Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operations Research

Enhancing Bitcoin Price Predictions: The Role of Denoising Autoencoders for Noise Reduction and Feature Selection

Author:	Luca Leimbeck Del Duca (591608)
Supervisor:	M. Mueller
Second assessor:	M. Grith
Date final version:	1st July 2024

Abstract

Bitcoin, a highly volatile financial asset, presents significant forecasting challenges due to its speculative nature and dependence on various factors. This paper proposes a two-stage framework to improve Bitcoin price prediction using three types of predictors: fundamental indicators, technical indicators, and lagged prices. In the first stage, a Denoising Autoencoder is employed to minimize noise and reduce the subset of potential predictors. To assess the importance of this model, its results are compared to those obtained using a traditional Autoencoder and no feature selection. In the second stage, these predictors are input into linear regression, Support Vector Regression, and Long Short-Term Memory models. Using data from August 2011 to May 2024, we evaluate the models based on mean absolute error, root mean squared error, and mean absolute percentage error, against a simple moving average benchmark. Our findings show that Long Short-Term Memory models exhibit greater performance than the other models. Furthermore, the Denoising Autoencoder significantly enhances model performance, validating its effectiveness in noise reduction and feature selection.

Contents

1 Introduction

Due to its anonymous, decentralized, inexpensive, and rapid peer-to-peer transaction system, the cryptocurrency market has become popular in recent years. Bitcoin, the most well-known example of a decentralized virtual currency, has seen tremendous growth. According to CoinMarketCap, as of June 2024 the total market capitalization of cryptocurrencies surpasses $2 trillion, with Bitcoin alone accounting for over 50% of the market share. Bitcoin differs from traditional fiat currencies issued by central banks as it is generated through calculations on a decentralized network known as the blockchain. Its rapid growth, fueled by increasing acceptance and adoption by major financial institutions and corporations, such as Tesla, which announced a $1.5 billion investment in Bitcoin in February 2021[1], has attracted significant interest from investors and traders. However, Bitcoin is characterized by extreme volatility and susceptibility to sudden price spikes and market manipulations. For instance, on November 6, 2021, Bitcoin's price fell over 30% within 24 hours after reaching a local all-time high of $64,000. Additionally, despite being relatively young, the cryptocurrency market has experienced several market cycles (bull and bear runs), shifts in public attention, and technological advancements. The determinants influencing Bitcoin's price in one phase might not necessarily influence subsequent phases. This volatility and uncertainty pose significant challenges for price forecasting models to accurately predict future prices, leading to the release of a plethora of papers that focusing on this problem..

Wang et al. (2012) sstate that the different price prediction approaches employed in the literature can be divided into statistical models and machine learning models. Preliminary studies primarily focused on applying statistical and econometric models to understand Bitcoin's price movement. Examples include the Error Correction Model (ECM) used by Wang et al. (2016), the Vector Autoregressive model (VAR) employed by Panagiotidis et al. (2018), and the autoregressive integrated moving average model (ARIMA) utilized by Abu Bakar and Rosbi (2017). However, Cheng et al. (2010) find that these approaches are only effective when dealing with linear problems. To address this limitation, recent research has shifted its focus to machine learning models, which tend to perform well on non-linear data. McNally et al. (2018) support this claim by comparing ARIMA with deep learning methods and finding that ARIMA models generally perform poorly in capturing the nonlinear dynamics of Bitcoin prices.

Furthermore, Lahmiri and Bekiros (2019) examine the performance of the Long-Short Term Memory (LSTM) model and generalized regression neural networks (GRNN) in predicting the daily exchange rates of three virtual currencies: Bitcoin, Digital Cash, and Ripple. Additionally, Mallqui and Fernandes (2019) use several machine learning techniques to forecast the closing price of Bitcoin, finding that Support Vector Regression (SVR) yield the best results. Chen et al. (2021) further expand on this finding by comparing the predictive power of the LSTM against that of the SVR across four different market stages, concluding that the former consistently outperforms the latter.

Numerous studies investigate the factors influencing cryptocurrency prices. Chen et al. (2021) examine previous prices, as well as fundamental and economic indicators, and conclude

[1] CNBC. (2021). Tesla buys $1.5 billion in bitcoin

that fundamental and economic indicators exhibit higher predictive performance compared to simply using lagged prices. Tripathi and Sharma (2023) extend this analysis by incorporating technical indicators as predictors. They underscore the critical importance of appropriate input data selection, feature selection, and noise reduction for successful Bitcoin forecasting. While various machine learning models, such as artificial neural networks and random forest models (Chen et al. (2021), have been employed to measure the importance of price determinants and reduce the subset of potential predictors, the application of a denoising autoencoder for noise reduction and feature selection in cryptocurrency forecasting remains unexplored. Vincent et al. (2008) introduce the denoising autoencoder model and demonstrate its capability to learn robust features from noisy data, highlighting its suitability for financial time series forecasting.

Building on these assertions, this paper focuses on constructing a hybrid framework to significantly improve Bitcoin's price predictions. Our objective is to address the following questions: Can we successfully reduce the inherent noise present in predictors and determine the most relevant and coherent factor attributes? What is the optimum combination of price forecasting model and price determinant category? How does the performance of these models change across distinct market stages?

We examine three different types of features: fundamental indicators, technical indicators, and lagged values of past Bitcoin prices. Additionally, we analyze the full set of these features combined. For the one-day-ahead Bitcoin price prediction, we propose four methods, including machine learning models, such as Support Vector Regression and long-short term memory networks, as well as linear regression and a simple moving average as a benchmark. The hyperparameters for machine learning models are optimized using Bayesian optimization. Given the high dimensionality of the feature sets, we utilize a Denoising Autoencoder for feature selection. While this model is designed to maximize reconstruction quality and minimize noise interference, we also investigate whether the selected features improve the predictive performance of our models. To evaluate this, we compare the predictive performance of models using Denoising Autoencoder-selected features against those using a traditional Autoencoder and those using no feature selection technique. This comprehensive approach results in a total of 36 prediction methods, applied across each time interval. The forecasting ability of each method is assessed in an out-of-sample setting based on three performance metrics: mean absolute error, root mean squared error, and mean absolute percentage error.

This research contributes to the existing literature in two significant ways. Firstly, a Denoising Autoencoder is used to determine a robust subset of potential predictors by selecting important data attributes present in each considered set of price determinants. To our knowledge, this deep learning model has never been applied in the financial literature as a feature selection technique for Bitcoin price determinants. Secondly, we expand the literature by considering the full set of price determinant categories combined, which could be beneficial given that feature selection techniques are used, thus joining the best feature attributes from each individual set.

The data used in this paper spans from August 2011 to May 2024 and is divided into four distinct intervals, all of which correspond to significant periods in Bitcoin's history. In each interval, the last 15% of the daily data is used as the out-of-sample set for performing one-step-

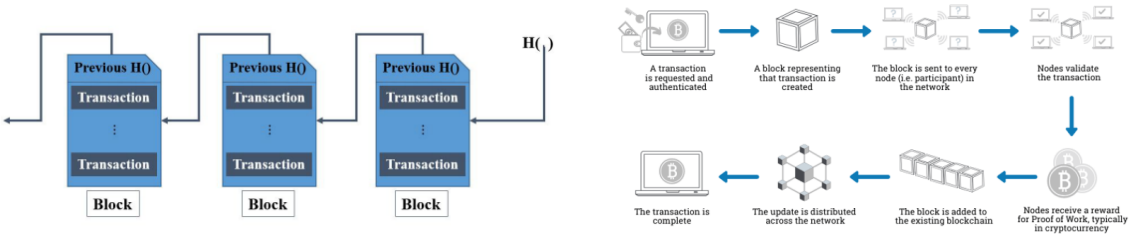
ahead forecasts. Our findings indicate that Support Vector Regression and Long Short-Term Memory models significantly outperform linear regression across all feature selection techniques. Furthermore, the use of an Autoencoder and a Denoising Autoencoder significantly improves model performance and shifts the best performing feature set from technical indicators to the combined full feature set. When comparing the two machine learning models, the Denoising Autoencoder demonstrates better predictive accuracy, further validating its noise reduction capabilities. Overall, the best performance across all metrics is consistently achieved by the Denoising Autoencoder combined with the full set of features. Conversely, the fundamental indicators feature set generally performs poorly, often failing to surpass the simple moving average.

The remainder of the paper is organized as follows. Section 2 presents the foundations of Bitcoin and a background review. In Section 3 we describe the data at hand, followed by a discussion of our methods in Section 3. In Section 4 we apply these models to the data and discuss the results. Finally, Section 5 summarizes our findings and concludes the paper.

2 Background Information

Cryptocurrencies, which are decentralized digital currencies secured by cryptography, have revolutionized the financial landscape by enabling peer-to-peer (P2P) transactions without the need for intermediaries such as banks and payment processors. The first and most well-known cryptocurrency is Bitcoin, created in 2008 by an individual or group under the pseudonym Satoshi Nakamoto (Nakamoto (2008)).

Bitcoin achieves decentralization through blockchain technology, a type of distributed ledger maintained by a network of computers called nodes or miners. These nodes use their hardware resources to perform complex mathematical procedures to verify transactions. While blockchain transactions are public, the pseudonymous nature of addresses ensures that users are not easily identifiable. However, maintaining a secure and consistent network of nodes presents challenges, including data consistency and protection against malicious activities. To secure the blockchain, Bitcoin employs a consensus mechanism known as Proof of Work (PoW). When a user initiates a transaction, miners compete to solve a mathematical puzzle, with the first to solve it earning the right to add a new block to the blockchain. This new block contains recent transactions and a hash from the previous block. As a reward, miners receive transaction fees and a block subsidy, the latter being the only source of new Bitcoin supply. Notably, while creating a block is resource-intensive, verifying its validity is relatively inexpensive. Therefore, if a miner attempts to add an invalid block, the network quickly rejects it, preventing the malicious miner from recovering their mining costs. Figure 1 illustrates this blockchain structure and formation process, where H() represents the hash of the previous block.



(a) Structure of a single block in the blockchain.

(b) Overall process of the blockchain formation.

Figure 1: The formation process of the Bitcoin blockchain.[2]

Despite its benefits, PoW comes with the drawback of a 51% attack, where a single entity or group controls more than 50% of the mining power. This control allows them to manipulate the blockchain, such as preventing new transactions from being confirmed. Additionally, scalability is a significant issue. The blockchain's fixed block size and block time limit the number of transactions it can handle per second, causing network congestion. This congestion leads to increased transaction fees and longer transaction times.

Given the influence of blockchain mechanics on the supply and demand of Bitcoin, this paper examines various blockchain-related variables. By analyzing these features, we aim to assess their impact on Bitcoin's price and evaluate their predictive power. Section 3 provides a detailed explanation of these variables.

3 Data

Bitcoin's price is the output target of all the predictive models considered in this study. The change in the price can be attributed to three categories: fundamental indicators, technical indicators, and univariate lagged prices. Following the methodology proposed by Tripathi and Sharma (2023), these three types of features are examined separately. Additionally, we analyze the full set of these features combined, which has not yet been done in the literature.

Li and Wang (2017) claim that fundamental indicators can be categorized into economic factors and technological factors. Exchange rates, commodities, and global macroeconomic factors haves significant effects on emerging economies (Gospodinov and Jamali (2015); Kandil and Mirzaie (2002); Mensi et al. (2013)). The cryptocurrency market can be regarded as one of these emerging economies, and thus, as shown by Gajardo et al. (2018) and Laboissiere et al. (2015), the price of Bitcoin is influenced by macroeconomic indicators, both in the long term and short term. In this study, we consider various global macroeconomic development factors, such as the S&P 500, NASDAQ 100, Dow 30, FTSE 100, Eurostoxx 50, SSE, VIX, gold index, and crude oil index. Additionally, Bitcoin's price is related to traditional currency markets (Dyhrberg (2015); Jang and Lee (2017); Kristjanpoller and Minutolo (2018)). Therefore, the exchange rates between the US Dollar (USD) and other major fiat currencies (CHF, EUR, GBP, JPY, and CNY) are also considered in the analysis. Moreover, the US Dollar Index, which measures the value of the USD against a weighted basket of currencies from US trade partners, is also taken into account. All public market data are obtained using the Python-based

[2]Figure (a) and (b) obtained from Jang and Lee (2017) and Sazu and Jahan (2022), respectively.

Yahoo YQL Finance API.

Bitcoin, however, lacks traditional financial fundamental indicators. Instead, technological factors are gathered using blockchain information. In this paper, blockchain information is collected from https://blockchain.info/ using a custom-built web scraper. Table 1 lists the factors considered, along with a short description of each.

Table 1: Description of fundamental indicators used in the study

Category	**Description**
Panel A: Blockchain Activity	
Average Block Size	Average amount of data processed in a block (MB) each day.
Confirmation Time	Median time to validate a transaction (minutes) per day.
Confirmed Transactions	Total confirmed transactions per day.
Transactions Excluding Popular	Total daily transactions excluding the top 100 addresses.
Output Value	Total value of all transaction outputs per day in USD.
Unspent Transaction Outputs	Daily value of transaction outputs that can be used as inputs in a new transaction.
Estimated Transaction Value	Estimated daily transaction value (excluding coins returned as change) in USD.
Panel B: Mining Information	
Total Hash Rate	Daily estimated terahashes per second.
Difficulty	Daily measure of mining difficulty.
Miners Revenue	Total daily USD value of block rewards and transaction fees.
Total Transaction Profitability	USD value of all transaction fees paid to miners per day.
Transaction Fees	Average daily transaction handling fee.
Cost per Transaction	Miners revenue divided by number of transactions per day.
Cost Percentage	Daily miners revenue as a percentage of transaction volume.
Panel C: Market Signals	
Market Value to Realised Value (MVRV)	Market capitalization divided by the average capitalization based on the last transfer price (per day).
Network Value to Transactions	Market capitalization divided by the total transaction volume in USD over the past 24 hours.

