STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **13**, May 2025, pp 1947–1971.
Published online in International Academic Press (www.IAPress.org)

Cryptocurrency Price Prediction with Genetic Algorithm-based Hyperparameter Optimization

N. Hafidi [1,*], Z. Khoudi [1], M. Nachaoui [1], S. Lyaqini [2]

[1] *EMI, Sultan Moulay Slimane University, Beni mellal, Morocco*
[2] *LIPIM Laboratory, Ecole Nationale des Sciences Appliquées, Sultan Moulay Slimane University, Beni Mellal, Morocco*

Abstract Accurate cryptocurrency price forecasting is crucial for investors and researchers in the dynamic and unpredictable cryptocurrency market. Existing models face challenges in incorporating various cryptocurrencies and determining the most effective hyperparameters, leading to reduced forecast accuracy. This study introduces an innovative approach that automates hyperparameter selection, improving accuracy by uncovering complex interconnections among cryptocurrencies. Our methodology leverages deep learning techniques, particularly Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, in conjunction with the Genetic Algorithm (GA) to optimize hyperparameters. We propose and compare two architectures, LAO and LOEE, utilizing these methods to enhance forecast accuracy and address the challenges of the cryptocurrency market. This cutting-edge approach not only improves forecasting capabilities but also provides valuable insights for managing cryptocurrency investments and conducting research. By automating hyperparameter selection and considering interconnections between cryptocurrencies, our approach offers a practical solution for accurate cryptocurrency price prediction in a dynamic market environment, benefiting both investors and academics.

Keywords cryptocurrency price predictions, time series, deep learning, hyperparameter optimization, genetic algorithm

DOI: 10.19139/soic-2310-5070-2035

1. Introduction

Traditional economic systems relied on institutions, such as banks, to facilitate payments, whether in cash or electronically. These intermediaries were central in overseeing transactions and completely controlled the financial exchange process. While effective for monetary transactions, this system posed limitations regarding transaction amounts and needed more crucial elements such as security, flexibility, transparency, and trust. The challenges posed by these limitations have sparked a demand for a system that eliminates intermediaries in financial transactions, enabling direct fund transfers between parties. Since the announcement of Blockchain technology, Cryptocurrencies have emerged as a worldwide sensation, captivating a large user base. This is primarily due to their reliance on a trust-based technological infrastructure, enabling the swift transfer of financial assets from any location with minimal delay. This is facilitated by the network's users, who provide essential authentication mechanisms. The evolution and adoption of blockchain technology have facilitated the rise of decentralized digital currencies, which operate independently of traditional financial institutions. This shift has fundamentally altered the financial landscape, with Bitcoin leading the way as the pioneering cryptocurrency. Its success has paved the way for developing numerous other cryptocurrencies, including Ethereum, Litecoin, Ripple, Dash, and many alternative coins [31, 40]. These cryptocurrencies, or virtual currencies, are digital assets that can be exchanged

*Correspondence to: N. HAFIDI (Email: nasreddine.hafidi@usms.ma), Sultan Moulay Slimane University, Beni mellal, Morocco

ISSN 2310-5070 (online) ISSN 2311-004X (print)

between individuals or groups. They function as a medium of exchange on a network secured by cryptographic algorithms [25, 44].

As cryptocurrencies are built on blockchain technology, they inherit its key properties, including decentralization, transparency, and immutability. This starkly contrasts traditional systems, where central authorities govern transactions. In cryptocurrency systems, the validation or confirmation of transactions relies on consensus algorithms. These algorithms play a crucial role in establishing trust among the involved parties, ensuring the integrity and security of the system. This means that the procedure for validating or confirming a transaction is based on consensus algorithms, which resolve trust problems between the parties involved in the system. The bitcoin was the first cryptocurrency created based on the concepts in Satoshi Nakamoto's paper [6]. Since then, numerous crypto-currencies have been introduced for various applications [17, 43]. These crypto-currencies are classified into three main areas: currency, platform, and application. Platform and application. Cryptocurrency prices depend on various factors such as the cost difficulty of mining, market trends, popularity, the price of other coins, stock markets, sentiment, and certain legal factors. Such large volatility predicts cryptocurrency prices, a more challenging problem than stock forecasting.

Previous attempts have been made to predict the price of cryptocurrencies based on historical data, while some research has analyzed the influence of other factors on price evolution. For instance, [26] proposes a time series analysis of those Bitcoin volatilizes and Ripple, supposing that the global stock indices, gold prices, and fear gauges partly determine them. Using Deep Learning methods, authors of [24] propose a deep-learning-based hybrid model to predict Litecoin and Zcash's price with the parent coin's inter-dependency. Conversely, The authors of [27] compare ARIMA, deep learning (LSTM), and Hybrid ARIMA-SVM in different scenarios for specific cryptocurrency prediction tasks.

Several attempts have been made to forecast cryptocurrency prices, with some research exploring the impact of various factors on price evolution. Authors of [22] proposed an LSTM and GRU-based hybrid cryptocurrency prediction scheme, focusing on Litecoin and Monero. On the other hand, a prediction is made on [20], using the (ARIMA) method, which could generate a high accuracy in short-term predictions. A financial time series forecasting model using a deep learning ensemble model was also introduced in [33]. Notably, these approaches have primarily concentrated on specific cryptocurrencies in their predictions, often overlooking the simultaneous prediction of multiple coins. Moreover, the manual selection of hyperparameters (of the used model) has been limited in these studies. The influence of the hyperparameters (as we will show in the section 3.2. Therefore, forecasting cryptocurrency prices has proven to be a challenging and essential task for researchers.

However, despite the advances in employing deep learning and other techniques for cryptocurrency price prediction, these approaches exhibit several notable limitations. One of the primary challenges is the inherent volatility of cryptocurrency markets, which often leads to models that struggle to generalize well across different periods or market conditions. Traditional prediction methods frequently rely on manual hyperparameter selection, which introduces subjectivity and often results in suboptimal model performance, failing to capture the complex and non-linear relationships in the data fully. Additionally, many studies have focused narrowly on specific cryptocurrencies or have employed static models that do not adapt well to the rapidly evolving market dynamics. This limitation underscores the need for more flexible and adaptive models to better account for the factors influencing cryptocurrency prices, such as market sentiment, regulatory news, technological developments, and macroeconomic indicators.

Our approach distinguishes itself by addressing these gaps, providing a comprehensive prediction model encompassing a broader spectrum of cryptocurrencies and factors in a more extensive range of volatilities, and offering a holistic perspective on the cryptocurrency market. This paper is arranged as follows: section 2 gives an in-depth analysis of the issue setting, delivering useful insights into the underlying basis of time series. Moving on, section 3 gives a detailed analysis of prior research, laying the way for our suggested approach in section 4,

where we introduce a unique technique for cryptocurrency prediction. This strategy combines a synergistic mix of Genetic Algorithms and deep learning approaches. Subsequently, section 5 shows the findings of our studies, supplemented by comparative analysis. To end, section 6 summarizes the results and provides prospective routes for further study.

2. Setting of the problem

We will start by exploring the importance of time series modeling in economics, industry, and finance and the transition to machine learning techniques. Then, we will introduce time series prediction, briefly discussing ARIMA models and RNNs as potential solutions.

2.1. Background

Time-series modeling has long been a central focus in academic research due to its success in machine learning [9], playing a crucial role in diverse fields such as economics [30], industry [21], and finance [28]. Traditionally, this field relied on parametric models like autoregressive (AR) [13], exponential smoothing [5], and structural time-series models [4], which were favored for their reliance on domain expertise. However, contemporary machine learning techniques offer an alternative, allowing the learning of temporal dynamics purely from data. With the surge in data availability and computational capabilities in recent years, machine learning has emerged as an indispensable component in the evolution of the next generation of time-series forecasting models, as evidenced in applications such as cryptocurrency prediction [30].

The Autoregressive Integrated Moving Average Model, often referred to as ARIMA [2], has been a standard method for time series forecasting for a long time, representing a comprehensive extension of the Autoregressive Moving Average (ARMA) model. ARIMA seamlessly integrates both Moving Average (MA) [1] and Autoregressive (AR) processes. The AR component of the model involves modeling dependencies between a current observation and a series of lagged observations. On the other hand, the MA component incorporates the relationship between observations and the residual error terms that result when applying a moving average model to these lagged observations. This integration allows ARIMA to capture a wide range of temporal behaviors in time series data, making it a versatile tool in forecasting. Some recent studies have shown the performance of ARIMA time series forecasting models in analyzing economic and financial time series [10, 18]. A hybridization of artificial neural networks and the ARIMA model was also proposed [7, 8]. On the other hand, a comparison [19] between LSTM as a deep learning-based algorithm and ARIMA was conducted.

2.2. Problem Formulation

Time-series forecasting models aim to predict future values of a target variable y_t at time t. Each observation represents a unique entity, such as sales of different products in retail or performance of various financial assets in the stock market, observed simultaneously. One-step forecasting approaches, in their simplest form, can be expressed as:

$$\widetilde{y}_{t+1} = f(y_{t-k:t}, x_{t-k:t}, s) \tag{1}$$

Here, $\widetilde{y}_{t+1}$ is the model's forecast, $y_{t-k:t}$ and $x_{t-k:t}$ denote the past k observations of the target variable and exogenous inputs, respectively, within a look-back window of k time steps. The static metadata associated with the entity is denoted by s, and $f(\cdot)$ denotes the prediction function learned by the model. While this survey focuses on univariate forecasting (i.e., 1-D targets), it is worth mentioning that the principles discussed can be extended to multivariate models without loss of generality. For simplicity, we omit the entity index ii in subsequent sections unless explicitly required.

