Instituto Tecnológico y de Estudios Superiores de Occidente

Recognition of official validity of higher education studies as set forth in ministerial agreement 15018, published in the Official Journal of the Federation on November 29, 1976.

Department of Mathematics and Physics

Master's Degree in Data Science

ITESO, Jesuit University of Guadalajara

Forecasting bitcoin's tendency with ConvLSTM architecture

DEGREE-EARNING PROJECT that, in order to earn the **DEGREE** of **MASTER IN DATA SCIENCE**

Is submitted by: **CESAR CONTRERAS GONZALEZ**

Advisor: **DR.SC. ROCIO CARRASCO NAVARRO**

Tlaquepaque, Jalisco. September, 2025.

Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Matemáticas y Física

Maestría en Ciencia de Datos

Predicción de la tendencia en el bitcoin usando una arquitectura ConvLSTM

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRO EN CIENCIA DE DATOS

Presenta: **CÉSAR CONTRERAS GONZÁLEZ**

Asesora: **DRA. ROCÍO CARRASCO NAVARRO**

Tlaquepaque, Jalisco. Septiembre de 2025.

ACKNOWLEDGMENTS

I wish to express my deepest gratitude to ITESO for its invaluable support and contribution to the development of this research. Similarly, I would like to extend my gratitude to all the professors and mentors who provided their knowledge, guidance, and experience throughout this academic process. A special thanks to my advisor, which guidance and advice were crucial in achieving the set objectives. Moreover, I cannot fail to mention the constant support and motivation provided by my peers and colleagues, who enriched this journey with their perspectives and camaraderie. Finally, from the bottom of my heart, I would like to thank my family and loved ones for their unwavering support, patience, and encouragement at every stage of this project.

AGRADECIMIENTOS

Deseo expresar mi más sincero agradecimiento al ITESO por su valioso apoyo y contribución al desarrollo de este trabajo. De igual manera, quisiera extender mi gratitud a todos los profesores y mentores que ofrecieron su conocimiento, guía y experiencia a lo largo de este proceso académico. Un agradecimiento especial a mi asesora, cuyas orientaciones y consejos fueron cruciales para alcanzar los objetivos propuestos. Además, no podría dejar de mencionar el apoyo constante y la motivación brindada por mis colegas y compañeros, quienes enriquecieron este viaje con sus perspectivas y camaradería. Finalmente, agradezco de corazón a mi familia y seres queridos por su incondicional apoyo, paciencia y aliento en cada etapa de este proyecto.

DEDICATORY

To my parents, whose unwavering faith, sacrifices, and love have been my guiding light throughout this journey. This accomplishment is not just mine, but yours as well. Thank you for being my strength, my inspiration, and my biggest supporters.

DEDICATORIA

A mis padres, cuya fe inquebrantable, sacrificios y amor han sido mi guía a lo largo de este viaje. Este logro no es solo mío, sino nuestro, gracias por ser mi fortaleza, mi inspiración y mi mayor apoyo.

ABSTRACT

This work presents a predictive model for real-time Bitcoin price forecasting, leveraging advanced machine learning techniques. The architecture combines Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (Bi-LSTM) networks, augmented with a scaled dot-product attention mechanism. This hybrid model aims to capture both spatial and temporal patterns in Bitcoin trading data, thereby improving the predictive accuracy.

The model is implemented using TensorFlow and Keras, featuring a custom loss function that blends Mean Squared Error with additional penalties for extreme value predictions. To optimize the model's hyperparameters, Keras Tuner's RandomSearch method is utilized, automating the selection process across a predefined hyperparameter space. The training employs the Nadam optimizer, with early stopping and learning rate reduction strategies to mitigate overfitting and accelerate convergence.

Out of 30 models trained during hyperparameter tuning, the best-performing model is selected based on the lowest validation loss. The resulting model serves as a robust tool for real-time Bitcoin price prediction, potentially aiding data-driven trading decisions.

RESUMEN

Este trabajo presenta un modelo predictivo sofisticado para la predicción en tiempo real del precio de Bitcoin, aprovechando técnicas avanzadas de aprendizaje automático. La arquitectura combina Redes Neuronales Convolucionales (CNN) y Redes de Memoria a Corto y Largo Plazo Bidireccionales (Bi-LSTM), complementadas con un mecanismo de atención de producto escalar. Este modelo híbrido tiene como objetivo capturar tanto patrones espaciales como temporales en los datos de comercio de Bitcoin, mejorando así la precisión predictiva.

El modelo se implementa usando TensorFlow y Keras, con una función de pérdida personalizada que combina el Error Cuadrático Medio con penalizaciones adicionales para predicciones de valores extremos. Para optimizar los hiperparámetros del modelo, se utiliza el método RandomSearch de Keras Tuner, automatizando el proceso de selección en un espacio de hiperparámetros predefinido. El entrenamiento emplea el optimizador Nadam, con estrategias de detención temprana y reducción de la tasa de aprendizaje para mitigar el sobreajuste y acelerar la convergencia.

De los 30 modelos entrenados durante el ajuste de hiperparámetros, se selecciona el modelo con mejor rendimiento en función de la menor pérdida de validación. El modelo resultante sirve como una herramienta para la predicción del precio de Bitcoin en tiempo real, lo que podría facilitar la toma de decisiones comerciales basadas en datos.

CONTENT TABLE

FIGURES LIST

TABLE LIST

ACRONYMS LIST

API	Application Programming Interface
ATR	Average True Range
CNN	Convolutional Neural Network
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transformation
LDF	Linear Diophantine fuzzy
LSTM	Long Short-Term Memory
MACD	Moving Average Convergence Divergence
MAE	Mean Square Error
MAPE	Mean Absolute Percentage
OBV	On-Balance Volume
OHLC	Open, High, Low, Close
RMSE	Root Mean Squared Error
RSI	Relative Strength Index
SMA	Smooth Moving Averange
SMA	Simple Moving Average
TF- IDF	Term Frequency-Inverse Document Frecuency

1. INTRODUCTION

In the next chapter there is going to be provided an examination of deep learning applications in financial forecasting, with particular focus on cryptocurrency markets. The discussion covers:

1. **Background**: Traditional statistical methods have limitations in capturing non-linear patterns in volatile financial data, leading to the adoption of deep learning approaches, especially Long Short-Term Memory (LSTM) networks.
2. **Problem Definition**: Cryptocurrency markets present unique forecasting challenges due to volatility influenced by news events and large traders, requiring models that can identify underlying "Core Market" trends.
3. **Justification**: There is a stablish market for this kind of tool that helps trades to operation.
4. **Objectives**: Development of prediction models with specific performance targets: MAE ≤ 0.134, MSE ≤ 0.026, RMSE ≤ 0.161, and MAPE ≤ 16.72% for minute-level predictions.
5. **Hypothesis**: Conventional market strokes have a possibility of being predicted if it treated under certain conditions (no news peaks or whales involve) with a time factor. LSTM can catch up with that temporality in the data for cryptocurrencies.
6. **State-of-the-Art Comparison**: Analysis of current approaches including Random Forest models, SVM implementations, and Transformer architectures for high-frequency trading.

1.1. Background

The rapidly evolving landscape of financial markets, coupled with the burgeoning volume of data, has necessitated the development of sophisticated forecasting models. Traditional statistical methods, while effective to a certain extent, often fall short in capturing the intricate non-linear patterns inherent in financial data. The past two decades have witnessed the rise of deep learning as a promising solution to this challenge.

Deep learning, a subset of machine learning, utilizes artificial neural networks with multiple layers to analyze various factors of data. Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Network (RNN), have emerged as a particularly effective tool in time series forecasting. Their unique architecture, designed to capture long-term dependencies in sequential data, makes them apt for predicting financial market movements.

However, the application of LSTMs in financial forecasting is not without challenges. Financial data is notorious for its volatility, noise, and susceptibility to external events, making predictions exceedingly complex. The successful employment of LSTMs requires meticulous data preparation, feature selection, and model tuning.

The recent integration of Convolutional Neural Networks (CNNs) with LSTMs offers a promising avenue for enhancing forecasting accuracy. While LSTMs excel at sequence prediction, CNNs are adept at spatial pattern recognition, making their combination potentially powerful for financial time series forecasting [1].

1.2. Problem

The financial markets, especially cryptocurrency markets like Bitcoin, are characterized by their volatile and unpredictable nature. Traditional forecasting methods often fall short in predicting sudden spikes or crashes, leading to substantial financial losses [2]. So, stepping away from the volatility point cause in mayor cases by news or whales in short or long terms, it follows a term call “Core Market” is what remains after excluding these influences like speculative trading or temporary shocks from news [3].

Whales, in the context of cryptocurrency, refer to individuals or entities that hold a significant portion of a particular cryptocurrency. These large holders can significantly influence market prices through their trading activities [4]. By excluding their influence, we aim to understand the behavior of the broader market participants.

News events, both positive and negative, can cause rapid and sometimes irrational price movements in cryptocurrency markets. These events can range from regulatory announcements to technological developments or macroeconomic factors. By stepping away from the volatility caused by news, we can focus on the long-term trends and patterns in the market [4].

1.3. Justification

The global financial markets have become increasingly intricate, with an escalating volume of transactions and a plethora of influencing factors [2]. Traditional models, while foundational, have demonstrated limitations in forecasting accuracy given the volatile nature of these markets [1]. An advanced model that can encapsulate the multifaceted dynamics of financial markets is imperative for better forecasting and risk management [4].

Accurate forecasting is not just a theoretical pursuit; it has tangible ramifications in the real world. Investors, financial institutions, policymakers, and individual traders base crucial decisions on market forecasts [5]. A slight improvement in predictive accuracy can translate into significant financial gains or risk mitigation [6].

Conventional time series models, like Autoregressive Integrated Moving Average (ARIMA) and Generalized AutoRegressive Conditional Heteroskedasticity (GARCH), have been widely used in financial forecasting [7][8]. However, they often assume linearity and can struggle with capturing non-linear patterns, especially in turbulent market conditions. Deep learning models, such as LSTMs, offer a solution to this, given their ability to model complex non-linear relationships [9].

The past decade has seen remarkable advancements in deep learning techniques. Their success in domains like image and speech recognition demonstrates their capability to capture intricate patterns from vast amounts of data. Financial time series forecasting stands to benefit immensely from these advancements, particularly when models are tailored for financial data's unique characteristics [10].

While LSTMs have demonstrated promise in time series forecasting, integrating them with other neural network architectures, like CNNs, can further enhance their efficacy. Such hybrid models can leverage the strengths of individual architectures, offering a more comprehensive approach to prediction [11].

Exploring and refining LSTM-based models for financial forecasting contributes to both academic literature and practical applications. For academia, it extends the body of knowledge around deep learning applications in finance. For practitioners, it provides a robust tool that can be implemented for better decision-making.

1.4. Objetives

General Objectives:

To develop a deep learning-based model optimized for predicting cryptocurrency prices with Mean Absolute Error (MAE) below 0.15, Mean Squared Error (MSE) below 0.03, and Mean Absolute Percentage Error (MAPE) below 17% for per-minute predictions.

Especific objectives:

Data Collection and Preprocessing:

- Acquire comprehensive time series data of cryptocurrency prices including OHLCV (Open, High, Low, Close, Volume) metrics at minute-level granularity.
- Implement data cleaning and normalization techniques to handle missing values, outliers, and scale features appropriately for deep learning models.

Feature Engineering:

- Develop a robust set of technical indicators including SMA, RSI, MACD, and Parabolic SAR to enhance the model's predictive capabilities.
- Implement cyclic encoding for temporal features to capture intraday and seasonal patterns in cryptocurrency price movements.
- Create rolling window features to capture relevant historical price patterns

Model Development:

- Design and implement hybrid deep learning architectures combining LSTM networks (unidirectional and bidirectional) with CNN layers
- Train models to achieve:
    - MAE ≤ 0.134
    - MSE ≤ 0.026
    - RMSE ≤ 0.161
    - MAPE ≤ 16.72%

Evaluation and Fine-tuning:

- Implement a systematic evaluation framework using walk-forward optimization with appropriate training/validation/test splits
- Apply hyperparameter optimization techniques including:
    - Early stopping with patience monitoring
    - Learning rate scheduling
    - Model checkpointing
- Analyze model performance across different market conditions to ensure robustness.