In addition to these blockchain-based factors, public attention is considered a significant predictor of Bitcoin's price (Dastgir et al. (2018); Polasik et al. (2015); Zhang et al. (2018)).

Public attention metrics include Twitter tweets, Wikipedia views, and particularly Google searches. For this study, we use Google Trends as a proxy for public attention, as it reflects the public's interest in learning about Bitcoin. The daily Google Trends index for the keywords *Bitcoin* and *BTC* (case insensitive) is retrieved from Google Trends using a custom Python web scraper.

Secondly, we analyze the predictive power of technical indicators. These indicators are constructed using Bitcoin's closing price and trading volume. The considered indicators are derived from those used by Tripathi and Sharma (2023) and can be categorized into three sub-categories: moving averages, momentum indicators, and oscillators. Table 2 lists all the considered technical indicators, along with a short description of each.

Table 2: Description of technical indicators used in the study

Category	Description
Panel A: Moving Averages	
Simple Moving Average	5 and 10-day average prices.
Exponential Moving Average (EMA)	9-day EMA, a type of weighted moving average that gives more importance to recent price data.
Volume Adjusted Moving Average	9-day Volume Adjusted Moving Average, where more significance is assigned to days with higher trading volume.
Moving Average Convergence Divergence	26-period EMA minus 12-period EMA.
Panel B: Momentum Indicators	
Rate-of-Change	Percent change in price from one period to the next.
Percent B	Closing price as a percentage of the lower and upper Bollinger Bands.
Chaikin Oscillator	Accumulation-distribution line of MACD.
Simple Momentum Indicator	Market momentum measured by continually taking price differences for 10 periods.
Panel C: Oscillators	
Triple Exponential Average	Oscillator used to identify oversold and overbought markets.
Relative Strength Index	Identifies overvalued or undervalued conditions by measuring the speed and magnitude of recent price changes.

Finally, to assess the predictive power of univariate lagged prices, the close prices from the previous 60 days are computed. It is important to note that cryptocurrencies are traded 24 hours a day, seven days a week, and do not have a standard daily start and close time/price. To address this, we consider the trading day to start at 00:00:00 Coordinated Universal Time (UTC) and end at 23:59:59 UTC.

The considered time period spans from August 2011 to May 2024, divided into four significant data intervals to capture different phases in Bitcoin's market evolution. The first period covers data from August 1, 2011, to June 30, 2015, including the early adoption phase of Bitcoin and significant events such as the Mt. Gox exchange hack and collapse. The second interval spans from July 1, 2015, to July 31, 2018, reflecting a period of substantial growth in the number of Bitcoin exchanges operating worldwide and a surge in Bitcoin transactions across most countries and regions. Additionally, this interval saw Bitcoin's price surge to nearly $20,000 by the end of 2017, followed by a substantial market correction, dropping its price to $4,500. The third period includes data from August 1, 2018, to November 30, 2021, encompassing the 2021 bull run and significant geopolitical events such as the COVID-19 pandemic. Finally, the fourth interval covers the data from December 1, 2021, to May 1, 2024. This period captures the most recent developments, including regulatory advancements, institutional adoption, Bitcoin ETF approvals, and political acceptance of Bitcoin and cryptocurrencies.[3]

These intervals have been chosen to capture different phases in Bitcoin's market evolution and ensure a comprehensive analysis of the predictive performance across various market conditions. Figure 2 shows Bitcoin's price and trading volume over these periods. The left panel illustrates

[3]Mt. Gox exchange hack, 2021 Cryptocurrency Bull Run, BTC ETF approval, 2024 US elections.

all considered time periods, while the right panel magnifies the first interval, providing a detailed view of the initial growth phase.

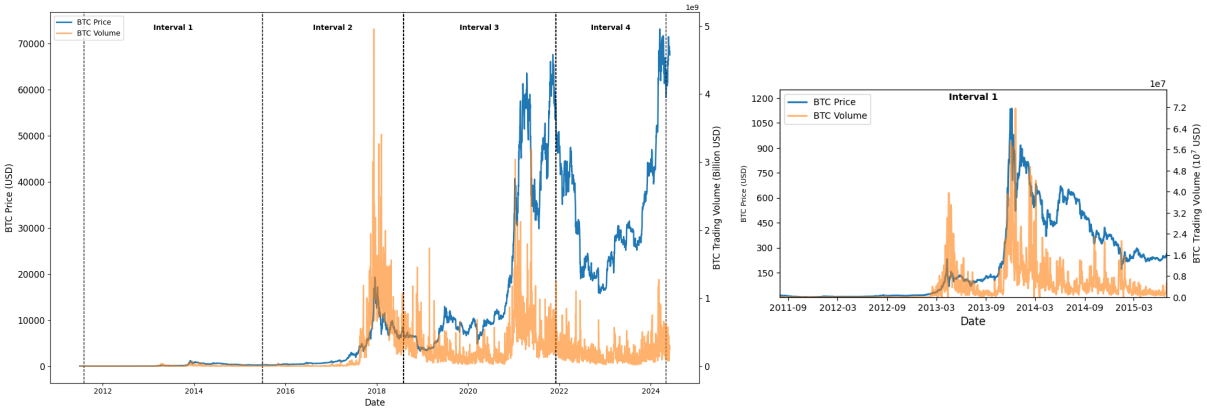


Figure 2: Bitcoin's price and trading volume. The left panel illustrates all intervals, while the right panel provides a detailed view of the first interval.

In this paper, missing cases are imputed using a linear interpolation method. Linear interpolation estimates missing values by connecting two known values with a straight line and assuming the missing value lies on this line. The data was standardized to have a mean of zero and a variance of one. Each interval was divided into three subsets for training, validation, and testing in the ratio of 70:15:15. The validation set was used to evaluate the model's performance on the training data and fine-tune the hyperparameters, thereby reducing potential overfitting and ensuring reliable test results on unseen data. Since we are working with time-series data, the split is not performed by randomly selecting observations but by maintaining the sequential nature of the data.

4 Methodology

To assess the predictive power of the various price determinant sets, we conduct a two-stage experiment on all predictor types. In the first stage, a feature selection technique is employed to identify the most relevant factor attributes and reduce noise. This process results in a subset of robust features. In the second stage, these robust features are fed into various regression models to predict Bitcoin's price. The following sections provide detailed guidance on the methodology for feature selection and price prediction, followed by an introduction to the metrics used to compare the performance of different models.

4.1 Feature Selection

To jointly reduce noise and select the most relevant, coherent, and non-redundant price determinants, we propose the use of a multi-layered Denoising Autoencoder (DAE). To highlight the DAE's capabilities in noise reduction and feature selection, we also compare its performance with models using no feature selection and those using a traditional Autoencoder. The DAE was introduced by Vincent et al. (2008) as an extension of the traditional Autoencoder model, such as the one proposed by Bengio et al. (2007). We replicate the main results from the former paper in Appendix A.

An Autoencoder is a specific architecture within artificial neural networks (ANNs) designed to represent input data in a lower-dimensional space, effectively extracting hidden features that aid in reconstructing the input and thus capturing only meaningful attributes. However, before delving into the specifics of an Autoencoder, it is crucial to discuss the foundational framework. ANNs are inspired by the way human brain neurons interact and perform computations. These networks consist of multiple layers, including an input layer, various hidden layers, and an output layer. Each neuron in one layer is connected to each neuron in the subsequent layer. A general neural network layer can be formulated as:

$$f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

where s is some activation function, $\mathbf{W}$ and $\mathbf{b}$ represent the connection-specific weight matrix and bias vector of the current layer, respectively, and $\mathbf{x}$ is the output vector from the previous layer. The processing of information in each layer and neuron is determined by the activation function. In this paper, the Rectified Linear Unit (ReLU) function is used. The ReLU function can be mathematically defined as:

$$ReLU(z) = \max(0, z).$$

This function introduces non-linearity into the model, enabling the network to learn complex patterns. Furthermore, ReLU is a popular choice for deep learning models since, for $x > 0$, it alleviates the vanishing gradient problem common with other activation functions, such as sigmoid and tanh. However, it is important to note that for $x < 0$, $\text{ReLU}(x) = 0$, which means no gradient signal is propagated and learning stops in those regions.

In these networks, the original data is fed into the first layer, and the processed information passes gradually through all the layers up to the last one. The output of the final layer is compared to the desired output using a loss function. Based on this difference, the weights and biases are iteratively updated through back-propagation. Back-propagation is an algorithm used to compute the gradient of the loss function with respect to each weight by applying the chain rule. This allows the network to adjust the weights and biases to minimize the loss, thereby reducing the error between the network's predictions and the actual values over time. The ultimate goal of the network is to learn the desired output for any possible input given to it. Figure 3 illustrates the general structure of an ANN.

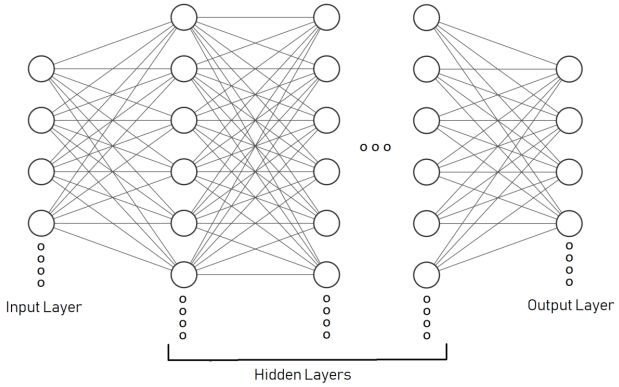


Figure 3: General ANN architecture.

However, relying solely on the reconstruction criterion is insufficient for ensuring the extraction of useful features, as it may result in the trivial solution of merely replicating the input. One strategy to avoid this is to use an Autoencoder (AE) and constrain the representation by a bottleneck. An AE is an unsupervised type of artificial neural network used mainly as a dimensionality reduction technique. Specifically, while the output and input have the same number of neurons, the hidden layer contains fewer neurons, creating a bottleneck architecture. The AE consists of an encoder, which compresses the input into a latent representation, and a decoder, which reconstructs the input from the latent space. This output is then compared to the original input using a loss function, such as the traditional Mean Squared Error. This forces the network to train the weight matrix and the bias vector in such a way that the hidden layer contains as much meaningful information as possible. Figure 4 visualizes a one-layered Autoencoder.

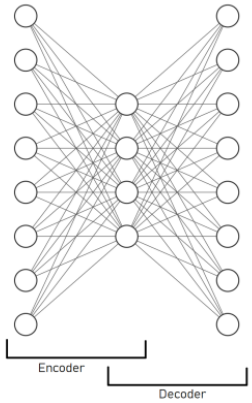


Figure 4: Basic structure of a one-layered Autoencoder.

Nonetheless, when the input is corrupted by noise, the basic Autoencoder falls short in extracting robust latent factors. Given that our model inputs comprise time series data of Bitcoin's price, blockchain fundamental factors, and price-related technical indicators, we suspect they might contain noise. Therefore, applying a Denoising Autoencoder is is more appropriate. The DAE maintains a structure very similar to that of a traditional Autoencoder, with the key difference lying in the input data. In contrast to the AE, where clean data is used, the DAE corrupts the input data by introducing noise.

There are multiple ways in which noise can be introduced, such as masking noise, where a fixed number of components are chosen at random and set to 0, or Gaussian noise, where values generated using a normal distribution with mean zero and a specified standard deviation are added to each element of the input. Given that Bitcoin price data and blockchain-related time series data often exhibit volatility and sudden changes, Gaussian noise is a suitable choice for our application. Gaussian noise effectively simulates these random fluctuations, ensuring that the extracted latent representations are robust against such variations, which are characteristic of financial time series data. Therefore, during training, noise is introduced in a Gaussian manner.

The model is trained to reconstruct a clean or repaired input from the corrupted one. This process can be formally formulated as follows. First, the DAE maps the corrupted input vector $\tilde{\mathbf{x}} \in \mathbb{R}^{d_0}$ to a latent representation $\mathbf{h} \in \mathbb{R}^{d_n}$, also known as the hidden representation. This is done by the encoder, f_E, which applies the following functions through multiple layers:

$$\begin{aligned}
\mathbf{h}^{(1)} &= r(\mathbf{W}_E^{(1)}\tilde{\mathbf{x}} + \mathbf{b}_E^{(1)}),\\
\mathbf{h}^{(2)} &= r(\mathbf{W}_E^{(2)}\mathbf{h}^{(1)} + \mathbf{b}_E^{(2)}),\\
&\vdots\\
\mathbf{h}^{(n)} = \mathbf{h} &= r(\mathbf{W}_E^{(n)}\mathbf{h}^{(n-1)} + \mathbf{b}_E^{(n)}),
\end{aligned}$$

where $r(\mathbf{a})$ represents the ReLU activation function. The terms $\mathbf{W}_E^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$, $\mathbf{b}_E^{(i)} \in \mathbb{R}^{d_i}$, and $\mathbf{h}^{(i)} \in \mathbb{R}^{d_i}$ represent the weight matrix, bias vector, and output vector of the i-th layer in the encoder, respectively. Note that $\mathbf{h}^{(0)} = \tilde{\mathbf{x}}$. After the encoding process, the decoder, f_D, reconstructs the latent representation $\mathbf{h}$ to an output vector $\mathbf{y} \in \mathbb{R}^{d_0}$. This can be formulated by the following equations through multiple layers:

$$\begin{aligned}
\mathbf{h}'^{(n-1)} &= r(\mathbf{W}_D^{(n-1)}\mathbf{h} + \mathbf{b}_D^{(n-1)}),\\
\mathbf{h}'^{(n-2)} &= r(\mathbf{W}_D^{(n-2)}\mathbf{h}'^{(n-1)} + \mathbf{b}_D^{(n-2)}),\\
&\vdots\\
\mathbf{h}'^{(1)} &= r(\mathbf{W}_D^{(1)}\mathbf{h}'^{(2)} + \mathbf{b}_D^{(1)}),\\
\mathbf{y} &= \mathbf{W}_D^{(0)}\mathbf{h}'^{(1)} + \mathbf{b}_D^{(0)},
\end{aligned}$$

where $r(\mathbf{a})$ represents the ReLU activation function. The terms $\mathbf{W}_D^{(i)} \in \mathbb{R}^{d_i \times d_{i+1}}$, $\mathbf{b}_D^{(i)} \in \mathbb{R}^{d_i}$, and $\mathbf{h}'^{(i)} \in \mathbb{R}^{d_i}$ represent the weight matrix, bias vector, and output vector of the i-th layer in the decoder, respectively. The choice of leaving out a non-linear activation function in the final layer is based on the need to compare the output directly with the clean, original input, which is generally unbounded even after standardization. If any of the aforementioned activation functions were used, the output would be restricted to a specific range: $[0, \infty]$ for ReLU, $[0, 1]$ for sigmoid, or $[-1, 1]$ for tanh. Figure 5 visualizes a general Denoising Autoencoder process.