2.3. AR and MA

Let $AR(n)$ represent a basic Autoregressive (AR) model of order n. This model can be expressed as a linear process given by:

$$x_t = c + \sum_{i=1}^{n} \mathcal{W}_i x_{t-i} + \epsilon_t \tag{2}$$

Here, x_t denotes the stationary variable, c is a constant, the terms in $\mathcal{W}_i$ are autocorrelation coefficients within the range $[1, 2, ..., n]$, and ϵ_t represents the residuals?a Gaussian white noise series with a mean of zero and a variance of $\sigma\epsilon^2$. Similarly, $MA(p)$ denotes the Moving Average (MA) model of order p, which can be expressed as follows:

$$x_t = \mu + \sum_{i=0}^{p} \theta_i \epsilon_{t-i} \tag{3}$$

with θ_i being the weights applied to the current and prior values of a stochastic term in the time series, where $\theta_0 = 1$. We make the assumption that ϵ_t follows a Gaussian white noise series with a mean of zero and a variance of σ_ϵ^2. By combining the Autoregressive (AR) and Moving Average (MA) models, we create an Autoregressive Integrated Moving Average (ARIMA) model of order (n, p):

$$x_t = c + \sum_{i=0}^{n} \mathcal{W}_i x_{t-i} + \epsilon_t - \sum_{i=0}^{p} \theta_i \epsilon_{t-i} \tag{4}$$

Here, $\mathcal{W}_i \neq 0$, $\theta i \neq 0$, and $\sigma_\epsilon^2 > 0$. The parameters n and p are called the Autoregressive (AR) and Moving Average (MA) orders, respectively. The foundation of ARIMA forecasting lies in its capacity to manage non-stationary time series data, a feat facilitated by its 'integrate' step. This step involves differencing the time series to transform a non-stationary series into a stationary one. The ARIMA model is typically denoted as $ARIMA(n, d, p)$, where d represents the differencing order.

A comparative study [14] examined the performance of newly developed deep learning-based algorithms for time series forecasting, including "Long Short-Term Memory (LSTM)" and Recurrent Neural Networks (RNN), compared to traditional approaches like the ARIMA model. The study found that deep learning-based algorithms outperformed conventional methods, particularly LSTM and RNN. Additionally, the study explored the use of evolutionary algorithms, such as the genetic algorithm, to optimize hyperparameters. This led to developing a robust predictive architecture tailored for precise price forecasting. Furthermore, the study investigated volatilities among different cryptocurrencies, revealing insights into their interconnected dynamics.

3. Related Works

We reviewed related works after discussing the basics of financial market modeling and deep learning's role in time series forecasting. Various studies have explored financial market modeling, revealing insights into price fluctuations. Authors like [23, 33, 32] have contributed. Artificial Neural Networks (ANN) and Autoregressive Integrated Moving Average (ARIMA) models are commonly used [34, 35]. Recent focus has been on Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) for forecasting. For instance, [37] proposes a hybrid ARIMA-LSTM model for better accuracy. We will delve into deep learning techniques, starting with RNNs for sequence prediction, especially in natural language processing. Then, we will discuss LSTM's ability to capture short and long-term dependencies. We will also cover stacked LSTM, which handles complex dependencies. Finally, we will address challenges in manual hyperparameter selection, proposing an automated approach.

3.1. Deep Learning techniques for time series forecasting

Recurrent Neural Networks (RNNs) are crucial in sequence prediction, especially in natural language processing [15]. They excel in handling temporal order and dependencies through feedback loops. RNNs consist of various units like Elman RNN (ERNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). Despite alternative variants such as depth-gated LSTM and bidirectional RNNs, their application in forecasting is limited. ERNN has shown comparable performance to gated units with faster training. Selecting optimal hyperparameters for Neural Networks remains a challenge. Our proposed architecture aims to automate hyperparameter optimization, ensuring superior results and minimal error.

Long Short-Term Memory (LSTM) networks address the limitations of traditional RNNs by introducing a specialized architecture to capture both short-term and long-term dependencies in sequential data. Originally proposed by Hochreiter and Schmidhuber [3], LSTMs consist of memory cells with self-connections in the recurrent hidden layer. These memory cells and adaptive gate units control the information flow within the network.

The structure of an LSTM memory cell c_t includes three gates: the input gate i_t, the forget gate f_t, and the output gate o_t. At each time step t, the input gate determines the information to be added to the cell state S_t (memory), the forget gate decides which information to discard from the cell state, and the output gate determines which information from the cell state will be used as output. The gates allow data to be filtered, discarded, or added, enabling LSTMs to capture short- and long-term correlation features within time series data. One significant advantage is the ability to address the vanishing gradient problem, which is crucial for improving the generalization performance of the network.

To increase the depth and capacity of LSTM networks, stacking LSTM layers is a common approach. The output of the $(L-1)$th LSTM layer at time t serves as input to the Lth layer. This stacking enhances the network's ability to capture complex dependencies within sequential data. The stacked LSTM structure can be described as follows: Let h_{Lt} represent the output of the$L$th layer at time t, and $h_{L-1,t}$ denote the output of the $(L-1)$th layer at the same time step. Each layer L produces a hidden state h_{Lt} based on the current output of the previous layer $h_{L-1,t}$ and time t.

The forget gate f_{Lt} of the Lth layer calculates the input for cell state c_{Lt-1} by

$$f_{Lt} = \sigma(W_{Lf}[h_{Lt-1}, h_{L-1,t}] + b_{Lf}),$$

where $\sigma(\cdot)$ is a sigmoid function, and W_{Lf} and b_{Lf} are the weights matrix and bias vector of layer L regarding the forget gate, respectively. Subsequently, the input gate i_{Lt} of the Lth layer computes the values to be added to the memory cell c_{Lt} by

$$i_{Lt} = \sigma(W_{Li}[h_{Lt-1}, h_{L-1,t}] + b_{Li}),$$

where W_{Li} is the weights matrix of layer L regarding the input gate. The output gate o_{Lt} of the Lth layer filters the information and calculates the output value by

$$o_{Lt} = \sigma(W_{Lo}[h_{Lt-1}, h_{L-1,t}] + b_{Lo}),$$

where W_{Lo} and b_{Lo} are the weights matrix and bias vector of the output gate in the L layer. Finally, the output of the memory cell is computed by $h_t = o_{Lt} \cdot \tanh(c_{Lt})$, where $\tanh(\cdot)$ is the hyperbolic tangent function, and

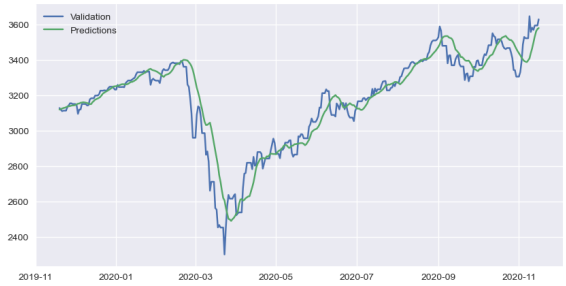
$$c_{Lt} = f_{Lt} \cdot c_{Lt-1} + i_{Lt} \cdot \tilde{c}_{Lt-1},$$

$$\tilde{c}_{Lt} = \tanh(W_{L\tilde{c}}[h_{Lt-1}, h_{L-1,t}] + b_{L\tilde{c}}).$$

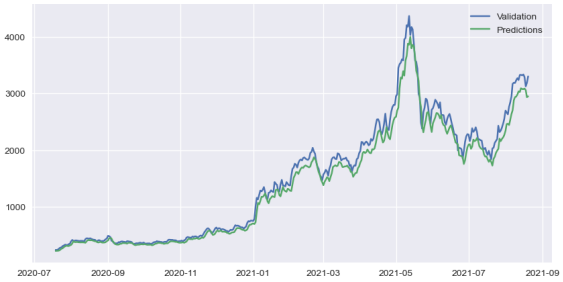
3.2. Exploring Hyperparameter Sensitivity in Traditional Price Prediction Models

In this section, we investigate the sensitivity of hyperparameters in traditional price prediction models, RNN, and LSTM networks. The selection of hyperparameters, such as the number of layers, learning rate, regularization parameter, and window size, plays a crucial role in the performance of these models. We propose an architecture that combines RNN, GRU, and LSTM with the Genetic Algorithm (GA) to automate the selection of hyperparameters for price prediction. We present our methodology, including data preparation, hyperparameter selection, and price prediction, and discuss the results of our experiments. Additionally, we analyze the impact of hyperparameters on prediction accuracy and highlight the challenges of manual hyperparameter selection in traditional time series forecasting.

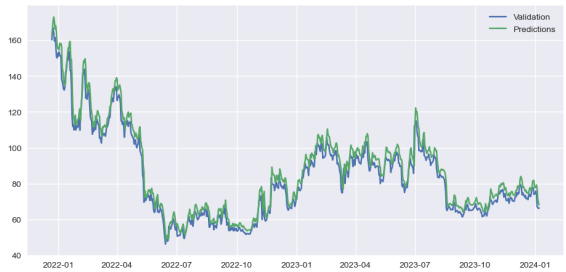
We will now show, in Figure 1, the results of price predictions for the studied coins using manual hyperparameter selection:



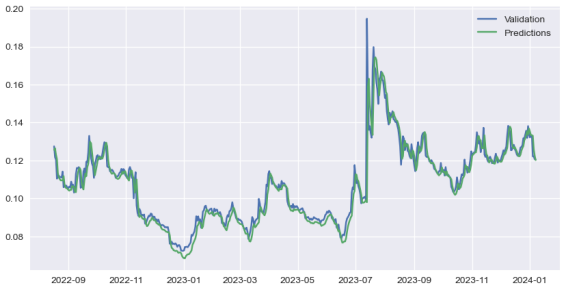
(a) BTC



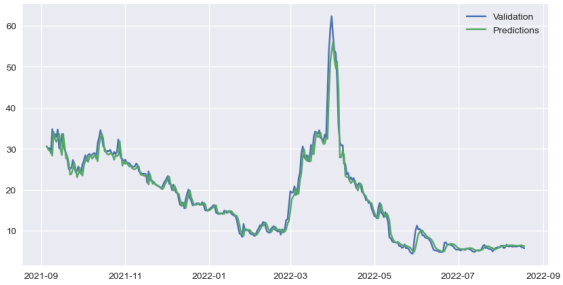
(b) ETH



(c) LTC



(d) XLM



(e) WAVES

Figure 1. Prices prediction of all used coins

After showing the prediction using a manual selection of the hyperparameter, We will analyze the error change over those hyperparameters to explore the impact of different hyperparameters on prediction accuracy and to

identify the set of hyperparameters that have the most significant influence on the model's effectiveness.

The Table 1 below shows how varying the 'window' parameter affects the training and validation errors. This parameter decides the model's historical data scope. A smaller window captures short-term trends well but may overlook long-term patterns, increasing errors. In contrast, a larger window offers a broader view but can add noise. These results highlight the importance of choosing an optimal window size for better prediction accuracy.