1.5. Hypothesis

Forecasting financial markets, given their volatile and non-linear nature, is a challenging endeavor. Traditional models have shown limitations in capturing intricate patterns in financial time series data. This work posits that deep learning models, specifically LSTMs, can provide superior forecasting accuracy when compared to conventional methods [2]. The following hypotheses are formulated:

1. Long Short-Term Memory (LSTM) networks will demonstrate a significantly higher accuracy in predicting financial market movements compared to traditional forecasting models.
2. Incorporating additional layers and neurons in the LSTM network will enhance the forecasting accuracy, but only up to a certain level of complexity after which the model might overfit.
3. Certain features, derived from historical financial data, will have a stronger influence on the LSTM's forecasting performance than others.
4. LSTMs, due to their ability to remember long-term dependencies, will exhibit a higher forecasting accuracy in real-time scenarios, especially during periods of high market volatility.
5. Combining LSTMs with other neural network architectures, such as CNNs for feature extraction, will further enhance the model's forecasting accuracy.

1.6. State of the Art

Cryptocurrency price prediction has seen a surge in research interest given the financial implications and the inherent complexity of crypto markets. Traditional methods, from ARIMA to GARCH, have been employed with varying degrees of success. However, with the advent of deep learning, models such as LSTMs and CNNs have emerged as frontrunners due to their ability to model non-linear sequences plus the batch size previous in the data to capture better the hidden patterns [12].

Deep learning approaches offer several key advantages over traditional statistical methods:

1. **Superior pattern recognition**: Deep neural networks excel at identifying complex patterns and relationships in data that traditional statistical models often miss. This is particularly valuable in cryptocurrency markets where price movements can be influenced by numerous interrelated factors.
2. **Non-linearity handling**: Cryptocurrencies exhibit highly non-linear behavior that traditional linear models struggle to capture. Deep learning architectures like LSTMs are specifically designed to model these complex non-linear relationships without requiring explicit feature engineering.
3. **Temporal dependency modeling**: LSTMs and other recurrent architectures can maintain memory of past events and learn long-term dependencies in sequential data, making them particularly suited for financial time series where past price movements influence future behavior.
4. **Feature hierarchy learning**: Deep networks can automatically learn hierarchical representations from raw data, extracting increasingly abstract features at deeper layers. This reduces the need for manual feature engineering while potentially discovering novel predictive patterns.
5. **Adaptability to market conditions**: Through continuous training, deep learning models can adapt to evolving market dynamics and regime changes, which is crucial in the rapidly evolving cryptocurrency landscape.

Recent works have focused on enhancing these models through attention mechanisms or transformer architectures, aiming to give models a "memory" of important events. Furthermore, external factors such as news sentiment analysis and social media trends have been integrated to capture market sentiment.

This research positions itself at the intersection of deep learning, feature engineering, and time granularity. By shifting focus to minute-based forecasting and fusing traditional technical indicators with modern architectures, this work aspires to push the boundaries of what's currently achievable in cryptocurrency price prediction.

Table 1 to Table 4 were synthesized the most relevant information from various research articles focused on Bitcoin price prediction using machine learning and deep learning techniques. These studies represent the current state of the art in cryptocurrency forecasting, each employing different methodologies, time scales, and feature sets to tackle the challenge of predicting Bitcoin's highly volatile price movements.

Table 1: Forecasting Mid-Price Movement of Bitcoin Futures Using Machine Learning [2]

Aspect	Details
Primary Focus	Classification of Bitcoin futures mid-price movement as "up" or "down"
Models Used	kNN, Logistic Regression, Naive Bayes, Random Forest, SVM, XGBoost
Data Characteristics	Bitcoin futures mid-price returns
Time Scale	5, 10, 15, 30, 60-minute intervals
Key Findings	• Random Forest: highest in-sample success (87%) • SVM: best out-of-sample forecasting (71%) • ML methods outperformed ARIMA (52%)
Performance Metrics	• Sign Prediction Ratio • Maximum Return • Total Return • Ideal Profit Ratio
Notable Features	Compared multiple train/test splits (0.7/0.3, 0.8/0.2, 0.9/0.1)

Table 2: Forecasting Bitcoin Price Direction with Random Forests [3]

Aspect	Details
Primary Focus	Predicting directional movement of Bitcoin and gold prices
Models Used	• Random Forests • Bagging • Logit model with boosting
Data Characteristics	• Bitcoin and gold prices • Technical indicators • Macroeconomic variables
Time Scale	Daily data from 2014-2021
Key Findings	• Tree-based methods effective for price direction prediction • Importance of macroeconomic factors (interest rates, inflation, volatility)
Performance Metrics	• Prediction accuracy Kappa statistic Positive/negative predictive values
Notable Features	Integration of macroeconomic factors alongside technical indicators

Table 3: Exploring the Advantages of Transformers for High-Frequency Trading [5]

Aspect	Details
Primary Focus	High-frequency Bitcoin-USDT log-return forecasting
Models Used	• HFformer (custom Transformer model) • LSTM (baseline)
Data Characteristics	High-frequency trading data
Time Scale	Very short-term (likely seconds to minutes)
Key Findings	• Transformer architectures outperform LSTM models for high-frequency trading • HFformer achieved higher cumulative PnL
Performance Metrics	• Cumulative Profit and Loss (PnL) • Backtesting performance
Notable Features	Specialized transformer architecture without position encoding

Table 4: Forecasting Bitcoin's Tendency with ConvLSTM Architecture (Current Research)

Aspect	Details
Primary Focus	Minute-level Bitcoin price prediction with hybrid deep learning
Models Used	• CNN-LSTM hybrid with attention • Bidirectional LSTM • Baseline comparisons (Linear Regression, Random Forest, Gradient Boosting)
Data Characteristics	• OHLCV minute data • Technical indicators • Temporal features
Time Scale	Minute-level with variable lookback periods (30-60 minutes)
Key Findings	• Hybrid architecture outperforms traditional models • Larger CNN filters (256, 512) improve performance • Lookback period affects model's ability to capture volatility
Performance Metrics	• MAE (≤ 0.134) • MSE (≤ 0.026) • RMSE (≤ 0.161) • MAPE (≤ 16.72%)
Notable Features	• ConvLSTM architecture for spatial-temporal learning • Attention mechanism • Specialized hyperparameter tuning

2. THEORICAL FRAMEWORK

This chapter provides an examination of the deep learning models employed in financial forecasting, with a particular focus on Long Short-Term Memory (LSTM) networks and gates architectures. The discussion addresses the following key areas:

1. **Time Series Component**: Analysis of financial data with temporal dependencies, it corroboration and where each data point is sequentially ordered over time.
2. **Model Architectures**: A detailed breakdown of LSTM units, including the roles of forget gates, input gates, and output gates. For Transformers, the self-attention mechanism and positional encoding are examined.
3. **Data Preprocessing**: Techniques specific to financial time series data, including normalization, standardization, and sequence creation for deep learning models.
4. **Feature Engineering**: Strategies for creating effective input features for financial forecasting, such as technical indicators and statistical measures derived.
5. **Training Considerations**: Batch sizes, learning rates, and other hyperparameters crucial for optimal model performance.
6. **Model Evaluation**: Metrics and methodologies for assessing the performance of deep learning models in financial forecasting contexts.

2.1. Time Series Analysis in Feature Finances Variables and their Relevance

Time series analysis provides a framework for understanding temporal data structures, especially in the finance sector where data points are sequentially ordered over time. Financial variables, due to their sequential nature, have been subjected to time series analysis to uncover patterns, predict future values, and understand their underlying dynamics [7].

One of the primary objectives in analyzing financial time series data is to understand and predict the future values of key variables, such as stock prices or exchange rates. Neural networks, specifically recurrent architectures like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), have shown promise in this domain due to their ability to capture long-term dependencies in sequential data [8].

Feature selection is paramount when dealing with financial variables. Some of the features commonly used in financial modeling include Simple Moving Averages (SMA), Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), and others. Each of these features provides unique insights. For instance, SMA can indicate the overall trend of a stock price, while RSI might signal overbought or oversold conditions [9].

By employing convolutional layers such as Conv1D before recurrent layers, the models benefit from an automated feature extraction process that efficiently captures local patterns within the input sequences, potentially enhancing model performance [10]-[11].

The relevance of these financial features lies in their ability to capture the intricate dynamics of financial markets. By effectively integrating these features into advanced models, we can achieve more accurate and insightful forecasts. Furthermore, with real-time data acquisition methods, such as WebSockets provided by platforms like CoinAPI, we can keep models up-to-date with the latest data, ensuring predictions remain relevant in rapidly changing markets [12].

2.1.1. Statistical Tests and Stationarity

In time series analysis, stationarity is a crucial concept. A stationary process has statistical properties, such as mean, variance, and autocorrelation, that are consistent over time. Most time series models assume stationarity, implying that the characteristics of the series do not change over time [13].

Stationarity is fundamental to time series analysis for several reasons:

- **Predictability**: If the statistical properties of a process do not change over time, it is easier to model. Models can assume that the underlying mechanics of the series will remain consistent in the future [14].
- **Valid Statistical Inference**: Many statistical tests and confidence intervals rely on the assumption of stationarity to produce reliable results.
- **Model Simplification**: Stationary series can be described with fewer parameters, making models more parsimonious and interpretable.

To determine whether a time series is stationary, various statistical tests can be employed. The most commonly used is the Augmented Dickey-Fuller (ADF) test, which examines the null hypothesis that a unit root is present in an autoregressive model characterizing the time series. If the null hypothesis is $\gamma = 0$, the time series is non-stationary is rejected, it suggests the time series is stationary (or trend-stationary) [15]-[16].

The equation for the test is:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \delta_2 \Delta y_{t-2} + \cdots + \delta_p \Delta y_{t-p} + \varepsilon_t \tag{1}$$

where Δy_t the difference series, y_t is the time series, ε_t is the error term, δ is the coefficient for the lagged difference terms, p the number of lagged difference terms included in the model, α the constant/intercept term and β the coefficient for the trend term t.

2.2. Long Short-Term Memory neural network models

LSTM networks are a specialized type of recurrent neural network (RNN) designed to address the problem of long-term dependencies and combat these gradient issues [28]. Unlike standard RNNs, LSTMs can "remember" information for extended periods [8].

2.2.1. Architecture of LSTM

The LSTM unit is composed of several key components:

1. **Memory Cell (Cell state):** This is the main component that acts as a conveyor belt of information throughout the sequence. It allows information to flow unchanged through the network represented as:

$$c_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2)$$

c_t represents the cell state (memory) at timestamp (t) which activation function is $\tanh$ with value range from -1 to 1 as the activation function.

2. **Gates:** These are structures that regulate the flow of information in and out of the memory cell. There are three types of gates:

   a) Forget gate decides what information from the memory cell should be discarded.

   $$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3)$$

   where f_t represents the forget gate, σ is the sigmoid function, W_f is the weight for the respective gate neurons, h_{t-1} is the output of the previous LSTM block (at timestamp $t-1$), x_t is the input at current timestamp and b_f is the biases for the forget gate.

   b) Input gate determines what new information will be stored in the memory cell.

   $$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

   where i_t represents the input gate, W_i is the weight for the respective gate neurons and b_i biases for the input gate

c) Output gate: controls which parts of the memory cell will be used to produce the output.

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

Where o_t represents the output gate, W_o is the weight for the respective gate (in this case o) neurons and b_o is the biases for the output gate

3. **Input layer:** Processes the current input and previous output to generate new candidate information for the memory cell.

This architecture allows LSTM to maintain relevant information over long periods, overcome the vanishing gradient problem that affects standard RNNs, and learn long-term dependencies in sequential data.

2.2.2. Functioning of Attention Mechanism

The attention mechanism allows the model to focus on different parts of the input sequence when producing an output. It calculates a context vector as a weighted sum of all input annotations, where the weights denote the importance of each input.
The attention weight α_{tj} for each input annotation a_j is calculated using:

$$\alpha_{tj} = \frac{\exp(score(h_t, a_j))}{\sum_{j=1}^{T_x} \exp(score(h_t, a_j))} \tag{6}$$

where score is a function that calculates how well the inputs around position j and the output at position t match [1], h_t is the hidden state at time step t and T_x the length of the input sequence-

2.3. Data Preprocessing

2.3.1. Normalization and Standardization

Before feeding data into most machine learning models, it's essential to scale the features so that they are centered around zero and have a unit variance. This ensures that no particular feature dominates others in terms of scale, leading to a more stable and faster convergence during training.

Normalization: Also call Min-Max Scaling which scales the data between a specified range (usually 0-1). It's given by the formula:

$$X_{norm} = \frac{X_{max} - X_{min}}{X - X_{min}} \tag{7}$$

where X_{max} and X_{min} are the maximum and minimum values of the feature, respectively. X is the input value.

For regression problems, it helps maintain the relative distribution of the target variable. On the other hand, in financial predictions, it can help in interpreting the model's output as a relative change.

