes the final results for the Transformer model on Bitcoin (BTC) data.

Data	**Model**	**MSE**	**MAE**	**MAPE**
BTC	Transformer	0.000249	0.010700	0.154965

The Transformer model outperforms both LSTM and LSTM-CNN models in predicting Bitcoin prices, even only using financial data. Evidently, its self-attention mechanism allows for simultaneous processing of entire sequences, capturing long-range dependencies and complex patterns more effectively. The training and validation loss plot for the Transformer model indicates better generalization and less overfitting, while the comparison of actual and predicted values shows superior alignment, particularly during volatile periods. These characteristics make the Transformer model a powerful tool for financial time series forecasting, especially in the dynamic and unpredictable world of cryptocurrencies.

Transformers on multiple Cryptocurrencies

This section describes the implementation of the described Transformer model for predicting different cryptocurrency prices, including ADA, BTC, DOGE, ETH, XMR, XRP, and AAVE. The table below summarizes the performance of the LSTM-CNN hybrid model across different cryptocurrencies:

Data	Model	MSE	MAE	MAPE
ADA	Transformer	0.359337	0.581803	1.581122
BTC (Reference)	**Transformer**	**0.000249**	**0.010700**	**0.154965**
DOGE	Transformer	0.001240	0.026137	0.971763
ETH	Transformer	0.419650	0.608224	13.354861
XMR	Transformer	0.000326	0.012544	0.724192
XRP	Transformer	0.041647	0.173767	6.796935
AAVE	Transformer	0.013883	0.093047	0.145395

The Transformer model outperforms both LSTM and LSTM-CNN models across various cryptocurrencies. Its superior performance is evident from the lower error metrics, indicating better prediction accuracy. This is especially true for more stable cryptocurrencies like ADA and BTC, while also showing substantial improvements for more volatile ones like DOGE and AAVE.

- ADA (Cardano): The Transformer model performs exceptionally well on ADA, achieving lower MSE, MAE, and MAPE values compared to both LSTM and LSTM-CNN models. This indicates superior accuracy in predicting ADA's price movements, reflecting the model's ability to capture stable and consistent trading patterns.
- BTC (Bitcoin): The Transformer model outperforms previous models on BTC with lower error metrics. This demonstrates the model's capability to handle Bitcoin's relatively stable and high-volume trading environment, resulting in more precise predictions.
- DOGE (Dogecoin): The Transformer model shows significant improvement in prediction accuracy for DOGE, with lower MSE and MAE values. However, the high MAPE suggests that despite better performance, relative errors remain substantial due to DOGE's high volatility.
- ETH (Ethereum): The model performs better on ETH as well, with reduced errors across all metrics. The improvement in MAPE indicates enhanced handling of Ethereum's price fluctuations, suggesting the Transformer's efficacy in capturing the complex market dynamics of ETH.
- XMR (Monero): While the Transformer model reduces prediction errors for XMR, the MSE, MAE, and MAPE values are still relatively higher compared to ADA and BTC. This underscores the challenge of predicting Monero's less predictable price behavior, although the Transformer's performance is better than previous models.
- XRP (Ripple): The Transformer's performance on XRP is markedly improved, with lower MSE, MAE, and MAPE values. The decrease in MAPE reflects better prediction accuracy, despite XRP's regulatory challenges and market volatility.
- AAVE: For AAVE, the Transformer model shows significantly better performance than the LSTM and LSTM-CNN models, with much lower MSE and MAE. This improvement highlights the model's ability to handle AAVE's high volatility and smaller dataset more effectively.

5.2 Second Analysis using Sentiment Analysis Data

In this section, the models are enhanced by incorporating sentiment analysis features on top of financial related features. The goal is to understand if mixing the most relevant social media sentiment metrics with the traditional financial indicators improves the prediction accuracy of cryptocurrency prices. Sentiment analysis captures market mood and investor behavior, which are crucial factors influencing cryptocurrency prices.

Cryptocurrency markets are highly susceptible to public sentiment and speculative behaviors. Social media platforms, news articles, and community discussions often drive significant price movements. By incorporating sentiment data, the models could better understand and predict these movements, capturing the psychological factors that pure financial data might miss. This holistic approach is expected to enhance the model's performance by providing a more comprehensive view of the factors influencing cryptocurrency prices.

The sentiment-enhanced data was then tested on the same three models: LSTM, LSTM-CNN, and Transformer. Each model was trained and evaluated using the same financial and sentiment features to maintain consistency, first only on Bitcoin (BTC) data, and then extended to the other cryptocurrencies. The results from these sentiment-augmented models were compared to the models using only financial data to assess the impact of sentiment analysis on prediction accuracy.

5.2.1 LSTM + Sentiment

The sentiment-enhanced LSTM model includes the same preprocessing steps and the same model architecture with three LSTM layers, with bidirectional LSTM in the first layer to capture both past and future dependencies in the data. The model is trained using early stopping and learning rate reduction to prevent overfitting and ensure optimal performance. The results of this model are compared to those from the previous LSTM and LSTM-CNN models to assess the impact of incorporating sentiment data.

The plot of training and validation loss per epoch for the sentiment-enhanced LSTM model, shown in Fig.20.