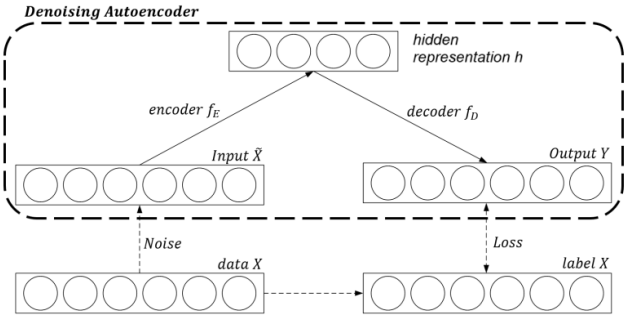


Figure 5: Visualization of the Denoising Autoencoder process, where $d_0 = 6$ and $d_n = 4$.

To extract robust features, the DAE and the Autoencoder are trained on the input data, with each observation corresponding to data for a single time step t. The encoder compresses each observation into a robust feature vector, effectively reducing noise and capturing the most relevant information. This results in one robust feature vector per observation, which is then used as input for price prediction models. This feature selection process is applied to each interval and for each of the four considered feature sets, giving a total of 32 selections. Additionally, early stopping is implemented to prevent overfitting during training. By monitoring the validation loss and halting the training if the loss does not improve for a specified number of epochs(known as the patience parameter), early stopping helps dynamically determine the optimal number of epochs, which is initially set to 1000. For this study, the patience parameter for early stopping is set to 15 epochs.

For both the DAE and AE, optimal model architecture and hyperparameter values must be determined. Given the computational expense of manual network tuning and conventional hyperparameter optimization methods, such as Grid Search and Randomized Search, we employ Bayesian optimization for each combination of interval and feature set. This technique leverages previously calculated hyperparameter values to guide the search for optimal parameters. While the standard deviation for the corruption process is fixed at 0.3, the learning rate, number of hidden layers, hidden dimensions, and batch size are tuned. Table 3 illustrates the considered bounds and values for each of these parameters.

Table 3: Autoencoder and Denoising Autoencoder hyperparameters search range used in Bayesian optimization

Hyperparameter	**Values**	**Description**
Learning Rate	[0.00001, 0.01]	Learning rate to adjust the weights of the network.
Number of Hidden Layers	[1, 4]	Number of hidden layers in the network, both in the encoder and decoder.
Hidden Dimensions	Proportions of input dimension: {3/4, 2/3, 1/2, 1/3, 1/4}	Number of units in each hidden layer.
Batch Size	{32, 64, 128}	Batch size for training.

4.2 Prediction Models

Now, the robust feature sets obtained from the DAE and the Autoencoder, as well as each set of features without feature selection, are used as inputs for various price prediction models. This study proposes the use of linear regression, a Long Short-Term Memory model, and Support Vector Regression to predict the one-day-ahead price of Bitcoin. The predictive performance of these models is compared to that of a simple moving average model, which serves as a benchmark. The following sections provide detailed explanations of the proposed models and their respective methodologies.

4.2.1 Linear Regression

Linear regression (LR) is a fundamental yet effective statistical method employed to model the relationship between a dependent variable and one or more regressors (also referred to as independent variables). The LR model assumes a linear relationship between the dependent variable and the independent variables, which, in this study, are Bitcoin's price and the set of input features, respectively. The model can be represented as:

$$y_t = \beta_0 + \beta_1 x_{1,t-1} + \beta_2 x_{2,t-1} + \cdots + \beta_p x_{p,t-1} + \epsilon_t,$$

where y_t denotes the price of Bitcoin at time t and $x_{i,t}$ represents the i^{th} independent variable at time t. The term β_0 is the intercept, which remains constant over time, while β_i represents the coefficient of the i^{th} selected feature. The error term at time t is denoted by ϵ_t. The coefficients are estimated using Ordinary Least Squares (OLS), which minimizes the sum of the squared differences between the observed and predicted values, aiming to find the best-fitting line that reduces prediction error.

To ensure the validity of the LR model, several assumptions must be satisfied: (1) Residuals must be independently and identically distributed, meaning no correlation between the residuals for any pair of observations must be present and the variance must be the same across time (homoskedasticity). (2) Residuals should be normally distributed. (3) Independent variables must not be highly correlated with each other (no multicollinearity). Violating these assumptions can lead to biased or inefficient predictions. Therefore, it is crucial to check these assumptions before relying on the model's predictions.

In this paper, the Durbin-Watson test is used to check for autocorrelation in residuals. The test statistic is given by:

$$D = \frac{\sum_{t=2}^{n}(e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2},$$

where e_t is the residual at time t and n denotes the number of observations. The test statistic D ranges between 0 and 4, where $D \approx 2$ indicates no autocorrelation, $D < 2$ indicates positive autocorrelation, and $D > 2$ indicates negative autocorrelation.

Furthermore, to detect heteroskedasticity, White's Test is used. The steps of the test can be summarized by the following pseudo-code:

Algorithm 1 White's Test for Heteroscedasticity

Step 1: Compute the squared residuals e_t^2.

Step 2: Construct the auxiliary regression model by regressing e_t^2 on the original independent variables, their squares, and their cross-products:

$$e_t^2 = \gamma_0 + \gamma_1 x_{1,t-1} + \gamma_2 x_{2,t-1} + \cdots + \gamma_p x_{p,t-1} + \gamma_{11} x_{1,t-1}^2 + \gamma_{12} x_{1,t-1} x_{2,t-1} + \cdots + \gamma_{pp} x_{p,t-1}^2 + u_t$$

Step 3: Obtain the R^2 from the auxiliary regression.

Step 4: Compute the test statistic:

$$W = nR^2 \sim \chi_k^2(\alpha)$$

where n is the number of observations, k denotes the degrees of freedom (number of regressors in the auxiliary regression), and α is the significance level.

Step 5: If W is greater than the critical value, reject the null hypothesis of homoskedasticity.

Finally, to assess the normality of residuals, we employ two methods: histograms and the Jarque-Bera test. The latter tests whether the skewness and kurtosis of the residuals match those of a normal distribution. Its test statistic is given by:

$$JB = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right)$$

where n is the number of observations, S is the skewness of the residuals, and K is the kurtosis of the residuals. This statistic is compared to the critical value from the chi-squared distribution with 2 degrees of freedom.

4.2.2 Support Vector Regression

Support Vector Regression (SVR) is a powerful machine learning model widely used for time series prediction in various fields, such as traffic flow prediction (Castro-Neto et al. (2009)), financial time series forecasting (Guo et al. (2018)), and Bitcoin price prediction (Chen et al. (2021)). SVR, introduced by Drucker et al. (1996) as an extension of the Support Vector Machine (SVM), differs from SVM by returning real-valued outputs rather than classifying inputs into discrete categories.

The goal of SVR is to minimize a normal vector $\mathbf{w}$ to the regression hyperplane while keeping the training errors within a certain threshold. Only errors that fall outside the margin from the fitted hyperplane are considered in the loss function. This is achieved by solving the following optimization problem:

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{t=1}^{n}(\xi_t + \xi_t^*), & \\
\text{s.t.} \quad & y_t - (\mathbf{w}\cdot\mathbf{x}_{t-1} + b) \leq \epsilon + \xi_t & t = 1, \ldots, n, \\
& (\mathbf{w}\cdot\mathbf{x}_{t-1} + b) - y_i \leq \epsilon + \xi_i^* & t = 1, \ldots, n, \\
& \xi_t, \xi_t^* \geq 0 & t = 1, \ldots, n,
\end{aligned}$$

where $\mathbf{w}$ represents the vector of all coefficients in the regression equation $y_i = \mathbf{w}\cdot\mathbf{x} + b$, b

is the bias term, ξ_t and ξ_t^* are slack variables representing the deviations from the ϵ-insensitive zone, C is the regularization constant, and ϵ is the width of the ϵ-insensitive zone (a tube within which errors are ignored). Figure 6 illustrates the main concept of SVR using the introduced notation.

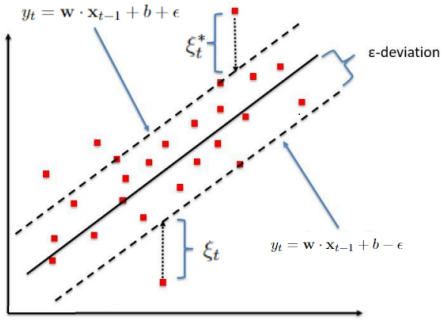


Figure 6: Main idea behind a Support Vector Regression.

If left as described, SVR performs linear regression. To model non-linearities, such as those present in the considered price determinants, SVR employs a kernel function, which replaces $\mathbf{x}_{t-1}$ inside the inner product $\mathbf{w} \cdot \mathbf{x}_{t-1}$ with a feature mapping function $\varphi(\cdot)$. Kernels implicitly map the input features into a higher-dimensional space where a linear separation is feasible. Following Chen et al. (2021), we utilize the Radial Basis Function (RBF) as a kernel. The RBF can be formally defined as:

$$K(\mathbf{x}_{t-1}, \mathbf{x}_{s-1}) = \exp\left(-\gamma \|\mathbf{x}_{t-1} - \mathbf{x}_{s-1}\|^2\right),$$

where γ determines the spread of the kernel.

To determine an optimal model specification, the hyperparameters are tuned using Bayesian optimization. The hyperparameters tuned for the SVR include C, ϵ, γ, and the sequence length. The sequence length refers to the number of previous time steps used as input features for predicting the next time step's Bitcoin price. Given the time-series nature of the data, selecting an appropriate sequence length is crucial for capturing temporal dependencies and improving predictive performance. Table 4 illustrates the considered bounds and values for each of these parameters.

Table 4: Support Vector Regression hyperparameter search space used in Bayesian optimization

Hyperparameter	**Values**	**Description**
C	[0.001, 1000]	Regularization parameter that controls the trade-off between achieving a low training error and a low testing error.
ϵ	[0.001, 1]	Width of the ϵ-insensitive zone, within which no penalty is associated in the training loss function.
γ	[0.001, 1]	Parameter for the Radial Basis Function kernel that determines its spread.
Sequence Length	[1, 60]	Number of previous time steps used as input features for predicting the next time step.

4.2.3 Long Short-Term Memory

Finally, we utilize a Long Short-Term Memory (LSTM) network. First introduced by Hochreiter and Schmidhuber (1996), LSTM is a type of recurrent neural network (RNN) designed to capture both long- and short-term dependencies, making it particularly effective for tasks involving time series or sequences.

Compared to traditional RNNs, LSTM networks incorporate special units called memory cells in their hidden layers. At each time step t, each memory cell contains three gates that regulate the flow of information and adjust the cell state: the input gate (i_t), the forget gate (f_t), and the output gate (o_t). This gating mechanism determines what information is retained and propagated to the next time step. The gates are functions of the input data, the previous hidden state, and their respective weights and biases. The gates and the cell states are computed using the following equations:

$$
\begin{aligned}
f_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
i_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
o_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

where $\mathbf{x}_t \in \mathbb{R}^{d_x}$ is the input vector at time t, $\mathbf{h}_{t-1} \in \mathbb{R}^{d_h}$ is the hidden state (output vector) from the previous cell, $c_t \in \mathbb{R}^{d_h}$ is the memory cell state vector at time t. The terms $\mathbf{W}_j \in \mathbb{R}^{d_h \times d_x}$, $\mathbf{U}_j \in \mathbb{R}^{d_h \times d_h}$, and $\mathbf{b}_j \in \mathbb{R}^{d_h}$ represent the weight matrices, recurrent weight matrices, and bias vectors for the respective gates, where $j \in \{f, i, o, c\}$. The functions σ and tanh are the sigmoid and hyperbolic tangent activation functions, respectively. The sigmoid function $\sigma(z)$ is defined as $\sigma(z) = \frac{1}{1+e^{-z}}$, mapping the input to a value between 0 and 1. The hyperbolic tangent function $\tanh(z)$ is defined as $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, mapping the input to a value between -1 and 1. The symbol $\odot$ denotes element-wise multiplication. Figure 7 illustrates the basic structure of such a cell using the introduced notation.

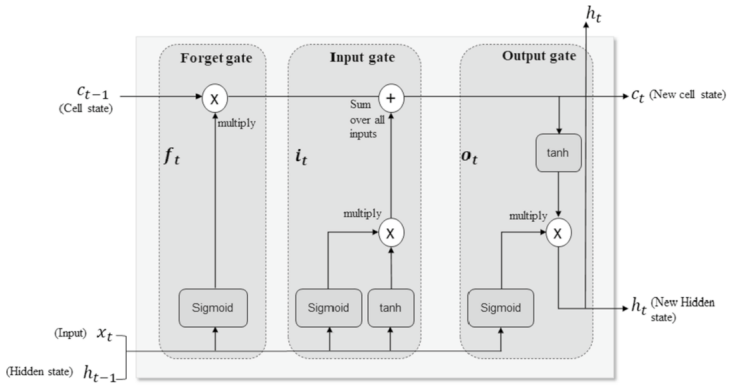


Figure 7: Memory cell used by the LSTM.[4]

Due to computational expense, we employ a simple LSTM layout with one hidden layer. In addition, we applied dropout regularization within the recurrent layer to reduce the risk of overfitting and used early stopping to dynamically determine the number of epochs for training during each study period. Dropout regularization involves randomly setting a fraction of the input units to zero during training, which is controlled by the dropout rate. This technique helps prevent the model from becoming too reliant on any particular set of neurons, thereby improving its generalization capability. Again, a patience parameter of 15 was used.