Standarization (Z-score normalization): This method scales the data based on the mean (μ) and standard deviation (σ) so that the new distribution will have a mean of 0 and a standard deviation of 1.

$$X_{std} = \frac{X - \mu}{\sigma} \tag{8}$$

For time series data, it's important to ensure that the scaling parameters like X_{max}, X_{min} μ the mean of the dataset and σ the standard deviation of the dataset is derived only from the training data to avoid data leakage [29].

2.3.2. Sequence Creation for LSTM

LSTM models require input data in sequences. For time series forecasting, the data needs to be transformed into overlapping sequences where each sequence corresponds to a set number of past observations.

The sliding window approach consists of moving a window of fixed size n over the dataset to generate subsequences that can be used as training instances. For example, with a window size of 3, the values at time steps $t-2, t-1$ would be used to predict $t+1$ [1].

Mathematical representation:
Given a time series $X = [x_1, x_2, \dots \dots .. x_T]$, with a window size n, the following input–output pairs are generated:

$$X^i = [x_i, x_{i+1}, \dots \dots .. x_{i+n-1}]$$

$$y^i = x_{i+n}$$

where $i = 1,2, \dots, T-n$. Here, n denotes the look-back period or the number of past time steps considered.

The resulting sequences must be reshaped into a three-dimensional tensor with the following structure:

- **Samples:** number of sequences generated.
- **Timesteps:** the window size n
- **Features:** number of input variables per timestep.

2.4. Feature engineering

Feature engineering involves creating new features from the existing data to enhance the model's predictive power. For financial time series data, some commonly engineered features include:

- RSI_price_close: Captures momentum and overbought/oversold conditions, complementing raw lagged price features by providing a normalized measure of price dynamics.
- Momentum: Offers a direct rate-of-change perspective, highlighting acceleration or deceleration in price movements.
- Lagged_price_close: Preserves essential temporal dependencies in the price series, providing strong predictive power without direct leakage.
- MACD: Adds a trend-following component by combining short- and long-term moving averages, useful for identifying shifts in underlying trend direction.
- Bollinger_Upper_Band: Encodes volatility-adjusted deviations from a moving average, helping to detect extreme price levels.
- Stochastic_Oscillator: Adds a relative positioning of the closing price within its recent range, useful for capturing reversal signals.
- OBV: Integrates price and volume to detect accumulation/distribution patterns not evident from price alone.
- ATR: Retained as a volatility indicator, capturing market turbulence beyond directional trends.
- SMA_volume_traded & RSI_volume_traded: Complement price-based signals with smoothed and momentum-based volume dynamics.
- Temporal and calendar variables (Minute, Hour, DayOfWeek, Holiday, Weekend): Introduced to capture intraday seasonality and event-driven effects.

Collectively, this refined feature set balances price momentum, trend, volatility, volume dynamics, and temporal effects, ensuring broad coverage of market behavior while minimizing redundancy.

2.5. XGBoost for feature selection

XGBoost, short for *Extreme Gradient Boosting*, is an optimized distributed gradient boosting library designed to be efficient, flexible, and portability [20]. Beyond its performance advantages, XGBoost provides built-in support for understanding the model by assessing the relevance or importance of each feature.

Feature importance is a key concept that connects interpretability, feature selection, domain knowledge, and model robustness.

Interpretability: Understanding which features are influential helps in interpreting the model. This is particularly important in domains like finance or healthcare where interpretability can be crucial [21].

Feature Selection: If a dataset has many features, determining which ones are most relevant can help in selecting a subset of the most important features, potentially improving model performance and reducing training time [22].

Domain Knowledge: In some cases, the importance of features can provide validation of domain knowledge. For example, in financial modeling, if a feature known to be crucial (like a key financial indicator) is also found to be important by XGBoost, it adds confidence to the model [23].

Reducing Overfitting: Features that do not contribute much to the predictive power can sometimes lead to overfitting. Identifying and removing them can help in building a more robust model [24].

XGBoost provides several methods to interpret feature importance as weight, Gain and Coverage.
Weight: This is the number of times a feature appears in a tree across all the trees in the model. It's the default measure of feature importance in XGBoost. Features that are used more frequently in the splits have higher importance [25].

Gain: It measures the improvement in accuracy brought by a feature to the branches it's on. The idea is to calculate the contribution brought by a feature to the model. It's the improvement in accuracy brought by a feature to the branches it's on.

$$\text{Gain} = \frac{1}{n}\sum(\text{Gini Index Before Split} - \text{Gini Index After Split}) \tag{10}$$

When using the Gain measure, the Gini Index is a key concept. The Gini Index for a node is:

$$\text{Gini Index} = 1 - \sum p_i^2 \tag{11}$$

Where p_i is the proportion of samples in class i at that node.

Coverage: Evaluates the relative quantity of observations affected by a feature. It's essentially an averaging operation, akin to the weight measure [26].

When using XGBoost's API, feature importance can be easily retrieved using the get_score method. Visualization tools like bar graphs or plots can then be used to represent the importance of each feature, providing a visual representation that can be useful for presentations or reports [27].

2.6. Training and Hyperparameters

2.6.1. Batch Size

Batch size is a fundamental hyperparameter in neural network training that determines how many samples are processed before the model's weights are updated. This hyperparameter plays a crucial role in how the network learns and ultimately affects both training efficiency and model performance [9].
A batch is the total number of samples that are passed through the network before the internal parameters (weights and biases) are updated shown in equation 12.

$$X_{batch} = \{x_1,\ x_2,\ x_3\ , \dots \dots \dots . x_b\} \tag{12}$$

There are three primary gradient descent techniques, each distinguished by their batch size approach:

Batch Gradient Descent: The whole dataset is passed to the network, and the parameters are updated once per epoch. Batch Size = Size of the dataset and for this experiment was 300 units.
Stochastic Gradient Descent (SGD): Each individual data point in the dataset is passed to the network, and the parameters are updated after each data point. Batch Size = 1
Mini-Batch Gradient Descent: A compromise between the two, a subset of the dataset is passed to the network, and the parameters are updated after each subset. 1 < Batch Size < Size of the dataset

Choosing the correct batch size can significantly influence a model's performance and training dynamics. Small batches, including Stochastic Gradient Descent (SGD), allow for more frequent updates, which may lead to faster convergence, and the noise in the gradient can help the model escape local minimum. However, they may also result in unstable training if the learning rate is not properly adjusted. On the other hand, large batches can better exploit hardware parallelization and provide more accurate gradient estimates, but they carry the risk of getting stuck in sharp minimizers that generalize poorly [9].

2.7. Model Evaluation

Model evaluation metrics quantify the discrepancy between predicted values and actual values, providing a numerical assessment of model performance. The choice of metric influences model selection and optimization, as different metrics emphasize different aspects of prediction error [1].

Mean Absolute Error (MAE)
MAE represents the average magnitude of errors without considering their direction. It treats all errors equally, making it a linear score where all individual differences are weighted equally [2].
MAE unlike squared errors is less sensitive to outliers, providing a more stable measure when the dataset contains anomalies. Additionally, MAE is expressed in the same units as the target variable, making it directly interpretable as the average prediction error. Also, when errors follow a Laplace distribution, MAE is the maximum likelihood estimator.

Mean Squared Error (MSE)
MSE measures the average squared difference between predictions and actual values, heavily penalizing larger errors due to the squaring operation.
MSE is differentiable everywhere, making it ideal for gradient-based optimization algorithms. Additionally, MSE decomposes into bias and variance components ($MSE = Bias^2 + Variance$), providing insight into model behavior. Also, when errors are normally distributed, minimizing MSE yields maximum likelihood estimates.

Root Mean Squared Error (RMSE)
RMSE is the square root of MSE, combining MSE's sensitivity with outliers with interpretability in the original units.
RMSE returns errors to the original scale of the data, facilitating comparison with MAE and actual values. For unbiased estimators, RMSE approximates the standard deviation of prediction errors. Like MSE, RMSE gives higher weight to large errors, making it suitable when large errors are particularly undesirable.

Mean Absolute Percentage Error (MAPE)
MAPE normalizes errors by actual values, expressing accuracy as a percentage independent of scale. Additionally, MAPE enables comparison across different datasets or variables with different units. Also, MAPE emphasizes the relative rather than absolute size of errors.
Percentage errors are intuitive for stakeholders and align with business metrics.

3. DEVELOPMENT METHODOLGY

This module will be presenting the independent variables and how adopts a systematic methodology, starting with data collection and preprocessing to create a feature-rich dataset comprising financial indicators and temporal variables, the detection and discard process using XGBoost of some variables since can cause a lick. The time window with an important lookback period. The architecture combines Convolutional Neural Networks (CNN) for spatial feature extraction and Bidirectional Long Short-Term Memory (Bi-LSTM) networks for capturing temporal dependencies, enhanced by an attention mechanism which is the principal model vs another baseline models like Linear Regression, Random Forest, and Gradient Boosting models. Hyperparameter optimization is conducted using Keras Tuner's RandomSearch. The model is trained using the Nadam optimizer, with strategies like early stopping and learning rate reduction to prevent overfitting. Performance is evaluated based on Mean Absolute Error and Mean Absolute Percentage Error.

1. **Deployment requirements**: Describe the data feed to the model, how the exploration and monitoring was, the financial indicators used and selection from the plenty created using feature engineering and the batch size plus the sequence length for the data before been passed to a model.
2. **Models**: Techniques used in the model, training specifications and metrics for evaluations.

3.1. Deployment requirements

3.1.1. Data Extraction

The required information as the independent variables are Open price, High Price, Low Price and Close Price (OHLC) that was been extracted using the Binance API for getting the historical of the bitcoin in hours. Having the range of 3 years for the training. Then another API was acquired from COINAPI (Which is a third API service webpage) where it can be called in minutes, in this case the historical was for a year but in lapses of minutes.

These calls were made with a created API using a start day and end day, can be downloaded in minutes or hours, creating a dataset concatenating the data of the calls from the services of APIcoin. Then it is passed in csv format.

3.1.2. Data Exploration

The initial dataset obtained from the API contained OHLCV (Open, High, Low, Close, Volume) data for Bitcoin at minute-level granularity, which drop for not been used in the feature/ independent variable. A comprehensive feature engineering approach was implemented to extract meaningful patterns from this raw data.

Closing price was chosen over adjusted closing price as the target variable because cryptocurrencies, unlike traditional securities, don't undergo corporate actions such as stock splits, dividends, or rights issues that would necessitate price adjustments. Additionally, the 24/7 nature of cryptocurrency markets eliminates the closing gaps typically addressed by adjusted close prices in traditional stock markets. The behavior of the target variable is visualized in Figure 1, which demonstrates the significant volatility and non-stationarity characteristics of Bitcoin prices.

Through Fourier analysis of the time series, was identify optimal cyclical patterns occurring at 24-hour intervals, which informed the decision on the ideal segmentation of data when implementing cyclical encoding, despite working with minute-level granularity for predictions.

Figure 1. The target variable for prediction's behavior, Close Price at 1 year segmented in minutes and the price ranges approximately from $16,000 to $31,000 throughout the time period.

The Dicky Fuller test for the close price variable shows a non-stationary behavior by the p value but the ADF said the opposite been that negative, so classic techniques like ARIMA cannot been used seen this kind of conflict in the relation of non and stationarity visible in the Figure 1.

One hyperparameter defined in the data preprocess for the LSTM is the time used to look how much data in the past the model should look to learn and the memory and has a pivotal role in determining the efficacy of this. During the experiments a long look back resulted in good inferences for sequences of hours, meanwhile a short is good for small inferences.

The experiments underscored that the dominant frequencies in the data should guide a choice of the look-back period. For instance, if it had exhibited a strong weekly cycle, a look-back period that encompasses an entire week (or multiple weeks) would be logical. This ensures that the model has access to complete cycles of dominant frequencies, allowing it to better internalize and predict based on these patterns that can be appreciated in the form of the graph from Figure 2.

The choice of look-back, guided by frequency domain insights, had direct implications on the model's architecture. Sequence lengths were tailored based on the look-back, impacting layers like LSTM or Conv1D which rely on sequential input. Furthermore, the attention mechanism, when employed, could focus on pivotal moments within the chosen sequence length, further enhancing the model's predictive prowess.