Window	2	6	8	16	20	30	40
Train Error	0.02219	0.02226	0.01445	0.012710241	0.015578	0.016234	0.04099
Validation Error	0.02838	0.04043	0.01566	0.01294743	0.018206	0.020463	0.05525

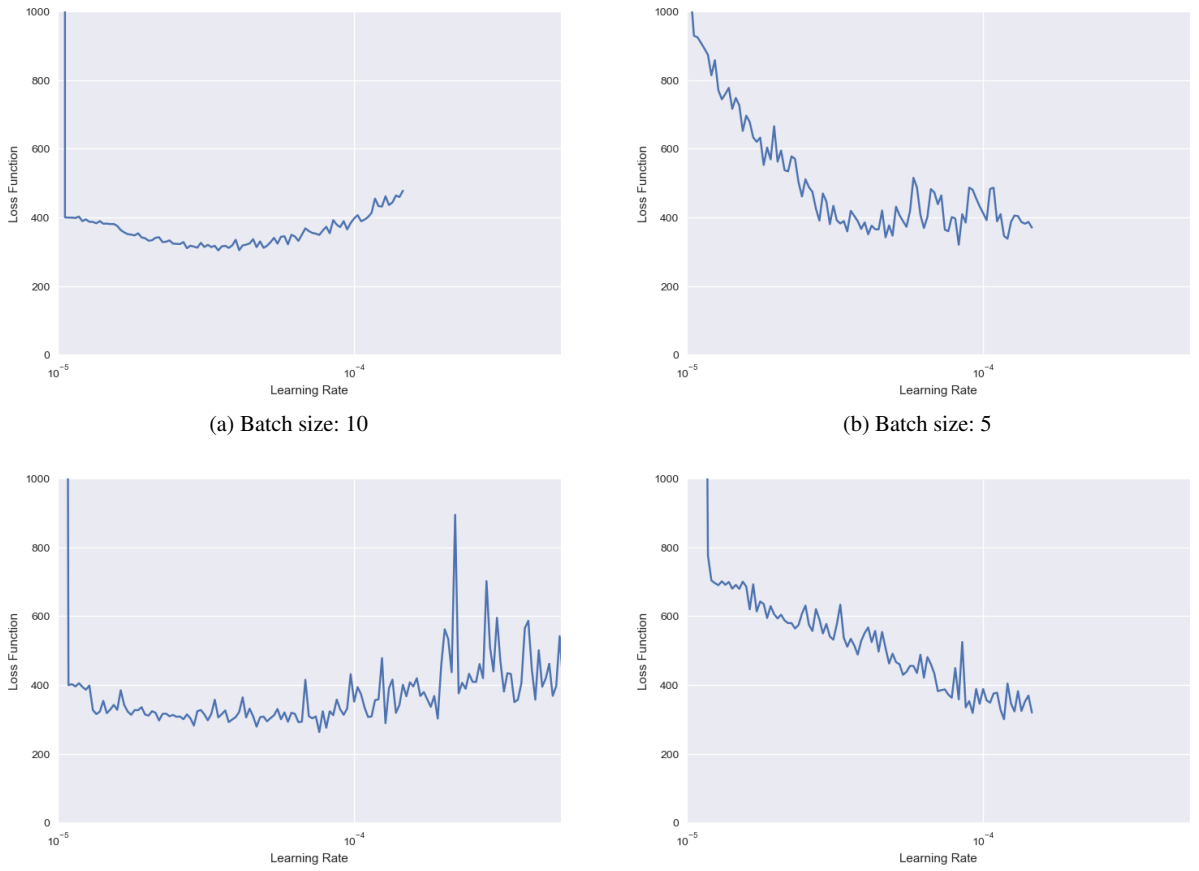
Table 1. Train and Validation Errors for Different Window Sizes, batch size : 10, 50 epochs and Learning rate $= 10^{-5}$

After analyzing the influence of the 'Window' hyperparameter, we continue exploring more hyperparameters with many models. We will now evaluate the prediction quality with different methods, including SimpleRNN, LSTM, Bi-LSTM, and LSTM, with changing epoch values; Table 2 shows the variation of those values with error.

		Currencies				
		BTC	**ETH**	**LTC**	**XLM**	**WAVES**
		Train - Valid	**Train - valid**	**Train - Valid**	**Train/Valid**	**Train - Valid**
Model - Epoch						
SIMPLE	50	0.0139 - 0.0033	0.0128 - 0.0437	0.0151 - 0.0410	0.0147 - 0.0279	0.0136 - 0.0112
	30	0.0114 - 0.0255	0.0137 - 0.0266	0.0126 - 0.0382	0.0143 - 0.0357	0.0109 - 0.0323
LSTM	50	0.0129 - 0.0194	0.0122 - 0.0272	0.0124 - 0.0181	0.0135 - 0.0303	0.0129 - 0.0239
	30	0.0132 - 0.0177	0.0159 - 0.0248	0.0172 - 0.0170	0.0174 - 0.0189	0.0191 - 0.0214
BI-LSTM	50	0.0137 - 0.0299	0.0140 - 0.0280	0.0139 - 0.0339	0.0126 - 0.0361	0.0153 - 0.0259
	30	0.0150 - 0.0275	0.0166 - 0.0360	0.0138 - 0.0334	0.0165 - 0.0428	0.0209 - 0.0423
GRU	50	0.0111 - 0.0238	0.0139 - 0.0333	0.0191 - 0.0416	0.0168 - 0.0324	0.0221 - 0.0228
	30	0.0119 - 0.0288	0.0147 - 0.0337	0.0122 - 0.0335	0.0120 - 0.0317	0.0132 - 0.0345

Table 2. Change of Error while changing epochs number with different method : SIMPLE, LSTM, Bi-LSTM and GRU

More importantly, hyperparameters influence the prediction. For instance, the learning rate, even a small change in its value, can change the convergence of the training model. We will now show a figure that shows the change of the loss function over the Learning Rate by changing other parameters, like the number of epochs, Batch size, etc., and keeping the learning rate value.



(a) Batch size: 10

(b) Batch size: 5

(c) Batch size: 15 and 200 epochs

(d) Batch size: 15

Figure 2. Change of Loss function over Learning Rate, using different hyperparameters

Figure 2 shows the changes due to a Small change in the learning rate value. It can be easy to get the interval that contains the good value and train our model with it, but this can be challenged if we have many hyperparameters that are combined and given to the model. So, without a method to get the set with good hyperparameters, this can lead to undesirable investments, given that they affect investors' budgets.

In conclusion, RNN and LSTM have many parameters, including the number of layers, learning rate, regularization parameter, and windows. All those parameters should be chosen carefully to obtain the desired result. In addressing the challenge of automating hyperparameter optimization, several studies have tackled this issue to automate and optimize the selection process. Notably, [36] employs the genetic algorithm to automate the search for hyperparameters. However, their study focuses on a limited set of hyperparameters, contrasting with our method, which aims to explore a broader range of hyperparameters to enhance prediction performance.

In this paper, we use the RNN and LSTM for price prediction, in combination with the Genetic Algorithm (GA), to propose an architecture to automate the selection of hyperparameters for price prediction. This architecture comprises three main steps: input data preparation and visualization, selecting the best hyperparameters, and predicting future prices. We can predict prices using RNN and LSTM, with a small error, as in the following figure, which shows the result of the predicted price of 5 coins, where our model was trained with the training dataset. Then, we predict the price for the validation data, which is necessary to find a way to determine the set of best hyperparameters to ensure good results of prediction.

4. Proposed approach

Our approach integrates time-series analysis with deep learning, specifically utilizing neural networks, to develop a predictive model for cryptocurrency prices. This innovative method goes beyond traditional approaches by combining RNN and LSTM with the Genetic Algorithm (GA) to automate hyperparameter selection, enhancing prediction accuracy and efficiency in the dynamic cryptocurrency market. Instead of relying on ARIMA, we harness the capabilities of RNN and Long Short-Term Memory (LSTM) networks, combined with a Genetic Algorithm, to automate hyperparameter selection for improved prediction accuracy. This section provides a detailed overview of our system and dataset, outlines the data prepossessing steps, and describes the design of our system tailored to meet specific requirements. We delve into the structural details of our learning algorithm, highlighting the integration of neural networks and Genetic Algorithm to establish an automatic architecture that optimizes hyperparameters and refines our model's solution, enhancing accuracy in forecasting within the dynamic cryptocurrency market.

4.1. Evaluation metrics

To evaluate our approach's effectiveness and reliability, we utilize various evaluation metrics. These metrics offer a thorough insight into the performance of our model, capturing the complexities of volatility and the relationships among all coins considered in our dataset. Key evaluation metrics used in our study include:

- $R2$-Score (Coefficient of Determination): The $R2$-Score is a coefficient of determination denoted as R^2, defined by the formula:
$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{5}$$
Here, SS_{res} represents the sum of squared residual errors, and SS_{tot} is the total sum of errors.
- Mean Absolute Percentage Error ($MAPE$): is a commonly used metric to measure the average absolute error in percentage, defined as:
$$MAPE = \frac{1}{n}\sum_{i=0}^{n}\frac{|A - F|}{A} \tag{6}$$
Where A represents the actual measurements, and F denotes the calculated future measurements.
- Root Mean Square Error ($RMSE$): $RMSE$ is the standard deviation of the residuals and measures the spread of these residuals. It is defined by the formula:
$$RMSE = \sqrt{\sum_{i=0}^{n}\frac{(\hat{y}_i - y_i)^2}{n}} \tag{7}$$
Here, $\hat{y}_i$ is the predicted value, and y_i is the actual value.
- Mean Squared Error (MSE): MSE quantifies the quantity of error in the model, given by:
$$MSE = \sum_{i=0}^{n}\frac{(\hat{y}_i - y_i)^2}{n} \tag{8}$$
- Mean Absolute Error (MAE): MAE measures the absolute difference between predicted and actual values, defined as:
$$MAE = \sum_{i=0}^{n}\frac{|\hat{y}_i - y_i|}{n} \tag{9}$$

These metrics collectively provide a comprehensive evaluation framework for our model's performance across various dimensions.

4.2. Dataset visualisation and data reprocessing

Consistent with established deep learning methodologies, our cryptocurrency prediction system is developed by training a model on preprocessed data. This data is systematically divided into two subsets: a training set, which enables the model to learn patterns, and a test set, which assesses the model's predictive accuracy. This methodological approach aligns with standard practices in deep learning, thereby ensuring the robustness and reliability of the cryptocurrency price prediction system.