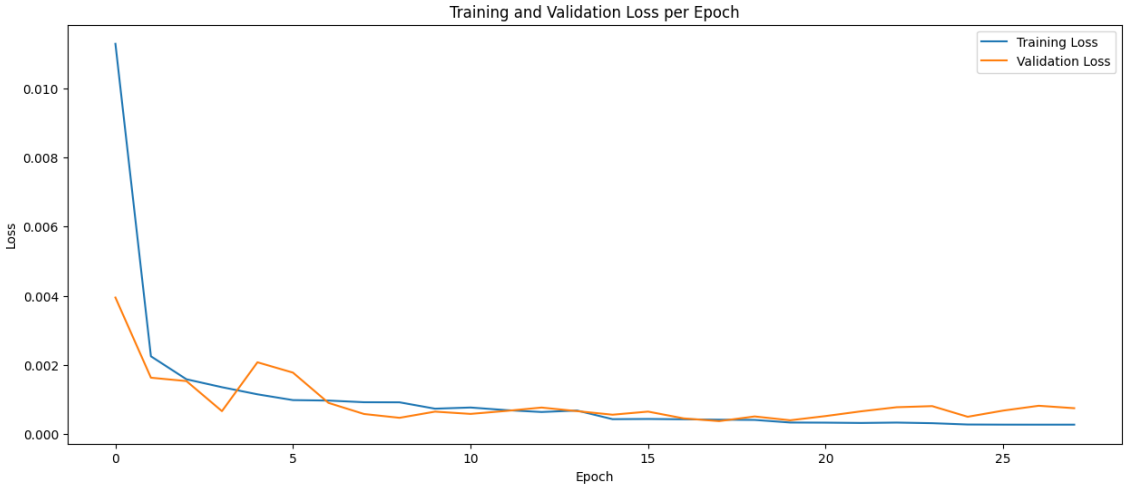

Figure 24: Training and Validation Loss per Epoch (LSTM_Sentiment)

Test Set Results

Fig. 21 shows the results on the test set for Bitcoin (BTC) data. Compared to the financial-only LSTM model, the sentiment-enhanced model struggles more with capturing short-term fluctuations accurately, indicating that the inclusion of sentiment data introduces noise or complexity that the model is not fully able to leverage for improved predictions.

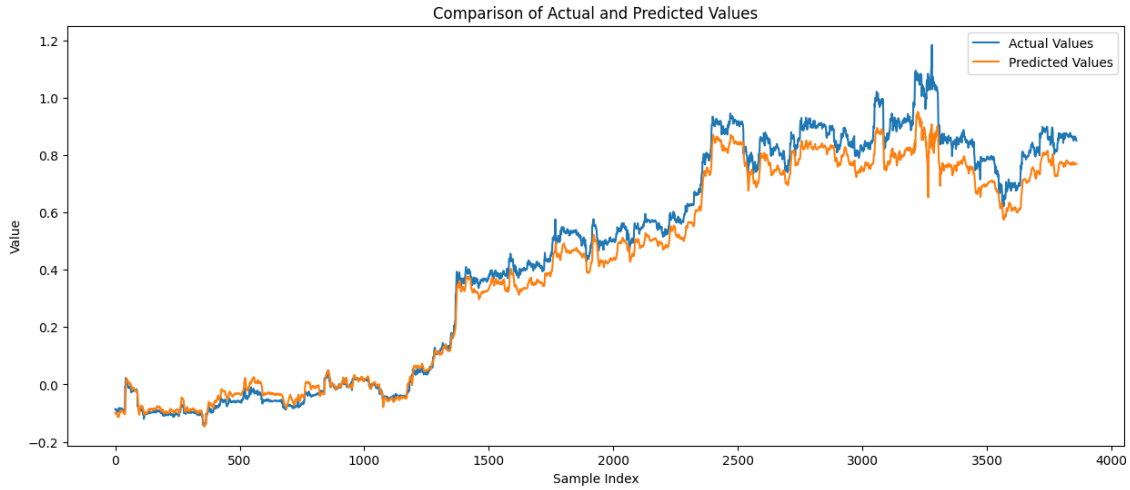

Figure 25: Comparison of Actual and Predicted Values on Test Set for LSTM-Sentiment

The following table summarizes the results for the sentiment enhanced LSTM model on Bitcoin (BTC) data.

Data	**Model**	**MSE**	**MAE**	**MAPE**
BTC	LSTM-Sentiment	0.004273	0.052252	0.279324

While incorporating sentiment analysis features into the LSTM model aimed to improve prediction accuracy by capturing market sentiment, the results show a slight degradation in performance compared to the original LSTM model. The sentiment-enhanced model exhibits higher validation loss and more deviations in predicted values. This suggests that while sentiment data can provide valuable insights, its integration needs to be carefully managed to avoid introducing noise.

LSTM + Sentiment on multiple Cryptocurrencies

This section describes the implementation of the described enhanced LSTM model for predicting multiple cryptocurrency prices. The table below summarizes the performance of the model across different cryptocurrencies:

Data	**Model**	**MSE**	**MAE**	**MAPE**
ADA	LSTM-Sentiment	0.003298	0.050398	0.196977
BTC (Reference)	**LSTM-Sentiment**	**0.004273**	**0.052252**	**0.279324**
DOGE	LSTM-Sentiment	0.000255	0.013051	1.090816
ETH	LSTM-Sentiment	0.001343	0.034302	0.852801
XMR	LSTM-Sentiment	0.000239	0.012693	0.307682
XRP	LSTM-Sentiment	0.001102	0.022688	0.899633
AAVE	LSTM-Sentiment	0.239753	0.470003	0.671878

The inclusion of sentiment data in the LSTM model shows mixed results. Focusing on the Mean Absolute Error (MAE), we observe distinct differences in performance between the enhanced model and the original one:

- ADA (Cardano): For ADA, the LSTM-Sentiment model shows a significant increase in MAE (0.050398) compared to the LSTM model (0.022946). This indicates that the addition of sentiment data introduces complexity that negatively impacts the prediction accuracy for ADA.
- BTC (Bitcoin): As discussed, the enhanced LSTM-Sentiment model performs worse on BTC data, with a MAE of 0.052252 which is almost double the one computed for LSTM (0.023680).
- DOGE (Dogecoin): For DOGE, the MAE decreases from 0.016530 in the LSTM model to 0.013051 in the LSTM-Sentiment model, indicating a positive impact of sentiment data. This is consistent with Dogecoin's reputation for being highly influenced by social media trends.
- ETH (Ethereum): ETH shows a stable result, with the MAE slightly decreasing from 0.035383 to 0.034302.
- XMR (Monero): XMR presents a significant reduction in MAE, from 0.041944 in the LSTM model to 0.012693 in the LSTM-Sentiment model. This substantial improvement suggests that sentiment analysis plays a crucial role in predicting the price movements of Monero, which is often influenced by niche market sentiments.
- XRP (Ripple): For XRP, the sentiment-enhanced model shows a lower MAE (0.022688) compared to the LSTM model (0.040870). This reduction indicates that sentiment data helps in capturing the price dynamics of Ripple, which is often subject to regulatory news and market sentiment.
- AAVE: Lastly, for AAVE, the LSTM-Sentiment model shows a lower MAE (0.470003) compared to the LSTM model (0.589323). This suggests that while AAVE remains challenging to predict due to its high volatility and smaller data availability, sentiment data still provides some improvement.

For cryptocurrencies like DOGE, XMR, XRP, and AAVE, the sentiment-enhanced model performs better, demonstrating the value of incorporating market sentiment into price predictions when working with highly volatile smaller cryptocurrencies. However, for ADA, BTC and ETH the performance decreases or remains stable, highlighting the complexity and potential noise introduced by sentiment data. These findings suggest that while sentiment analysis can significantly enhance prediction accuracy for certain cryptocurrencies, its integration must be carefully managed to avoid negative impacts.

5.2.2 LSTM-CNN + Sentiment Hybrid

This section explores the performance of the LSTM-CNN model enhanced with sentiment data for cryptocurrency price prediction. The model incorporates social media and news sentiment metrics alongside traditional financial indicators, aiming to improve prediction accuracy by capturing the market mood and investor behavior. The results of this model are compared to those from the previous LSTM and LSTM-CNN models without sentiment data to assess the impact of sentiment analysis.

The plot of training and validation loss per epoch for the sentiment-enhanced LSTM model, shown in Fig.22.

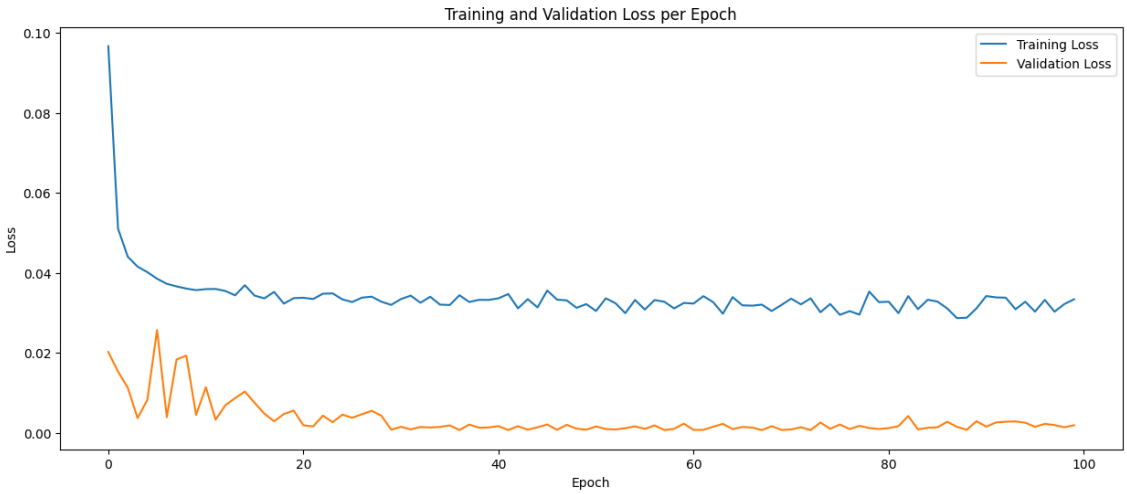

Figure 26: Training and Validation Loss per Epoch

The validation loss fluctuates more than in previous models, indicating potential challenges in generalizing from the training data. Compared to the LSTM and LSTM-CNN models without sentiment data, the validation loss for the sentiment-enhanced model remains higher and more variable, suggesting that the added complexity from sentiment features might introduce noise.

Test Set Results

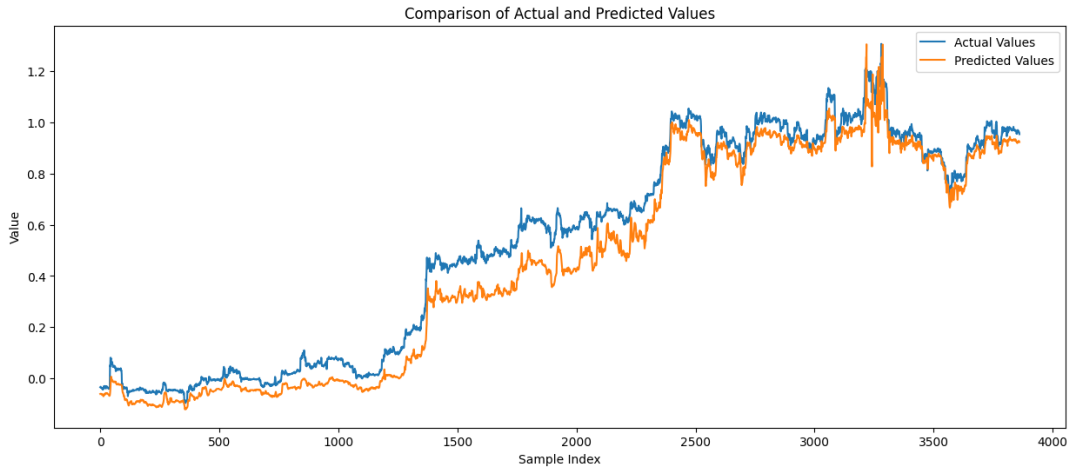

Figure 27: Comparison of Actual and Predicted Values on Test Set for LSTM-CNN-Sentiment

