Quantitative Finance and Economics

QFE, 8(1): 153–209.
DOI: 10.3934/QFE.2024007
Received: 26 November 2023
Revised: 13 March 2024
Accepted: 21 March 2024
Published: 26 March 2024

https://www.aimspress.com/journal/QFE

Research article

Managing extreme cryptocurrency volatility in algorithmic trading: EGARCH via genetic algorithms and neural networks

David Alaminos[1,*], M. Belén Salas[2,3] and Ángela M. Callejón-Gil[2,3]

[1] Department of Business, University of Barcelona, Barcelona, Spain
[2] Department of Finance and Accounting, University of Málaga, Málaga, Spain
[3] Cátedra de Economía y Finanzas Sostenibles, University of Málaga, Málaga, Spain

* **Correspondence:** Email: alaminos@ub.edu.

Abstract: The blockchain ecosystem has seen a huge growth since 2009, with the introduction of Bitcoin, driven by conceptual and algorithmic innovations, along with the emergence of numerous new cryptocurrencies. While significant attention has been devoted to established cryptocurrencies like Bitcoin and Ethereum, the continuous introduction of new tokens requires a nuanced examination. In this article, we contribute a comparative analysis encompassing deep learning and quantum methods within neural networks and genetic algorithms, incorporating the innovative integration of EGARCH (Exponential Generalized Autoregressive Conditional Heteroscedasticity) into these methodologies. In this study, we evaluated how well Neural Networks and Genetic Algorithms predict "buy" or "sell" decisions for different cryptocurrencies, using F1 score, Precision, and Recall as key metrics. Our findings underscored the Adaptive Genetic Algorithm with Fuzzy Logic as the most accurate and precise within genetic algorithms. Furthermore, neural network methods, particularly the Quantum Neural Network, demonstrated noteworthy accuracy. Importantly, the X2Y2 cryptocurrency consistently attained the highest accuracy levels in both methodologies, emphasizing its predictive strength. Beyond aiding in the selection of optimal trading methodologies, we introduced the potential of EGARCH integration to enhance predictive capabilities, offering valuable insights for reducing risks associated with investing in nascent cryptocurrencies amidst limited historical market data. This research provides insights for investors, regulators, and developers in the cryptocurrency market. Investors can utilize accurate predictions to optimize investment decisions, regulators may consider

implementing guidelines to ensure fairness, and developers play a pivotal role in refining neural network models for enhanced analysis.

Keywords: emerging cryptocurrencies; EGARCH; genetic algorithms; neural networks; algorithmic trading; quantum computing; deep learning.

JEL Codes: C63, C45, G15, G12

1. Introduction

In recent times, financial institutions have increasingly included cryptocurrencies in their investment portfolios. Cryptocurrencies have garnered significant attention since their inception in 2009, emerging as a significant player in the global financial market (Paule-Vianez et al., 2020; Yang et al., 2020). One noteworthy aspect of cryptocurrencies is their exclusion of intermediaries from financial institutions, leading to a reduction in transaction costs. Consequently, unlike other financial assets, cryptocurrencies operate without a higher authority. Additionally, cryptocurrencies' values are secured by an algorithm that enables the tracking of all transactions (Abakah et al., 2020). The primary advantages of cryptocurrencies include their transparency and 24/7 availability due to the decentralized nature of the cryptocurrency market. Besides, cryptocurrency transactions do not occur in a physical location; instead, they are recorded on a public, open ledger known as the blockchain (Grobys and Sapkota, 2020; Vo and Yost-Bremm, 2020). Consequently, many hedge funds and asset managers have started incorporating cryptocurrency-related assets into their investment portfolios and strategies. This integration has led to a notable surge in interest and activity in both cryptocurrencies and cryptocurrency trading (Fang et al., 2022).

Cryptocurrency markets, regarded as part of the space of alternative investments, are garnering attention given recent technological advances. This growing investor interest makes cryptocurrency markets an important asset class for traders and researchers alike (Akyildirim et al., 2021). The analysis of cryptocurrency market predictions emerges as a valuable tool for investors, as it can enhance the synchronisation of their investment decisions, mitigate risks, and adjust portfolios to reduce potential losses. Moreover, these models may attract more investors to cryptocurrency markets, boosting liquidity and potentially contributing to market efficiency by reducing information asymmetry and facilitating faster price adjustments. Increasing confidence in automated trading strategies based on neural network predictions could drive trading activity, underscoring the complexity and importance of studying cryptocurrency prediction accuracy in terms of performance, market efficiency, risk management, and market dynamics.

In addition, it is interesting to comprehend the liquidity interconnectivity within the cryptocurrency market and to scrutinize the disparities between emerging cryptocurrencies and established ones, like Bitcoin and Ethereum, concerning liquidity, trading volume, and supply. Cryptocurrencies have demonstrated significant potential, leading to an increase in trading volumes in cryptocurrency markets and indicating a substantial improvement in liquidity levels. According to Hasan et al. (2022), liquidity interconnectivity is more noticeable in the short-term timeframe

compared to the medium and long-term perspectives. Trimborn et al. (2020) investigate the potential benefits of incorporating cryptocurrencies into optimised risk portfolio holdings, considering them as promising investment assets due to their rapid growth. On the contrary, these digital currencies often display higher volatility, possess long tails in their distribution patterns, and exhibit relatively limited liquidity, creating a challenging investment landscape. Their investigations indicate that Bitcoin (BTC), being the earliest and most prevalent cryptocurrency, is given zero importance when liquidity constraints are not a factor in both risk and volatility assessments. Consequently, BTC might not be the most appealing cryptocurrency when considering risk-return optimization. This underscores the significance of incorporating cryptocurrencies other than BTC in constructing a portfolio.

In the dynamic landscape of recent years, a cohort of emerging cryptocurrencies is making significant strides for diverse reasons. Decentraland stands out as a thriving virtual reality platform on the Ethereum blockchain, catering to content creators and businesses through its MANA and LAND tokens (Guidi and Michienzi, 2022). LuckyBlock has swiftly become a notable player, achieving a remarkable US$1 billion market capitalization and pioneering transparency and fairness in gaming, attracting a substantial investor base (Mirkamol and Mansur, 2023). SafeMoon is gaining traction in the DeFi space, incorporating reflection, LP acquisition, and burn functions while ambitiously aiming to establish an NFT exchange coupled with charity initiatives (Hassan et al., 2022). Avalanche, positioned as a formidable competitor to Ethereum, offers heightened transaction output through its three blockchains—X-Chain, C-Chain, and P-Chain (Makarov and Schoar, 2022). SeeSaw prioritizes security with its blockchain implementation, ensuring technical robustness and resilience to tampering or hacking (Murray-Rust et al., 2023). King Cardano introduces itself as the first auto-claim ADA token, boasting high yields and contributing to the open market by buying back 3% of each transaction (King and Koutmos, 2021). Binamon focuses on community growth, offering a gaming metaverse with digital monsters across multiple games for users to earn tokens and passive income. Kasta facilitates instant and free cryptocurrency transfers via QR codes, with the KASTA Token playing a pivotal role in driving the adoption of cryptocurrencies (King and Koutmos, 2021). Last, X2Y2 emerges as a decentralized NFT marketplace, connecting crypto investors to various wallets and addressing challenges posed by the established NFT trading platform, OpenSea, since its launch in February 2022 (Makridis et al., 2023). After this analysis about the different strengths and utilities, we have chosen these cryptocurrencies for our study.