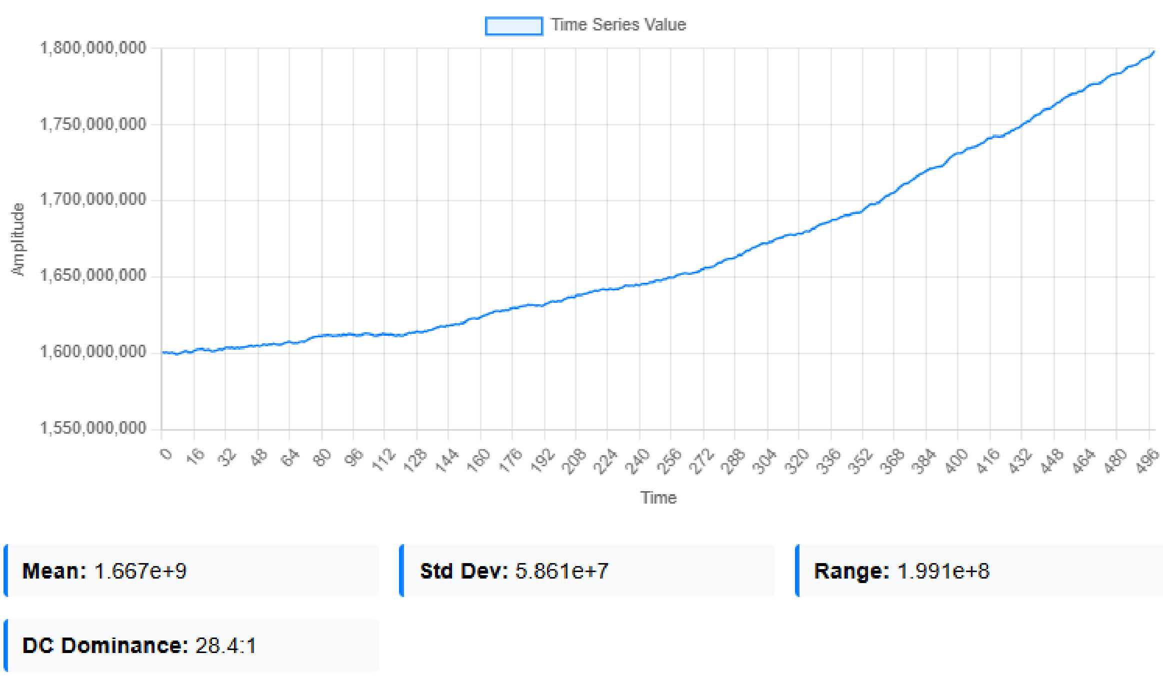

Figure 2. The graph shows that the events which contain the most valuable information in a particular period are in big frequencies, it means that in periods of 24 hours you could see the patterns meanwhile, if you used small sequences like 3 or 4 hours won't be recognized.

A XGBoost was harnessed not only for its predictive prowess but also for its innate ability to rank features based on their importance. After training the model, it provided a ranked list of features based on their *importance*. This ranking is derived from the number of times a feature is used to split the data across all trees. Leading to some variables that were dependent on and must been taken out.

The feature importance rankings provided by XGBoost were cross-referenced with the results from correlation analysis. Features that ranked highly in both XGBoost's importance metric and the correlation analysis were identified as critical for the prediction model. This dual validation approach ensured a more robust feature selection process, potentially increasing model accuracy.
The iterative nature of this process, as illustrated in Figures 3, 4, and 5, revealed important insights, for example, some engineered features showed high importance initially but were later identified as being dependent on other variables in this case the OHLC. These dependent variables were subsequently removed to prevent redundancy and potential overfitting which was detected with MAE and MAPE.

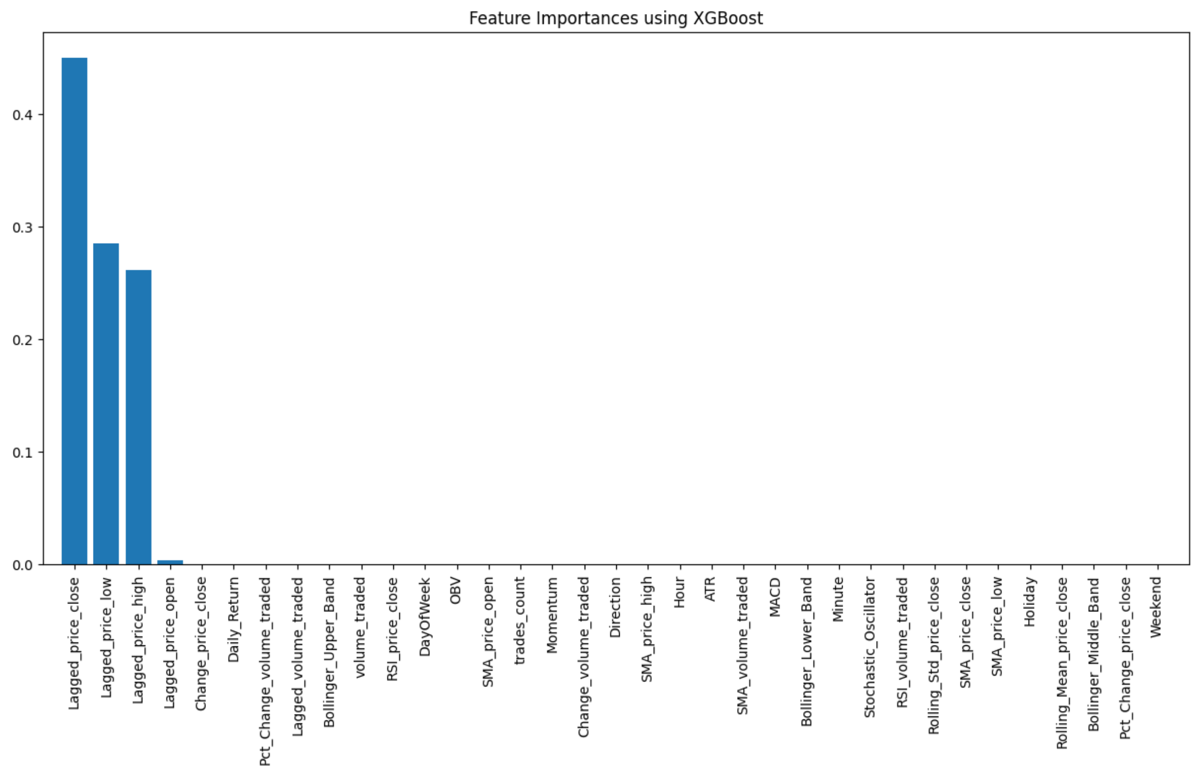

Figure 3. Feature Importances using XGBoost with the all the set of variables.

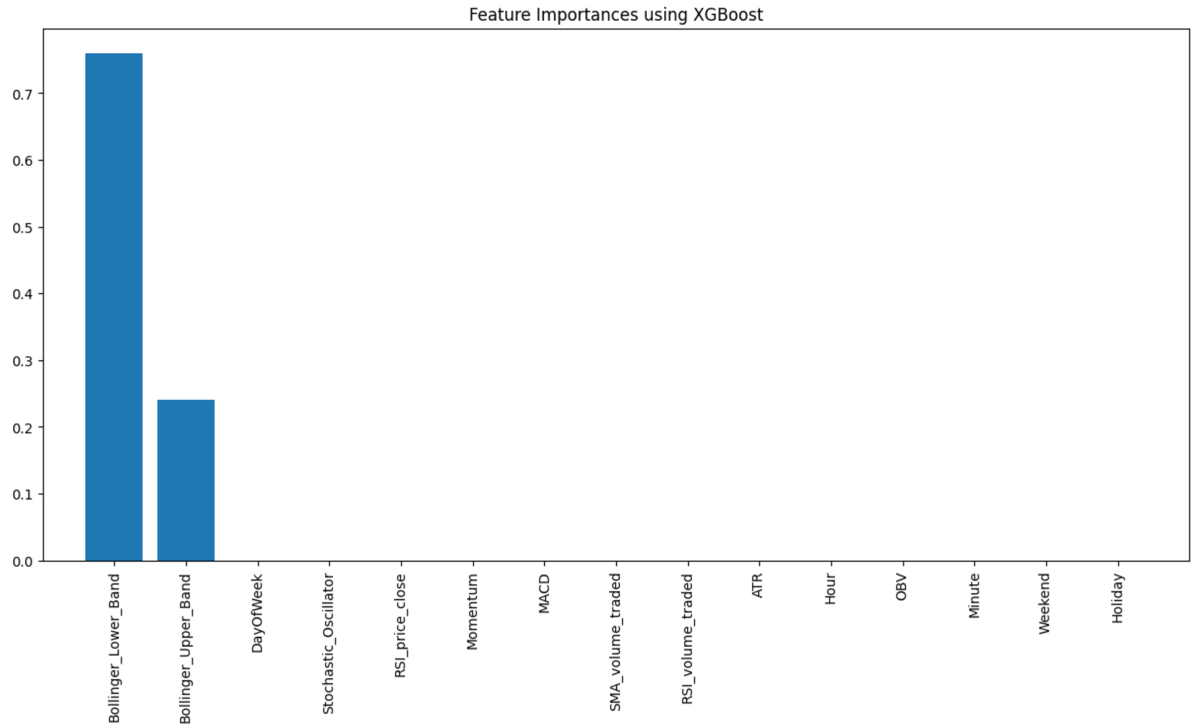

Figure 4. Feature Importances using XGBoost after delating Lagged price high, closed and low.

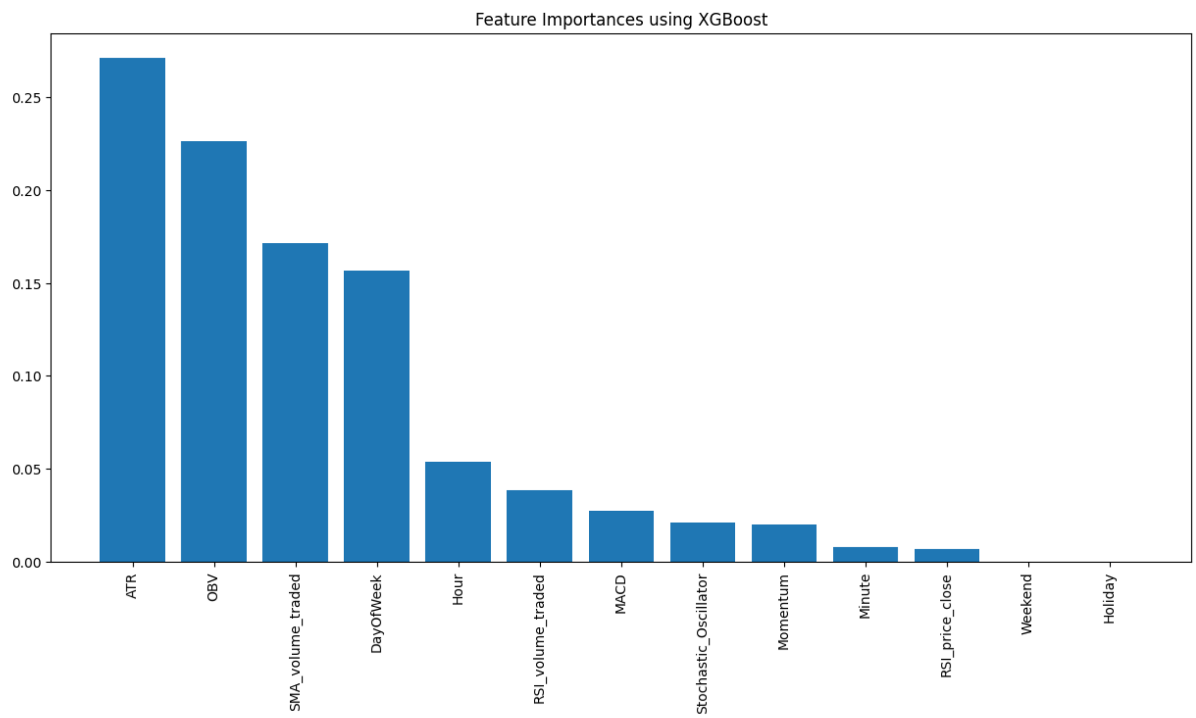

Figure 5. The feature relevance based on the XGBoost method and how we were choosing the best set of variables base on the graphs.

Through iterative feature selection, was identified that the optimal predictor set evolved across multiple stages of refinement. Initially, volatility indicators (ATR) and volume metrics (OBV, SMA_volume_traded) along with temporal features (DayOfWeek) provided the strongest signals. After removing redundant variables, Bollinger Bands emerged as predictors, likely capturing price boundaries and volatility regimes. In the final iteration, a carefully selected set of lagged price variables proved most effective, indicating that recent price history properly transformed to avoid multicollinearity contains the most valuable predictive information for Bitcoin price movements.