The dataset used in this study focuses on the S&P 500 index, a vital indicator of the U.S. stock market, representing the performance of 500 leading companies across various sectors. It includes daily records with essential features such as opening and closing prices, highs, lows, and trading volume. Our analysis concentrated on daily closing prices commonly used in financial forecasting. To ensure accuracy and reliability, we applied preprocessing steps, including data cleaning, normalization, and transformation, which enhanced the reproducibility of our results and provided a solid foundation for future research in stock price prediction.

Our approach organizes the dataset into distinct files, each corresponding to a specific cryptocurrency, such as 'Bitcoin,' 'Ethereum,' 'Litecoin,' 'Stellar,' and 'Waves.' This dataset encapsulates the price variations of these coins from 2017 to 2021, providing a comprehensive view of the cryptocurrency market. The Figure 3 shows the coins prices visualization:

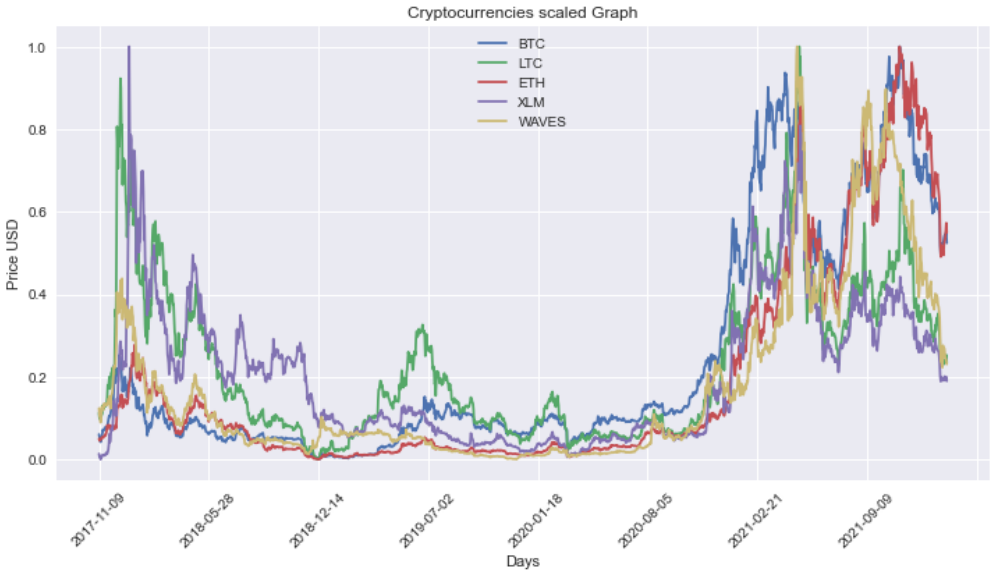


Figure 3. Variation of dally prices

The daily simple return measures the dollar variation in the price of a cryptocurrency as a percentage of the previous day's closing price. A positive return indicates an increase in the cryptocurrency's value, while a negative return signifies a decrease. In the Figure 3, a graphical representation will illustrate the percentage changes over days for all the cryptocurrencies considered. For additional examples illustrating the relationship between Daily Simple Returns (DSR) and some cryptocurrency prices, refer to [29], which explores extreme dependence and correlation within high-frequency cryptocurrency data.

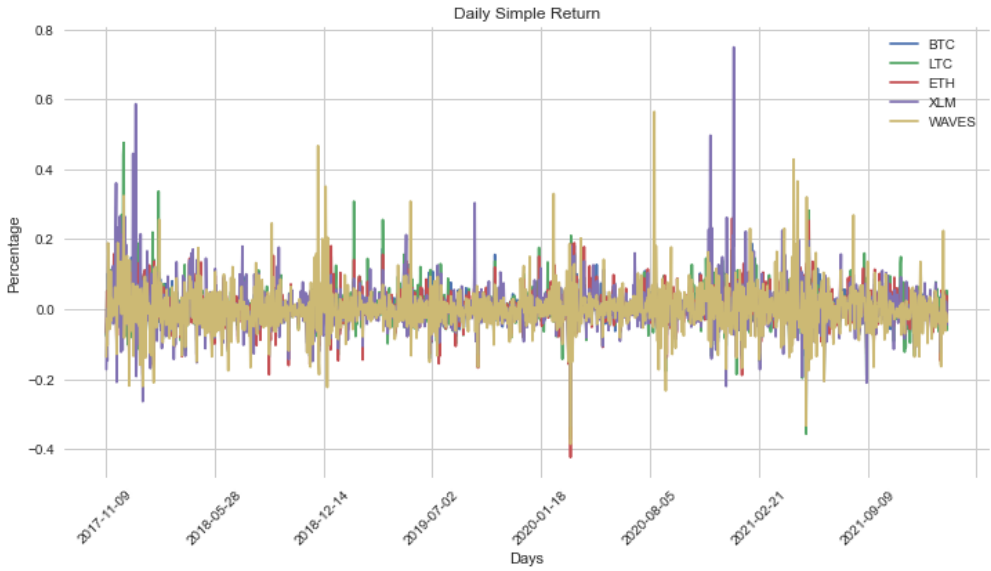


Figure 4. Daily Simple Return

As shown on the Figure 4, the coins are changed over time because of the relations and volatilization between all coins; this will allow us to analyze the volatilizes of all coins. The following table shows the volatilizes of each coin :

Coin	Volatilise of cryptocurrency
BTC	0.041077
LTC	0.057684
ETH	0.051653
XLM	0.067722
WAVES	0.067430

Table 3. Volatilies of coins

Building upon the insights gained from the volatility (Table 3), we now shift our focus to a visual exploration of the correlation between Daily Simple Returns (DSR). The upcoming figure provides a graphical representation, offering a more nuanced understanding of the relationships and trends among the various cryptocurrencies. This transition allows us to delve deeper into the market dynamics and explore how these digital assets interact over time.

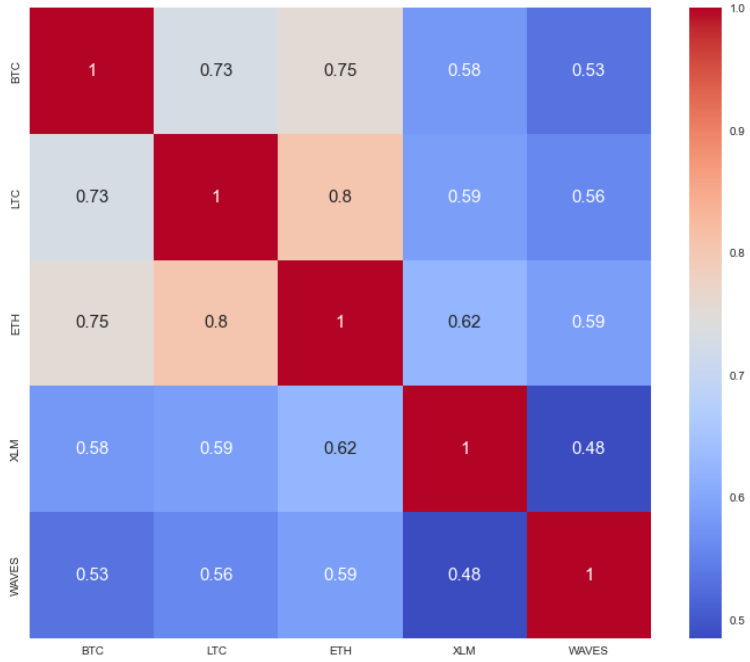


Figure 5. Heatmap showing the average correlation values across DSR for all feature variables.

The correlation matrix depicted in Figure 5 visually represents the correlation coefficients between variables. Notably, it highlights a robust positive correlation between cryptocurrencies, including BTC, LTC, and ETH. This observation implies that corresponding movements in the others mirror fluctuations in one targeted cryptocurrency. As we shift our focus to Transaction Volumes, the subsequent section aims to explore the trading activity throughout our time-frame visually. Transaction Volumes, reflecting the number of units (stocks, bonds, etc.) exchanged in a single day, offer a pivotal perspective on market dynamics. The Table 4 will further elucidate and contextualize the transactional aspects, providing valuable insights into the intensity and patterns of trading activities.

	BTC	LTC	ETH	XLM	WAVES
TTV	39106884742661	4219291800175	19170292239155	765839471707	111694330830

Table 4. Total Transaction Volume (TTV) for cryptocurrencies.

The findings from the Table 4 confirm a discernible relationship between the transaction volumes of Bitcoin (BTC) and Ethereum (ETH). This correlation suggests that changes in the transactional activities of these two cryptocurrencies are interconnected, implying a potential proximity in predictive outcomes. To further illustrate this relationship, the forthcoming box plot in Figure 6 presents a visual example of the distribution of BTC transaction volumes concerning ETH. This graphical representation, exemplifying a box plot, provides insights into the comparative distribution and potential patterns in transaction volumes between these prominent cryptocurrencies.

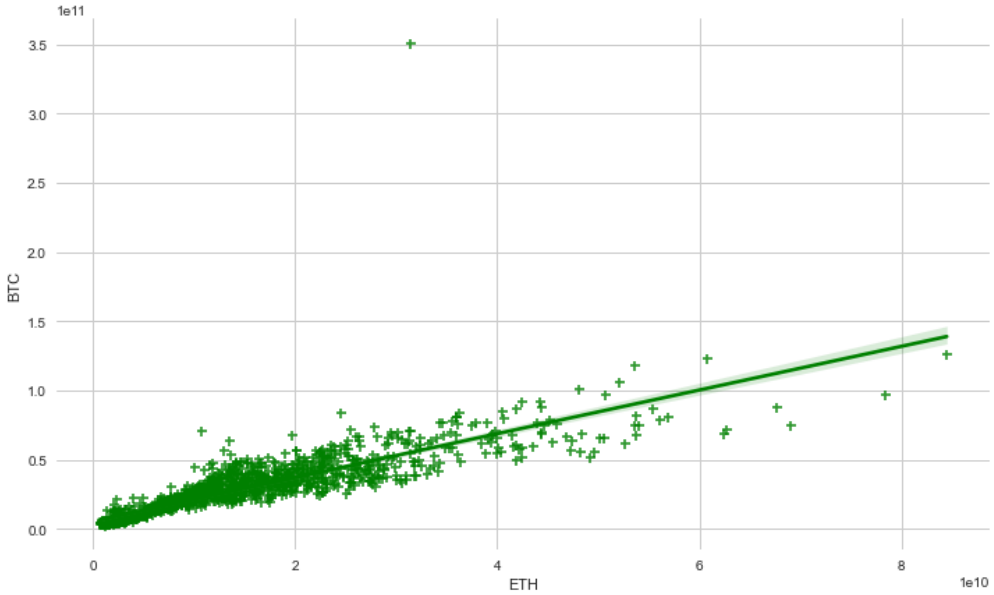


Figure 6. Percentage of the total transactions.