Since the inception of Bitcoin in 2009, numerous cryptocurrencies have come into existence, making them a regular feature of financial newsletters. Such currencies have attracted interest from both financial institutions and central banks due to their novel technology drivers and their ability to perturb the prevailing financial institutions' frameworks (Nica et al., 2022). The growing popularity and acceptance of new emerging cryptocurrency markets demand dedicated strategies for maximizing the investment return potential through "learning" algorithms (Petukhina et al., 2021). Furthermore, given the aforementioned characteristics of cryptocurrencies and, above all, their speculative nature and volatility, the cryptocurrency market offers the opportunity to study the dynamics of algorithms from the point of view of human psychology (Vo and Yost-Bremm, 2020; Fang et al., 2022).

The majority of traders and analysts prefer employing programmatic trading techniques in the cryptocurrency markets. Technological advancements have revolutionized the way investors engage in financial markets. Algorithmic trading (AT), which involves "the use of computer algorithms to

automatically make certain trading decisions, submit orders and manage them after submission," exemplifies these technological shifts (Frino et al., 2020). The impact of AT on market quality is deemed significant by researchers, industry professionals, and policymakers. According to Hendershott et al. (2011), AT reduces trading costs and enhances quote information. Moreover, the revenues of liquidity providers also see an increase with AT, although this effect appears to be transient. AT is predominantly active over brief time intervals, spanning from minutes to hours or days. As commercial timeframes contract even more, dwindling to levels of a minute, second, or millisecond, the trading evolves into a form termed high-frequency trading (HFT). Within HFT, algorithms take on the exclusive role of managing the negotiation process (Vo and Yost-Bremm, 2020). In essence, financial trading requires AT to systematically assess the environment for appropriate and swift decisions, especially when continuous data monitoring is not in place (Aloud and Alkhamees, 2021).

Researchers have extensively studied algorithmic trading, encompassing algorithms guided by both fundamental and technical indicator analyses, as well as those reinforced by machine learning techniques. In the realm of financial applications, Machine Learning (ML) is assuming a progressively vital role, according to Goldblum et al. (2021). Their assertion extends to Deep Learning (DL), a subset of ML techniques honing in on deep neural networks. DL algorithms are advancing, exhibiting the capacity to train on intricate datasets and forecast outcomes. The present landscape sees a surge in active investment by various financial entities, including hedge funds, investment banks, retailers, and contemporary FinTech providers, as they strive to develop expertise in data science and ML (Goodell et al., 2021). Indeed, finance professionals are increasingly trusting automated ML systems for AT, such as fraud recognition systems that detect illegal trades (Abdallah et al., 2016; Bhattacharyya et al., 2011; Milana and Ashta, 2021), risk systems that pass/fail loans (Binns, 2018), in HFT applications that take time-scale decisions that humans are not able to verify (Hendershott et al., 2011; Arévalo et al., 2016; Klausand Elzweig , 2017; Borovkova and Tsiamas, 2019; Egger et al., 2020), and integration with other emerging technologies including cryptocurrencies and blockchain (Milana and Ashta, 2021).

The most common DL technologies used in cryptocurrency trading have been Convolutional neural networks (CNN), Gated Recurrent Units (GRU), Multiplayer perceptron (MLP), and Long short-term memory (LSTM). Many advanced deep neural network models have been implemented by researchers in cryptocurrency trading. Recent research highlights the success of models utilizing these architectures for modelling and predicting financial time series, which includes cryptocurrencies (Fang et al., 2022). Livieris et al. (2020) presented a model, CNN-LSTM, specifically designed for accurately predicting gold prices. Similarly, Lu et al. (2020) suggested a method based on CNN-LSTM for predicting stock prices.

On a different note, the genetic algorithm (GA) is a widely recognized metaheuristic algorithm that draws inspiration from the biological development process. It operates on the principles of chromosome representation and fitness selection. Significant benefits of genetic algorithms are that they are efficient in handling non-stationary data and that there is no need to adopt a particular distribution of data (Huang et al., 2019; Katoch et al., 2021). According to Drachal and Pawłowski 2021), there are research studies that use GA from a theoretical and practical perspective, but the latter is mostly used in engineering sciences, and very little in economics or finance. Besides, studies based on finance focus on optimization questions and do not deal with time series forecasting. Therefore, possible applications of GA could be in finance, in particular bankruptcy prediction, risk management,

and financial forecast (Sezer et al., 2020; Bustos and Pomares-Quimbaya, 2020). Such authors as (Alameer et al., 2019; Weng et al., 2018), applied GA to obtain the estimation of the variables of the model like the Adaptive Neuro-Fuzzy Inference System (ANFIS). Besides, GA has proven effective in predicting commodity prices with models as support vector machines (SVM). Although GAs have many promising applications, they could be enhanced because GAs remain a huge area for further development and can be improved.

To address the void in this research domain, we compare DL and quantum methods in neural networks and genetic algorithms techniques applied to the algorithmic trading of the youngest cryptocurrencies, which we have considered: Decentraland, LuckyBlock, SafeMoon, Avalanche, SeeSaw Protocol, Binamon, Kasta, and X2Y2. Blockchains are one of the most significant emerging technologies and, according to Saad et al. (2019), will play a central role in the future configuration of our society. The market capitalization of cryptocurrencies constitutes the main feature representing the importance of the market. Since the introduction of Bitcoin in 2009, there has been huge growth in the blockchain ecosystem, led by conceptual and algorithmic innovations and the emergence of a great number of new cryptocurrencies. A key attribute of the cryptocurrency market is its notable volatility, a trait that escalates with the emergence of new cryptocurrencies. As some new cryptocurrencies generate millionaires for crypto investors, while others determine massive losses for crypto portfolios, an analysis of emerging cryptocurrencies is necessary. Therefore, we should evaluate which new cryptocurrencies are secure, stable, safe, and non-fluctuating, as some new cryptocurrencies deal with the weaknesses of older ones, with better performance, scalability, and schedulability (Marzo et al., 2022). Thus, in our research, we have chosen as a sample the main cryptocurrencies that have increased their exchange in the market in 2022. These new cryptocurrencies have been on the market for only a few years and hence with little historical data to analyze the effectiveness of the application of computational algorithms with trading strategies. Moreover, they are the major cryptocurrencies that appear on cryptocurrency data sites such as CoinMarketCap, CoinGecko, and Bloomberg Cryptocurrencies Insights. In addition, some recent papers have proposed the analysis of other new cryptocurrencies, which benefits cryptocurrency traders and investors in making investment and trading decisions, as most have analyzed Bitcoin and Ethereum (Nica et al., 2022; Petukhina et al., 2021).

We expect to make at least four further contributions to the literature. First, most existing research studies have analyzed the largest cryptocurrency Bitcoin; however, several authors have proposed the analysis of other cryptocurrencies as future lines of research. Thus, (Vo and Yost-Bremm, 2020) conclude that future studies ought to widen the scope of the analysis to include other cryptocurrencies to check whether the High Frequency Trading strategy is applicable across all cryptocurrencies. Nakano et al. (2018) and Corbet et al. (2020) conclude that building a multi-asset portfolio of various cryptocurrencies would appear to be worthy and not focus only on bitcoin. In addition, Gerritsen et al. (2020) suggest that additional future investigation should target other leading new cryptocurrencies, which comes at the benefit of investors and operators in cryptocurrencies to make informed investment and trading decisions. Moreover, Gradojevic et al. (2023) state that, in forthcoming research, it would be valuable to expand the scope to encompass alternative cryptocurrencies and include a broader range of data frequencies, provided high-frequency data is available. Finally, Adcock and Gradojevic (2019), Akyildirim et al. (2021) and Ren et al. (2022) suggest that other cryptocurrencies could also be studied, since, new cryptocurrencies given that new cryptocurrencies are often overlooked in the existing body

of literature. Therefore, our study responds to this call by analyzing the eight new emerging cryptocurrencies markets of 2022.