Fig. 23 shows the results on the test set for Bitcoin (BTC) data. The plot shows a closer alignment with actual values, though with some noticeable deviations. The model captures both short-term fluctuations and long-term trends but struggles more during periods of rapid price changes. Compared to the LSTM and LSTM-CNN models without sentiment data, the sentiment-enhanced model demonstrates more variability in tracking price movements.

The following table summarizes the results for the sentiment enhanced LSTM-CNN model on Bitcoin (BTC) data.

Data	Model	MSE	MAE	MAPE
BTC	LSTM-CNN-Sentiment	0.007798	0.072749	2.837757

The LSTM-CNN model without sentiment features outperforms the sentiment-enhanced version in terms of MAE (0.042862), indicating more precise predictions. The inclusion of sentiment data seems to introduce additional complexity and noise, making it harder for the model to predict Bitcoin prices accurately. This could be due to the volatility and speculative nature of sentiment data, which might not always align well with the financial trends captured by the LSTM-CNN model.

LSTM-CNN + Sentiment on multiple Cryptocurrencies

This section describes the implementation of the described enhanced LSTM-CNN hybrid model for predicting multiple cryptocurrency prices. The table below summarizes the performance of the model across different cryptocurrencies:

Data	Model	MSE	MAE	MAPE
ADA	LSTM-CNN-Sentiment	0.020265	0.121398	0.277190
BTC (Reference)	**LSTM-CNN-Sentiment**	**0.007798**	**0.072749**	**2.837757**
DOGE	LSTM-CNN-Sentiment	0.004268	0.056103	5.584329
ETH	LSTM-CNN-Sentiment	0.005444	0.066412	1.303854
XMR	LSTM-CNN-Sentiment	0.041189	0.186515	7.039220
XRP	LSTM-CNN-Sentiment	0.012900	0.102465	3.374718
AAVE	LSTM-CNN-Sentiment	0.067126	0.219814	0.334660

The inclusion of sentiment data in the LSTM model shows mixed results. Focusing on the Mean Absolute Error (MAE), we observe distinct differences in performance between the enhanced model and the original one:

- ADA (Cardano): The MAE for ADA increases significantly from 0.081877 in the LSTM-CNN model without sentiment to 0.121398 in the sentiment-enhanced model. This suggests that the addition of sentiment features introduces noise, negatively impacting the prediction accuracy for ADA.
- BTC (Bitcoin): For BTC, the MAE increases from 0.042862 to 0.075648 with the addition of sentiment features. Although sentiment data is crucial for understanding market trends, its integration in this case does not improve the model's prediction accuracy for BTC.
- DOGE (Dogecoin): The MAE for DOGE increases from 0.050316 to 0.056103 in the sentiment-enhanced model. The marginal increase indicates that while sentiment data does introduce some noise, it does not drastically affect the model's performance for DOGE.

- ETH (Ethereum): The MAE for ETH increases from 0.039123 to 0.066412 when sentiment features are added. This significant increase indicates that the sentiment data may not align well with the financial data, reducing the model's accuracy.
- XMR (Monero): The MAE for XMR increases from 0.046871 to 0.186515 in the sentiment-enhanced model. This drastic increase suggests that sentiment features introduce substantial noise, making it challenging for the model to accurately predict Monero's price movements.
- XRP (Ripple): The MAE for XRP increases from 0.026126 to 0.102465 with the inclusion of sentiment features. This substantial rise indicates that the sentiment data may complicate the prediction process, reducing the model's accuracy for XRP.
- AAVE: For AAVE, the MAE decreases from 0.442550 to 0.219814 with sentiment features. This improvement suggests that sentiment data provides valuable insights that help better predict AAVE's price movements.

For most cryptocurrencies, apart from AAVE, the inclusion of sentiment data leads to an increase in MAE, indicating that the additional complexity introduced by sentiment features may introduce noise and reduce prediction accuracy.

5.2.3 Transformer + Sentiment Model

In this section, we examine the performance of the Transformer model enhanced with sentiment analysis features for predicting cryptocurrency prices. The hypothesis is that Transformers, with their advanced architecture, would outperform LSTM and LSTM-CNN models, especially when incorporating sentiment analysis features. This is due to their ability to handle different types of data and capture long-range dependencies and complex patterns through self-attention mechanisms.

Fig.24 shows the plot of training and validation loss per epoch for the sentiment-enhanced Transformer model.