Finally, to determine the optimal model architecture and hyperparameters for the LSTM, Bayesian optimization is employed. The hyperparameters tuned for the LSTM include the learning rate, dropout rate, hidden dimensions, sequence length, and batch size. Table 5 illustrates the considered bounds and values for each of these parameters.

Table 5: Long Short-Term Memory model hyperparameter search space used in Bayesian optimization

Hyperparameter	**Values**	**Description**
Learning Rate	[1e-5, 1e-2]	Learning rate to adjust the weights of the network.
Dropout Rate	[0.001, 0.25]	Fraction of input units to drop for preventing overfitting.
Hidden Dimensions	{8, 16, 32, 64, 128, 256, 512}	Number of units in the hidden layer.
Sequence Length	[1, 60]	Number of previous time steps used as input features for predicting the next time step's Bitcoin price.
Batch Size	{32, 64, 128}	Batch size for training.

4.3 Performance Metrics

To evaluate the predictive performance of the hybrid frameworks, the following metrics are used: mean absolute error (MAE), root mean squared error (RMSE), and mean absolute percentage error (MAPE). These performance metrics are formally defined as follows:

$$\begin{aligned}
\text{MAE} &= \frac{1}{m}\sum_{t=1}^{m}|y_t - \hat{y}_t|, \\
\text{RMSE} &= \sqrt{\frac{1}{m}\sum_{t=1}^{m}(y_t - \hat{y}_t)^2}, \\
\text{MAPE} &= \frac{100}{m}\sum_{t=1}^{m}\left|\frac{y_t - \hat{y}_t}{y_t}\right|,
\end{aligned}$$

where m represents the number of samples in the test set, and y_t and $\hat{y}_t$ are the actual and predicted values at time t, respectively. Note that lower values of MAE, MAPE, and RMSE indicate more accurate predictions and are thus more desirable. A model that occasionally predicts erratic values will have a higher RMSE value, even if it has a lower MAE or MAPE. Therefore, models should be evaluated using all three metrics to ensure a comprehensive assessment of their performance.

[4]Figure 7 obtained from Tripathi and Sharma (2023).

5 Results

In this section, we present the results of our numerical experiments. We compare the performance of all models and evaluate them in comparison with the benchmark model. All experiments are conducted on a Windows 11 machines using Python 3.10 and a NVIDIA T4 GPU. To conduct the hyperparameter tuning using Bayesian optimization we used the Optuna package with 100 trials. Detailed results of the suggested hyperparameters are provided in Appendix C.

5.1 Benchmark

For the moving average model, we employed several rolling windows, specifically 2, 3, 4, 5, 6, 7, 14, 30, and 60 days. This approach resulted in multiple benchmark models. The detailed results for these models are presented in Table 16 in the Appendix. In the following sections, we will compare the performance of the proposed hybrid frameworks with that of the 2-day moving average model, as this is the best performing benchmark. Table 6 shows the performance measures for this model.

Table 6: Performance of the Best Performing Benchmark Model (2-day Moving Average)

Metric	**Interval 1**	**Interval 2**	**Interval 3**	**Interval 4**
MAE	8.96	396.11	2004.32	1751.36
RMSE	14.03	531.74	2461.02	2409.29
MAPE (%)	3.57	4.82	4.44	3.10

Note: This table contains the performance measures for the best performing benchmark model, the 2-day moving average. Intervals are defined as follows: Interval 1 - August 1, 2011 to June 30, 2015, Interval 2 - July 1, 2015 to July 31, 2018, Interval 3 - August 1, 2018 to November 30, 2021, Interval 4 - December 1, 2021 to May 1, 2024. Abbreviations: MAE - Mean Absolute Error, RMSE - Root Mean Squared Error, MAPE - Mean Absolute Percentage Error. MAE and RMSE are in US Dollars, while MAPE is in percentage.

5.2 Hybrid Frameworks

We compute the predictions for the LR, SVR, and LSTM models across different combinations of feature selection techniques and price determinant categories, and discuss their numerical results. The detailed results for these models across each interval are presented in the Appendix. However, given that the intervals all generally show a similar pattern, we focus on the results from interval 4, as this represents the most recent and significant data. In Table 7, the performance metrics of the most interesting hybrid frameworks are shown. Models that make use of univariate lagged prices are not reported as they do not provide any significant meaning.

Table 7: Performance Metrics for Linear Regression, SVR, and LSTM - Interval 4 (December 1, 2021 to May 1, 2024)

Panel A: Linear Regression									
	TI_{NO}	**FI_{NO}**	**FS_{NO}**	**TI_{AE}**	**FI_{AE}**	**FS_{AE}**	**TI_{DAE}**	**FS_{DAE}**	**2-MA**
MAE	1727.85	2728.08	2055.37	1689.24	2398.24	1402.68	1442.28	**1207.35**	1751.36
RMSE	2150.85	3393.29	2542.55	2190.54	2928.54	1768.62	1886.94	**1552.78**	2409.29
MAPE	3.10	4.98	3.76	2.99	4.38	2.58	2.61	**2.19**	3.10
Panel B: Support Vector Regression (SVR)									
	TI_{NO}	**FI_{NO}**	**FS_{NO}**	**TI_{AE}**	**FI_{AE}**	**FS_{AE}**	**TI_{DAE}**	**FS_{DAE}**	**2-MA**
MAE	1405.54	2218.93	1590.00	1320.37	1923.28	1102.81	1073.66	**952.87**	1751.36
RMSE	1790.06	3085.19	2080.56	1611.94	2655.85	1377.39	1347.26	**1226.43**	2409.29
MAPE	2.49	3.95	2.81	2.38	3.37	2.00	1.94	**1.71**	3.10
Panel C: Long Short-Term Memory (LSTM)									
	TI_{NO}	**FI_{NO}**	**FS_{NO}**	**TI_{AE}**	**FI_{AE}**	**FS_{AE}**	**TI_{DAE}**	**FS_{DAE}**	**2-MA**
MAE	1097.23	1759.28	1214.74	987.96	1475.57	871.36	821.02	**732.21**	1751.36
RMSE	1408.96	2184.13	1600.11	1307.40	1949.88	1170.29	1023.25	**945.55**	2409.29
MAPE	1.98	3.14	2.18	1.78	2.71	1.58	1.47	**1.31**	3.10

Note: This table presents the performance metrics for each hybrid framework for interval 4 (December 1, 2021 to May 1, 2024). The metrics reported are Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). MAE and RMSE are in US dollars, while MAPE is in percentage. Each index indicates one feature selection technique (NO for No Feature Selection, AE for Autoencoder, and DAE for Denoising Autoencoder) used for the specific feature set (TI - Technical Indicators, FI - Fundamental Indicators, FS - Full Set of Features). The last column includes the 2-day Moving Average (2-MA) for comparison. The bold values represent the best performing model for each metric within each panel.

The results indicate that, when comparing between price prediction models (panels), SVR generally performs better than the simple LR model across all feature selection techniques, showing average improvements of 22.30% in MAE, 20.78% in RMSE, and 23.17% in MAPE. When comparing LSTM to LR, the improvements are even more pronounced, with average improvements of 39.87% in MAE, 38.75% in RMSE, and 40.13% in MAPE. The underperformance of the LR can be attributed to the fact that both machine learning model are more sophisticated and have a better ability to capture complex patterns, but also to the fact that, as show in Appendix D, none of the LR models across all intervals (including interval 4) meet the three assumptions for model validity, affecting the reliability of these models. The underperformance of the LR can be attributed to the fact that both machine learning model are more sophisticated and have a better ability to capture complex patterns. Additionally, as shown in Appendix E, none of the LR models across all intervals (including interval 4) meet the three assumptions for model validity, affecting the reliability of these models.

Furthermore, in the LR case, only four models outperform the 2-day moving average benchmark in terms of MAPE: the frameworks that combine technical indicators or the full set of features with a DAE or AE feature selection technique. If we consider the MAE and RMSE, not only do the aforementioned models outperform the benchmark, but the model utilizing the

technical indicators feature set without any feature selection also beats the benchmark. On the other hand, the 2-day moving average model is surpassed by most frameworks that make use of SVR and LSTM. Only some price prediction models that solely use the fundamental indicators feature set, such as the SVR when no feature selection or the DAE is used, and the LSTM when no feature selection is used, do not beat the benchmark, indicating the underperformance of this feature set compared to the others..

When comparing between feature selection techniques, some interesting results can be seen. For all panels, the best performing model under no feature selection is consistently the technical indicators feature set. This is likely due to the presence of noise and unnecessary predictors in the full set of features, which can lead to overfitting and decreased model performance. This problem is largely mitigated by the use of an Autoencoder and a Denoising Autoencoder, where the frameworks using the full set of features consistently outperform the other combinations. The former model aims to reduce the dimensionality of the data, thus finding the best data attributes of each feature set. Hence, by combining the three feature sets, the autoencoder effectively captures the meaningful attributes of each single feature set and merges them together. The DAE further improves this by minimizing the present noise, thus providing even more accurate forecasts. To illustrate this noise reduction achieved by the DAE, Figure 8 compares the predictions achieved using the DAE and the AE.

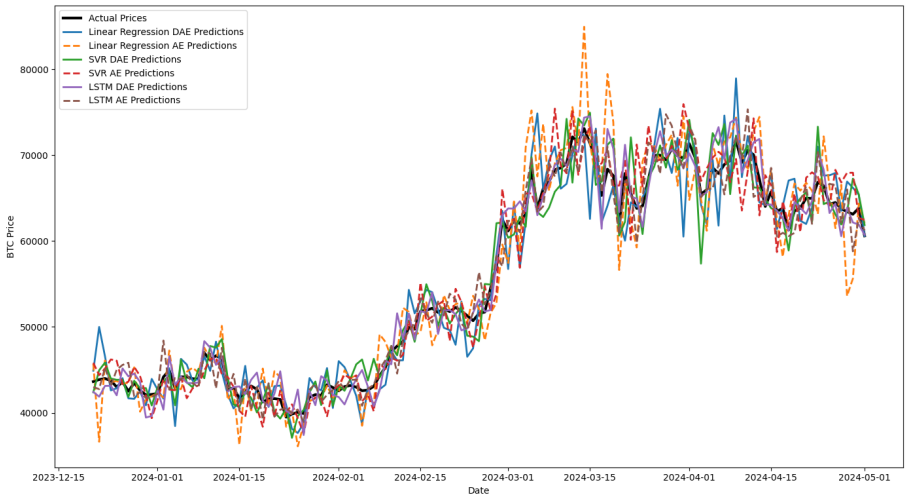


Figure 8: DAE (solid lines) and AE (dashed lines) predictions for interval 4 and fundamental indicators

Figure 8 shows that DAE-based models (solid lines) generally provide smoother predictions compared to their AE-based counterparts (dashed lines), highlighting the effectiveness of the noise minimization and capturing essential patterns in the data.

Overall, the Denoising Autoencoder combined with the full set of features consistently provided the best performance across all metrics. This finding is significant as it suggests that incorporating advanced feature selection techniques like Denoising Autoencoders can substantially improve the accuracy of predictive models. However, although using a DAE comes with the benefit that it captures meaningful data attributes while maximizing reconstruction quality and minimizing noise interference, the selected feature attributes are latent, thus not allowing us to know which exact features are important in each time interval or derive their feature importance.

6 Conclusion

This paper aims to determine the optimal input-model combination for one-day-ahead Bitcoin price forecasting and examines the potential benefits of employing a Denoising Autoencoder for enhanced predictive power. By utilizing a hybrid framework, we investigate the predictive performance of three distinct types of features: fundamental indicators, technical indicators, and lagged values of past Bitcoin prices, as well as the combined full set of these features. In the first stage, a Denoising Autoencoder is used as a noise reduction and feature selection method to obtain a subset of robust predictors. In the second stage, these predictors are fed into four predictive models: a simple moving average, a linear regression model, a Support Vector Regression, and a Long Short-Term Memory network. The main objective is to compare the predictive performance of models using Denoising Autoencoder-selected features against those using a traditional Autoencoder and those employing no feature selection technique.

Using data from August 2011 to May 2024, divided into four distinct intervals to capture the dynamic changes in Bitcoin, we evaluate the input-model combinations based on mean absolute error, root mean squared error, and mean absolute percentage error. Our findings reveal that Support Vector Regression and Long Short-Term Memory models significantly outperform linear regression across all feature selection techniques. The underperformance of linear regression can be attributed to its failure to meet the three assumptions for model validity and its limited ability to capture complex patterns in the data. Furthermore, when no feature selection is applied, the technical indicators feature set consistently outperforms the other feature sets, as well as the combined set.

Both the Autoencoder and Denoising Autoencoder feature selection techniques improve model performance by effectively capturing meaningful data attributes and minimizing noise. With the use of these two machine learning models, the best performing models switch from using the technical indicators feature set to the combined full feature set. When comparing the two models, the Denoising Autoencoder demonstrates better predictive accuracy, further validating its role in reducing noise in the inputs. Overall, the best performance across all metrics is consistently achieved by the Denoising Autoencoder combined with the full set of features. Conversely, the fundamental indicators feature set generally performs poorly, often failing to surpass the benchmark. However, our models do not manage to exhibit a better predictive performance than the state-of-the-art results, which use different feature selection and noise reduction techniques.