Second, we compare innovative techniques of ML, not used in the emerging cryptocurrency market until now. Aloud and Alkhamees (2021) suggest as a future direction for investigation to evaluate the application of Reinforcement Learning algorithmic trading to various emerging markets; for instance, the currency market and cryptocurrencies. For their part, Goodell et al. (2021) conclude that the rise of Artificial Intelligence (AI) and ML has led to future research opportunities to build innovative FinTech business models supported by technology for finance. Examples of core FinTech innovations include blockchain and cryptocurrencies, digital trading, and advisory systems. Besides, further research is needed to comprehend the many faces of FinTech and the implications of AI and ML in the emerging field. Jia et al. (2019) conclude that, although deep learning has demonstrated excellent performance on many signal-processing problems, the association of deep learning and reinforcement learning in financial quantitative trading remains to be studied more. Also, Maghsoodi (2023) recommends for future developments in cryptocurrency markets the application and comparison of ML and DL models like Long Short-Term Memory, Deep Convolutional Networks, Adaptive-Network-based Fuzzy Inference System, and Artificial Neural Networks. Last, Corbet et al. (2020) propose as future research extended technical trading analyses applying more complicated technical trading rules and comparing the results.

In the third step, we introduce additional non-financial indicators like volatility (Bollinger Band), market strength (Klinger Oscillator), and trend (Trend Elimination Oscillator), proposed by Vo and Yost-Bremm (2020) as a potential area for future investigation. Gradojevic et al. (2023) also suggest exploring a broader range of technical indicators and covering different data frequencies, assuming the availability of high-frequency data. Demir et al. (2019) showcase how seamlessly integrating technical indicators into Day-ahead electricity market predictions significantly improves the predictive accuracy of machine learning models. They highlight the enhanced explanatory capacity provided by non-financial indicators, emphasizing that technical analysis, as an analytical approach, focuses on examining statistical trends in historical market data to predict upcoming price movements. Technical indicators, a tool in technical analysis, analyze market data using various formulas to identify complex price patterns, indicating situations of excessive buying or selling, deviations from a central trend, or proximity to support and resistance lines. Moreover, they demonstrate that technical indicators can offer early indications of future price movements, thereby boosting the predictive capability of models forecasting financial time series and showcasing the explanatory effectiveness of technical indicators across stock markets. c present evidence that technical indicators provide directional information about a security's price, assisting in predicting security prices and simplifying machine learning. Consequently, they conclude that technical indicators wield substantial explanatory power in stock markets. Alonso-Monsalve et al. (2020) conclude that when analysing the change in the value of cryptocurrencies in relation to the US dollar, the use of convolutional neural networks (CNNs) can be beneficial for predicting market trends at short time intervals. However, their research is based on a limited set of technical indicators. Expanding the scope of these indicators would likely reveal the possibility of improving prediction performance. Therefore, it would be advisable for future research to consider exploring this avenue to achieve more robust results.

Fourth, our study covers data and algorithms that could be replicated in more examples of high frequency trading. According to Akyildirim et al. (2021), the majority of research on cryptocurrency

markets focuses on daily returns, but the analysis at higher frequencies is commonly overlooked and should be considered in future research. Furthermore, Adcock and Gradojevic (2019) propose further research to investigate the profitability of other non-linear high-frequency technical trading in cryptocurrency markets. In their 2021 study, Briola et al. (2021) demonstrate the effective training and application of Deep Reinforcement Learning models within the realm of high-frequency trading. They explore the impact of three distinct state definitions on the out-of-sample performance of agents, discovering that having awareness of the mark-to-market profitability of their present position significantly enhances both the overall profit and loss equation and the reward for an individual trade. Alonso-Monsalve et al. (2020) suggest that future studies could explore incorporating higher frequency data into their analyses. They argue that by increasing the number of features used at high return frequencies, it may be possible to develop models with improved predictive accuracy, particularly at shorter time intervals. We evaluate the performance of Neural Networks and Genetic Algorithms in predicting "buy" or "sell" decisions for various cryptocurrencies. Three key metrics—F1 score, Precision, and Recall—are used to measure accuracy. The study encompasses multiple classifiers across different cryptocurrencies, with micro-means used to consolidate results.

The subsequent sections of this paper are organized as follows. Section 2 furnishes a review of the existing literature. Section 3 outlines the methodology employed in the study. The indicators and data utilized in the research are elaborated in Section 4. Section 5 highlights the obtained results and findings. Section 6 shows the discussion of results. Finally, Section 7 concludes by elucidating the reached conclusions.

2. Literature review

The literature on cryptocurrencies has garnered significant attention in recent times and has been the subject of a variety of research. Some studies have addressed issues concerning their prices, like substantial price volatility (Katsiampa et al., 2019: Bianchi et al., 2022), pricing efficiency (Sensoy, 2018; Corbet and Katsiampa, 2018; Mensi et al., 2019), price determination, and predictability (Bouri et al., 2019; Panagiotidis et al., 2018; Giudici and Abu-Hashish, 2019). Panagiotidis et al. (2018) analyze 21 variables that impact bitcoins returns, concluding that the most crucial factors are the intensity of Google searches, gold yields, and political uncertainty. They state that European economic policy uncertainty, the NIKKEI index, and negative feedback from the Google trend appear to be drivers of Bitcoin. Bianchi et al. (2022) illustrate that, as far as the gains from the short-term investment strategy represent the returns from offering liquidity, their findings indicate that market makers generally achieve greater expected returns when there is heightened concern about adverse selection on both sides of the trade. Furthermore, they reveal that the diminished market liquidity observed during periods of increased volatility appears to be, at least in part, due to the elevated risk premiums necessary for offering liquidity.

Other authors have investigated the factors determining cryptocurrency market liquidity. For example, Westland (2023) examined five hypotheses regarding liquidity and the factors that influence it, using a selection of cryptocurrencies that make up approximately 90% of trading volume and market capitalization, making the findings broadly applicable. The research findings are derived from the analysis of a concise yet convincing group of cryptocurrencies. These findings are supported by calculated statistics such as liquidity and inferred transaction fees, which were derived from extensive

datasets. This author concluded that price serves as an effective predictor of liquidity, while volume can act as a surrogate for liquidity, although there was some contradictory evidence regarding this hypothesis. The study did not find supporting evidence for the notion that interchangeability between cryptocurrencies predicts liquidity. Trimborn et al. (2020) delve into the advantages of integrating cryptocurrencies into risk-optimized portfolios, considering the inherent liquidity constraints of the cryptocurrency market. They propose the Liquidity Bounded Risk-return Optimization method, which introduces an additional liquidity constraint based on the expected investment amount. Application of this approach to monthly and weekly adjusted portfolios, encompassing constituent stocks from the S&P 100, the Barclays Capital US Aggregate Index (US-Bonds Index), and the S&P GSCI (Commodities Index), alongside cryptocurrencies, yields significant improvements in the relationship between volatility/risk and return when employing both Markowitz and CVaR models. Bianchi et al. (2022) highlight that limited liquidity supply, as evidenced by heightened idiosyncratic volatility and broader spreads in the Treasury-EuroDollar rate (TED) and bid-ask spreads, corresponds to increased anticipated returns on investment strategies. They conclude that liquidity providers tend to focus on demanding higher expected returns and risk premiums for lower-quality assets. Hasan et al. (2022) examine liquidity connectivity dynamics in the cryptocurrency market using six major cryptocurrencies: Bitcoin (BTC), Litecoin (LTC), Ethereum (ETH), Ripple (XRP), Monero (XMR), and Dash. Their analysis reveals a moderate level of liquidity connectivity among these cryptocurrencies, with BTC and LTC exerting notable influence. The authors determine that liquidity connectivity is more prominent in the short-term compared to the medium and long-term perspectives.