4.3. Hyperparameter optimization

After the dataset visualization, we identify the more critical hyperparameters significantly influencing price prediction. The hyperparameter space explored by the GA was carefully defined with the following ranges and types:

- **Learning Rate**: Ranges from 0.0001 to 0.5, sampled on a logarithmic scale.
- **Number of Epochs**: Set between 5 and 200, to balance computational efficiency and model convergence.
- **Window Size (for LSTM models)**: Varies between 3 and 20, accounting for different time horizons in sequence data.
- **Batch Size**: Ranges from 16 to 128, impacting the model's training stability and speed.
- **Dropout Rate**: Varies between 0.1 and 0.95, used to prevent overfitting by randomly dropping units during training.
- **Regularization parameter**: Applied with values ranging from 0.00001 to 0.1, used to penalize large weights and control model complexity.
- **Recurrent Dropout (for RNN and LSTM models)**: Ranges from 0.1 to 0.95, used to drop units in recurrent connections to combat overfitting.

Then comes bilevel optimization using GA, which can be resumed by constructing two levels of problems as optimization ones, with one of the two problems being a constraint of the other.

Among Evolutionary Algorithms (EA), probably the best known is the Genetic Algorithm (GA) [12]; the GA was receiving remarkable attention all over the world [16]. Even previously obtained predictions show us a stable performance. However, some sensitive hyperparameters can affect the quality of the result, more the fact that the best hyperparameter for a coin can not give a good result for others, and, as we are dealing with the budget of a company, we need to reduce the error and give traders a correct result, the thing that will lead us to use the Genetic Algorithm (GA) for the selection of the best hyperparameter to use while training our model. The parameter called

ĺearning_rateís an important parameter that can affect the quality of predicted results.

However, in neural networks, certain parameters, known as hyperparameters, are not updated autonomously during training. These parameters are not the variables that need to be tuned or optimized through neural network training; instead, they are set based on prior knowledge by the developer or automatically by external model mechanisms. Examples include the learning rate, number of epochs, generalization variables, and window size in the LSTM model. Hyperparameters remain constant throughout training but can significantly impact the algorithm's accuracy.

Evolutionary algorithms operate on the premise of managing a population of individuals in an environment with finite resources. The underlying concept involves fostering a competitive atmosphere where individuals vie for these resources, leading to natural selection akin to "survival of the fittest." A quality function is introduced in the context of optimization problems, and a set of candidate solutions representing elements of the function's domains is randomly generated. Subsequently, the quality function is applied to assess the fitness of these candidates, with higher values indicating superior fitness.

The evolutionary process unfolds by selecting some of the most promising candidates to propagate the next generation. This propagation involves the application of recombination and mutation. Recombination, executed on two or more selected candidates (referred to as parents), generates one or more new candidates (the children). On the other hand, mutation is applied to a single candidate, creating a new candidate. The dynamic interplay of these mechanisms mirrors the genetic processes observed in biological organisms.

Genetic algorithms (GAs) are adaptive techniques for tackling search and optimization problems. These algorithms mimic the genetic state of a population by integrating operators such as natural selection, mutation, and crossover. In the forthcoming sections, we will delve into applying the genetic algorithm as a fundamental component of our methodology, leveraging its adaptive nature to enhance our approach to predicting cryptocurrency prices.

The Genetic Algorithm was implemented with the following operators:

- **Selection**: We employed a tournament selection method, where individuals are selected based on their fitness relative to a randomly chosen subset of the population.
- **Crossover**: A single-point crossover was applied, where offspring inherit portions of their genetic material from two parent solutions.
- **Mutation**: A Gaussian mutation was used to introduce variability, where small changes are made to individual parameters based on a normal distribution.

The implementation of GAs is distinguished from classical optimization methods by several key characteristics, which contribute to their effectiveness:

1. **Population-Based Search:** Unlike classical methods that use a single-point search, GAs operate on a population of solutions. This parallel search mechanism enables the algorithm to explore multiple regions of the solution space simultaneously, thereby increasing the likelihood of discovering the global optimum.
2. **Versatility in Parameter Coding:** GAs are versatile in encoding parameter sets, making them suitable for various problems. This adaptability allows GAs to be applied to diverse optimization challenges, demonstrating their broad applicability.
3. **Randomness in Search Process:** Incorporating stochastic elements like crossover and mutation guides the search process towards optimal regions. This randomness differentiates GAs from deterministic methods and helps overcome local optima to achieve better solutions.

Algorithm [12] is a simplified pseudo-code of the genetic algorithm.

Algorithm 1 Genetic Algorithm

Input: Population size, mutation rate, crossover rate, termination criteria
Output: Best solution
Initialization: Create initial random population

while Termination criteria not met **do**
  Evaluation: Assess fitness of each individual
  Selection: Choose the fittest individuals for reproduction
  Crossover: Combine pairs of individuals to produce offspring
  Mutation: Apply random changes to offspring
  Replacement: Form new population from parents and offspring
end while

In summary, the application of Genetic Algorithms (GAs) in our study involves precisely implementing selection, crossover, and mutation operators to optimize hyperparameters for predicting cryptocurrency prices. The unique characteristics of GAs, such as their population-based search and adaptability, underscore their effectiveness in handling complex optimization tasks. GAs operate with a population of candidate solutions, use selection, crossover, and mutation operators to evolve solutions, and are particularly suited for problems where traditional optimization methods may fall short.

Literature supports the efficacy of GA-based optimization techniques in locating the global optimum region, which is attributed to their inherent parallelism. These algorithms exhibit resilience in handling non-convex problems and do not necessitate the objective function to be differentiable. These well-established properties have led to the widespread adoption of GA as a fundamental solution approach.

By using the neural network models for the price prediction and the GA for the optimization of hyperparameters, we will compare two architectures for the proposed contribution.

4.3.1. Learning On Each Epoch (LOEE)

In this method, we integrate a Genetic Algorithm with a neural network to dynamically adjust the learning rate at the end of each epoch, progressively fine-tuning it to approach the optimal value. The effectiveness of this method is closely tied to the architecture of the neural network employed, which includes multiple LSTM layers, dropout mechanisms, and specific activation functions.

It is important to note that this approach focuses exclusively on optimizing the learning rate as the primary hyperparameter. Due to the iterative nature of the learning rate adjustment at each epoch, other hyperparameters, such as the number of LSTM units, dropout rates, and activation functions, remain fixed throughout the training phase. This targeted optimization strategy ensures a focused approach without the complexities of simultaneously adjusting multiple hyperparameters.

Network Structure:

- **Layers:** The model comprises three LSTM layers, each followed by a dropout layer to prevent overfitting.
- **Layer Configuration:** The LSTM layers have 128 units each, with a dropout rate of 0.2 applied to each LSTM layer.
- **Activation Functions:** We used the *tanh* activation function within the LSTM units and a *sigmoid* activation function for the final output layer.

- **Output Layer:** A dense layer with a single neuron and a *sigmoid* activation function for binary classification.

- **Learning Rate Range:** The Genetic Algorithm searches for the optimal learning rate within a predefined range (e.g., [0.001, 0.1]).
- **Number of Epochs:** The training is conducted for a fixed number of epochs (e.g., 100 epochs) to ensure adequate learning.
- **Regularization:** Dropout with a rate of 0.2 is applied to each LSTM layer to mitigate overfitting.

The detailed system architecture of the LOEE method is depicted in Figure 7. This figure illustrates how the neural network, combined with the Genetic Algorithm, iteratively adjusts the learning rate at the end of each epoch. The architecture showcases the flow from input data through the LSTM layers, the application of the dropout mechanism, and the final output layer, with the Genetic Algorithm optimizing the learning rate dynamically based on the performance metrics.

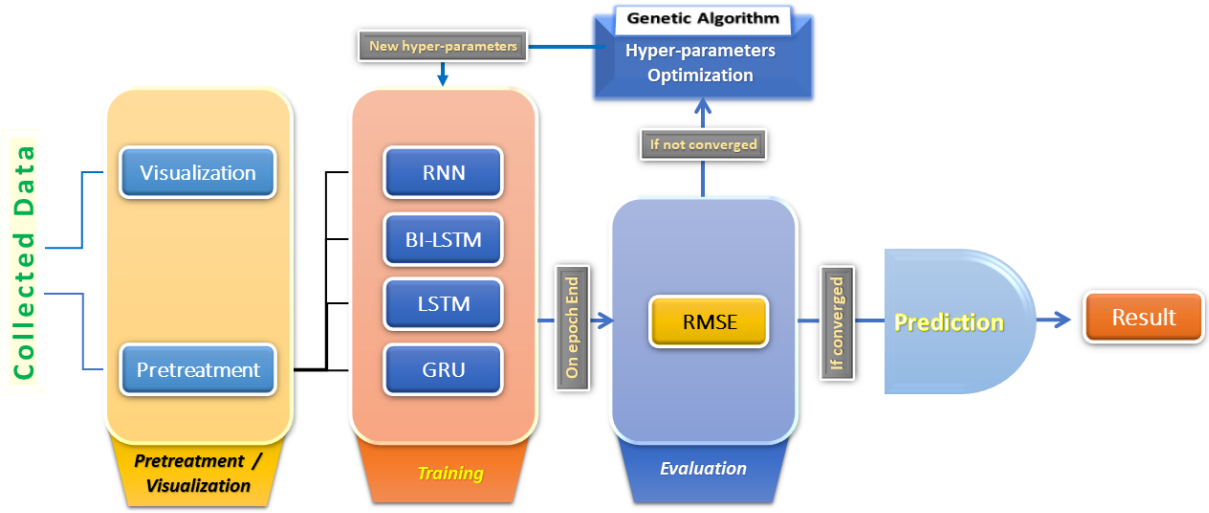


Figure 7. System architecture

While the training, to get an idea about the process of convergence of this method and to use a fixed number of epochs, the following Figure 8 will show the evolution of the loss function at the end of each epoch:

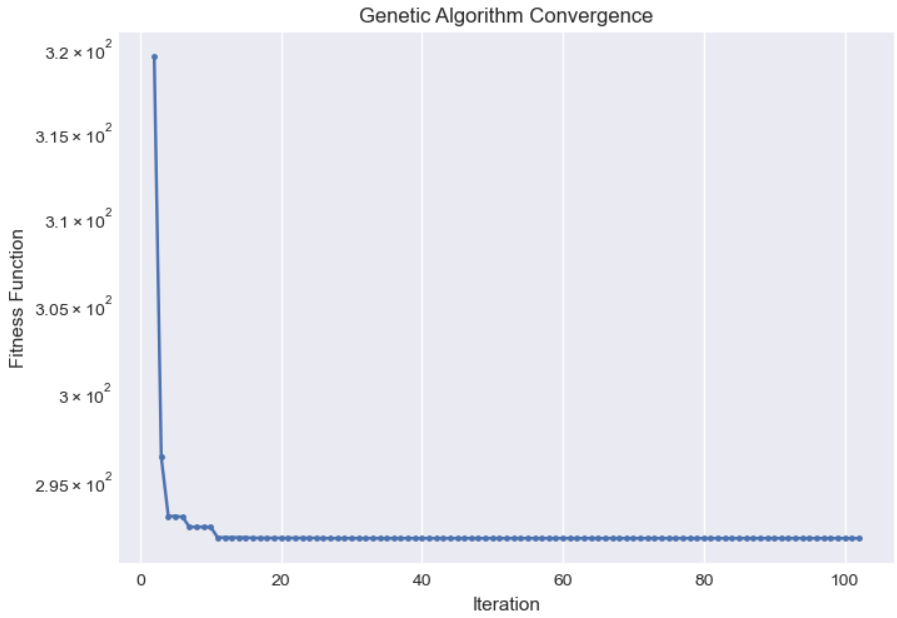


Figure 8. LOEE Convergence

4.3.2. Learning At End (LAE)

In the Learning At End (LAE) method, we use the Genetic Algorithm to optimize a set of hyperparameters by evaluating the model's performance after each complete training cycle. Unlike the LOEE method, where only the learning rate is updated iteratively, the LAE method allows for simultaneously optimizing multiple hyperparameters. This method requires more converging iterations, as each cycle adjusts and evaluates several parameters.

For the LAE method, we define search intervals for various hyperparameters, while some are kept fixed to ensure consistency. Specifically, activation functions and certain model configurations are predetermined to maintain uniformity. In contrast, hyperparameters such as the learning rate, number of epochs, batch size, dropout rate, and regularization parameter are explored within specified ranges. This approach allows for a comprehensive evaluation of potential hyperparameter values.

The architecture of the neural network used in LAE is detailed as follows:

- **Learning Rate:** We search within a range from 10^{-5} to 10^{-1}, allowing the algorithm to explore a broad spectrum of values to find the optimal learning rate.
- **Number of Epochs:** The range of epochs is set between 5 and 200. This range helps balance between underfitting and overfitting, ensuring sufficient training without excessive computational cost.
- **Batch Size:** We explore batch sizes from 16 to 128, as different batch sizes can significantly impact training dynamics and generalization performance.
- **Dropout Rate:** The dropout rate is varied between 0.1 and 0.9 to help prevent overfitting by regularizing the model.
- **Regularization Parameter:** We investigate regularization parameters from 10^{-5} to 10^{-1}, which helps control model complexity and prevent overfitting.

The Figure 9 illustrates the architecture of the LAE method and shows how the hyperparameters are managed and optimized.

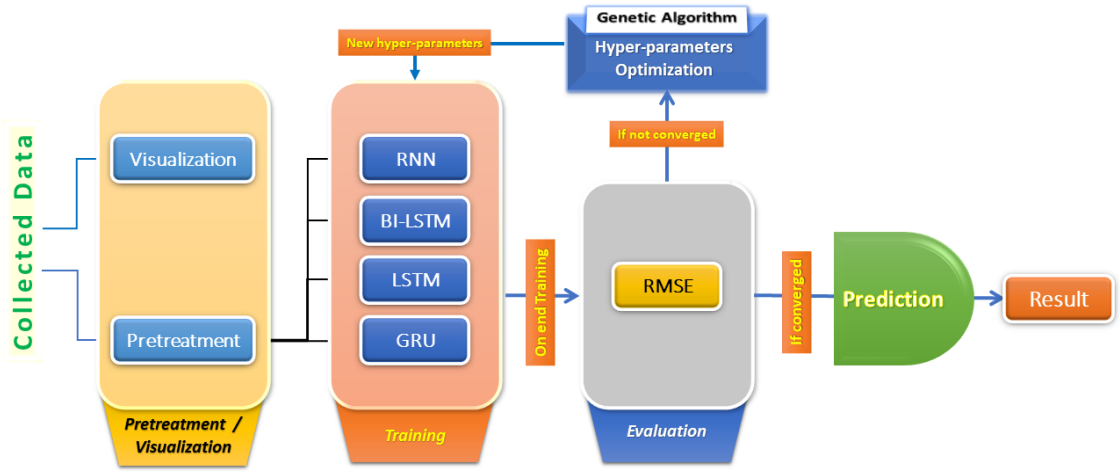


Figure 9. System architecture

In contrast to the LOEE method, which focuses on optimizing a single parameter during training, the LAE method involves optimizing multiple hyperparameters simultaneously. This approach typically requires more iterations to converge, as demonstrated in Figure 10, which shows an example of the convergence behavior for this method.

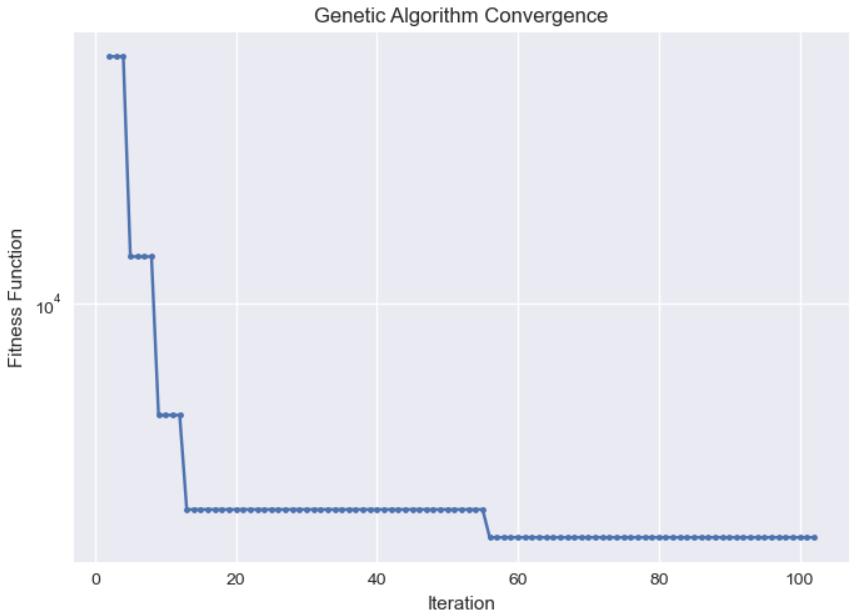


Figure 10. Convergence of LAE Methode

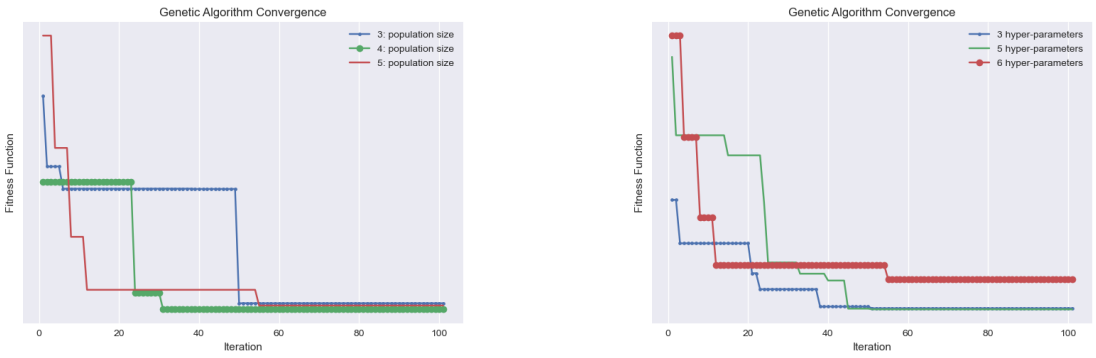
4.3.3. comparison between LOEE and LAE As we said, on the LOEE method, only one hyperparameter was optimized and obtained at the end; on the other hand, with LAE, we can optimize many hyperparameters; for instance, the number of used epochs cannot be changed on the other methods (LOEE), but in this one (LAE), as we evaluate and update at the end of all training, we can change the number of epoch with additional hyperparameters. These things make this one more efficient than the other. This comparison leads us to our proposed method. We will use the LAE method to optimize the set of all sensitive hyperparameters.

5. Results and discussion

In this section, after the dataset, Daily Simple Return (DSR), and Volatilities are visualized, we will show our experiment results in prediction and price using a Time Series and Neural network model.