Future research could extend this study by exploring the applicability of these findings to other cryptocurrencies, such as Ethereum, or cryptocurrency indexes. Additionally, incorporating more advanced machine learning algorithms may provide further insights into improving Bitcoin price prediction models. Expanding the analysis to include non-linear cost models and other cost mitigation strategies could also enhance the practical applicability of these models in real-world trading scenarios. It is also important to note that this study employed a limited search range for the hyperparameters of the deep learning methods, particularly restricting the Long Short-Term Memory network to a single hidden layer. Hence, considering a broader hidden structure for this model to improve its performance could be of academic interest.

References

Abu Bakar, N. and Rosbi, S. (2017). Autoregressive integrated moving average (arima) model for forecasting cryptocurrency exchange rate in high volatility environment: A new insight of bitcoin transaction. *International Journal of Advanced Engineering Research and Science*, 4(11):130–137.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19.

Castro-Neto, M., Jeong, Y., Jeong, M. J., and Han, L. (2009). Aadt prediction using support vector regression with data-dependent parameters. *Expert Systems with Applications*, 36(2):2979–2986.

Chen, W., Xu, H., Jia, L., and Gao, Y. (2021). Machine learning model for bitcoin exchange rate prediction using economic and technology determinants. *International Journal of Forecasting*, 37(1):28–43.

Cheng, C.-H., Chen, T.-L., and Wei, L.-Y. (2010). A hybrid model based on rough sets theory and genetic algorithms for stock price forecasting. *Information Sciences*, 180(9):1610–1629.

Dastgir, S., Demir, E., Downing, G., Gozgor, G., and Chi, Q. (2018). The causal relationship between bitcoin attention and bitcoin returns: Evidence from the bitcoin attention index. *Finance Research Letters*, 26:41–49.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., and Vapnik, V. (1996). Support vector regression machines. *Advances in neural information processing systems*, 9.

Dyhrberg, A. H. (2015). Bitcoin, gold and the dollar–a garch volatility analysis. *Finance Research Letters*, 16:85–92.

Gajardo, G., Kristjanpoller, W., and Minutolo, M. C. (2018). Does bitcoin exhibit the same asymmetric multifractal cross-correlations with crude oil, gold and djia as the dollar-exchange rates? *Chaos, Solitons & Fractals*, 109:195–205.

Gospodinov, N. and Jamali, I. (2015). Commodity prices, commodity currencies, and global economic developments. *Review of International Economics*, 23(3):592–621.

Guo, Y. et al. (2018). Financial time series forecasting using support vector machine. *Expert Systems with Applications*, 39:9298–9304.

Hochreiter, S. and Schmidhuber, J. (1996). Lstm can solve hard long time lag problems. *Advances in neural information processing systems*, 9.

Jang, H. and Lee, J. (2017). An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information. *IEEE access*, 6:5427–5437.

Kandil, M. and Mirzaie, I. A. (2002). Exchange rate fluctuations and economic activity in developing countries: theory and evidence. *Journal of Economic Development*, 27(1):85–108.

Kristjanpoller, W. and Minutolo, M. C. (2018). Volatility of bitcoin and its role as a medium of exchange and a store of value. *Empirical Economics*, 54(2):543–577.

Laboissiere, A., Fernandes, R. A., and Lage, M. (2015). A learning approach to assess the impact of macroeconomic variables on bitcoin price. *Procedia Computer Science*, 55:125–132.

Lahmiri, S. and Bekiros, S. (2019). Cryptocurrency forecasting with deep learning chaotic neural networks. *Chaos, Solitons Fractals*.

Li, X. and Wang, C. A. (2017). The technology and economic determinants of cryptocurrency exchange rates: The case of bitcoin. *Decision support systems*, 95:49–60.

Mallqui, D. C. and Fernandes, R. A. (2019). Predicting the direction, maximum, minimum and closing prices of daily bitcoin exchange rate using machine learning techniques. *Applied Soft Computing*, 75:596–606.

McNally, S., Roche, J., and Caton, S. (2018). Predicting the price of bitcoin using machine learning. In *2018 26th euromicro international conference on parallel, distributed and network-based processing (PDP)*, pages 339–343. IEEE.

Mensi, W., Beljid, M., Boubaker, S., and Managi, S. (2013). Global financial crisis and stock market contagion: Evidence from brics markets. *International Review of Economics & Finance*, 25:202–220.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Panagiotidis, T., Stengos, T., and Vravosinos, O. (2018). On the determinants of bitcoin returns: A lasso approach. *Finance Research Letters*, 27:235–240.

Polasik, M., Piotrowska, A. I., Wisniewski, T. P., Kotkowski, R., and Lightfoot, G. (2015). Price fluctuations and the use of bitcoin: An empirical inquiry. *International Journal of Electronic Commerce*, 20(1):9–49.

Sazu, M. H. and Jahan, S. A. (2022). Impact of blockchain-enabled analytics as a tool to revolutionize the banking industry. *Data Science in Finance and Economics*, 2(3):275–293.

Tripathi, B. and Sharma, R. K. (2023). Modeling bitcoin prices using signal processing methods, bayesian optimization, and deep neural networks. *Computational Economics*, 62(4):1919–1945.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103.

Wang, J., Xue, Y., and Liu, M. (2016). An analysis of bitcoin price based on vec model. In *2016 international conference on economics and management innovations*, pages 180–186. Atlantis Press.

Wang, J.-J., Wang, J.-Z., Zhang, Z.-G., and Guo, S.-P. (2012). Stock index forecasting based on a hybrid model. *Omega*, 40(6):758–766.

Zhang, W., Wang, P., Li, X., and Shen, D. (2018). The dynamic correlation between bitcoin and traditional financial assets: Evidence from china. *Physica A: Statistical Mechanics and its Applications*, 510:685–692.

Appendix

A Replication of Vincent et al. (2008)

In this section we replicate one of the main results from the paper of Vincent et al. (2008) and show the correct performance of the used Denoising Autoencoder (DAE). We restrict ourselves to the same data set as used by the authors, namely the MNIST data set, and the same type of noise, namely masking-noise. The MNIST data set contains 70,000 grayscale images of handwritten digits from 0 to 9, with each image being a 28x28 pixel grid, totaling 784 pixels per image. In the application of this set used in the paper, the data is divided into a training, validation, and test set, which contain 10000, 2000, 50000 examples, respectively.

The used DAE contains 3 hidden layers. Here, the authors specify that the model selection procedure chose an overcomplete first hidden layer representation of size 2000. However, no number of neurons for the other two layers is given. In addition to this, the optimal hyperparameters are unspecified in the original paper. Thus, we make the following decisions:

1. The second and third layer have a representation of size 320 and 128, respectively.
2. The DAE is symmetric. In other words, the encoder and decoder have the same number of hidden layers, with the number of neurons in each layer of the decoder mirroring those of the encoder.
3. Given its common usage and vanishing gradient problem solving nature, a Rectified Linear Unit (ReLU) activation function is used in each intermediate hidden layer.
4. The last hidden layer of the decoder uses a sigmoid activation function. This is done in order to match the range of the output to the inputs, which are bounded between [0,1].
5. The learning rate, destruction rate used for the masking noise, batch size, and number of epochs equal 0.0001, 0.4, 64, and 200, respectively.

We show the clean images, inputs (noisy images), and the outputs (reconstructed images) of the model in Figure 9. The empirical results support the conclusion that the DAE effectively reconstructs the clean images from a corrupted input with a small test loss (loss = 0.0147).

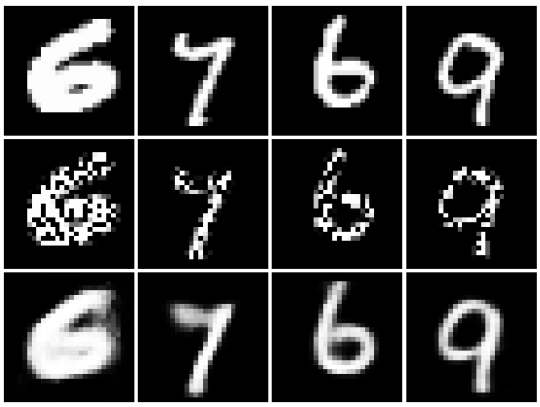


Figure 9: Example images showing the original (first row), noisy (second row), and reconstructed images (third row) from the model.

B Bayesian Optimization Results

Table 8: Optimal hyperparameters for the used Autoencoder models

Panel A: Interval 1 (August 1, 2011 to June 30, 2015)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.001	0.01	0.0001
Number of Hidden Layers	1	1	1	4
Number of Units	7	18	37	55, 54, 63, 49
Batch Size	128	128	128	128
Panel B: Interval 2 (July 1, 2015 to July 31, 2018)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.01	0.001	0.001	0.0001
Number of Hidden Layers	1	1	1	1
Number of Units	7	19	40	63
Batch Size	64	64	32	32
Panel C: Interval 3 (August 1, 2018 to November 30, 2021)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.001	0.0001	0.0001
Number of Hidden Layers	1	1	1	1
Number of Units	8	19	39	63
Batch Size	64	128	128	128
Panel D: Interval 4 (December 1, 2021 to May 1, 2024)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.001	0.001	0.0001
Number of Hidden Layers	1	1	1	1
Number of Units	7	19	34	49
Batch Size	32	64	32	32

Note: The table contains the optimal hyperparameters Autoencoder models cross different intervals and feature sets. Parameters optimized include learning rate (learning rate to adjust the weights of the network), number of hidden layers (number of hidden layers in the network, both in the encoder and decoder), number of units (number of units in each hidden layer), and batch size (batch size for training). Abbreviations: Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Combined - Full set of features.

Table 9: Optimal hyperparameters for the used Denoising Autoencoder models

Panel A: Interval 1 (August 1, 2011 to June 30, 2015)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.001	0.001	0.001
Number of Hidden Layers	1	1	1	2
Number of Units	6	16	30	60, 56
Batch Size	32	32	128	128
Panel B: Interval 2 (July 1, 2015 to July 31, 2018)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.001	0.001	0.0001
Number of Hidden Layers	1	1	1	1
Number of Units	7	18	40	55
Batch Size	32	128	32	32
Panel C: Interval 3 (August 1, 2018 to November 30, 2021)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.001	0.0001	0.0001
Number of Hidden Layers	1	1	1	1
Number of Units	8	19	37	63
Batch Size	64	128	128	128
Panel D: Interval 4 (December 1, 2021 to May 1, 2024)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.001	0.001	0.0001
Number of Hidden Layers	1	1	1	1
Number of Units	6	16	39	60
Batch Size	128	32	32	32

Note: The table contains the optimal hyperparameters Autoencoder models cross different intervals and feature sets. Parameters optimized include learning rate (learning rate to adjust the weights of the network), number of hidden layers (number of hidden layers in the network, both in the encoder and decoder), number of units (number of units in each hidden layer), and batch size (batch size for training). Abbreviations: Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Combined - Full set of features.

Table 10: Optimal hyperparameters for Support Vector Regression using Autoencoder features

Panel A: Interval 1 (August 1, 2011 to June 30, 2015)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	277.146	0.658	0.814	0.127
ϵ	0.056	0.181	0.409	0.189
γ	0.005198	0.013565	0.076918	0.003908
Sequence Length	2	2	36	5
Panel B: Interval 2 (July 1, 2015 to July 31, 2018)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	711.529	1.753	0.933	404.404
ϵ	0.223	0.019	0.565	0.788
γ	0.001014	0.001001	0.003236	0.001024
Sequence Length	22	1	1	3
Panel C: Interval 3 (August 1, 2018 to November 30, 2021)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	567.983	384.714	366.018	719.999
ϵ	0.109	0.119	0.004	0.543
γ	0.003455	0.001206	0.001012	0.001398
Sequence Length	1	1	24	4
Panel D: Interval 4 (December 1, 2021 to May 1, 2024)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	33.710	0.697	5.158	5.910
ϵ	0.029	0.747	0.160	0.232
γ	0.002924	0.027617	0.001009	0.001175
Sequence Length	2	1	1	1

Note: The table contains the optimal hyperparameters for Support Vector Regression models with Autoencoder features across different intervals and feature sets. Parameters optimized include C (regularization parameter), ϵ (width of the ϵ-insensitive zone), γ (spread parameter for the Radial Basis Function kernel), and Sequence Length (number of previous time steps used as input features for predicting the next time step). Abbreviations: Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Combined - Full set of features.

Table 11: Optimal hyperparameters for Long Short-Term Memory using Autoencoder features

Panel A: Interval 1 (August 1, 2011 to June 30, 2015)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.01	0.001	0.01	0.001
Dropout Rate	0.153	0.033	0.158	0.244
Hidden Dimensions	128	128	32	128
Sequence Length	46	6	57	7
Batch Size	128	128	64	128
Panel B: Interval 2 (July 1, 2015 to July 31, 2018)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.01	0.01	0.01
Dropout Rate	0.186	0.056	0.201	0.068
Hidden Dimensions	512	512	512	512
Sequence Length	58	51	21	7
Batch Size	32	32	32	32
Panel C: Interval 3 (August 1, 2018 to November 30, 2021)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.001	0.001	0.0001
Dropout Rate	0.099	0.154	0.175	0.213
Hidden Dimensions	256	512	512	512
Sequence Length	48	47	53	5
Batch Size	32	32	64	32
Panel D: Interval 4 (December 1, 2021 to May 1, 2024)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.01	0.01	0.01	0.001
Dropout Rate	0.160	0.147	0.093	0.054
Hidden Dimensions	128	256	128	256
Sequence Length	18	32	1	24
Batch Size	128	32	128	32

Note: The table contains the optimal hyperparameters for Long Short-Term Memory models with Autoencoder features across different intervals and feature sets. Parameters optimized include Learning Rate, Dropout Rate (fraction of input units to drop for preventing overfitting), Hidden Dimensions (number of units in the hidden layer), Sequence Length (number of previous time steps used as input features for predicting the next time step), and Batch Size (batch size for training). Abbreviations: Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Lagged Prices - Lagged prices, Combined - Full set of features.