Various studies explore the influence of specific factors on the Bitcoin exchange rate and returns. Demir et al. (2018) delve into the impact of economic policy uncertainty, while Zhu et al. (2017) focus on economic indicators such as the consumer price index and the US dollar index. Other drivers examined include market sentiment and Bitcoin popularity, as investigated by Makrichoriti and Moratis (2016), as well as Polasik et al. (2015). Additionally, researchers like Feng et al. (2018), Bouri et al. (2019 and 2019), Vidal-Tomas et al. (2018), and Mensi et al. (2019) study broader market behavior. Atsalakis et al. (2019) take a different approach by exploring the predictive and classification ability of Bitcoin. They propose the use of a hybrid neuro-fuzzy controller system named PATSOS, consisting of two sub-systems of the Adaptive Neuro-Fuzzy Inference System (ANFIS). Their findings demonstrate that the neuro-fuzzy PATSOS model outperforms both the ANFIS model and the Artificial Neural Networks model, offering higher returns for potential investors. This conclusion is supported through validation involving three alternative cryptocurrencies, specifically Ethereum, Litecoin, and Ripple.

Most studies on the cryptocurrency market have been based primarily on Bitcoin (Kristoufek, 2015; Panagiotidis et al., 2018; Adcock and Gradojevic, 2019; Atsalakis et al., 2019; Huang et al., 2019; Gerritsen et al., 2020; Gradojevic et al., 2023). Kristoufek (2015) examines the Bitcoin price formation and this author observes that conventional fundamental factors, including usage in trade, money supply, and price level, exert an influence on the long-term price of Bitcoin. In addition, their results show that the prices of bitcoin follow investor interest in the cryptocurrency, and bitcoin no longer represents a safe investment. Some studies reported a decline in trust in the integrity of the cryptocurrency market and revealed the existence of possible crime and fraud in this system (Gandal et al., 2018; Griffin and Shams, 2018; Corbet et al., 2020).

Despite the wealth of literature on Bitcoin, other cryptocurrencies have received less attention. Akyildirim et al. (2021) conduct an in-depth analysis of the twelve most liquid cryptocurrencies, including Bitcoin Cash (BCH), Bitcoin (BTC), Dash (DSH), EOS (EOS), Ethereum Classic (ETC), Ethereum (ETH), Iota (IOT), Litecoin (LTC), OmiseGO (OMG), Monero (XMR), Ripple (XRP), and Zcash (ZEC). Their findings reveal consistent predictability in the expected return direction for cryptocurrency markets at daily or minute time scales, achieving classification accuracies of up to a 69% success rate. Bouri et al. (2020) explore various cryptocurrencies and uncover a connection wherein explosive prices in one cryptocurrency drive similar movements in another. Bouri et al. (2019) investigate the role of trading volume in forecasting yields and volatility across different cryptocurrencies, finding that trading volume is valuable for predicting extreme returns but less so for forecasting future volatility in a smaller set of cryptocurrencies. In a recent study, Mensi et al. (2019) examine the intraday efficiency of Bitcoin and Ethereum, noting Bitcoin's greater inefficiency during general uptrends and downtrends. Maghsoodi (2023) addresses the cryptocurrency allocation problem and real multi-scenario trading experience using a hybrid approach that combines the Prophetic Prediction Model (PFM) and cluster analysis. We demonstrate the effectiveness of integrating PFM and CLUS-MCDA approaches in resolving big data problems with multiple scenarios simultaneously. Hasan et al. (2022) delve into liquidity dynamics interactions within the cryptocurrency market, analyzing a dataset encompassing six major cryptocurrencies: Bitcoin (BTC), Litecoin (LTC), Ethereum (ETH), Ripple (XRP), Monero (XMR), and Dash. Their static analysis reveals moderate interdependence in liquidity among cryptocurrencies, with BTC and LTC exerting significant influence on overall interconnectedness. They identify distinct liquidity clusters, comprising BTC, LTC, and XRP, and another cluster with ETH, XMR, and Dash. In their frequency domain examination, they find higher liquidity interconnections in the shorter term compared to the medium and long term, with BTC, LTC, and XRP being major contributors to short-term fluctuations, while ETH assumes this role in the longer term.

Another strand of literature concerns the methodology applied in the cryptocurrency market. Some studies employ Artificial Neural Networks (ANN) with technical analysis (Gerritsen et al., 2020; Adcock and Gradojevic, 2019; Huang et al., 2019; Corbet et al., 2020; Nakano et al., 2018). Corbet et al. (2020) examine diverse dimensions of Moving Average (MV) strategies in cryptocurrencies. They sampled this technique to high-frequency Bitcoin returns, at the one-minute frequency. They conclude that some technical rules may show signs of predictive power; nevertheless, more consideration needs to be given to questions like market liquidity and transaction costs. Nakano et al. (2018) assess the effectiveness of a deep learning artificial neural network (D-ANN) model incorporating technical indicators during the period from 2016 to 2018. Their Bitcoin predictions were evaluated over a 15-minute horizon, employing a trading strategy that utilized a more extensive set of technical indicators compared to the study by Corbet et al. (2020). They found that certain technical trading strategies could yield a positive excess return in comparison to the buy-and-hold strategy, even when factoring in reasonable and realistic trading expenses. Adcock and Gradojevic (2019) employ an ANN model with technical indicators, generating daily horizon predictions across the data range from 2011 to 2018. They establish the accuracy of their forecasts for the BTC/USD exchange rate, both in point forecast and density terms. Gerritsen et al. (2020) leverage daily price data spanning from July 2010 to January 2019, demonstrating that specific technical analysis trading rules, particularly the trading range breakout, possess meaningful predictive capabilities

for Bitcoin prices. This results in significant excess returns, providing valuable insights for Bitcoin traders and investment decisions.

Numerous studies dedicated to predicting and trading cryptocurrencies leverage deep learning methodologies. Atsalakis et al. (2019) introduce a neuro-fuzzy trading system incorporating lagged Bitcoin prices as model inputs, surpassing simpler neuro-fuzzy and artificial neural network (ANN) models in performance. In a trading simulation, signals from their model outperform a naïve buy-and-hold strategy by 71.21% in terms of investment yields. Adcock and Gradojevic (2019) explore the predictability of twelve cryptocurrencies at daily and minute frequencies using machine learning classification algorithms such as support vector machines, logistic regression, artificial neural networks, and random forests. On average, these algorithms achieve an accuracy ranging from 55% to 65% at daily or minute frequencies, with support vector machines proving most effective. Gradojevic et al. (2023) evaluate the predictability of BTC/USD exchange rate returns at hourly and daily forecast horizons, employing Support Vector Machine and Random Forest methods, with the Random Forest approach performing exceptionally well. Similarly, Madan et al. (2015) advocate for binomial regressions, support vector machines, and random forest algorithms to forecast random signs of Bitcoin price movement. Lahmiri and Bekiros (2019) utilize deep learning techniques to showcase their effectiveness in forecasting daily returns for Bitcoin, Digital Cash, and Ripple, revealing that Long-short term memory neural network topologies (LSTM) significantly outperform generalized regression neural architecture. Finally, through machine learning optimization, Greaves and Au (2015) achieve a classification accuracy of approximately 55% for predicting upward and downward movements in Bitcoin prices.

In summary, we can conclude that no previous literature has approached the analysis of the cryptocurrency market by making a comparison between deep learning and quantum methods in neural networks and genetic algorithms used in the algorithmic trading of emerging cryptocurrencies. We, therefore, aim to cover this lack of literature.