For this process of discarding and accepting variables, the flow described in Figure 6 follows a systematic sequence. The process begins with Data Extraction from the API as the starting point of the analysis pipeline.
Following data extraction, Data Exploration serves as the central hub of the analysis, where initial understanding of the dataset structure and characteristics is established.
From the data exploration phase, the workflow branches into several analytical components: Test for Correlation is conducted to identify relationships between variables and detect potential multicollinearity issues that could affect model performance.
The Augmented Dickey-Fuller test is applied to assess stationarity through p-value analysis. The resulting p-values determine whether the time series data requires transformation or differencing before further analysis. This step is critical because non-stationary data can produce misleading results in subsequent modeling.
Frequency Domain Analysis is performed to examine cyclical patterns and periodic behaviors within the time series data, providing insights into underlying temporal structures that may not be apparent in the time domain.
Finally, Variable Importance scoring using XGBoost is employed to identify which features have the strongest predictive power for the target variable. Variables demonstrating high importance scores are retained, while those with minimal impact are discarded from the model.

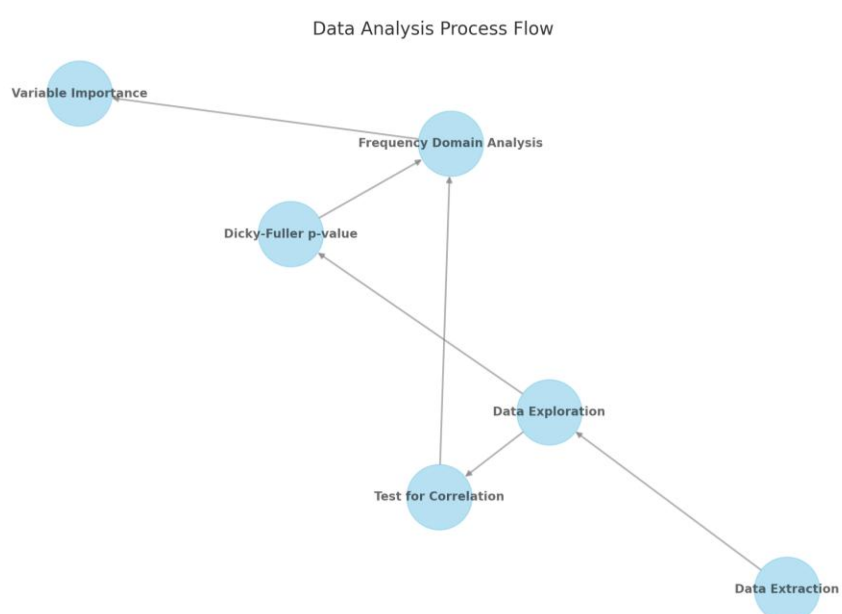

Figure 6. Iteration for discarding and recreating the features followed in this work.

3.1.3. Data Preprocessing

In this work, a sliding window technique will be used to prepare the input data for the LSTM model. This technique involves creating overlapping sequences of historical data, where each sequence is used to predict the next point in time.

The sliding window is implemented as follows:

1. A fixed window size is defined, which in our case is 1440 for minute-by-minute data, representing a full day of trading.
2. We iterate over the data, creating sequences where each contains observations from the previous 1440 minutes.
3. Each sequence is used as input to predict the value of the next minute.

This approach is adopted for several reasons:

1. Capturing temporal patterns: It allows the LSTM model to learn from patterns that develop over time, crucial in the 24/7 cryptocurrency market.
2. Consistency with the sequential nature of LSTMs: LSTM models are designed to process sequential data, and this method provides exactly that type of input.
3. Flexibility in prediction: It allows predictions at different time horizons by adjusting the window size.

The resulting format of the input data is [N, 1440, X], where N is the number of sequences, 1440 is the number of time steps in each sequence (corresponding to a full day), and X is the number of input features/variables per time step.

3.2. Model

The model proposed in this work is a deep learning architecture integration Convolutional Neural Networks (CNN) with Bidirectional Long Short-Term Memory Networks (Bi-LSTM), enhanced by a scaled dot-product attention mechanism. This hybrid architecture is specifically designed to capture both spatial patterns through convolutional layers and temporal dependencies through recurrent layers. The attention mechanism allows the model to focus on specific moments in the time sequence that are most relevant for prediction, significantly improving its predictive capability in the volatile cryptocurrency market. This comprehensive approach facilitates the detection of complex patterns in Bitcoin historical data, providing more accurate predictions of future price trends.

Architectural Components:

1. Input Layer: Accepts time sequences of multiple financial indicators and temporal variables.

2. Convolutional Layer: A 1D convolutional layer for initial feature extraction, followed by max-pooling to reduce dimensionality.
3. Bi-LSTM Layers: Two layers of Bidirectional LSTM units to capture forward and backward dependencies in the time sequence.
4. Attention Mechanism: Scaled dot-product attention to weigh the importance of different time steps.
5. Dense Layers: Fully connected layers for final feature representation and prediction.

Input Variables:

- Historical Bitcoin prices (Open, High, Low, Close)
- Trading volume
- Technical indicators (e.g., RSI, MACD)
- Temporal features (e.g., day of week, hour of day)

Output:

- Predicted Bitcoin price for the next time step

To evaluate the performance of the proposed model, it was implemented several baseline models for comparison that can be seen on Figure 14 from chapter 5.2:

1. Linear Regression
2. Random Forest
3. Gradient Boosting

These models were trained and tested on the same dataset as our proposed CNN-BiLSTM model with attention. The performance metrics of all models were compared to demonstrate the effectiveness of our approach.

3.2.1. Model implementation

The model is implemented using TensorFlow and Keras libraries, providing a seamless experience for model development, training, and deployment. Specialized Keras layers and functionalities are utilized to implement the advanced hybrid architecture combining CNN and Bi-LSTM components.

The model employs Keras Tuner's RandomSearch method to perform automated hyperparameter optimization across a predefined search space. Key hyperparameters include:

- Learning rate: optimized within the range $[1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}]$
- Number of LSTM units: ranging from 20 to 120 neurons
- Dropout rates: systematically tested for optimal regularization
- Batch size: evaluated for optimal training efficiency

A hybrid loss function is implemented, combining MSE with additional penalties for extreme value predictions to improve model robustness and prevent outlier-driven training instability. Multiple neural network architectures were systematically evaluated, varying in number of layers (both convolutional and recurrent), units per layer configuration, activation functions and regularization techniques

The hyperparameter search space was carefully defined to balance computational efficiency with thorough exploration. This systematic approach ensures that the model configuration is optimized for the specific characteristics of Bitcoin price prediction while maintaining generalization capabilities.

Following the initial hyperparameter optimization phase, the best-performing architecture was selected based on validation performance metrics and subsequently fine-tuned for the final implementation.

3.2.2. Training

The model training process employs a comprehensive strategy designed to optimize performance while preventing overfitting and ensuring robust convergence.
Training is carried out using the Nadam optimizer, an enhanced variant of the Adam optimizer that incorporates Nesterov momentum. This optimizer was selected for its superior performance in handling the non-stationary nature of financial time series data and its ability to accelerate convergence in sparse gradient scenarios.

Training Parameters

- **Batch Size**: 32 samples per batch.
- **Maximum Epochs**: 300 epochs.
- **Initial Learning Rate**: Optimized through hyperparameter tuning ($\text{range: } 1 \times 10^{-5} \text{ to } 1 \times 10^{-3}$)

Early Stopping

The training process implements early stopping to prevent overfitting:

- **Monitoring Metric**: Validation loss.
- **Patience**: 5 consecutive epochs without improvement.
- **Restoration**: Best model weights are automatically restored when training stops.

Learning Rate Reduction

An adaptive learning rate reduction strategy is employed to fine-tune the optimization process:

- **Trigger Condition**: Validation loss plateaus for a specified number of epochs
- **Reduction Factor**: Learning rate is multiplied by 0.2.
- **Minimum Threshold**: 1×10^{-4} to prevent excessively slow convergence.

- **Rationale**: Allows the optimizer to take smaller steps in flat loss landscape areas while maintaining computational efficiency.

Regularization Techniques
Multiple regularization methods are applied to prevent overfitting:
- **L1 and L2 Regularization**: Applied to LSTM layers with hyperparameters optimized through Keras Tuner.
- **Dropout**: Implemented in dense layers to reduce co-adaptation of neurons.
- **Batch Normalization**: Applied to stabilize training and improve convergence.

Model Checkpointing
- **Criterion**: Best validation loss performance.
- **Frequency**: Every epoch where validation performance improves.
- **Purpose**: Ensures preservation of optimal model weights.
-

Learning Rate Scheduling
- **Type**: ReduceLROnPlateau scheduler.
- **Monitoring**: Validation loss.
- **Purpose**: Dynamic adjustment of learning rate based on training progress.

The model employs mini-batch stochastic gradient descent, processing data in batches of 32 samples. This approach balances computational efficiency with gradient estimate quality.

Batch Size Considerations
Experimental observations revealed important relationships between batch size and model performance:
- **Small Batch Sizes**$(< \mathbf{32})$: Led to early overfitting due to high variance in gradient estimates and insufficient pattern capture in the temporal sequences
- **Large Batch Sizes**$(> \mathbf{64})$: Demonstrated better pattern recognition capabilities, particularly for capturing long-term temporal dependencies in the time series data
- **Optimal Range**: Batch size of 32 was selected as it provided the best balance between training stability and computational efficiency

3.2.3. Inference

Once the model is trained and validated, it is deployed for real-time Bitcoin price prediction. The model takes a sequence of the latest financial indicators and temporal variables as input and outputs a predicted price, aiding in data-driven trading decisions.

MAPE measures the size of the errors in percentage terms. It’s the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

The R^2 metric provides an indication of the goodness of fit of a set of predictions to the actual values. In statistical literature, this measure is called the coefficient of determination.

The research team reduced the oscillation of big numbers of MAPE by adding a CNN in the model architecture, a learning rate scheduler for avoiding overfitting. Then a Time Distributed layer for the sensibility to n_steps (that are how many predicting do we need).

Prediction: The trained LSTM model was used to predict Bitcoin prices for a specific future date (e.g., coinapi.get_bitcoin_data $("2023-05-01T00:00:00", "2023-07-28T10:50:00")$. The LSTM model can make predictions by taking in a sequence of past data. For the specific predictions, the data for the past 'n' hours was fed into the model, which then outputs a prediction for the next minute.

Post-processing of Predictions: The predictions generated by the LSTM model were in the scaled domain (because the LSTM was trained on scaled data). To interpret these predictions, they were converted back to the original price scale. This was done using the inverse_transform function of the scaler that was initially used to scale the data.

Now, for the general overview of the work done, Figure 7 shows a simplification of the steps.

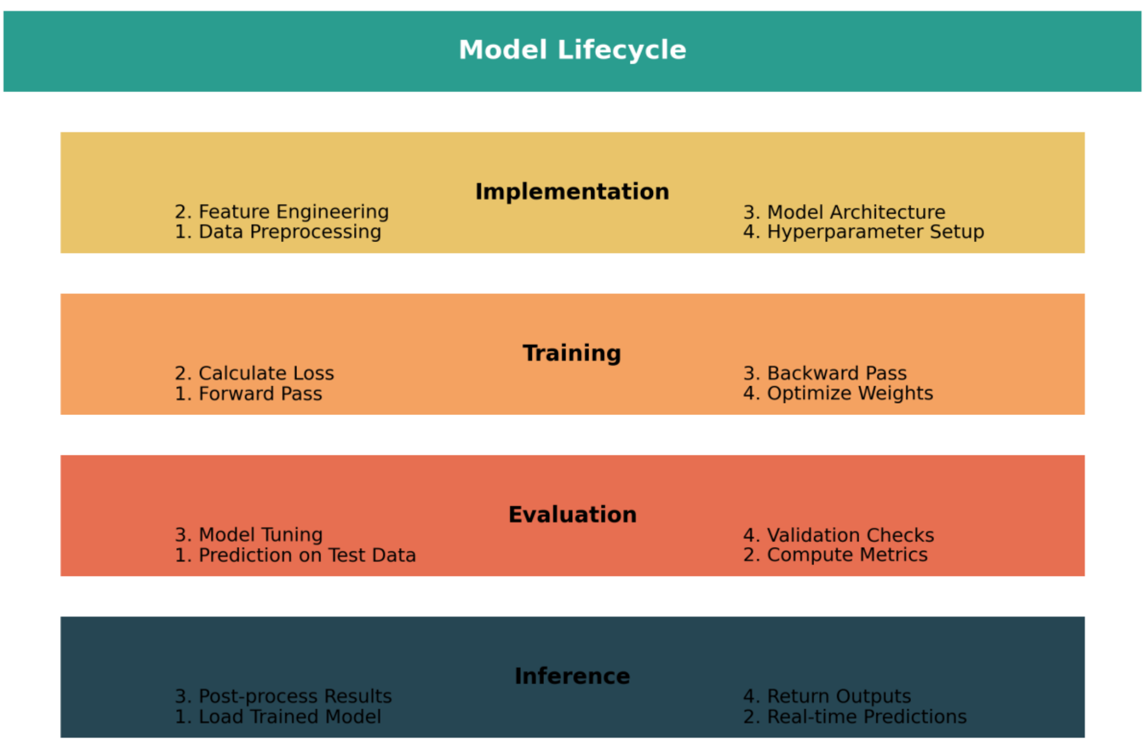

Figure 7. The methodology implemented (proper creation)

4. Results and Discussion

Bitcoin price prediction has emerged as a critical challenge in financial forecasting due to the cryptocurrency's inherent volatility and complex market dynamics. Traditional forecasting methods often struggle to capture the non-linear patterns and temporal dependencies present in high-frequency cryptocurrency data. This research presents a sophisticated deep learning approach that combines Convolutional Neural Networks (CNNs) with Bidirectional Long Short-Term Memory (Bi-LSTM) networks, enhanced by an attention mechanism, to predict Bitcoin prices at minute-level granularity. The hybrid architecture leverages the spatial feature extraction capabilities of CNNs and the temporal dependency modeling of LSTMs to achieve superior predictive performance compared to conventional baseline models. Through experimentation with various hyperparameters, including lookback periods and filter sizes, this study demonstrates the model's effectiveness in capturing both high-volatility market conditions and stable market periods, contributing to the advancement of cryptocurrency price forecasting methodologies.