Table 12: Optimal hyperparameters for Support Vector Regression using Denoising Autoencoder features

Panel A: Interval 1 (August 1, 2011 to June 30, 2015)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	21.506	11.491	0.197	25.047
ϵ	0.176	0.006	0.033	0.001
γ	0.001910	0.001467	0.003745	0.001037
Sequence Length	1	1	1	1
Panel B: Interval 2 (July 1, 2015 to July 31, 2018)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	463.406	12.077	301.601	2.127
ϵ	0.220	0.937	0.019	0.786
γ	0.001060	0.001744	0.001303	0.001010
Sequence Length	1	1	1	34
Panel C: Interval 3 (August 1, 2018 to November 30, 2021)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	19.941	43.116	5.957	746.917
ϵ	0.557	0.287	0.052	0.887
γ	0.001015	0.001015	0.001042	0.004653
Sequence Length	25	1	1	1
Panel D: Interval 4 (December 1, 2021 to May 1, 2024)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	56.640	6.396	75.596	940.550
ϵ	0.006	0.863	0.005	0.624
γ	0.001398	0.005865	0.001001	0.001298
Sequence Length	3	1	4	1

Note: The table contains the optimal hyperparameters for Support Vector Regression models with Denoising Autoencoder features across different intervals and feature sets. Parameters optimized include C (regularization parameter), ϵ (width of the ϵ-insensitive zone), γ (spread parameter for the Radial Basis Function kernel), and Sequence Length (number of previous time steps used as input features for predicting the next time step). Abbreviations: Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Lagged Prices - Lagged prices, Combined - Full set of features.

Table 13: Optimal hyperparameters for Long Short-Term Memory using Denoising Autoencoder features

Panel A: Interval 1 (August 1, 2011 to June 30, 2015)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.01	0.001	0.001	0.001
Dropout Rate	0.172	0.088	0.111	0.127
Hidden Dimensions	16	128	256	256
Sequence Length	56	57	40	60
Batch Size	32	64	128	64
Panel B: Interval 2 (July 1, 2015 to July 31, 2018)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.0001	0.01	0.01
Dropout Rate	0.087	0.107	0.068	0.011
Hidden Dimensions	512	512	512	256
Sequence Length	26	1	59	3
Batch Size	32	32	32	32
Panel C: Interval 3 (August 1, 2018 to November 30, 2021)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.01	0.0001	0.001	0.001
Dropout Rate	0.074	0.250	0.224	0.017
Hidden Dimensions	256	512	512	512
Sequence Length	15	2	54	9
Batch Size	32	32	64	32
Panel D: Interval 4 (December 1, 2021 to May 1, 2024)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.01	0.01	0.01	0.001
Dropout Rate	0.161	0.192	0.194	0.196
Hidden Dimensions	128	8	16	256
Sequence Length	42	19	53	43
Batch Size	128	128	64	32

Note: The table contains the optimal hyperparameters for Long Short-Term Memory models with Denoising Autoencoder features across different intervals and feature sets. Parameters optimized include Learning Rate, Dropout Rate (fraction of input units to drop for preventing overfitting), Hidden Dimensions (number of units in the hidden layer), Sequence Length (number of previous time steps used as input features for predicting the next time step), and Batch Size (batch size for training). Abbreviations: Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Lagged Prices - Lagged prices, Combined - Full set of features.

Table 14: Optimal hyperparameters for Support Vector Regression without feature selection

Panel A: Interval 1 (August 1, 2011 to June 30, 2015)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	7.493	1.113	41.982	0.245
ϵ	0.009	0.008	0.005	0.009
γ	0.001037	0.004214	0.001881	0.004289
Sequence Length	1	1	4	1
Panel B: Interval 2 (July 1, 2015 to July 31, 2018)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	758.355	13.758	110.158	505.466
ϵ	0.350	0.043	0.032	0.377
γ	0.001006	0.001003	0.505448	0.001001
Sequence Length	12	1	1	1
Panel C: Interval 3 (August 1, 2018 to November 30, 2021)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	40.329	6.058	1.725	2.296
ϵ	0.353	0.126	0.355	0.999
γ	0.001229	0.001459	0.001276	0.001750
Sequence Length	1	1	12	29
Panel D: Interval 4 (December 1, 2021 to May 1, 2024)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
C	6.054	78.078	651.819	0.517
ϵ	0.007	0.008	0.085	0.102
γ	0.001539	0.001174	0.001016	0.001021
Sequence Length	3	3	2	1

Note: The table contains the optimal hyperparameters for Support Vector Regression models without feature selection across different intervals and feature sets. Parameters optimized include C (regularization parameter), ϵ (width of the ϵ-insensitive zone), γ (spread parameter for the Radial Basis Function kernel), and Sequence Length (number of previous time steps used as input features for predicting the next time step). Abbreviations: Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Lagged Prices - Lagged prices, Combined - Full set of features.

Table 15: Optimal hyperparameters for Long Short-Term Memory without feature selection

Panel A: Interval 1 (August 1, 2011 to June 30, 2015)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.01	0.001	0.0001	0.0001
Dropout Rate	0.171	0.038	0.115	0.153
Hidden Dimensions	32	256	512	512
Sequence Length	7	40	1	7
Batch Size	128	32	64	32
Panel B: Interval 2 (July 1, 2015 to July 31, 2018)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.01	0.01	0.01	0.01
Dropout Rate	0.073	0.096	0.013	0.198
Hidden Dimensions	512	512	512	512
Sequence Length	46	23	9	2
Batch Size	32	32	32	32
Panel C: Interval 3 (August 1, 2018 to November 30, 2021)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.001	0.0001	0.01	0.001
Dropout Rate	0.036	0.086	0.173	0.081
Hidden Dimensions	512	512	512	512
Sequence Length	11	1	48	13
Batch Size	32	32	32	64
Panel D: Interval 4 (December 1, 2021 to May 1, 2024)				
Parameter	**Tech. Ind.**	**Fund. Ind.**	**Lagged Prices**	**Combined**
Learning Rate	0.01	0.01	0.001	0.001
Dropout Rate	0.133	0.231	0.085	0.072
Hidden Dimensions	128	512	512	512
Sequence Length	53	45	1	37
Batch Size	128	32	128	32

Note: The table contains the optimal hyperparameters for Long Short-Term Memory models without feature selection across different intervals and feature sets. Parameters optimized include Learning Rate, Dropout Rate (fraction of input units to drop for preventing overfitting), Hidden Dimensions (number of units in the hidden layer), Sequence Length (number of previous time steps used as input features for predicting the next time step), and Batch Size (batch size for training). Abbreviations: Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Lagged Prices - Lagged prices, Combined - Full set of features.

C Detailed Results Tables

Benchmark

Table 16: Benchmark results for different intervals

Panel A: Interval 1 (August 1, 2011 to June 30, 2015)									
	2-day	**3-day**	**4-day**	**5-day**	**6-day**	**7-day**	**14-day**	**30-day**	**60-day**
MAE	**8.96**	9.43	9.90	10.36	10.89	11.50	14.73	20.73	28.80
RMSE	**14.03**	14.76	15.50	16.10	16.75	17.42	21.26	28.98	41.04
MAPE	**3.57**	3.76	3.96	4.15	4.37	4.62	5.86	8.20	11.68
Panel B: Interval 2 (July 1, 2015 to July 31, 2018)									
	2-day	**3-day**	**4-day**	**5-day**	**6-day**	**7-day**	**14-day**	**30-day**	**60-day**
MAE	**396.11**	426.81	468.65	504.54	538.89	571.70	735.75	1014.62	1241.31
RMSE	**531.74**	578.65	620.64	658.61	694.43	727.58	899.95	1172.26	1497.51
MAPE	**4.82**	5.18	5.69	6.13	6.57	6.98	9.07	12.76	16.07
Panel C: Interval 3 (August 1, 2018 to November 30, 2021)									
	2-day	**3-day**	**4-day**	**5-day**	**6-day**	**7-day**	**14-day**	**30-day**	**60-day**
MAE	**2004.32**	2045.92	2106.60	2212.68	2322.22	2409.47	3017.42	4783.72	7479.74
RMSE	**2461.02**	2529.50	2640.27	2783.86	2927.19	3054.61	3843.74	5772.26	8486.99
MAPE	**4.44**	4.54	4.67	4.89	5.11	5.29	6.50	10.31	16.72
Panel D: Interval 4 (December 1, 2021 to May 1, 2024)									
	2-day	**3-day**	**4-day**	**5-day**	**6-day**	**7-day**	**14-day**	**30-day**	**60-day**
MAE	**1751.36**	1940.38	2120.89	2290.16	2447.69	2596.81	3108.91	4470.00	7078.40
RMSE	**2409.29**	2594.90	2803.85	2993.73	3155.76	3316.18	4167.15	6218.88	9414.61
MAPE	**3.10**	3.43	3.74	4.03	4.31	4.57	5.47	7.64	11.79

Note: The table contains benchmark results using a h-day moving average where h includes 2, 3, 4, 5, 6, 7, 14, 30, and 60 days. MAE and RMSE are in US Dollars, while MAPE is in percentage. The bold values represent the best performance for each metric across different window sizes. Abbreviations: MAE - Mean Absolute Error, RMSE - Root Mean Squared Error, MAPE - Mean Absolute Percentage Error.

Linear Regression

Table 17: Linear regression results for interval 1

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**8.41**	15.98	9.53	12.35
RMSE	**10.86**	19.98	12.06	15.52
MAPE (%)	**3.24**	6.12	3.65	4.78
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	7.06	13.35	8.61	**5.10**
RMSE	8.82	17.13	10.98	**6.31**
MAPE (%)	2.75	5.02	3.26	**1.98**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	5.47	11.71	7.44	**4.14**
RMSE	6.95	14.49	9.61	**5.28**
MAPE (%)	2.11	4.48	2.85	**1.59**

Table 18: Linear regression results for interval 2

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**385.64**	498.68	423.06	432.42
RMSE	**489.30**	641.54	540.59	535.31
MAPE (%)	**4.82**	6.19	5.17	5.39
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**306.30**	408.29	387.05	309.42
RMSE	**387.96**	505.45	481.16	387.85
MAPE (%)	**3.81**	5.09	4.84	3.81
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	286.68	386.02	368.80	**275.84**
RMSE	374.76	490.31	491.68	**348.71**
MAPE (%)	3.58	4.73	4.48	**3.43**

Note: These table present the performance metrics of linear regression models across different feature sets and feature selection techniques for interval 1 (August 1, 2011 to June 30, 2015) and interval 2 (July 1, 2015 to July 31, 2018), respectively. MAE and RMSE are given in US dollars, while MAPE is given in percentage. The bold values represent the best performance for each feature selection technique. Abbreviations: MAE - Mean Absolute Error, RMSE - Root Mean Squared Error, MAPE - Mean Absolute Percentage Error, Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Combined - Full set of features.

Table 19: Linear regression results for interval 3

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**2072.25**	2485.10	2286.23	2308.84
RMSE	**2723.51**	3234.50	3006.53	2949.23
MAPE (%)	**4.44**	5.49	4.89	4.85
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	1845.87	2263.47	2193.57	**1474.04**
RMSE	2382.48	2939.07	2806.13	**1995.82**
MAPE (%)	3.98	4.87	4.63	**3.10**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	1674.62	1946.79	1925.37	**1251.24**
RMSE	2181.44	2480.22	2464.43	**1630.78**
MAPE (%)	3.60	4.19	4.22	**2.65**

Table 20: Linear regression results for interval 4

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**1727.85**	2728.08	2589.19	2055.37
RMSE	**2150.85**	3393.29	3328.60	2542.55
MAPE (%)	**3.10**	4.98	4.68	3.76
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	1689.24	2398.24	2298.58	**1402.68**
RMSE	2190.54	2928.54	2962.40	**1768.62**
MAPE (%)	2.99	4.38	4.17	**2.58**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	1442.28	2201.88	2123.65	**1207.35**
RMSE	1886.94	2828.88	2704.55	**1552.78**
MAPE (%)	2.61	3.94	3.78	**2.19**

Note: These tables present the performance metrics of linear regression models across different feature sets and feature selection techniques for interval 3 (August 1, 2018 to November 30, 2021) and 4 (December 1, 2021 to May 1, 2024), respectively. MAE and RMSE are given in US dollars, while MAPE is given in percentage. The bold values represent the best performance for each feature selection technique. Abbreviations: MAE - Mean Absolute Error, RMSE - Root Mean Squared Error, MAPE - Mean Absolute Percentage Error, Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Combined - Full set of features.

Support Vector Regression

Table 21: Support Vector Regression results for interval 1

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**7.02**	14.40	7.79	10.61
RMSE	**9.24**	17.99	9.47	13.55
MAPE (%)	**2.67**	5.56	3.02	4.05
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	5.72	11.76	7.42	**4.27**
RMSE	7.20	14.80	9.62	**5.49**
MAPE (%)	2.22	4.48	2.85	**1.62**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	4.45	10.52	6.08	**3.34**
RMSE	5.62	13.23	7.74	**4.23**
MAPE (%)	1.71	4.01	2.33	**1.28**

Table 22: Support Vector Regression results for interval 2

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**308.83**	405.36	329.82	328.71
RMSE	**389.16**	500.92	415.06	429.79
MAPE (%)	**3.84**	5.02	4.12	4.12
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	235.88	329.65	321.35	**207.17**
RMSE	298.20	409.55	401.59	**256.65**
MAPE (%)	2.92	4.14	3.98	**2.54**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	222.62	297.32	288.99	**207.17**
RMSE	271.77	379.79	368.57	**256.65**
MAPE (%)	2.76	3.67	3.56	**2.54**

Note: These tables present the performance metrics of Support Vector Regression models across different feature sets and feature selection techniques for interval 1 (August 1, 2011 to June 30, 2015) and interval 2 (July 1, 2015 to July 31, 2018), respectively. MAE and RMSE are given in US dollars, while MAPE is given in percentage. The bold values represent the best performance for each feature selection technique. Abbreviations: MAE - Mean Absolute Error, RMSE - Root Mean Squared Error, MAPE - Mean Absolute Percentage Error, Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Combined - Full set of features.