3. Methodology

As previously indicated, our analysis of emerging cryptocurrencies trading employs a variety of methods, aiming to establish a resilient model. This model undergoes testing not only through a single categorization technique but also through those proven successful in prior literature and diverse domains. Specifically, our research incorporates a diverse set of neural network approaches, such as Convolutional Neural Networks-Long Short Term Memory with EGARCH Integration, Gated Recurrent Unit-Convolutional Neural Networks with EGARCH Integration, Quantum Neural Networks with EGARCH Integration, Deep Recurrent Convolutional Neural Networks with EGARCH Integration, and Quantum Recurrent Neural Networks with EGARCH Integration. Additionally, our methodology includes Genetic Algorithms, featuring Adaptive Boosting and Genetic Algorithm with EGARCH Integration, Adaptive Neuro-Fuzzy Inference System-Quantum Genetic Algorithm with EGARCH Integration, Adaptive Genetic Algorithm with Fuzzy Logic with EGARCH Integration, Quantum Genetic Algorithm with EGARCH Integration, and Support Vector Machine-Genetic Algorithm with EGARCH Integration. The subsequent sections provide a succinct overview of the procedural aspects inherent in each of these classification techniques.

3.1. Neural networks

3.1.1. Convolutional Neural Networks-Long Short-Term Memory (CNN-LSTM) with EGARCH Integration

Utilizing the capabilities of both CNN and LSTM, a forecasting model for value is crafted based on CNN-LSTM. CNN, originally proposed by LeCun et al. (1998), functions as a feed-forward neural network and has exhibited proficiency in image and natural language processing (Kim and Kim, 2019). Its success extends to time-series forecasting, where local sensing and weight distribution help efficiently reduce the parameter count, thereby enhancing the learning efficiency of the model (Qin et al., 2018). Comprising two key components, namely the convolution layer and the clustering layer, CNN houses multiple convolution kernels, with its calculation formula provided in Equation (1). Following the convolution operation, features are extracted from the data, leading to high-dimensional separated characteristics. To address this issue and mitigate the training cost of the network, a clustering layer is introduced directly after the convolution layer to reduce the dimensionality of the features. The CNN layer typically consists of a convolutional layer with multiple convolution kernels, as defined by Equation (1):

$$l_t = tnh\ (x_t * k_t + b_t) \tag{1}$$

being l_t the output value after convolution, t_{nh} is the activation function, x_t represents the input vector, k_t stands for the weight of the convolution kernel, and b_t means the bias of the convolution kernel.

Introduced by Hochreiter and Schmidhuber in 1997, LSTM was specifically designed to tackle the persistent challenges of exploding and vanishing gradients in Recurrent Neural Networks (RNNs), as highlighted by Ta et al. (2020). Extensively employed in various tasks like speech detection, sentiment analysis, and text processing, LSTM stands out for its distinctive memory capabilities, enabling it to make relatively accurate predictions (Gupta and Jalal, 2020). The LSTM architecture consists of three essential components: the forgetting gate, the input gate, and the output gate. The computation of the LSTM model unfolds through the following steps:

1. The output at the previous time step and the input at the current time step are processed by the forgetting gate, producing an output according to Equation (2):

$$f_t = \sigma\left(W_f.\left[h_{t-1},\ x_t\right] + b_f\right) \tag{2}$$

being the value range of f_t (0,1), W_f represents the weight of the forget gate, and b_f symbols the bias of the forget gate, x_t stands for the input current time value, and h_{t-1} means the last output value.

2. The output at the previous time step and the current input are processed by the input gate, resulting in the output and the input gate's candidate status, as defined by Equations (3) and (4):

$$i_t = \sigma(W_i.\left[h_{t-1}, x_t\right] + b_i) \tag{3}$$

$$\tilde{C_t} = tanh(W_c.\left[h_{t-1}, x_t\right] + b_c) \tag{4}$$

where the value range of i_t is (0,1), W_i is the weight of the input gate, b_i represents the bias of the

input gate, W_c represents the weight of the candidate input gate, and b_c symbols the bias of the candidate input gate.

3. The current cell state is updated as follows:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5}$$

being the value range of C_t (0,1).

4. At time t, the values of the output gate are determined by taking the output h_{t-1} and the input, x_t as it input. The resulting output o_t is then established by the output gate:

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{6}$$

where the value range of o_t is (0,1), W_o represents the weight of the output gate, and b_o symbols the bias of the output gate.

5. The ultimate computation of the LSTM output involves the consideration of both the output from the output gate and the cell state, as specified in Equation (7):

$$h_t=o_t * tanh(C_t) \tag{7}$$

Integrating EGARCH into CNN-LSTM enhances financial modeling by combining EGARCH's volatility modeling with CNN-LSTM's dynamic relationship capturing capabilities, promising improved accuracy in forecasting volatility and risk. EGARCH, a model for capturing conditional volatility, is integrated into the CNN-LSTM model as an additional feature. The EGARCH equation, which captures the conditional variance (ht), is as follows:

$$h_t = \omega + \alpha|r_{t-1}| + \beta h_{t-1} \tag{8}$$

where h_t represents the conditional variance at time t, ω is the constant term, α and β are parameters controlling the impact of past returns (r_{t-1}) and past conditional variances (h_{t-1}). The use of the absolute value (r_{t-1}) reflects the asymmetric nature of EGARCH, capturing how positive and negative returns affect volatility differently.

The output from the EGARCH model, which is the conditional variance (ht), can be combined with the LSTM output. This is typically achieved by treating (ht) as an additional feature and appending it to the LSTM output.

$$\text{CombinedOutputt=[LSTMOutputt,ht]} \tag{9}$$

where CombinedOutputt is the output at time t that incorporates both the LSTM output and the EGARCH conditional variance (ht). This approach treats ht as an additional feature and appends it to the LSTM output.

The combined output from the EGARCH and LSTM models is run via a completely linked level, and the model's output is compared to actual values to calculate the prediction error. The CNN-LSTM-EGARCH model provides a comprehensive approach for financial time series forecasting, integrating feature extraction, sequential modeling, and conditional volatility capture for improved accuracy in predictions. After combining EGARCH and LSTM outputs, the combined output run via a completely linked level The model's output is then compared to actual values to calculate the prediction error.

$$\text{ModelOutputt=f(CombinedOutputt*W+b)} \tag{10}$$

being ModelOutputt represents the model's output at time t, CombinedOutputt is the combined output from the EGARCH and LSTM models, W signifies the weight matrix in the completely linked level, b denotes the bias vector in the completely linked level, and f is the activation function, typically used to introduce non-linearity.

$$\text{PredictionErrort} = \left| \text{ActualValuet} - \text{ModelOutputt} \right| \tag{11}$$

where PredictionErrort measures the absolute difference between the actual value at time t and the model's output at the same time. This error quantifies the accuracy of the model's predictions.

3.1.2. Gated Recurrent Unit- Convolutional Neural Networks (GRU-CNN) with EGARCH Integration

In this method, we aim to develop a powerful forecasting model by combining EGARCH, a model for conditional volatility, with the hybrid neural network framework that unifies the Gated Recurrent Unit (GRU) and Convolutional Neural Network (CNN) modules. This integrated approach allows us to capture both temporal dependencies and spatial features in load data for more accurate predictions. RNN utilizes hidden layers to retain information from earlier time points, shaping the output based on both current states and memories from previous instances. The structure of the unrolled RNN is illustrated as follows:

$$\begin{aligned} a^{\langle t \rangle} &= g_1\left(\omega_{aa} a^{\langle t-1 \rangle} + \omega_{ax} x^{\langle t-1 \rangle} + b_a\right) \\ \hat{y}^{\langle t \rangle} &= g_2\left(\omega_{ay} a^{\langle t \rangle} + b_y\right) \end{aligned} \tag{12}$$

let $a^{\langle t \rangle}$ represent the output of a single hidden layer at time t. The weight matrices for the hidden layer, input, and output are denoted as $\omega_{aa}^{\langle t \rangle}$, $\omega_{ax}^{\langle t \rangle}$, and $\omega_{ay}^{\langle t \rangle}$ a respectively. Additionally, b_a and b_y represent the bias vectors for the single hidden layer and the output, respectively. The symbols g_1 and g_2 signify the non-linear activation functions.