5.1. Genetic Algorithm LOEE and LAE

In this section, we will show the result of using the Genetic Algorithm (GA) in optimizing the set of hyperparameters of the RNN and LSTM models. Each model has its own set of hyperparameters, so we will run the GA on each model to optimize the parameters. The following figure will show the convergence of the fitness function of the GA in each model:

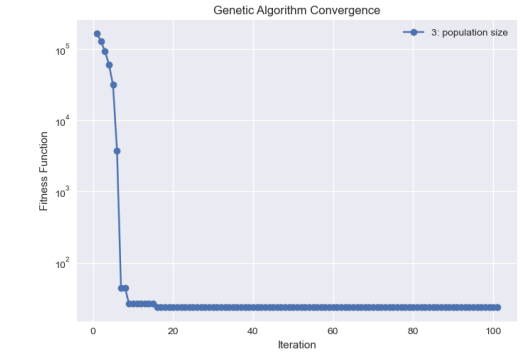


(a) Convergence over size of population

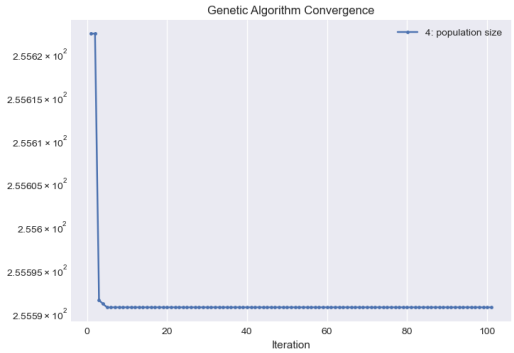
(b) Convergence over hyperparameters number

Figure 11. LAE: Convergence of the Genetic Algorithm

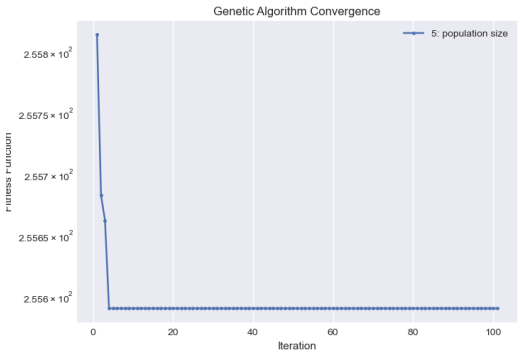
The Figure 11 presented below, we illustrate the convergence behavior of the Genetic Algorithm with, using the LAE Architecture, respect to variations in both the number of hyperparameters and the population size. Sub-figure (a) explores the influence of adjusting the population size while optimizing six hyperparameters. Conversely, in Sub-figure (b), the Genetic Algorithm's parameters remain constant, and we observe the impact of altering the number of hyperparameters targeted for optimization. This analysis provides insights into how the algorithm converges under different configurations, shedding light on the interplay between hyperparameter tuning and population size.



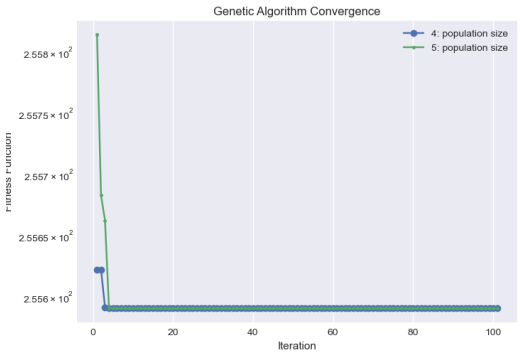
(a) 3: size of population



(b) 4: size of population



(c) 5: size of population



(d) 4 and 5: size of populationu

Figure 12. LOEE: Convergence of the Genetic Algorithm

In the depicted Figure 12, we showcase the convergence patterns of the Genetic Algorithm alongside the LOEE method, which focuses on tuning a single hyperparameter, namely the learning rate, within the LOEE. Each distinctive curve in the graph corresponds to a different setting of the population size parameter within the Genetic Algorithm. The diverse population sizes are systematically examined to understand their impact on the convergence behavior of the LOEE method, providing valuable insights into the interplay between learning rate optimization and genetic algorithm population size.

5.1.1. Detailed Comparison of LAE and LOEE Architectures

In this section, we delve deeper into the comparative performance of the LAE and LOEE architectures, focusing on the impact of hyperparameter optimization on convergence and prediction results. The primary distinction between these methods is their approach to hyperparameter tuning: LOEE optimizes a single hyperparameter, while LAE allows for the optimization of multiple hyperparameters simultaneously. We will analyze and compare these methods on several fronts: their handling of hyperparameter optimization, effects on convergence speed, and influence on prediction accuracy. This analysis will provide insights into each approach's practical benefits and limitations, emphasizing the advantages of a more flexible and expansive optimization strategy over a more constrained one.

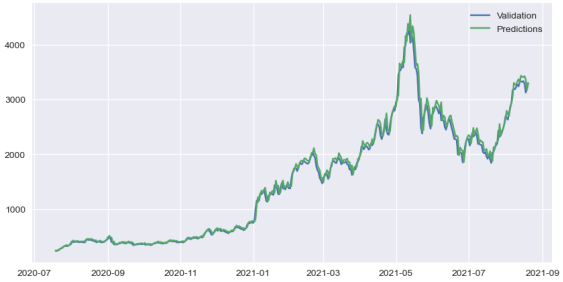
The LAE method allows for the optimization of multiple hyperparameters simultaneously, including but not limited to learning rate, number of epochs, batch size, dropout rate, regularization parameters, and recurrent dropout. This flexibility facilitates a more comprehensive exploration of the hyperparameter space. However,

this broader scope can lead to slower convergence, as optimizing a more extensive set of hyperparameters involves a more complex search process. This is illustrated by the convergence plots in Figure 11. Sub-figure (a) shows the effect of varying the population size on the convergence of the LAE method when optimizing multiple hyperparameters. The slower convergence observed in this sub-figure reflects the increased complexity and thoroughness of the hyperparameter search compared to methods with a more limited scope, such as LOEE, which focuses on a single hyperparameter.

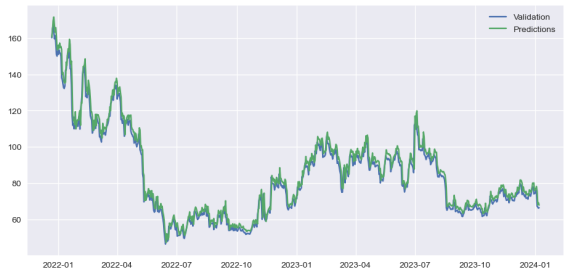
In contrast, the LOEE method optimizes a single hyperparameter, precisely the learning rate. As depicted in Figure 12, the LOEE approach involves fewer hyperparameters and shows a slower convergence pattern. Sub-figure (a) in this figure displays convergence behaviors across different population sizes, revealing that while the optimization of the learning rate is effective, the limited number of hyperparameters restricts the overall flexibility and efficiency of the method.

Overall, the detailed analysis of convergence and prediction performance underscores the LAE approach's superiority in flexibility and effectiveness in hyperparameter optimization. The ability to optimize a broader range of hyperparameters and achieve rapid convergence contributes to the enhanced predictive accuracy and robustness of the LAE method over the LOEE method.

(a) BTC



(b) ETH



(c) LTC

(d) XLM

(e) WAVES

Figure 13. Prices prediction of all used coins

5.2. Results of Predected Prices

Now, transitioning to the prediction results, on Figure 13, we present a series of Sub-figures, each dedicated to a specific cryptocurrency. For each coin, the depicted graphs illustrate a side-by-side comparison of the actual price trajectory and the predicted values generated through our methodology. This visual representation offers a comprehensive view of the model's performance across various cryptocurrencies, providing a clear insight into the accuracy and effectiveness of our predictive approach. The predicted prices testify to the model's ability to capture and forecast the intricate dynamics of each cryptocurrency's market behavior.

A distinct narrowing of the disparity between predicted and validation data is readily apparent upon meticulous comparison with previous outcomes achieved without hyperparameter optimization. Furthermore, a substantial decrease in errors has been observed, a natural outcome of the documented impact of hyperparameters. This emphasizes the robust efficacy of the optimization process in fine-tuning the model's performance and elevating its precision. The discernible improvements affirm the pivotal role of hyperparameter optimization in enhancing the overall accuracy and reliability of the model.

Reference	Model	Cryptocurrency	Results
[42]	**ARIMA**	BTC	RMSE = 1718.339
		ETH	RMSE = 136.605
[38]	**LSTM**	BTC	RMSE = 410.399 MAPE = 1.1234%
		ETH	RMSE = 59.507 MAPE = 1.5498%
[39]	**LSTM**	BTC	RMSE = 297.97 MAPE = 2.75%
	CNN	BTC	RMSE = 261.90 MAPE = 2.03%
This Paper	**LSTM**	BTC	RMSE = 40.58 MAPE = 1.08% MAE = 39.05
		ETH	RMSE = 28.13 MAPE = 0.74% MAE = 26.75

Table 5. Comparison with other methods

Our results align with existing research, demonstrating that hyperparameter optimization can significantly enhance model performance. However, the initial model configuration and complexity influence the extent of improvement. To underscore the effectiveness of our approach, Table 5 presents a comparative analysis of our model's performance metrics, specifically RMSE and MAPE, against the current state-of-the-art. The results show that our models consistently outperform the leading methods across these critical metrics, highlighting the

advantages of genetic algorithm-based optimization.

The results presented in the table provide a comparative analysis of forecasting performance for various models applied to Bitcoin (BTC) and Ethereum (ETH). The table highlights the performance of the ARIMA, LSTM, and CNN models in predicting cryptocurrency prices. As noted in [42], the ARIMA model shows high RMSE values, indicating less accuracy in BTC and ETH price predictions. In contrast, the LSTM model, explored in [38] and [39], demonstrates improved forecasting capabilities, with RMSE and MAPE values indicating better performance for BTC. The CNN model, evaluated in [39], further enhances accuracy, particularly for BTC. Our study, using LSTM and GA, reveals superior performance with lower RMSE, MAPE, and MAE values for both BTC and ETH, showcasing the effectiveness of our approach in optimizing hyperparameters and capturing the complex interconnections among cryptocurrencies. This improvement underscores the potential of combining deep learning techniques with automated hyperparameter selection to address forecasting challenges. These results support the efficacy of our methodology in enhancing predictive accuracy and provide valuable insights for investors and researchers in the field.

6. Conclusion and future work

This paper underscores the pivotal role of hyperparameters in shaping the efficacy of price-prediction models, presenting an integrated framework that merges the Genetic Algorithm and Deep Learning to forecast cryptocurrency prices. Our study encompasses a diverse array of cryptocurrencies, acknowledging their volatilities, with the overarching objective of elevating the quality of predictions across the board. As delineated in the results section, our experimental findings unequivocally affirm the robustness and precision of our approach, demonstrating its aptitude for delivering accurate predictions across multiple coins. This research contributes significantly to the broader landscape of cryptocurrency prediction by addressing existing model limitations and proposing innovative solutions. Notably, our exploration delves into the realm of neural networks, elucidating the impact of hyperparameters on their performance. Through the strategic amalgamation of these insights with the Genetic Algorithm, we introduce two distinct architectures, LOEE and LAE, whose comparative analysis guides the selection of the superior model. Subsequently, the chosen architecture is leveraged for price prediction, culminating in our methodological advancements. As part of our future endeavors, we envision expanding this study to incorporate more cryptocurrencies and delve into alternative neural network architectures, further enriching the depth and scope of our predictive models.