Table 23: Support Vector Regression results for interval 3

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**1744.08**	2023.86	1745.42	1534.91
RMSE	**2295.14**	2546.54	2263.08	1969.87
MAPE (%)	**3.72**	4.41	3.77	3.34
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	1453.26	1697.35	1587.60	**1164.61**
RMSE	1780.55	2074.33	1973.30	**1484.90**
MAPE (%)	3.12	3.71	3.45	**2.52**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	1258.18	1547.98	1510.34	**921.26**
RMSE	1603.78	1905.71	1860.28	**1153.37**
MAPE (%)	2.75	3.33	3.28	**2.01**

Table 24: Support Vector Regression results for interval 4

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**1405.54**	2218.93	2074.04	1590.00
RMSE	**1790.06**	3085.19	2697.62	2080.56
MAPE (%)	**2.49**	3.95	3.68	2.81
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	1320.37	1923.28	1894.91	**1102.81**
RMSE	1611.94	2655.85	2391.42	**1377.39**
MAPE (%)	2.38	3.37	3.38	**2.00**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	1073.66	1728.98	1605.18	**952.87**
RMSE	1347.26	2255.35	2003.10	**1226.43**
MAPE (%)	1.94	3.07	2.94	**1.71**

Note: These tables present the performance metrics of Support Vector Regression models across different feature sets and feature selection techniques for interval 3 (August 1, 2018 to November 30, 2021) and 4 (December 1, 2021 to May 1, 2024), respectively. MAE and RMSE are given in US dollars, while MAPE is given in percentage. The bold values represent the best performance for each feature selection technique. Abbreviations: MAE - Mean Absolute Error, RMSE - Root Mean Squared Error, MAPE - Mean Absolute Percentage Error, Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Combined - Full set of features.

Long Short-Term Memory

Table 25: Long Short-Term Memory results for interval 1

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**5.84**	13.37	7.03	9.13
RMSE	**7.51**	16.90	8.77	11.37
MAPE (%)	**2.24**	5.02	2.75	3.50
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	4.79	10.52	6.45	**3.52**
RMSE	6.04	13.66	8.70	**4.44**
MAPE (%)	1.83	4.02	2.41	**1.34**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	3.37	9.25	4.93	**2.66**
RMSE	4.25	11.46	6.17	**3.48**
MAPE (%)	1.30	3.57	1.89	**1.01**

Table 26: Long Short-Term Memory results for interval 2

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**239.30**	332.23	260.04	246.57
RMSE	**304.66**	419.22	331.41	301.66
MAPE (%)	**2.97**	4.12	3.24	3.07
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	197.66	264.08	249.42	**179.59**
RMSE	250.37	337.16	309.73	**228.71**
MAPE (%)	2.45	3.27	3.09	**2.20**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	184.95	237.11	220.86	**155.45**
RMSE	227.18	302.32	280.41	**202.04**
MAPE (%)	2.31	2.91	2.71	**1.89**

Note: These table present the performance metrics of Long Short-Term Memory models across different feature sets and feature selection techniques for interval 1 (August 1, 2011 to June 30, 2015) and interval 2 (July 1, 2015 to July 31, 2018), respectively. MAE and RMSE are given in US dollars, while MAPE is given in percentage. The bold values represent the best performance for each feature selection technique. Abbreviations: MAE - Mean Absolute Error, RMSE - Root Mean Squared Error, MAPE - Mean Absolute Percentage Error, Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Combined - Full set of features.

Table 27: Long Short-Term Memory results for interval 3

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**1328.70**	1650.55	1427.64	1215.41
RMSE	**1744.57**	2153.51	1829.63	1566.21
MAPE (%)	**2.83**	3.48	3.05	2.61
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	1078.61	1400.24	1354.92	**889.60**
RMSE	1382.75	1750.22	1786.02	**1153.55**
MAPE (%)	2.37	3.01	2.89	**1.89**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	1013.45	1216.65	1225.85	**699.97**
RMSE	1283.36	1496.93	1592.25	**889.22**
MAPE (%)	2.16	2.66	2.62	**1.51**

Table 28: Long Short-Term Memory results for interval 4

Panel A: No Feature Selection				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	**1097.23**	1759.28	1640.07	1214.74
RMSE	**1408.96**	2184.13	2165.84	1600.11
MAPE (%)	**1.98**	3.14	2.97	2.18
Panel B: Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	987.96	1475.57	1503.73	**871.36**
RMSE	1307.40	1949.88	1873.64	**1170.29**
MAPE (%)	1.78	2.71	2.76	**1.58**
Panel C: Denoising Autoencoder				
	Tech. Ind.	**Fund. Ind.**	**Lagged Prices**	**Combined**
MAE	821.02	1350.70	1286.01	**732.21**
RMSE	1023.25	1707.42	1647.33	**945.55**
MAPE (%)	1.47	2.45	2.35	**1.31**

Note: These tables present the performance metrics of Long Short-Term Memory models across different feature sets and feature selection techniques for interval 3 (August 1, 2018 to November 30, 2021) and 4 (December 1, 2021 to May 1, 2024), respectively. MAE and RMSE are given in US dollars, while MAPE is given in percentage. The bold values represent the best performance for each feature selection technique. Abbreviations: MAE - Mean Absolute Error, RMSE - Root Mean Squared Error, MAPE - Mean Absolute Percentage Error, Tech. Ind. - Technical indicators, Fund. Ind. - Fundamental indicators (economic and blockchain data), Combined - Full set of features.

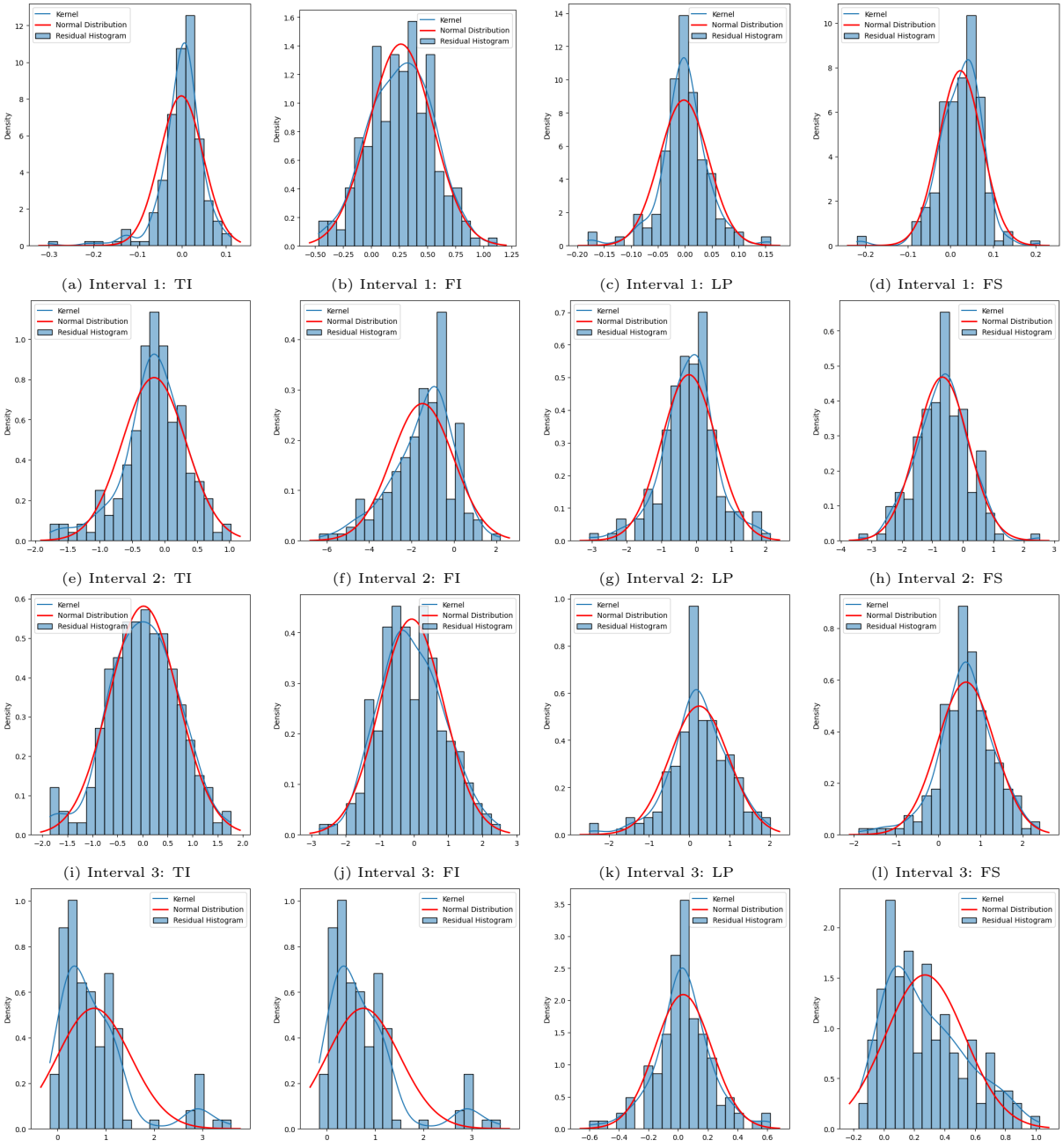
D Linear Regression Validity Checks

In this section, the validity of the linear regression model is analyzed. We choose to use the Durbin-Watson test to test for autocorrelation in residuals, White's test to detect heteroskedasticity, and histograms and the Jarque-Bera test to assess the normality of residuals. In Section 4.2.1 a detailed explanation of these tests is given. In the following table and figure the results of the tests and the histograms are presented.

Table 29: Linear regression validity checks test statistics

Panel A: Interval 1 (August 1, 2011 to June 30, 2015)				
Feature Set	**DW**	**WT**	**JB**	**Decision**
Technical Indicators	0.93	190.89	741.62	Reject
Fundamental Indicators	0.39	216.00	0.36	Reject
Lagged Prices	1.79	216.00	88.47	Mixed
Full Set Features	1.25	216.00	144.68	Reject
Panel B: Interval 2 (July 1, 2015 to July 31, 2018)				
Feature Set	**DW**	**WT**	**JB**	**Decision**
Technical Indicators	0.99	134.22	19.36	Reject
Fundamental Indicators	0.33	170.00	15.53	Mixed
Lagged Prices	1.81	170.00	15.92	Mixed
Full Set Features	1.02	170.00	3.81	Mixed
Panel C: Interval 3 (August 1, 2018 to November 30, 2021)				
Feature Set	**DW**	**WT**	**JB**	**Decision**
Technical Indicators	1.21	84.84	1.14	Accept
Fundamental Indicators	0.78	184.00	0.60	Reject
Lagged Prices	1.58	184.00	17.18	Mixed
Full Set Features	0.84	184.00	22.26	Reject
Panel D: Interval 4 (December 1, 2021 to May 1, 2024)				
Feature Set	**DW**	**WT**	**JB**	**Decision**
Technical Indicators	1.00	106.30	14.20	Mixed
Fundamental Indicators	0.06	133.00	140.21	Reject
Lagged Prices	2.27	133.00	12.21	Mixed
Full Set Features	0.45	133.00	10.87	Reject

Note: This table shows the test statistics for the three tests used to check the validity of the linear regression model across different intervals and feature sets. The decision column indicates whether the model passed the tests, with 'Mixed' indicating cases where some tests passed and others did not. The significance level for the tests was 5%. Abbreviations: DW - Durbin-Watson test, WT - White's test, JB - Jarque-Bera test. Here, accept is used for *fail to reject the null hypothesis*, which indicates that the null hypothesis cannot be proven.



(a) Interval 1: TI (b) Interval 1: FI (c) Interval 1: LP (d) Interval 1: FS

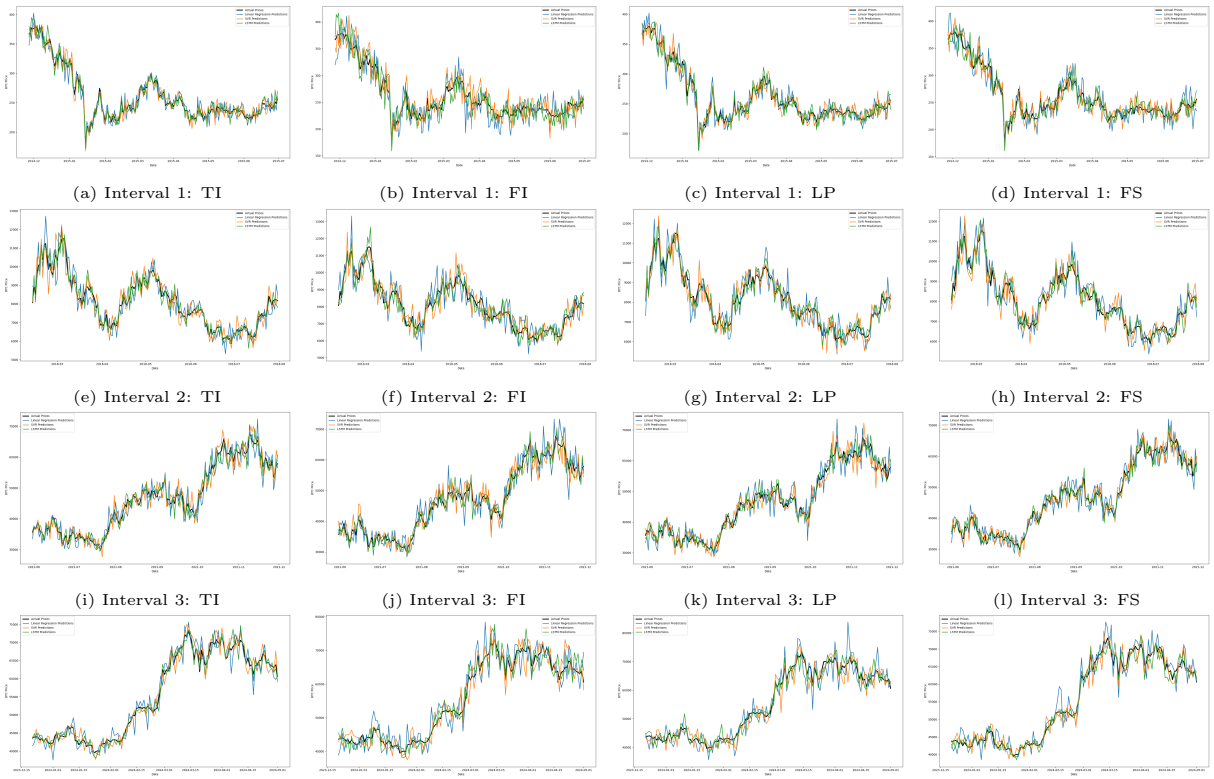
(e) Interval 2: TI (f) Interval 2: FI (g) Interval 2: LP (h) Interval 2: FS