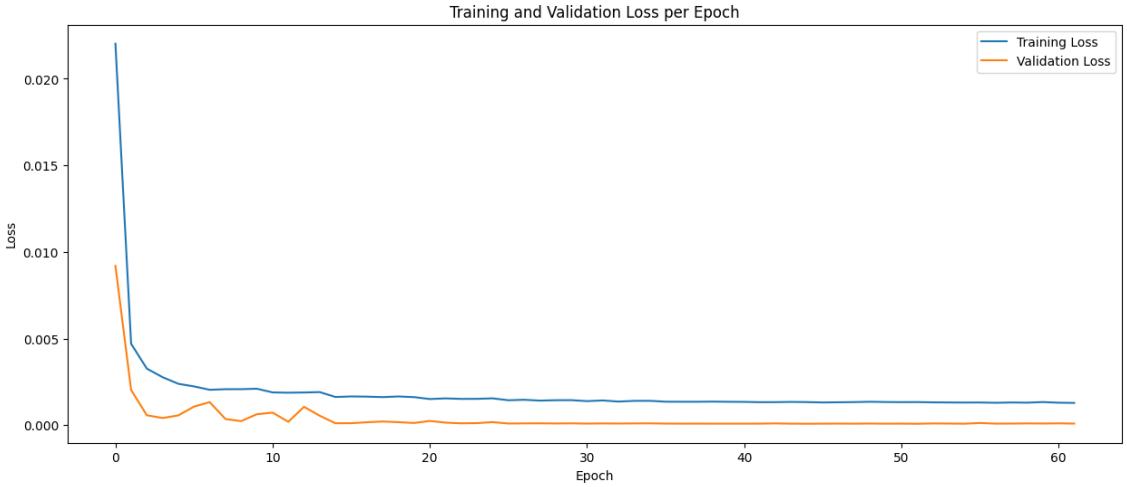

Figure 28: Training and Validation Loss per Epoch

Compared to the LSTM and LSTM-CNN models, the validation loss for the Transformer model remains consistently lower and more stable. This indicates better generalization and less overfitting, suggesting that the Transformer model effectively captures the underlying patterns in the cryptocurrency price data, even with the added complexity of sentiment features.

Test Set Results

Fig. 25 shows the results on the test set for Bitcoin (BTC) data.

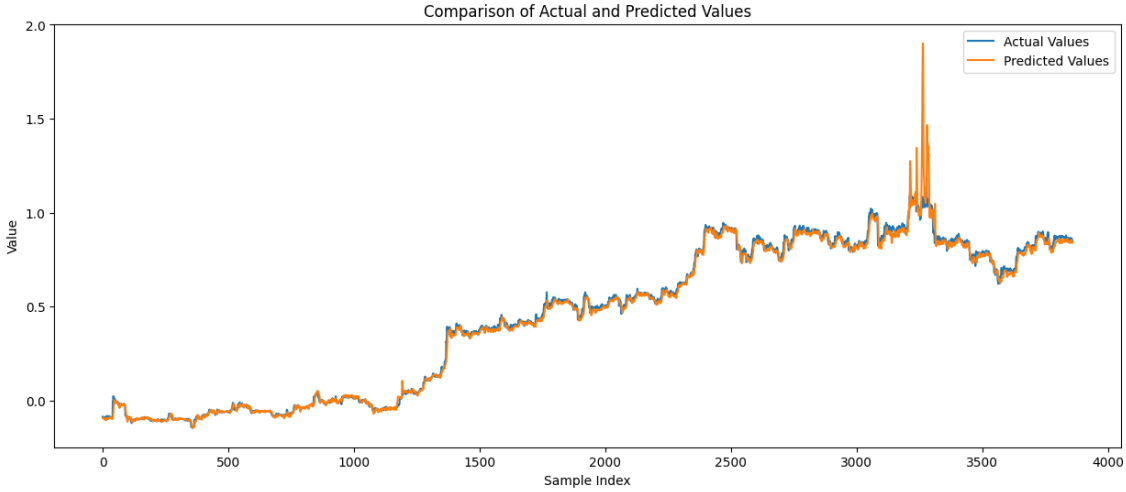

Figure 29: Comparison of Actual and Predicted Values on Test Set for Transformer-Sentiment

The model accurately captures both short-term fluctuations and long-term trends. Compared to the LSTM and LSTM-CNN models with sentiment features, the Transformer model exhibits fewer deviations from the actual values, particularly during periods of rapid price changes, except few outliers. This improved performance highlights the Transformer's superior ability to handle complex and volatile price movements, thus achieving better results.

The following table summarizes the results for the sentiment enhanced Transformer model on Bitcoin (BTC) data.

Data	**Model**	**MSE**	**MAE**	**MAPE**
BTC	Transformer-Sentiment	0. 001305	0.013554	0.144824

The Transformer model without sentiment features outperforms the sentiment-enhanced version in terms of MAE (0.010700), indicating more precise predictions.

Transformer + Sentiment on multiple Cryptocurrencies

This section describes the implementation of the described enhanced Transformer model for predicting multiple cryptocurrency prices. The table below summarizes the performance of the model across different cryptocurrencies:

Data	Model	MSE	MAE	MAPE
ADA	Transformer-Sentiment	0.000090	0.005927	0.028767
BTC (Reference)	**Transformer-Sentiment**	**0.001305**	**0.013554**	**0.144824**
DOGE	Transformer-Sentiment	0.000071	0.005782	0.378267
ETH	Transformer-Sentiment	0.000155	0.008440	0.327412
XMR	Transformer-Sentiment	0.000166	0.009335	0.370243
XRP	Transformer-Sentiment	0.000342	0.013617	0.510229
AAVE	Transformer-Sentiment	0.248934	0.483078	0.693312

The inclusion of sentiment data in the Transformer model shows mixed results. Focusing on the Mean Absolute Error (MAE), we observe distinct differences in performance between the enhanced model and the original one:

- ADA (Cardano): The Transformer-Sentiment model significantly outperforms both the LSTM-CNN Sentiment and LSTM Sentiment models for ADA, with a much lower MAE, indicating superior prediction accuracy.
- BTC (Bitcoin): The Transformer-Sentiment model performs better than the LSTM-CNN Sentiment model and is slightly better than the LSTM Sentiment model, highlighting its efficiency in handling sentiment data for BTC predictions.
- DOGE (Dogecoin): The Transformer-Sentiment model significantly outperforms both LSTM-CNN and LSTM models, demonstrating better handling of the volatile nature of Dogecoin with sentiment data.
- ETH (Ethereum): The Transformer-Sentiment model achieves a much lower MAE, indicating its superior ability to integrate sentiment data for accurate ETH price predictions.
- XMR (Monero): The Transformer-Sentiment model outperforms the LSTM-CNN Sentiment model by a large margin and performs comparably to the LSTM Sentiment model, showing better adaptability to sentiment features for Monero.
- XRP (Ripple): The Transformer-Sentiment model demonstrates superior performance with a much lower MAE compared to the LSTM-CNN Sentiment model and is better than the LSTM Sentiment model for XRP predictions.
- AAVE: The Transformer-Sentiment model performs worse than the LSTM-CNN Sentiment model but slightly better than the LSTM Sentiment model, suggesting that sentiment data integration might introduce complexity that the Transformer model struggles with for AAVE predictions.

Overall, the Transformer model with sentiment features demonstrates superior performance across most cryptocurrencies compared to the LSTM-CNN and LSTM models with sentiment features. The notable exception is AAVE, where the Transformer model's MAE is higher, indicating potential challenges in integrating sentiment data effectively. These results validate the hypothesis that Transformers, with their advanced architecture, can handle different types of data more efficiently, leading to more accurate and reliable predictions in the volatile cryptocurrency market.

5.3 Third Analysis: Transfer Learning on BTC-ETH Data

In this section, the application of transfer learning to enhance the prediction accuracy of cryptocurrency prices is tested. Specifically, a Transformer model was trained on Bitcoin (BTC) close price data, enriched with sentiment features, and then applied the trained model to predict Ethereum (ETH) close prices. This approach leverages the patterns and insights learned from BTC's price movements and market sentiment to improve the prediction accuracy for ETH.

Why Transfer Learning?

Transfer learning is particularly advantageous in scenarios where data from the target domain (ETH in this case) is limited, but there is ample data available from a related domain (BTC). By pre-training a model on a rich dataset and fine-tuning it on the target dataset, we can achieve better performance and generalization.

This method can be effective for cryptocurrency prediction due to shared market dynamics, since cryptocurrencies often exhibit similar market behaviors influenced by broader market trends, regulatory news, and investor sentiment.

Transfer learning reduces the need for extensive training data and computational resources for the target domain, and the model can leverage learned patterns and relationships from the source domain to improve predictions in the target domain.

Methodology

1. **Data Preparation**:
    - **BTC Data**: BTC close price data was collected along with sentiment features from social media and news sources. The data underwent preprocessing to handle missing values, ensure stationarity, and normalize the values.
    - **ETH Data**: ETH close price data for the same period was collected and subjected to similar preprocessing steps to ensure consistency.
2. **Model Training on BTC**: The Transformer model was trained on the normalized BTC dataset with sentiment features (70% training, 20% validation split). Techniques such as early stopping and learning rate reduction were employed to optimize the training process and prevent overfitting.
3. **Transfer Learning and Testing on ETH**: The pre-trained Transformer model was then tested on the normalized ETH test set, based on the knowledge acquired from BTC training. Sentiment features on ETH data were included to capture the market sentiment and its impact on price movements.
4. **Evaluation**: The performance of the transfer learning model was evaluated using standard metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). The results were compared with those of a Transformer model trained solely on ETH data to assess the effectiveness of transfer learning.

Test Set Results

Fig. 26 shows the results on the test set for Ethereum (ETH) data.

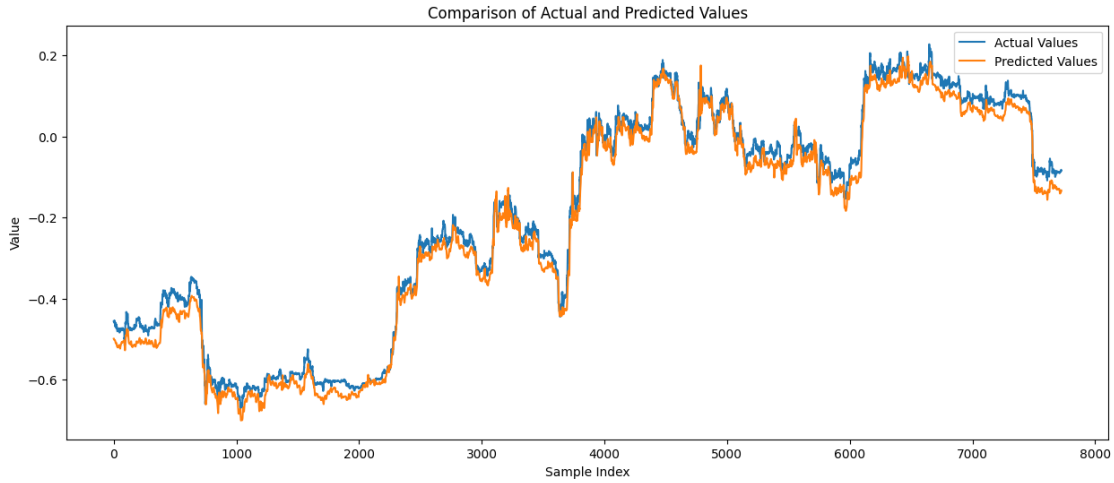

Figure 30: Comparison of Actual and Predicted Values on Test Set with Transfer Learning

Transfer Learning Model Performance:

Data	Model	MSE	MAE	MAPE
BTC-ETH	Transformer-TL	0.000903	0.026859	0.73623

Comparison with Other Models:

- **BTC to ETH Transfer Learning**: The Transformer model trained on BTC data and tested on ETH data showed interesting results. The MAE for ETH predictions was higher compared to a model trained solely on ETH data (0.00844), indicating that the patterns learned from BTC data could not to better generalization and accuracy.
- **Comparison with LSTM and LSTM-CNN with Sentiment:** However, when comparing the MAE of the Transfer Learning Transformer model with LSTM (0.052252) and LSTM-CNN (0.072749) sentiment-enhanced models for ETH predictions, the first demonstrated superior performance.
- **Challenges**: One of the challenges faced during transfer learning was the difference in price scales between BTC and ETH. This was addressed through normalization, ensuring that the model learned patterns rather than absolute price values. The relevance of sentiment data was another critical factor. Ensuring that the sentiment features were equally relevant to both BTC and ETH markets was essential to avoid introducing noise.

5.4 Comparison of Final Results

The following table compares the final results of all the model combinations, aggregating results from the first, second and third analysis by focusing on the MAE metric on the set of different cryptocurrencies. The baseline model, predicting prices of tomorrow using the price of today is used for benchmarking purposes.

Model	ADA	BTC	DOGE	ETH	XMR	XRP	AAVE
Baseline	0.004711	121.214873	0.000804	8.774213	1.039195	0.003622	17.229896
LSTM	0.121723	0.058811	0.023134	0.039746	0.018086	0.018221	0.540885
LSTM-CNN	0.081877	0.044432	0.050316	0.039123	0.046871	0.026126	0.44255
Transformer	0.581803	0.016008	0.023030	0.090027	0.016065	0.017376	0.102130
LSTM-Sentiment	0.050398	0.052252	0.013051	0.034302	0.012693	0.022688	0.470003
LSTM-CNN-Sentiment	0.121398	0.072749	0.056103	0.066412	0.186515	0.102465	0.219814
Transformer-Sentiment	**0.005927**	**0.013554**	**0.005782**	**0.00844**	**0.009335**	**0.013617**	0.483078

6 Conclusions

In this master thesis, we set out to explore the predictive capabilities of various models on cryptocurrency prices, with a particular focus on enhancing these models with sentiment analysis data. The core objective was to determine whether advanced models, such as Transformers, which are typically used for natural language processing tasks, could outperform traditional time series models like LSTM and LSTM-CNN hybrids when applied to financial data, especially when augmented with sentiment features.

Extensive preprocessing steps were undertaken to address issues like missing values and non-stationarity, which are common in financial data. Techniques such as differencing and moving averages were employed to make the data suitable for modeling. Sentiment analysis features were incorporated from social media platforms and articles, hypothesizing that market sentiment could provide additional predictive power beyond traditional financial indicators.

6.1 Findings

- LSTM (Long Short-Term Memory): This model was chosen for its proficiency in handling sequential data and capturing long-term dependencies, two essential qualities for predicting cryptocurrency prices which often exhibit temporal dependencies.
- LSTM-CNN (Convolutional Neural Network): This hybrid model was tested to leverage the strengths of both LSTM for sequential data and CNN for capturing local patterns in data. This hybrid approach aimed to improve the model's ability to predict price movements by capturing both temporal and spatial dependencies.
- Transformers: The main point of the master thesis was to test their hypothesized superior capabilities in handling sequential data through self-attention mechanisms. This model was

expected to excel due to its ability to weigh different parts of the input data, potentially making it highly effective for financial time series forecasting, especially when integrating sentiment analysis features.

The evaluation was split on two parallel tracks, one for models working with financial-only data, and the other for the sentiment-enhanced models.

- Traditional Models: The three models using only financial data provided a baseline for comparison. These models performed reasonably well but had limitations in capturing the high volatility and sudden market shifts typical of cryptocurrencies.
- Sentiment-Enhanced Models: Integrating sentiment analysis features showed mixed results across different models and cryptocurrencies. While some benefited from the additional sentiment data, others experienced increased noise, leading to worse performance.

The LSTM model showed robust performance on stable cryptocurrencies like BTC and ETH. However, their effectiveness diminished with more volatile cryptocurrencies like DOGE and XMR. Incorporating sentiment data into LSTM model generally did not improve performance and often led to higher error metrics, suggesting that the added complexity and noise from sentiment data could not be effectively managed by the LSTM architecture.

The LSTM-CNN hybrid model performed better than pure LSTM model on volatile cryptocurrencies, thanks to the CNN's ability to capture local patterns in the data. However, the integration of sentiment data did not consistently enhance performance. In fact, for most cryptocurrencies, the sentiment-enhanced LSTM-CNN model showed higher Mean Absolute Error (MAE) compared to their purely financial counterparts, indicating that sentiment data introduced noise rather than useful signals.

6.2 Results Framework

The following table addresses the findings against the previously defined hypotheses. It is shown how Transformer models performed compared to traditional models both with financial data only and integrating sentiment analysis features.

Hypothesis	**Results**
H1: Transformer models will outperform traditional time series forecasting models in predicting cryptocurrency prices due to their ability to efficiently process sequences with self-attention mechanisms, allowing them to manage volatility and non-stationarity more effectively than traditional models and thus leading to improved forecasting accuracy.	R1: The Transformer model outperformed LSTM and LSTM-CNN models across the board, with best results obtained on more stable cryptocurrencies like BTC. The self-attention mechanism of Transformers proved highly effective in capturing the intricate patterns and dependencies in the financial data.
H2: The integration of sentiment analysis into forecasting models will enhance predictive accuracy, uncovering latent signals that are not captured by traditional financial indicators alone, thereby providing a more comprehensive understanding of market dynamics.	R2: The inclusion of sentiment data generally enhanced the performance of the models, especially with Transformers: in fact, the Transformer model with sentiment data achieved the lowest MAE (0.013554 for BTC), demonstrating its superior predictive capabilities.