1. **Performance Metric**: MAE, MSE, MAPE, RMSE for checking the model performance during train and test.
2. **Baseline Models**: Compare different models against the main one.
3. **Real Time Scenarios**: Predict 15 minutes) also plotting the results) and then see the actual plots in the web page of binance.
4. **Discussion**: Effects noticed during the results, predictions and training, the final model architecture and why LSTM domain over other models.

4.1. Performance Metrics

The LSTM model for predicting Bitcoin prices was evaluated using key metrics:

4.1.1. Loss (MSE)

Training MSE: The initial loss at the onset of training, 0.0493770, represents the model's initial struggle to capture the underlying patterns of the dataset. As the epochs progress, this loss undergoes a significant decline, signifying the model's increasing proficiency in understanding the intricacies of the training data. The consistent reduction in MSE indicates that the neural network's weight adjustments, facilitated by backpropagation, are effectively minimizing the gap between the predicted and actual outputs.

Validation MSE: The notably low starting point of 0.0030960 for the validation loss is suggestive of the model's immediate capacity to generalize its learnings from the training set to previously unseen data as can be seen in Figure 8 The relatively stable trajectory of the validation loss across epochs underscores the model's robustness and its resilience against overfitting.

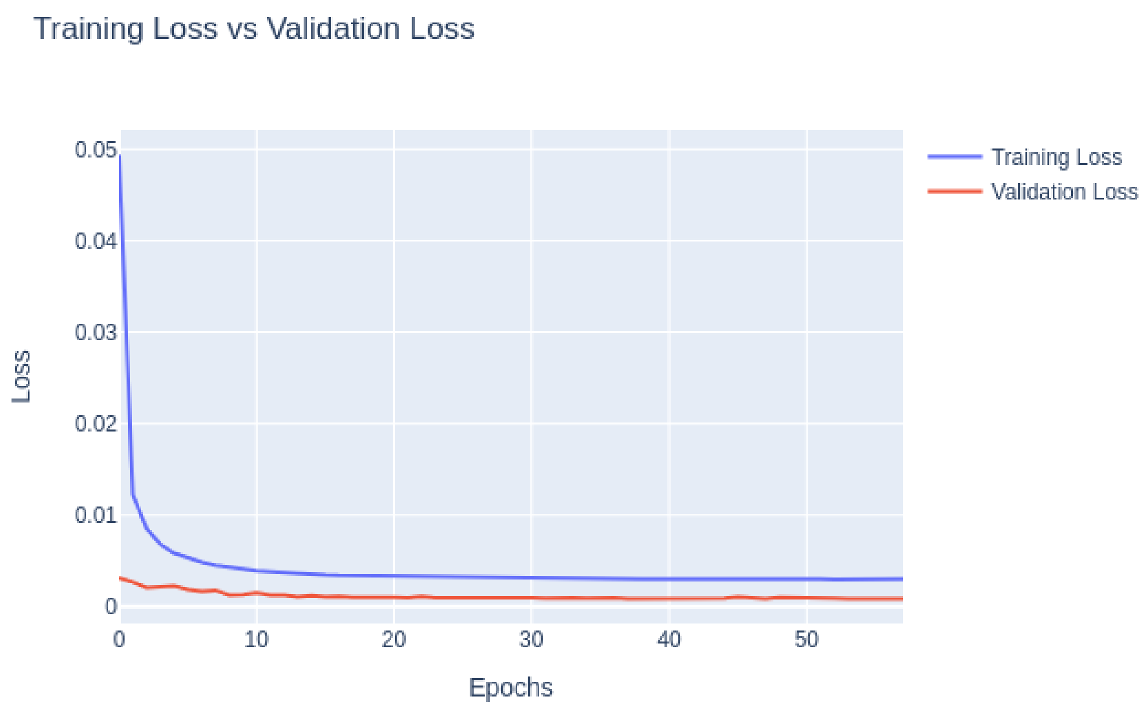

Figure 8. Training and loss validation from the model output.

4.1.2. Mean Absolute Error (MAE)

Training MAE: Beginning at 0.1736820, the gradual decrement in the training MAE denotes the model's growing precision in approximating actual values seen in Figure 9 As the absolute discrepancies between predicted and true outcomes shrink, it becomes evident that the model is increasingly aligning its predictions closer to the actual data distribution.

Validation MAE: The initial validation MAE of 0.0391870 is a testament to the model's capability to make predictions with minimal absolute deviations on the validation set. Its consistent low magnitude underlines the model's aptitude to maintain prediction accuracy on new datasets.

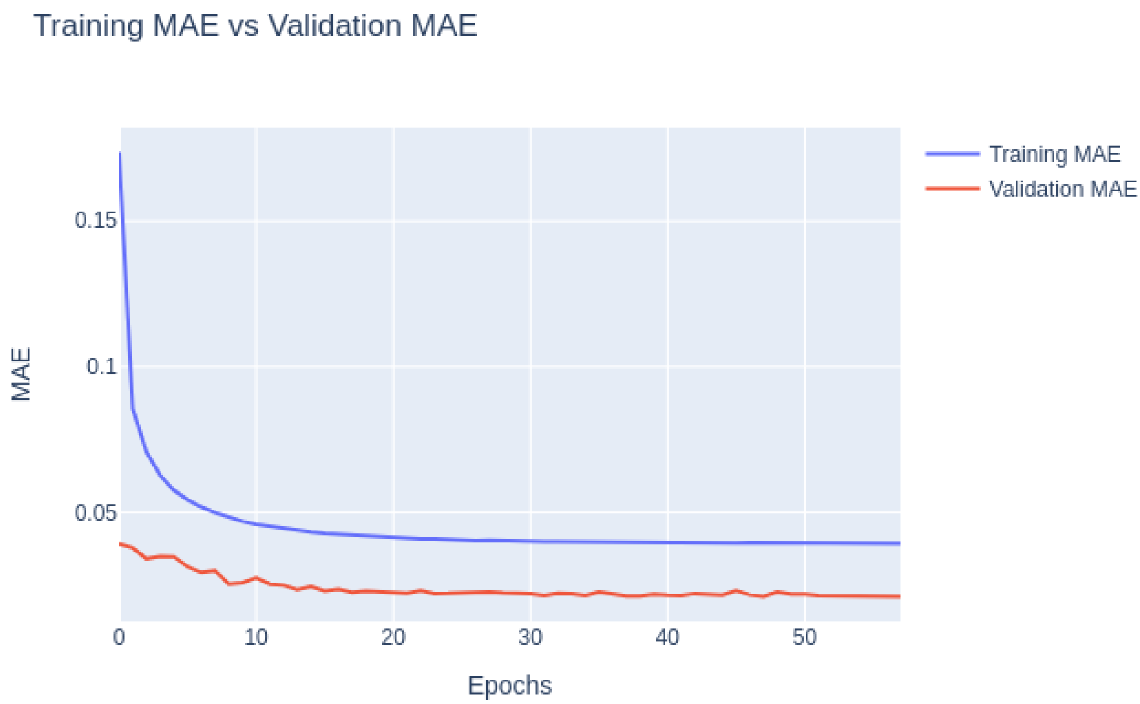

Figure 9. Training and validation MAE of the main model.

4.1.3. Mean Absolute Percentage Error (MAPE)

Training MAPE: The high initial value of MAPE denotes, in percentage terms, the extent of deviation between the model's predictions and actual values. As training progresses, this percentage error decreases, albeit with fluctuations. Such oscillations could be attributed to the model's continuous adjustments to understand non-linear patterns in the data.

Validation MAPE: The stark difference between training and validation MAPE, especially in initial epochs, accentuates the model's inherent capability to generalize well. A low MAPE for validation data is indicative of the model's predictions being close to actual values, in relative terms as can be seen in Figure 10.

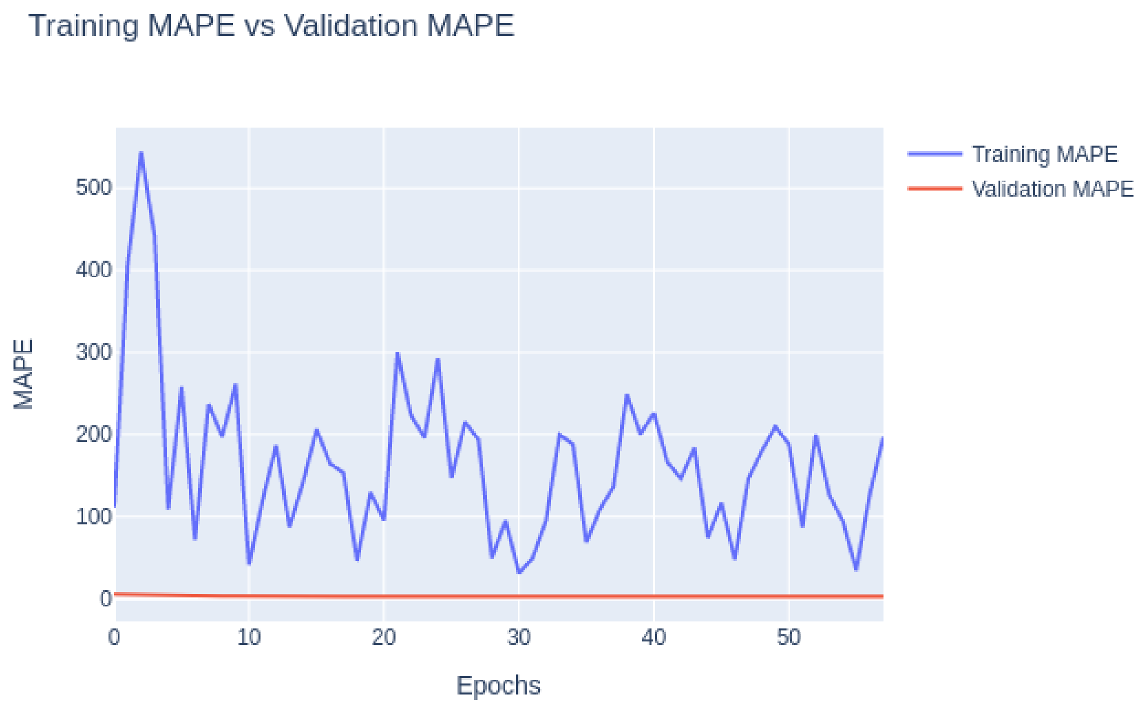

Figure 10. Training and validation data of the metric MAPE.

4.1.4. Root Mean Squared Error (RMSE)

Training and Validation RMSE: The RMSE, often considered a more sensitive metric due to its quadratic nature, effectively captures the model's prediction errors' magnitude. The diminishing RMSE values for both training and validation datasets reiterate the model's progressive refinement in capturing the underlying data dynamics. A lower RMSE is emblematic of a model's higher predictive accuracy, and its trajectory here is indicative of the LSTM network's efficacy in this temporal prediction task shown in Figure 11.