The GRU presents a streamlined RNN structure, serving as a variation of the LSTM. Unlike the LSTM, the GRU comprises two gates – the update gate and reset gate – whereas the LSTM integrates three gates, encompassing the forget gate, input gate, and output gate (Gao et al., 2019). The GRU equations are:

$$\begin{aligned} \Gamma_u &= \sigma\left(\omega_u\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right), \\ \Gamma_r &= \sigma\left(\omega_r\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_r\right) \\ \tilde{c}^{\langle t \rangle} &= tanh\left(\omega_c\left[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right), \\ c^{\langle t \rangle} &= (1 - \Gamma_u) * c^{\langle t-1 \rangle} + \Gamma_u * \tilde{c}^{\langle t \rangle}, \end{aligned} \tag{13}$$

being ω_u, ω_r, and ω_c the training weight matrix of the update gate, the reset gate, and the candidate activation $\tilde{c}^{\langle t \rangle}$, respectively and b_u, b_r, and b_c stand for the bias vectors.

This matrix is structured based on the sensors' locations and the chronological sequence of time. The representation of the spatiotemporal matrix is as follows:

$$x = \begin{bmatrix} X_1(1) & \cdots & X_n(n) \\ \vdots & \ddots & \vdots \\ X_k(1) & \cdots & X_k(n) \end{bmatrix} \tag{14}$$

We denote k as the k^{th} smart sensor, n as the n^{th} time sequence, and $X_k(n)$ as the data recorded by the k^{th} a smart sensor at time n. To extract the charging characteristics from the spatiotemporal matrix, CNN was employed to process the information within the matrix. The results from the CNN module are linked to a completely linked level for additional processing. The EGARCH equation for capturing conditional variance (ht) is given as:

$$h_t = \omega + \alpha|r_{t-1}| + \beta h_{t-1} \tag{15}$$

being h_t represents the conditional variance at time t, ω is the constant term, α and β are parameters controlling the impact of past returns (r_{t-1}) and past conditional variances (h_{t-1}). The use of the absolute value (r_{t-1}) reflects the asymmetric nature of EGARCH, capturing how positive and negative returns affect volatility differently.

After combining the outputs of the GRU-CNN model with the EGARCH component, the subsequent step involves processing this combined information through a completely linked level. This is instrumental in the forecasting process and produces the final forecasting outcomes.

$X_{combined}$ represents the combined output from the GRU-CNN and EGARCH integration. The fully connected layer can be represented using a weighted sum and activation function, where W_{FC} represents the weights, b_{FC} represents the biases, and F is the activation function. The output $Y_{forecast}$ is the final load forecasting outcome:

$$Y_{forecast} = F(W_{FC} \cdot X_{combined} + b_{FC}) \tag{16}$$

where $X_{combined}$ is the combined information from the GRU-CNN and EGARCH integration, W_{FC} stands for the weight matrix for the completely linked level, b_{FC} means the bias vector for the completely linked level, and F is the activation function applied to the weighted sum.

This equation signifies the pivotal role of the completely linked level in the forecasting process, where it transforms the combined information into the final forecasting outcomes. The fully connected layer computes the final load forecasting outcomes by taking into account all the information provided by the GRU-CNN model and EGARCH. This includes temporal dependencies, spatial features, and conditional volatility information.

3.1.3. Quantum Neural Networks (QNN) with EGARCH Integration

The fusion of CNNs and quantum computing has the potential to yield a computational approach with robust predictive capabilities (Wan et al., 2017). In quantum computing, a qubit serves as the smallest unit of information, representing a probabilistic state. A qubit can exist in either a "1" or "0" state or any superposition of the two (Gonçalves, 2019). The state of a qubit is defined in Equation (17):

$$|\psi\rangle = \alpha|o\rangle + \beta|1\rangle \tag{17}$$

We denote α and β the numbers of the amplitude of the corresponding states like $|\alpha|^2 + |\beta|^2 = 1$. It is formed as a pair of numbers. $\begin{bmatrix}\alpha\\ \beta\end{bmatrix}$ The angle θ means the specification of geometrical aspects, defined like: $\cos(\theta) = |\alpha| \; and \; \sin(\theta) = |\beta|$ (Alaminos, Esteban and Salas, 2023). Quantum gates can be applied to adjust probabilities through weight enhancement. An illustration of this is presented in formula (18) with the example of a rotation gate:

$$U(\Delta\theta) = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \tag{18}$$

The qubit state can be upgraded by employing the aforementioned quantum gate. The implementation of the spin gate on a qubit is provided below:

$$\begin{bmatrix} \alpha' \\ \beta' \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{19}$$

The process begins with a quantum hidden neuron in the state $|o\rangle$, setting up the superposition as outlined in Formula (20).

$$\sqrt{p}|O\rangle + \sqrt{1-p}|1\rangle \ \ with\ O \leq |p| \leq 1 \tag{20}$$

Here, "p" denotes the random probability of initiating the system in the state$|O\rangle$. The classical neurons are induced through the generation of random numbers, as discussed in the works of Sensoy (2018) and Katsiampa et al. (2019). The output from the quantum neuron is established as per Equation (21).

$$v_j = f\left(\sum_{i=1}^{n} w_{ji} \ * \ x_i\right) \tag{21}$$

Considering "f" as a sigmoid or Gaussian function that depends on the specific problem, the description of the network's output is provided in the subsequent formula:

$$y_k = f\left(\sum_{j=1}^{l} w_{jk} \ * \ v_j\right) \tag{22}$$

We assume the fully connected layer in the QNN processes the combined inputs, which now include ht from EGARCH, to compute the final load forecasting outcomes. We denote the fully connected layer's output as "z_k" (Alaminos et al., 2020). The weighted sum (z_k) of the inputs (v_j) is calculated for each output neuron k in the fully connected layer. This is a linear combination of the inputs, and the weights (w_{jk}) determine the strength of the connections.

$$z_k = \sum_{j=1}^{l} \omega_j k * v_j \tag{23}$$

The weighted sum (z_k) is passed through an activation function (f_k), which is typically a problem-dependent sigmoid or Gaussian function (Alaminos et al., 2020). The choice of activation function depends on the specific problem and design of the QNN.

$$v_k = f_k(z_k) \tag{24}$$

The output from the completely linked level represents the forecasted value for a particular time step or prediction horizon.

$$y_k = v_k \tag{25}$$

To train the QNN, an objective function (E_k) is defined to measure the error between the predicted value (y_k) and the actual or desired output (o_k). The most common objective function applied is the

Mean Squared Error (MSE). The target output is denoted byo_k and the adjustment of the output layer weight is detailed in Formulas (26) and (27):

$$E_k^2 = \frac{1}{2} |y_k - o_k|^2 \tag{26}$$

To update the weights (w_{jk}) in the completely linked level, the backpropagation algorithm is typically employed. The weight update for a given connection (w_{jk}) can be calculated as:

$$\Delta w_{jk} = \eta e_k f' v_j \tag{27}$$

where E_k^2 represents the squared error between the network's output and the desired output, Δw_{jk} is the update for the weight between neuron j and output neuron k, η is the learning rate, e_k is the error signal for output neuron k, and $f' v_j$ is the derivative of the activation function applied to the output of neuron j (Alaminos et al., 2020).