However, it is essential to recognize our approach's limitations. One potential challenge is the risk of overfitting, particularly given cryptocurrency markets' inherent volatility and unpredictability. Additionally, the robustness of our model in adapting to sudden changes in market conditions remains an area for further exploration. Despite these challenges, our framework demonstrates strong potential for accurate predictions, but future work should focus on enhancing its adaptability and resilience.

In future work, we plan to expand and refine the application of the Genetic Algorithm by integrating a broader range of hyperparameters, including those that are not linear or continuous. For instance, we will explore hyperparameters such as activation functions, types of recurrent layers (e.g., LSTM or GRU), and categorical parameters like optimization algorithms. Additionally, we aim to incorporate the existing relationships between different cryptocurrencies and assess their influence by integrating these interdependencies as features in our prediction models. This approach will allow us to capture the intricate market dynamics more effectively, potentially enhancing the accuracy and robustness of our predictive models.

Statements and Declarations

- Funding: Not applicable

- Conflict of interest: The authors declare that they have no confict of interest
- Ethics approval: Not applicable
- Consent to participate: Not applicable
- Consent for publication: Not applicable
- Availability of data and materials: The data used this study are available from https://www.kaggle.com/datasets/arashnic/time-series-forecasting-with-yahoo-stock-price
- Code availability: The code that support the findings of this study are available from the corresponding author upon reasonable request
- Authors' contributions: Not applicable

REFERENCES

1. P. R. Winters, *Forecasting sales by exponentially weighted moving averages*, *Management Science*, vol. 6, no. 3, pp. 324–342, 1960.
2. G. E. P. Box, G. M. Jenkins, C. C. Reinsel, G. M. Ljung, *Time series analysis: forecasting and control*, *San Francisco: Holden Bay*, 1976.
3. S. Hochreiter, J. Schmidhuber, *Long short-term memory*, *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
4. A. C. Harvey, *Forecasting, structural time series models and the Kalman filter*, *Cambridge university press*, 1990.
5. E. S. Gardner Jr, *Exponential smoothing: The state of the art*, *Journal of forecasting*, vol. 4, no. 1, pp. 1–28, 1985.
6. S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*,
7. M. Khashei, M. Bijari, *A novel hybridization of artificial neural networks and ARIMA models for time series forecasting*, *Applied soft computing*, vol. 11, no. 2, pp. 2664–2675, 2011.
8. M. Khashei, M. Bijari, G. A. R. Ardali, *Hybridization of autoregressive integrated moving average (ARIMA) with probabilistic neural networks (PNNs)*, *Computers & Industrial Engineering*, vol. 63, no. 1, pp. 37–45, 2012.
9. G. Bontempi, S. Ben Taieb, Y.-A. Le Borgne, *Machine learning strategies for time series forecasting*, *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures 2*, pp. 62–77, 2013.
10. A. A. Ariyo, A. O. Adewumi, C. K. Ayo, *Stock price prediction using the ARIMA model*, *2014 UKSim-AMSS 16th international conference on computer modelling and simulation*, pp. 106–112, 2014.
11. F. Tschorsch, B. Scheuermann, *Bitcoin and beyond: A technical survey on decentralized digital currencies*, *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.
12. K. Sastry, D. Goldberg, G. Kendall, *Genetic algorithms*, *Search methodologies: Introductory tutorials in optimization and decision support techniques*, pp. 97–125, 2005.
13. S. M. Idrees, M. A. Alam, P. Agarwal, *A Prediction Approach for Stock Market Volatility Based on Time Series Data*, *IEEE Access*, vol. 7, pp. 17287–17298, 2019.
14. P. T. Yamak, L. Yujian, P. K. Gadosey, *A comparison between arima, lstm, and gru for time series forecasting*, *Proceedings of the 2019 2nd international conference on algorithms, computing and artificial intelligence*, pp. 49–55, 2019.
15. Y. Yu, X. Si, C. Hu, J. Zhang, *A review of recurrent neural networks: LSTM cells and network architectures*, *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.
16. M. Nachaoui, L. Afraites, A. Laghrib, *A regularization by denoising super-resolution method based on genetic algorithms*, *Signal Processing: Image Communication*, vol. 99, p. 116505, 2021.
17. M. Torky, A. E. Hassanein, *Integrating blockchain and the internet of things in precision agriculture: Analysis, opportunities, and challenges*, *Computers and Electronics in Agriculture*, vol. 178, p. 105476, 2020.
18. M. Pirani, P. Thakkar, P. Jivrani, M. H. Bohara, D. Garg, *A comparative analysis of ARIMA, GRU, LSTM and BiLSTM on financial time series forecasting*, *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, pp. 1–6, 2022.
19. S. Siami-Namini, N. Tavakoli, A. S. Namin, *A comparison of ARIMA and LSTM in forecasting time series*, *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pp. 1394–1401, 2018.
20. I. M. Wirawan, T. Widiyaningtyas, M. M. Hasan, *Short term prediction on bitcoin price using ARIMA method*, *2019 International Seminar on Application for Technology of Information and Communication (iSemantic)*, pp. 260–265, 2019.
21. Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, M. S. Hossain, *Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach*, *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2020.
22. M. M. Patel, S. Tanwar, R. Gupta, N. Kumar, *A deep learning-based cryptocurrency price prediction scheme for financial institutions*, *Journal of information security and applications*, vol. 55, p. 102583, 2020.
23. O. B. Sezer, M. U. Gudelek, A. M. Ozbayoglu, *Financial time series forecasting with deep learning: A systematic literature review: 2005–2019*, *Applied soft computing*, vol. 90, p. 106181, 2020.
24. S. Tanwar, N. P. Patel, S. N. Patel, J. R. Patel, G. Sharma, I. E. Davidson, *Deep learning-based cryptocurrency price prediction scheme with inter-dependent relations*, *IEEE Access*, vol. 9, pp. 138633–138646, 2021.
25. M. S. Ferdous, M. J. M. Chowdhury, M. A. Hoque, *A survey of consensus algorithms in public blockchain systems for crypto-currencies*, *Journal of Network and Computer Applications*, vol. 182, p. 103035, 2021.
26. R. K. Malladi, P. L. Dheeriya, *Time series analysis of cryptocurrency returns and volatilities*, *Journal of Economics and Finance*, vol. 45, no. 1, pp. 75–94, 2021.

27. T. K. Toai, R. Senkerik, I. Zelinka, A. Ulrich, V. T. X. Hanh, V. M. Huan, *ARIMA for Short-Term and LSTM for Long-Term in Daily Bitcoin Price Prediction*, *International Conference on Artificial Intelligence and Soft Computing*, pp. 131–143, 2022.
28. C. Gourieroux, J. Jasiak, *Financial econometrics: Problems, models, and methods*, vol. 2, 2022, *Princeton University Press*.
29. S. Chan, J. Chu, Y. Zhang, S. Nadarajah, *An extreme value analysis of the tail relationships between returns and volumes for high frequency cryptocurrencies*, *Research in International Business and Finance*, vol. 59, p. 101541, 2022.
30. R. P. Masini, M. C. Medeiros, E. F. Mendes, *Machine learning advances for time series forecasting*, *Journal of economic surveys*, vol. 37, no. 1, pp. 76–111, 2023.
31. J. Almeida, T. C. Gonçalves, *A systematic literature review of investor behavior in the cryptocurrency markets*, *Journal of Behavioral and Experimental Finance*, p. 100785, 2023.
32. K. Gajamannage, Y. Park, D. I. Jayathilake, *Real-time forecasting of time series in financial markets using sequentially trained dual-LSTMs*, *Expert Systems with Applications*, vol. 223, p. 119879, 2023.
33. K. He, Q. Yang, L. Ji, J. Pan, Y. Zou, *Financial time series forecasting with the deep learning ensemble model*, *Mathematics*, vol. 11, no. 4, p. 1054, 2023.
34. X. Zhang, C. Zhong, J. Zhang, T. Wang, W. W. Y. Ng, *Robust recurrent neural networks for time series forecasting*, *Neurocomputing*, vol. 526, pp. 143–157, 2023.
35. X. Wang, Y. Kang, R. J. Hyndman, F. Li, *Distributed ARIMA models for ultra-long time series*, *International Journal of Forecasting*, vol. 39, no. 3, pp. 1163–1184, 2023.
36. A. A. Oyedele, A. O. Ajayi, L. O. Oyedele, S. A. Bello, K. O. Jimoh, *Performance evaluation of deep learning and boosted trees for cryptocurrency closing price prediction*, *Expert Systems with Applications*, vol. 213, p. 119233, 2023.
37. J. Zhang, H. Liu, W. Bai, X. Li, *A hybrid approach of wavelet transform, ARIMA and LSTM model for the share price index futures forecasting*, *The North American Journal of Economics and Finance*, vol. 69, p. 102022, 2024.
38. Hamayel, Mohammad J and Owda, Amani Yousef, *A novel cryptocurrency price prediction model using GRU, LSTM and bi-LSTM machine learning algorithms*, *MDPI*, vol. 2, no. 4, pp. 477–496, 2021.
39. Li, Yan and Dai, Wei, *Bitcoin price forecasting method based on CNN-LSTM hybrid neural network model*, *The journal of engineering*, vol. 2020, no. 13, p. 344–347, 2020.
40. Khoudi, Zakaria, Mourad Nachaoui, and Soufiane Lyaqini., *Identifying the contextual factors related to the reading performance of Moroccan fourth-grade students from a Machine Learning-based Approach.*, *Education and Information Technologies*, 29.3 (2024): 3047-3073.
41. Fauzi Al Giffary, Novan and Sulianta, Feri, *Prediction Of Cryptocurrency Prices Using LSTM, SVM And Polynomial Regression*, *arXiv e-prints*, 2024.
42. Fleischer, Jacques Phillipe and von Laszewski, Gregor and Theran, Carlos and Parra Bautista, Yohn Jairo, *Time series analysis of cryptocurrency prices using long short-term memory*, *MDPI*, vol. 15, no. 7, p. 230, 2022.
43. Lyaqini, S., and M. Nachaoui, *Diabetes prediction using an improved machine learning approach*, *Math. Model. Comput*, 8.1 (2021): 726-735.
44. Lyaqini, S., Hadri, A., Ellahyani, A., and Nachaoui, M., *Primal-dual algorithm for solving the nonsmooth Twin SVM.*, *Engineering Applications of Artificial Intelligence*, 128 (2024): 107567.