H3: Transfer learning will demonstrate generalizability across different cryptocurrencies, leveraging learned patterns from one market to predict another with reduced model training requirements and similar performance.	R3: The application of transfer learning in this context was explored, by training the sentiment-enhanced Transformer model on BTC data to predict ETH close prices. The results showed good prediction accuracy, validating the effectiveness of this approach. The MAE for ETH predictions was higher compared to a model trained solely on ETH data (0.00844), but still lower than LSTM and LSTM-CNN models.

6.3 Final Thoughts

There are several challenges that should be taken into consideration regarding this field of machine learning, and the sector of cryptocurrency. First of all, cryptocurrencies are inherently volatile, making accurate predictions challenging. The introduction of sentiment data added another layer of complexity, which was not always beneficial.

More complex models like Transformers handled the additional data better, but they also required careful tuning and more computational resources. The integration of sentiment analysis features proved to be a double-edged sword. While potentially valuable, these features need to be carefully curated and tested to ensure they add predictive value without introducing significant noise. In fact, the original dataset contained a lot of sentiment analysis features which resulted to be meaningless, or just plain noise which would highly degrade the models' results.

This thesis demonstrates the potential of advanced models like Transformers in the field of financial forecasting, particularly for cryptocurrencies. While traditional models like LSTM and LSTM-CNN can perform well under certain conditions, their ability to handle complex data is limited compared to Transformers. The integration of sentiment analysis features, although challenging, shows promise when managed correctly. Adding too many features, often not meaningful, proved to be drastically damaging to the predictive capabilities of the models, introducing noise.

Future research could focus on optimizing sentiment data integration and exploring other advanced models to further improve prediction accuracy in volatile markets like cryptocurrencies, as well as extending transfer learning to other cryptocurrency pairs to further enhance the robustness and applicability of transfer learning in financial forecasting.

These insights are not only academically significant but also hold practical implications for developing more sophisticated and accurate financial forecasting tools, which are crucial for investors and traders navigating the unpredictable world of cryptocurrency markets.

7 Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need," 2023.

[2] S. Ahmed, I. E. Nielsen, A. Tripathi, S. Siddiqui, R. P. Ramachandran and G. Rasool, "Transformers in Time-Series Analysis: A Tutorial," 2023.

[3] H. Zhao, M. Crane and M. Bezbradica, "Attention! Transformer with Sentiment on Cryptocurrencies Price Prediction," School of Computing, Dublin City University, Dublin, Ireland.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[5] Z. Zheng, S. Xie, H. Dai, X. Chen and H. & Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *In 2017 IEEE International Congress on Big Data (BigData Congress) (pp. 557-564)*, IEEE, 2017.

[6] C. Catalini and J. S. Gans, "Some simple economics of the blockchain," *National Bureau of Economic Research,* no. No. w22952, 2016.

[7] F. Schär, "Decentralized finance: On blockchain- and smart contract-based financial markets.," vol. Federal Reserve Bank of St. Louis Review, no. 103(2), pp. 153-174. , 2021.

[8] R. Shumway and D. Stoffer, "ARIMA Models," in *Time Series Analysis and Its Applications*, Springer Texts in Statistics. Springer, Cham., 2017, p. 75–163.

[9] J. Brownlee, "How to Develop Convolutional Neural Network Models for Time Series Forecasting," *Machine Learning Mastery.*

[10] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition.," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[11] Z. Zeng, R. Kaur, S. Siddagangappa, S. Rahimi, T. Balch and M. Veloso, "Financial Time Series Forecasting using CNN and Transformer," *J. P. Morgan AI Research,* 2023.

[12] E. Xi, S. Bing and Y. Jin, "Capsule Network Performance on Complex Data.," 2017.

[13] C. Wang, Y. Chen, S. Zhang and Q. Zhang, "Stock market index prediction using deep Transformer model," in *Expert Systems with Applications*, Elsevier, 2022.

[14] A. H. Ribeiro, K. Tiels, L. A. Aguirre and T. Schönl, "Beyond exploding and vanishing gradients: analysing rnn training using attractors and smoothness.," *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics,* vol. 108, no. Proceedings of Machine Learning Research, p. 2370–2380, 26–28 Aug 2020.

[15] J. El Zini, Y. Rizk and M. Awad, " An optimized parallel implementation of non-iteratively trained recurrent neural networks.," *Journal of Artificial Intelligence and Soft Computing Research,* 2021.

[16] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network.," *Physica D: Nonlinear Phenomena,* 2020.

[17] X. Caosen, L. 1. Jingyuan, F. Bing and L. Baoli, "A Financial Time-Series Prediction Model Based on Multiplex Attention and Linear Transformer Structure," *MDPI,* vol. Applied Sciences, 2023.

[18] G. Sababipour Asl, "Stock Volatility Forecasting with Transformer Network," *The University of Manitoba,* 2023.

[19] E. Zivot and J. Wang, "Rolling Analysis of Time Series," in *Modeling Financial Time Series with S-Plus*, Vols. Modeling Financial Time Series with S-Plus, New York, NY, Springer, 2003, p. 299–346.

[20] Yahoo Inc., "Yahoo Finance," [Online]. Available: https://finance.yahoo.com/lookup.

A Nonno Lucio e Nonna Ida