(i) Interval 3: TI (j) Interval 3: FI (k) Interval 3: LP (l) Interval 3: FS

(m) Interval 4: TI (n) Interval 4: FI (o) Interval 4: LP (p) Interval 4: FS

Figure 10: Histograms comparing the density to the normal distribution for each combination of interval and feature set. The blue line represents the kernel density estimate, and the red line represents the normal distribution with the same mean and standard deviation as the residuals. (TI = Technical Indicators, FI = Fundamental Indicators, LP = Lagged Prices, FS = Full Set)

E Graphs and Figures



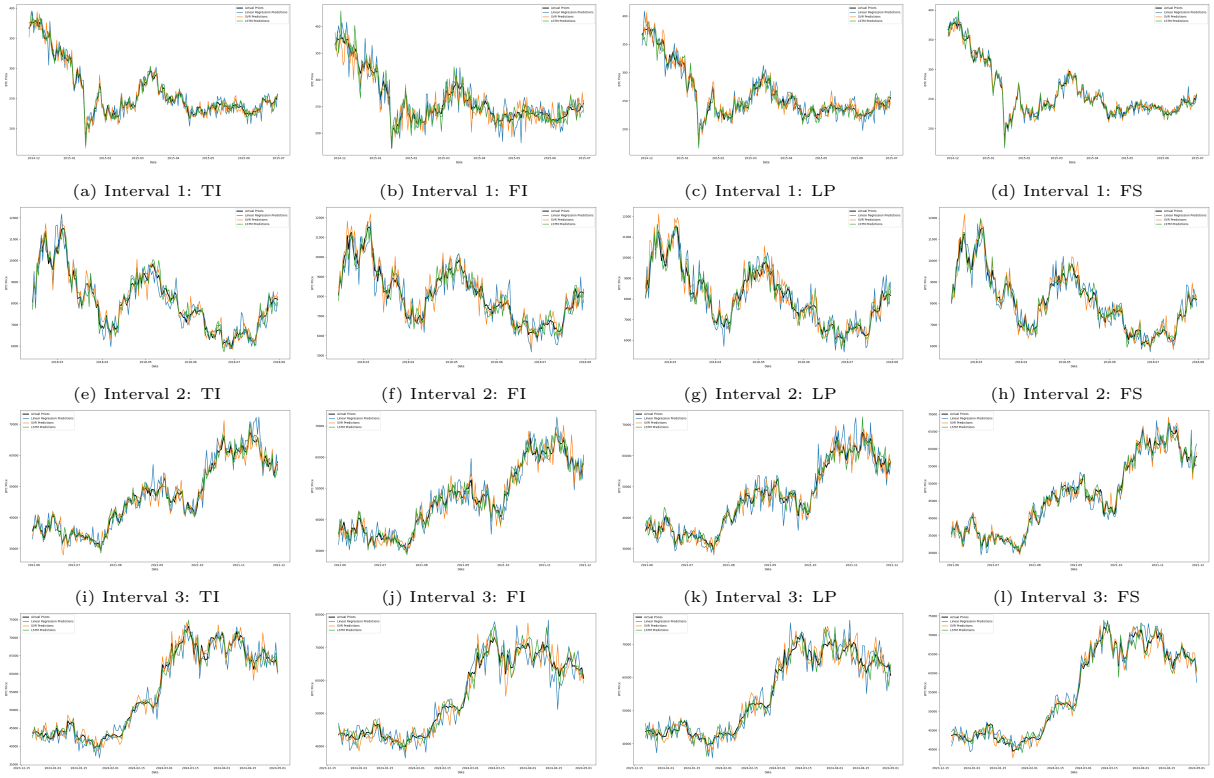
(a) Interval 1: TI (b) Interval 1: FI (c) Interval 1: LP (d) Interval 1: FS

(e) Interval 2: TI (f) Interval 2: FI (g) Interval 2: LP (h) Interval 2: FS

(i) Interval 3: TI (j) Interval 3: FI (k) Interval 3: LP (l) Interval 3: FS

(m) Interval 4: TI (n) Interval 4: FI (o) Interval 4: LP (p) Interval 4: FS

Figure 11: Predicted vs Real Price when no feature selection is applied (TI = Technical Indicators, FI = Fundamental Indicators, LP = Lagged Prices, FS = Full Set)



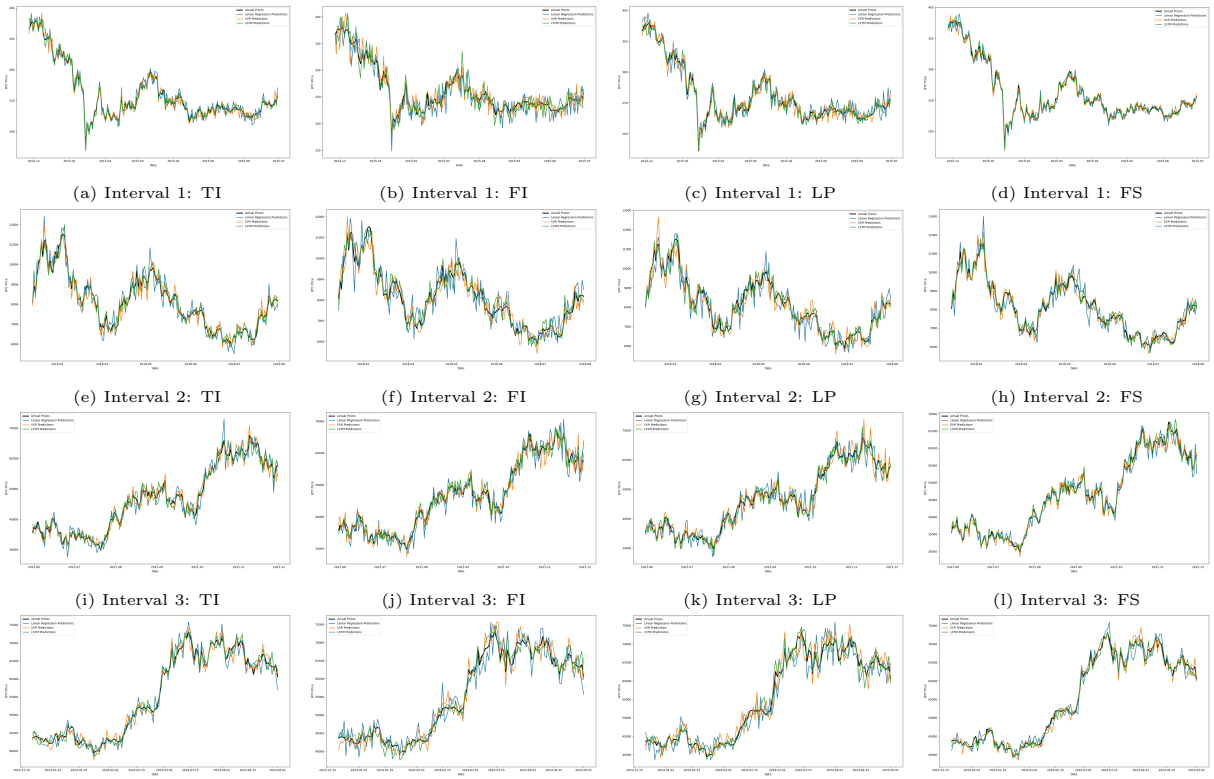
(a) Interval 1: TI (b) Interval 1: FI (c) Interval 1: LP (d) Interval 1: FS

(e) Interval 2: TI (f) Interval 2: FI (g) Interval 2: LP (h) Interval 2: FS

(i) Interval 3: TI (j) Interval 3: FI (k) Interval 3: LP (l) Interval 3: FS

(m) Interval 4: TI (n) Interval 4: FI (o) Interval 4: LP (p) Interval 4: FS

Figure 12: Predicted vs Real Price when AE models are applied (TI = Technical Indicators, FI = Fundamental Indicators, LP = Lagged Prices, FS = Full Set)



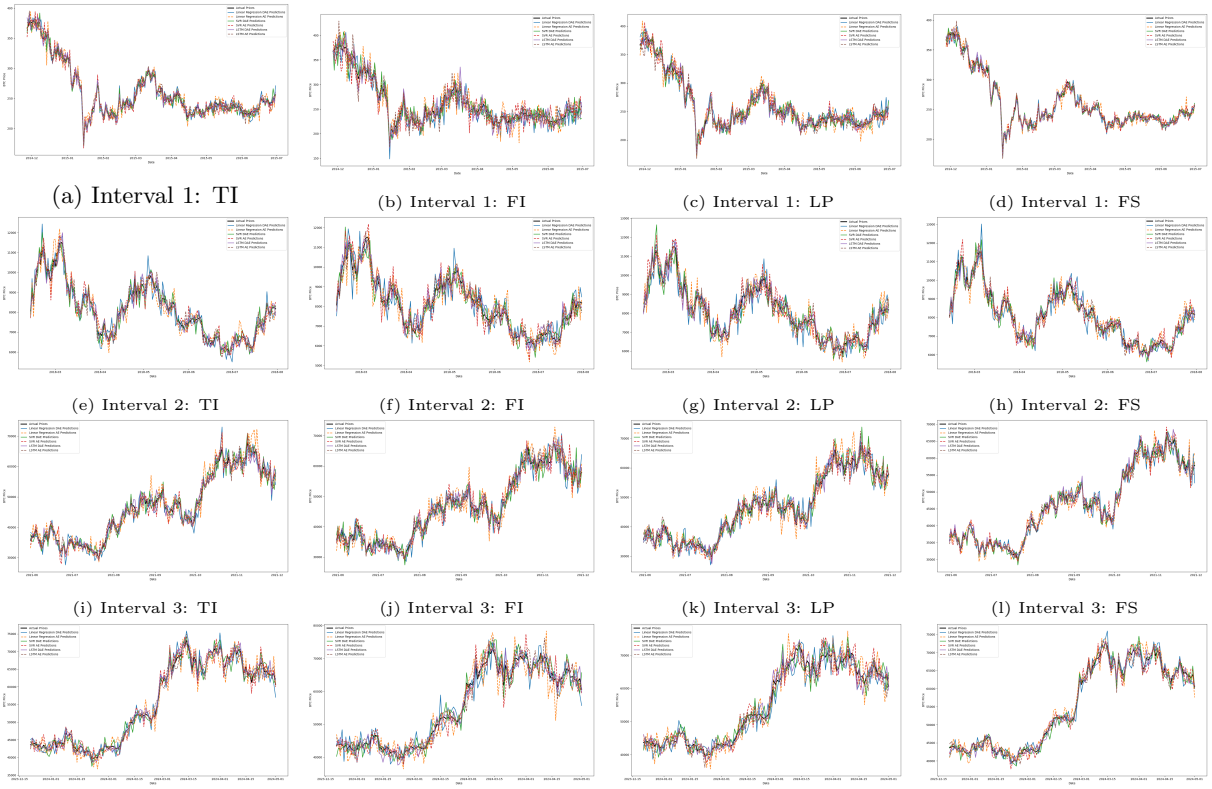
(a) Interval 1: TI (b) Interval 1: FI (c) Interval 1: LP (d) Interval 1: FS

(e) Interval 2: TI (f) Interval 2: FI (g) Interval 2: LP (h) Interval 2: FS

(i) Interval 3: TI (j) Interval 3: FI (k) Interval 3: LP (l) Interval 3: FS

(m) Interval 4: TI (n) Interval 4: FI (o) Interval 4: LP (p) Interval 4: FS

Figure 13: Predicted vs Real Price when DAE models are applied (TI = Technical Indicators, FI = Fundamental Indicators, LP = Lagged Prices, FS = Full Set)



(a) Interval 1: TI (b) Interval 1: FI (c) Interval 1: LP (d) Interval 1: FS

(e) Interval 2: TI (f) Interval 2: FI (g) Interval 2: LP (h) Interval 2: FS

(i) Interval 3: TI (j) Interval 3: FI (k) Interval 3: LP (l) Interval 3: FS

(m) Interval 4: TI (n) Interval 4: FI (o) Interval 4: LP (p) Interval 4: FS

Figure 14: DAE vs AE across all models and intervals (TI = Technical Indicators, FI = Fundamental Indicators, LP = Lagged Prices, FS = Full Set)

F Code Documentation

The code consists of one Python Notebook. Initially, we replicate the findings of Vincent et al. (2008). After that, the extension part starts. First, we fetch, preprocess, and split the price determinants using custom web scrapers, as well as the TA library and the Yahoo YQL Finance API. This part of the code can be found under the Data section. Next, under the Methodology section the used the definitions for the Denoising Autoencoder, Autoencoder, linear regression model, Support Vector Regression model, and Long Short-Term Memory network are given. Hyperparameter tuning is performed using Optuna. We follow up by selecting the most important (and robust) features. Finally, the extracted features, as well as the non-feature selected price determinants (full and noisy set), are used as input to construct the price predictions. The models' predictions are compared against a simple moving average benchmark.