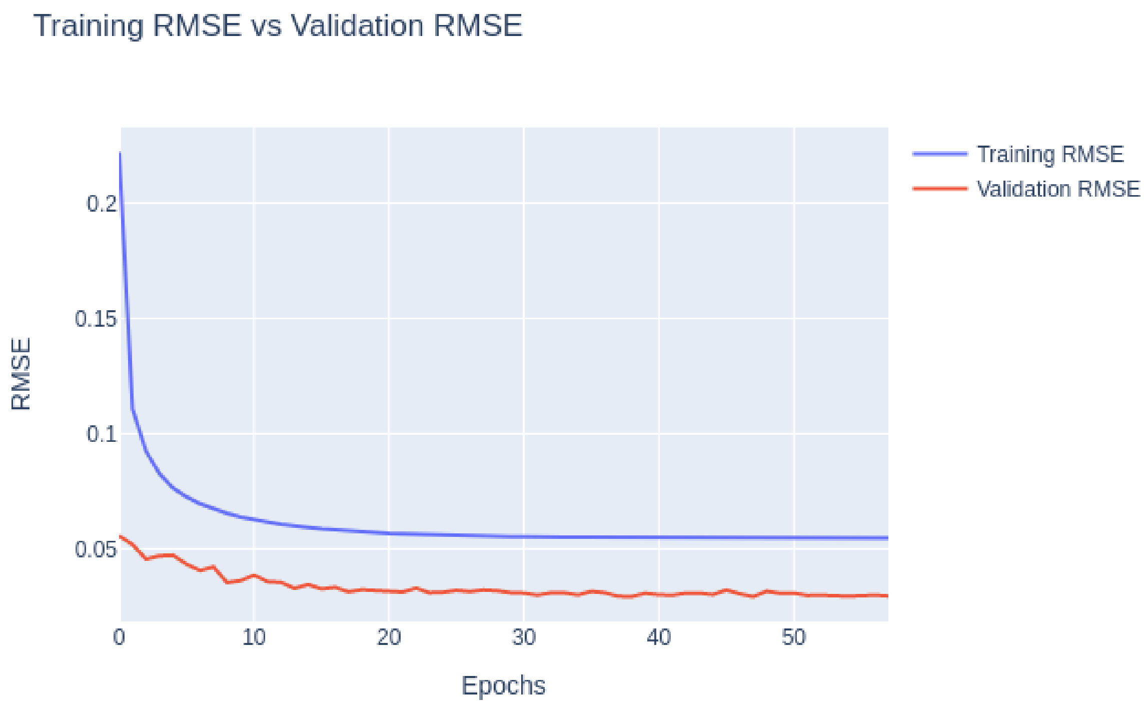

Figure 11. RMSE performance of the model.

4.1.5. Discussion

Based on the metrics, the model shows a relatively small error in its predictions. This suggests that the model has learned a significant portion of the underlying structure in the Bitcoin price data and can predict future prices with reasonable accuracy. However, there are a few things to keep in mind:

While the model demonstrates strong performance on test data, it must acknowledge the inherent complexity of financial markets. Cryptocurrencies like Bitcoin operate in an ecosystem characterized by extreme volatility, where prices can fluctuate dramatically in response to regulatory announcements, technological developments, macroeconomic shifts, and social media sentiment. This complex interplay of factors creates a challenging environment for any predictive model.

The transition from controlled testing environments to real-world deployment presents additional challenges. Although the model shows promising results in backtesting scenarios, its true resilience will only be revealed when faced with extreme market conditions that may not have been represented in our historical training data. This necessitates ongoing monitoring of the model's predictions and a commitment to regular retraining as new market data becomes available, ensuring the model remains calibrated to evolving market dynamics.

Looking ahead, we see several pathways for enhancing our model's performance. Fine-tuning hyperparameters could optimize the model's sensitivity to market patterns, while incorporating additional features such as sentiment analysis from financial news or social media could capture dimensions of market psychology doesn't present in technical indicators alone. Furthermore, exploring more sophisticated architectural approaches might unlock even greater predictive accuracy, particularly in capturing the non-linear relationships that define cryptocurrency markets.

In the grand tapestry of neural network training, the metrics paint a compelling narrative. The model, initiated with a rudimentary understanding of the data's complexities, gradually evolves, refining its predictive prowess. The consistent performance on validation data, juxtaposed against the training metrics, offers a reassuring testament to the model's robustness and its resistance to overfitting.
In the realm of deep learning, where the interplay of weights, biases, and activations crafts the predictive alchemy, these metrics serve as a compass, guiding us through the model's journey from naive approximations to sophisticated predictions. The model's commendable performance could be further enhanced by delving deeper into hyperparameter optimization, exploring alternate architectures, or incorporating advanced regularization strategies.

4.2. Baseline models

Figure 12. Graphs of comparing the baseline models against our main model.

The model implemented during this work showed in Figure 12 has the lowest MAE and RMSE, indicating its superior performance compared to the baseline models (gradient boosting, random forest, Decision Tree and so on).

Among the baseline models, Linear Regression and Random Forest tend to perform better in terms of MAE and RMSE, respectively.

Gradient Boosting has the highest errors, suggesting that it might not be the best choice for this specific dataset or task without further tuning.

While the LSTM-based model demonstrated superior performance, it inherently possesses greater complexity due to its architecture. Such intricacy might render it less interpretable than simpler models like Linear Regression.
Linear Regression, despite its simplicity and interpretability, can't capture the non-linear patterns in the data as effectively as models like Random Forest or LSTM, which could explain its inferior performance in this context.
Random Forest, being a collection of decision trees, can capture complex patterns and provide a balance between performance and interpretability. Its ability to rank features based on their importance can offer insights into which variables most influence Bitcoin prices.

The LSTM's superior performance on the validation set suggests that it can generalize well to unseen data, making it a robust choice for this dataset.
Random Forest's inherent bootstrapping and bagging techniques make it less prone to overfitting, which might explain its decent performance in comparison to Gradient Boosting.
Gradient Boosting, if not carefully regularized, can sometimes be overfit to the training data, which might be a factor in its higher validation errors.

The LSTM model's ability to incorporate sequence-based information gives it an edge in predicting time series data. This temporal understanding is something traditional models like Linear Regression lack.
Random Forest and Gradient Boosting can handle a mix of numerical and categorical features without much preprocessing. However, their inability to inherently understand temporal sequences could be a limitation in time series forecasting.

While Gradient Boosting underperformed in this context, it shouldn't be dismissed outright for other tasks. With the right feature engineering and hyperparameter tuning, it has the potential to outperform many other algorithms.
Considering ensemble techniques, which combine predictions from multiple models, could be a future direction to achieve even better performance. Combining the strengths of models like LSTM and Random Forest might yield a more robust and accurate predictor.

4.3. Real time scenarios

During the test it was noticed that looking back parameter adjusted in 30 minutes was sensitive for a high volatility market (when the sell/buy movement is on constant change in a single minute) been able for detecting the violent uppers and downs. This phenomenon can be seen in Figures 13 and 14 also in 15 and 16.

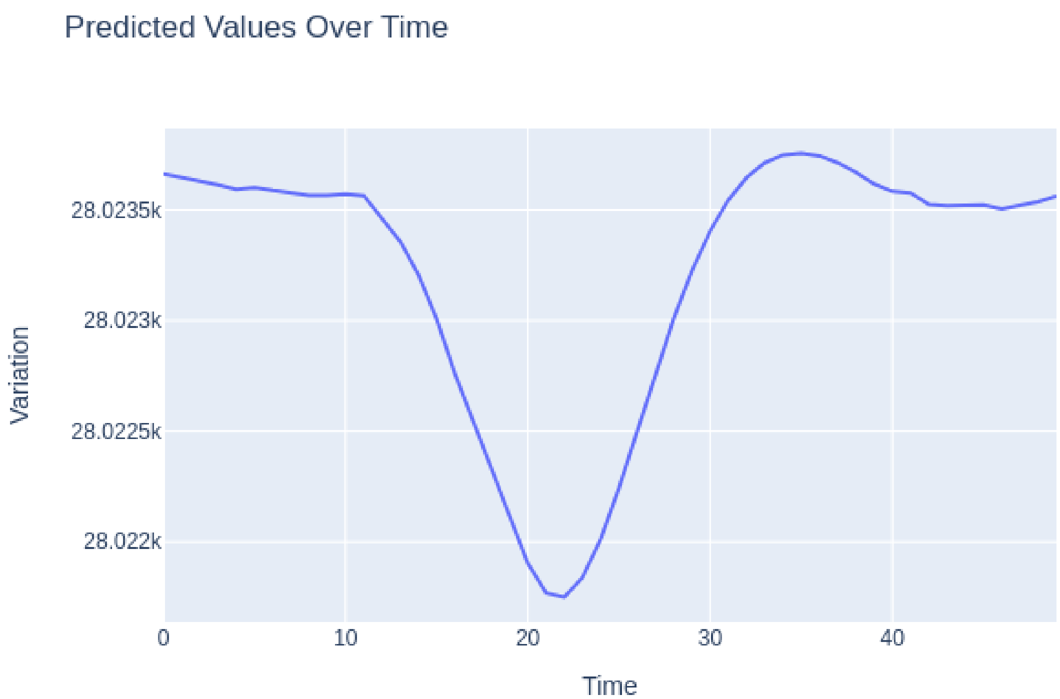

Figure 13. Prediction of the model for the next 50 minutes.

Figure 14. Binance's trading view [35] candle plot of the period time forecasting in the figure 5.6

Meanwhile a lookback of 60 minutes can capture the patterns of a stable market that sustain large uppers or downs that can be observed in the Figures 15 and 16

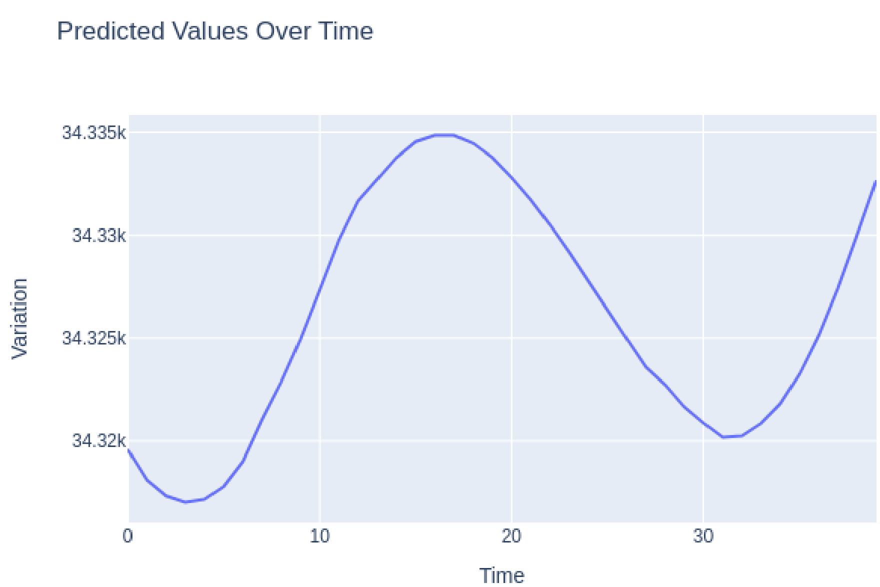

Figure 15. Prediction of the model for the next 35 minutes.

Figure 16. Binance trading view [35] date respect the tendencies predicted in the graph 5.7.

This was noticed using the same feature engineering and model architecture. The main change in the preprocessing part where the data is gives the array structure in the look back hyperparameter at 30 minutes or 60 minutes.

The LSTM model's predictions can be visualized to see if they align with the expected Elliott Wave patterns. If the model's predictions show a 5-3 wave pattern, it could indicate that the model has captured the natural rhythms present in the Bitcoin market.

The lookback parameter's effect on the model's performance can be related to how far back in the wave cycle you need to go to accurately predict the next wave [33]. For instance, if a lookback of 30 (representing a month of data) provides the best results, it might indicate that the most recent month's data is the most relevant in predicting future price movements, aligning with the idea that the most recent waves (or cycles) are the most indicative of future movements.

Table 5 shows the results of a training variation in the filters sections of the convolutional neural networks.

Epochs	Validation Loss	Validation MAE	Validation MAPE	R^2	Accuracy
10	0.001722	0.033702	4.110091	0.898653	0.958899
20	0.000799	0.023768	2.983370	0.931672	0.970166
30	0.001112	0.028373	3.559777	0.922300	0.964402
40	0.001963	0.033884	4.046715	0.892141	0.959533
50	0.000687	0.021484	2.681084	0.942778	0.973189
60	0.001452	0.030971	3.802332	0.905259	0.961977
70	0.001206	0.030195	3.800965	0.890715	0.961990
80	0.000867	0.025325	3.189284	0.940052	0.968107
90	0.001184	0.027691	3.404013	0.929436	0.965960
100	0.000897	0.024279	2.995291	0.916399	0.970047

Table 5: Sample of evaluation metric during training per epoch.