The process of updating weights using backpropagation is typically applied iteratively to minimize the error (E_k) and improve the accuracy of forecasting. This integration of EGARCH with the QNN allows the forecasting model to consider both the temporal dependencies captured by the QNN and the conditional volatility information provided by EGARCH, resulting in enhanced forecasting accuracy. The specific equations and implementation details may vary based on the QNN architecture and the integration of EGARCH.

3.1.4. Deep Recurrent Convolutional Neural Network (DRCNN) with EGARCH Integration

RNNs have found success in various domains, particularly in time series forecasting, thanks to their significant predictive capabilities. The conventional RNN framework is organised around its output, which is influenced by its prior predictions (Mahajan, 2011). Formulas (28) and (29) provide the means to obtain an input sequence vector "x," the hidden states of a recurrent layer "s," and the output of a singular hidden layer "y."

$$s_t = \sigma(W_{xs} x_t + W_{ss} s_{t-1} + b_s) \tag{28}$$

$$y_t = o\left(W_{so} s_t + b_y\right) \tag{29}$$

the weights from the input layer "x" to the hidden layer "s," from the hidden layer to itself, and from the hidden layer to its output layer are denoted as W_{xs}, W_{ss}, and W_{so}, respectively. Additionally, b_y represent the biases of the hidden layer and output layer. In formula (30), σ and o are indicated as symbols for the activation functions.

$$STFT\{z(t)\}(\tau, \omega) = \int_{-\infty}^{+\infty} z(t)\omega\left(t - \tau\right)e^{-j\omega t} dt\Big) \tag{30}$$

in this context, z(t) represents the vibration signals, while, ω(t) symbolises the Gaussian window function centred around 0. The complex function T(τ, ω) defines the vibration signals in terms of both time and frequency. To determine the hidden layers through the convolutional operation, we employ Formulas (31) and (32).

$$S_t = \sigma(W_{TS} * T_t + W_{SS} * S_{t-1} + B_s) \tag{31}$$

$$Y_t = o(W_{YS} * S_t + B_y) \tag{32}$$

being W the convolution kernels.

For the formation of a deep architecture, stacking the recurrent convolutional neural network (RCNN) results in the DRCNN, as proposed by Huang and Narayanan in 2017. In this amalgamation, the ultimate segment of the model incorporates a supervised learning layer, as determined by Formula (33).

$$\hat{r} = \sigma(W_h * h + b_h) \tag{33}$$

where W_h represents the weight and b_h the bias, respectively. To optimize parameter learning, stochastic gradient descent is utilized (Ma and Mao, 2019). Assuming that the actual data at time "t" is denoted as "r," the loss function is outlined in Formula (34).

$$L(r, \hat{r}) = \frac{1}{2}\|r - \hat{r}\|_2^2 \tag{34}$$

To integrate EGARCH, we will append its conditional variance (h_t) to the input of the DRCNN model. The augmented input vector at time t becomes [x_t,h_t].

After combining the outputs of the DRCNN model with the EGARCH component, the subsequent step involves processing this combined information through a completely link level to produce the final load forecasting outcomes.

Z_k is denoted as the weighted sum in the completely link level, V_k represents the output after applying the activation function, and Y_k denotes the final forecasting outcome.

$$Z_k = \sum_{j=1}^{l} \omega_{jk} \cdot v_j \tag{35}$$

$$V_k = f_k(Z_k) \tag{36}$$

$$Y_k = V_k \tag{37}$$

The choice of activation function f_k depends on the specific problem and design of the DRCNN model. To train the DRCNN with EGARCH, a similar process as described in the previous response would be followed, involving the definition of an objective function to measure the error and the use of backpropagation to update the weights in the fully connected layer. This integration allows the DRCNN model to consider both the temporal dependencies captured by the DRCNN architecture and the conditional volatility information provided by EGARCH, leading to an enhanced load forecasting model.

We denote the predicted values by the DRCNN-EGARCH model as $\hat{r}_k$ at time step k and the actual values as r_k. The objective function measures the error between the predicted and actual values. A common choice for regression problems like forecasting is objective function of Mean Squared Error (MSE):

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(\hat{r}_k - r_k)^2 \tag{38}$$

In this step, we denote N as the total number of time steps. The weights in the fully connected layer are updated using the gradient descent optimization algorithm. The gradient descent update rule for weight w_{ij} in the completely link level is given by:

$$\omega_{ij}^{(t+1)} = \omega_{ij}^{(t)} - \eta \frac{\partial MSE}{\partial \omega_{ij}} \tag{39}$$

We denote η as the learning rate, and $\frac{\partial MSE}{\partial \omega_{ij}}$ as the partial derivative of the MSE with respect to weight ω_{ij}. The partial derivative is computed applying the chain rule of calculus and backpropagation:

$$\frac{\partial MSE}{\partial \omega_{ij}} = -\frac{2}{N}\sum_{k=1}^{N}(\hat{r}_k - r_k)\frac{\partial \hat{r}_k}{\partial \omega_{ij}} \tag{40}$$

The derivative $\frac{\partial \hat{r}_k}{\partial \omega_{ij}}$ involves the activations and outputs of the layers in the DRCNN-EGARCH model and depends on the specific architecture and activation functions used. This backpropagation process is applied iteratively over the training dataset to minimize the MSE and improve the model's forecasting performance.

3.1.5. Quantum Recurrent Neural Network (QRNN) with EGARCH Integration

A quantum system with n qubits resides in the n-fold Hilbert space, represented by the tensor product $\mathcal{H} = (\mathbb{C}^2)^{\otimes d}$ resulting in a dimension of 2^d. A quantum state is denoted by a unit vector $\psi \in \mathcal{H}$, commonly expressed in quantum computing using bracket notation $|\psi\rangle \in \mathcal{H}$ |; its conjugate transpose is represented as $\langle\psi| = |\psi\rangle^{\dagger}$. The inner product $\langle\psi|\psi\rangle = \|\psi\|_2^2$ signifies the square of the 2-norm of ψ. $[\psi\rangle\langle\psi]$ then denominates the outer product, producing a tensor of rank 2. The computational basis states are represented by $|0\rangle$ = (1, 0), $|1\rangle$ = (0, 1). Compound basis states, such as $|01\rangle = |0\rangle \otimes |1\rangle$ = (0, 1, 0, 0), are formed by taking the tensor product of individual states.

Therefore, a quantum gate serves as a unitary operation $\mathbb{U}$ on $\mathcal{H}$, where the operation significantly acts on a subset $\mathbb{S} \subseteq$ [n] of qubits, tdenoted by $\mathbb{U}\epsilon\mathbb{S}\mathbb{U}(2^{|\mathbb{S}|})$ To operate on $\mathcal{H}$ we extend $\mathbb{U}$ to act as identity on the remainder of the space, for instance $\mathbb{U}_{\mathbb{S}}\otimes 1_{[n]\setminus\mathbb{S}}$. In a quantum circuit, the first gate, denoted as R(θ) represents a unitary operation on a qubit, acting on the second qubit from below and dependent on the parameter θ. The dotted line extending from the gate signifies a "controlled" operation. For instance, if the control acts solely on a single qubit, the gate corresponds to the single block-diagonal unitary map $|0\rangle\langle 0| \otimes 1 + |1\rangle\langle 1|\otimes\mathbb{R}(\theta) = 1 \oplus \mathbb{R}(\theta)$, indicating "if the control qubit is in state $|1\rangle$ apply $\mathbb{R}(\theta)$". Gate sequences are computed as matrix products in quantum circuits.