As the number of epochs increases, the model iteratively refines its internal parameters to better capture the underlying patterns in the Bitcoin price data. This iterative refinement is evident in the decreasing trend observed in the validation loss, MAE, and MAPE metrics. The marginal fluctuations in these metrics across epochs underscore the challenges in predicting volatile financial data and the inherent randomness in financial markets.

Validation Loss, Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) took place while predicting the exact future price of Bitcoin is inherently impossible due to market complexities. What's remarkable about this model is its ability to accurately capture price tendencies and directional movements. As demonstrated in the prediction figures, the model successfully identifies upward and downward trends with high precision. The predictions closely track actual price movements, with the model adapting effectively to different market conditions through its adjustable lookback periods - 30 minutes for volatile markets and 60 minutes for more stable conditions.

This close alignment between predicted and actual price tendencies makes the model particularly valuable for trading decisions, where understanding the direction of price movement often matters more than the exact price point. The model's low error metrics translate to reliable directional guidance that can inform effective trading strategies in the volatile cryptocurrency market.

By the 100th epoch, the model seems to have achieved a balance between learning from the training data and generalizing the validation data. This is evident in the convergence of metrics like validation loss and MAE. However, the slight variations in metrics like MAPE underscore the complexities inherent in predicting financial time series data.

Table 6 has the difference in the model architecture of the convolutional neural networks of the filters, where change from 256 and 512 to 32 and 128 respectively.

Epoch	Validation Loss	Validation MAE	Validation MAPE	R^2	Accuracy
10	0.001025	0.023624	2.826907	0.901141	0.971731
20	0.001413	0.028624	3.422966	0.874705	0.965770
30	0.001357	0.028704	3.465295	0.886363	0.965347
40	0.000696	0.019887	2.416371	0.941952	0.975836
50	0.000904	0.023382	2.841277	0.919490	0.971587
60	0.000963	0.022562	2.705092	0.933600	0.972949
70	0.000988	0.025441	3.132796	0.910997	0.968672
80	0.001679	0.032385	3.909254	0.857223	0.960907
90	0.000812	0.021795	2.655726	0.931931	0.973443
100	0.001560	0.029292	3.478340	0.864825	0.965217

Table 6: Set of values from the other configuration of the model architecture.

The optimal model for Bitcoin price prediction is the hybrid CNN-LSTM architecture with larger filter sizes (256 and 512), trained for precisely 40 epochs. This model demonstrates superior performance across all key evaluation metrics compared to traditional baseline models like Linear Regression, Random Forest, and Gradient Boosting.

The model achieves its peak performance at epoch 40 with the following results:

- Validation Loss (MSE): 0.000696
- Mean Absolute Error (MAE): 0.019887
- Mean Absolute Percentage Error (MAPE): 2.416371%
- Accuracy: 0.975836 (97.58%)

These metrics indicate that the model can explain approximately 94.2% of the variance in Bitcoin price movements while maintaining prediction errors below 2.5% on average. The model begins to show signs of overfitting if trained beyond 40 epochs, as evidenced by the noticeable deterioration in performance metrics at epoch 80.

The proposed architecture offers significant advantages, as the combination of convolutional layers for spatial pattern recognition with LSTM layers for capturing temporal dependencies results in a powerful hybrid model capable of identifying both short-term price fluctuations and long-term market trends. Regarding filter optimization, larger filters (256 and 512 units) outperform smaller ones (32 and 128), suggesting that larger filters are more effective in capturing macro patterns in Bitcoin price movements. Similarly, tuning the lookback parameter enhances the model's performance by adapting to market volatility: a 30-minute window is recommended for highly volatile markets, while a 60-minute window is more suitable for stable conditions. To strengthen the quality of input data, feature engineering was carried out through rigorous testing with XGBoost and correlation analysis, ensuring the use of only the most relevant financial indicators while avoiding noise from redundant variables. Finally, it was found that larger batch sizes benefit the model by enabling it to capture broader patterns in the data and reducing the risk of being misled by small and potentially noisy subsets.

4.4. Discussion

Despite careful data preparation, the model was affected by noise introduced by certain features, as indicated by XGBoost's feature importance rankings. Four main categories of problematic features were identified. First, scale variations were present in variables such as OBV and SMA_volume_traded, which operated on scales drastically different from price-based indicators like SMA_price_close and Momentum, potentially skewing predictions even after normalization. Second, range-limited indicators, such as RSI_price_close and the Stochastic Oscillator, which are bounded between 0 and 100, created difficulties when the model failed to account for these constraints. Third, sensitivity issues were observed in relative measures like ROC, Momentum, and MFI, which displayed heightened responsiveness to price fluctuations and disproportionately influenced outputs when dominant in the learning process. Finally, smoothing effects introduced by moving averages (SMA_price_close, MA_50, MA_200) risked masking short-term variations essential for accurate prediction. These findings highlight the importance of not only preparing features correctly but also understanding how their inherent characteristics influence the learning dynamics of neural networks in financial forecasting tasks.

5. CONCLUSIONS AND FUTURE WORK

5.1. Conclusions

Despite the model's strong performance, the researchers acknowledged the inherent unpredictability of financial markets and the importance of staying within theoretical frameworks like Core Market [32]. Cryptocurrencies are influenced by a myriad of factors beyond historical price data, including regulatory changes, technological advancements, and market sentiment. While the model provides valuable insights, it should be considered as one tool among many in the decision-making process of traders and investors.

The LSTM-based approach to Bitcoin price prediction offers a solid foundation for understanding market trends based on historical data. As the field of cryptocurrency forecasting continues to evolve, the insights gained from this study contribute to the ongoing dialogue about the most effective methods for navigating these volatile markets. The future of financial forecasting likely lies in the thoughtful combination of various approaches, each offering unique strengths in unraveling the complex tapestry of market behavior.

The selection for the variables during the feature engineering not only depends on the 7 postulates of the assumptions of linear regression (linearity, independence, homoscedasticity, multivariate normality, no multicollinearity, no autocorrelation, no endogeneity), since they could be good variables been evaluated with those parameters but can cause noise for others during the inference part.

The hyperparameter look back, that is how many minutes the model looks at the past is also important for dealing with the market volatility since a higher value worse against this event describe.

The batch size played an important role for the learning curve of the model, since small chunks of data lost some patterns of the own data.

The number of epochs needed for a model that stays outside under and overfitting, turns out to be 40, this number was estimated doing multiple training. This is also backed up when the inferences were made, a model with less of 40 epochs gave results indicating underfitting seen as peaks during inferences, meanwhile, more epochs resulted in a straight-line indicating overfitting.

CNN importance aims to avoid the problem of overfitting, since the bigger the filters, the better inferences we had. During experimentation this phenomenon was noted, where filters of 256 units or bigger were useful.

LSTM configuration was determined by experimentation with a large range of parameters, the best model was taken and then reduced the area for the experimentation. Also adjust the hyperparameter like dropout and how many neurons are in the LSTM with the Nadam optimizer. Finding this configuration took most of the simulation time even having batch training.

Macroevents and microevents were another important point that became one of the main explanations for the observation observed during inferences and how to adjust the model.

All these features are considered for a model which deals with the trend of bitcoin, using it for another cryptocurrency could require adjusting.

This model was on static way, a dynamic model could add some interesting things for analyzing the market like 3 minutes into the future.

Looking ahead, several avenues for further research and improvement were identified. Exploring the integration of sentiment analysis, as mentioned in some of the literature reviewed, could provide additional insights into market dynamics. Furthermore, investigating the potential of more advanced architectures, such as Transformer models, in the context of cryptocurrency price prediction could yield interesting comparisons with the LSTM-based approach.

5.2. Future work

Looking ahead, several avenues for further research and improvement were identified. Exploring the integration of sentiment analysis, as mentioned in some of the literature reviewed, could provide additional insights into market dynamics. Furthermore, investigating the potential of more advanced architectures, such as Transformer models, in the context of cryptocurrency price prediction could yield interesting comparisons with the LSTM-based approach.

Implement a coefficient which comes from sentiment analyze using news, for that could be an API that gets the financial data news per minute using the actual dates from the dataframe used, then cleaning, applying an embedding like finBert (specialize in financial text), then doing TF-IDF for getting the main topics discussed in those news and a LDF that allows to classify that news in categories (the 3 main positive, neutral and negative with some dummies which deals with news outside those 3 categories).

6. BIBLIOGRAPHY

[1] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, "Data preprocessing for supervised leaning," International Journal of Computer Science, vol. 1, no. 2, pp. 111-117, 2006.

[2] Forecasting Bitcoin Price Using Machine Learning (2022) by Md. Ashraful Alam, Md. Saiful Islam, and Md. Ashikuzzaman

[3] Syed Abul Basher a , Perry Sadorsky b,∗ a Department of Economics East West University, Aftabnagar Main Road, Dhaka, 1219, Bangladesh b Schulich School of Business, York University, 4700 Keele Street, Toronto, Ontario, Canada M3J 1P3

[4] Sahana N. B, Haritha GB. (2023). Cryptocurrency Price Prediction using Twitter Sentiment Analysis. Retrieved from arXiv.

[5] Lisitsyn, N., Gervais, A., Bilokon, P., & Barez, F. (2023). Exploring the Advantages of Transformers for High-Frequency Trading. Retrieved from arXiv.

[6] Frost, A. J., & Prechter, R. R. (1985). "Elliott Wave Principle: Key to Market Behavior." New Classics Library.

[7] G.E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, "Time series analysis: forecasting and control," John Wiley & Sons, 2015.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

[10] J. J. Murphy, "Technical analysis of the financial markets: A comprehensive guide to trading methods and applications," Penguin, 1999.

[11] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 2267-2273, 2018.

[12] CoinAPI.io Documentation, "WebSocket API | EMS API," CoinAPI.io, 2022.

[13] P. S. P. Cowpertwait and A. V. Metcalfe, "Introductory Time Series with R," Springer, 2009.

[14] E. P. Box, G. M. Jenkins, and G. C. Reinsel, "Time Series Analysis: Forecasting and Control," John Wiley & Sons, 2013.

[15] D. A. Dickey and W. A. Fuller, "Distribution of the Estimators for Autoregressive Time Series with a Unit Root," Journal of the American Statistical Association, vol. 74, no. 366a, pp. 427-431, 1979.

[16] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Mathematics of Computation, vol. 19, no. 90, pp. 297-301, 1965.

[17] R. N. Bracewell, "The Fourier Transform and its Applications," McGraw-Hill, 1986.

[18] J. P. Cunningham and M. Y. Byron, "Dimensionality reduction for large-scale neural recordings," Nature Neuroscience, vol. 17, pp. 1500-1509, 2014.
[19] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
[20] D. Louppe et al., "Understanding variable importances in forests of randomized trees," Advances in Neural Information Processing Systems, pp. 431–439, 2013.
[21] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," Journal of Machine Learning Research, vol. 3, pp. 1157–1182, 2003.
[22] J.H. Friedman, "Greedy function approximation: A gradient boosting machine," Annals of Statistics, pp. 1189–1232, 2001.
[23] A. Blum and P. Langley, "Selection of relevant features and examples in machine learning," Artificial Intelligence, vol. 97, no. 1-2, pp. 245–271, 1997.
[24] T. Chen, T. He, M. Benesty, V. Khotilovich, and Y. Tang, "xgboost: eXtreme Gradient Boosting," R package version 0.6-4, 2015.
[25] S. Ribeiro, S. Singh, and C. Guestrin, "Model-agnostic interpretability of machine learning," arXiv preprint arXiv:1606.05386, 2016.
[26] M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," Advances in Neural Information Processing Systems, pp. 4765–4774, 2017.
[27] R. Kohavi and G. H. John, "Wrappers for feature subset selection," Artificial Intelligence, vol. 97, no. 1-2, pp. 273-324, 1997.
[28] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE transactions on neural networks, vol. 5, no. 2, pp. 157-166, 1994.
[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.
[30] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques. Elsevier, 2011.
[31] Binance, "Bitcoin Trading View," [Online], Available at: https://www.binance.com/en/trade/BTC_USDT [cited 2023 October 26].
[32] Malkiel, B. G., & Fama, E. F. (1970). EFFICIENT CAPITAL MARKETS: A REVIEW OF THEORY AND EMPIRICAL WORK. The Journal of Finance, 25(2), 383–417. doi:10.1111/j.1540-6261.1970.tb00518.x
[33] Neely, G., & Hall, E. (1990). Mastering Elliott Wave: Version 2.0 (2nd ed.). Windsor Books.