The projective measurements of a single qubit are represented by a Hermitia 2× 2 matrix P, for instance, M$|1\rangle\langle 1|$ = diag(0, 1); where the complementary outcome is then denoted as $M^{\perp}$ = 1 −M. These measurements are quantified in meters within the circuit. For a given quantum state $|\psi\rangle$, the post-measurement state is $M|\psi\rangle/p$ /p with probability $p = \|M|\psi\|_2$. This probability also serves as the post-selection likelihood, ensuring the measured result M. This likelihood can be elevated close to 1 through approximately ~ $\sqrt{1/p}$ rounds of amplitude amplification, as proposed by Grover in 2005.

In the training process, we employ quantum amplitude amplification (Guerreschi, 2019) on the output pathways to ensure accurate measurement of the correct token from the training data at each

step. However, non-linear behavior is achievable in quantum mechanics. As an example, a single-qubit gate R(θ)= exp(iYθ) for the Pauli matrix Y (Nielsen, and Chuang, 2001), acts as:

$$R(\theta) = exp\left(i\theta \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\right) = \begin{pmatrix} cos\,\theta & sin\,\theta \\ -sin\,\theta & cos\,\theta \end{pmatrix} \tag{41}$$

specifically, it resembles a rotation within the two-dimensional space defined by the computational basis vectors of a single qubit, $\{|0\rangle, |1\rangle\}$. Although the rotation matrix itself maintains linearity, it is noteworthy that the state amplitudes — $\cos\theta$ and $\sin\theta$— exhibit non-linear dependence on the angle θ. When elevating the rotation to a controlled operation cR(i, θ_i) contingent upon the i^{th} qubit of a state $|x\rangle$ for $x \in \{0,1\}^n$, we attain the mapping

$$R(\theta_0)cR(1,\theta_1)\ldots cR(n,\theta_n)|x\rangle|0\rangle = |x\rangle(cos(\eta)|0\rangle + sin(\eta)|1\rangle) \;\; for$$
$$\eta = \theta_0 + \sum_{i=1}^{n} \theta_i\, x_i \tag{42}$$

Therefore, this equates to a rotation through an infinite transformation of the basis vector $|x\rangle$with $x = \{x_{1,\ldots,}x_n\} \in \{0,1\}^n$, influenced by a parameter vector $\theta = (\theta_0, \theta_1, \ldots, \theta_n)$. This procedure extends linearly to both the base and target state superpositions. Due to the nature of f R(θ) all alterations in amplitude introduced are real-valued.

The non-linear transformation of cosine amplitudes is inherent in a controlled operation. However, the sine function exhibits a less sharply defined profile, resembling that of a linear rectified unit and closely resembling a sigmoidal activation function. This activation function introduces a parameter ord (order) with a value greater than or equal to 1, influencing the tilt and thereby affecting the resulting amplitude. In the case of pure states, this quantum neuron undergoes rotation by an f(θ)=arctan(tan(θ)^2ord), where ord ≥ 1 represents the order of the neuron. Assuming an affine transformation η for the input bitstring x_i, as depicted in formula (43), this rotation is then translated into the amplitudes. This approach highlights the nuanced interplay between non-linear transformations, activation functions, and rotational dynamics in the quantum neuron model.

$$cos\big(f(\eta)\big) = \frac{1}{\sqrt{1+tan(\eta)^{2x2^{ord}}}} \;\; and \;\; sin\big(f(\eta)\big) = \frac{tan(\eta)^{2^{ord}}}{\sqrt{1+tan(\eta)^{2x2^{ord}}}} \tag{43}$$

This transformation emerges from standardising the operation $|0\rangle \mapsto \longrightarrow cos(\theta)^{2^{ord}}|0\rangle + sin(\theta)^{2^{ord}}|1\rangle$ as is evident. The circuit for ord = 1 is illustrated on the left, and for ord = 2, it is depicted on the right. Higher orders can be constructed recursively. For control in superposition, as in a state$|x\rangle + |y\rangle/\sqrt{2}$, the method is ineffective when $x \neq y$ for two bit-strings of length n. In such cases, the amplitudes within the overlap will hinge on the success outcome. A strategy known as fixed-point oblique amplitude amplification (Tacchino et al., 2019) is employed. This technique essentially involves post-selecting the measurement result 0 while maintaining the unitarity of the operation with arbitrary precision.

In this paper, we enhance the capabilities of this quantum neuron by introducing a greater number of check terms. Specifically, η as defined in formula (44) is an affine transformation of the boolean vector $x = \{x_{1,\ldots,}x_n\}$ for $x_i \in \{0, 1\}$. With the introduction of multi-control gates—each having its own parameterized rotation denoted by a multi-index θ_I that varies based on the qubits $i \in I$ upon which the gate conditions—we have the flexibility to incorporate higher-degree polynomials. In other words

$$\eta' = \theta_0 + \sum_{i=1}^{n} \theta_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \theta_{ij} x_i x_j + \cdots = \sum_{\substack{I \subseteq [n] \\ |I| \leq d}} \theta_I \prod_{i \in I} x_i \quad (44)$$

With "d" representing the degree of the neuron, for instance, when d = 2 and n = 4, an illustration of a checked rotation is provided as an example that elevates to this higher-order transformation η' on the bit string x_i. Consequently, boolean logic operations of higher degrees can be directly encoded within a single conditional rotation. For instance, an AND operation between two bits x_1 and x_2 can be succinctly represented as $x_1 x_2$. To implement the constructed QRNN cell, it necessitates an iterative application of the QRNN cell to a sequence of input words $in_1, in_2, \ldots, in_L$. The outgoing lanes out_i denote a discrete distribution measuring p_i over the class labels (Alaminos et al., 2022).

In the intersection of financial forecasting and quantum-inspired machine learning, a pioneering method unfolds through the integration of EGARCH volatility forecasting into a Quantum Recurrent Neural Network (QRNN). The process involves embedding EGARCH output into the input sequence, employing a QRNN cell for dynamic updates, and optimizing with Cross-Entropy Loss. Uniquely, quantum principles like Amplitude Amplification and Repetition-to-Success circuits augment the training, providing a novel perspective. Post-selection techniques add adaptability based on outcomes.

In this step, the EGARCH conditional volatility (h_t) is integrated into the input sequence at each time step. The input at time t ($input_t$) is constructed as a concatenation of the original input (in_t) and the EGARCH conditional volatility (h_t)

$$input_t = [in_t, h_t] \quad (45)$$

This combined input is then fed into the subsequent layers of the QRNN.

Within the Quantum Recurrent Neural Network (QRNN), the hidden state (stst) dynamically evolves by integrating EGARCH volatility (h_t). The sigmoid activation function orchestrates this update, blending the previous hidden state (s_{t-1}), current input (x_t), and EGARCH information through weighted matrices (W_{ss}, W_{sx}) and a bias term (b_s). This integration ensures the nuanced representation of EGARCH dynamics within the evolving hidden state.

$$s_t = \sigma(W_{ss} s_{t-1} + W_{sx} x_t + b_s) \quad (46)$$

The QRNN's predictive prowess emanates from the output (y_t), crafted through a softmax transformation of the updated hidden state (s_t). Fueled by the weight matrix W_{ys} and bias term b_y, this process synthesizes EGARCH-informed information into a probability distribution. This outcome encapsulates the network's ability to distill EGARCH-informed insights into actionable predictions, a pivotal aspect in enhancing financial modeling precision.

$$y_t = softmax(W_{ys} s_t + b_y) \quad (47)$$

In the intricate tapestry of financial modeling, the integration of EGARCH (Exponential Generalized Autoregressive Conditional Heteroskedasticity) stands as a pivotal element for enhanced volatility forecasting. Within this framework, the Cross-Entropy Loss function takes center stage, serving as the compass for training precision. The loss function, defined as

$$Loss = -\sum_t \sum_i out_i \cdot log\ (y_{t,i}) \quad (48)$$

The subsequent weight updates (W_{ss},W_{sx},W_{ys},b_s,b_y) reflect the network's adaptability to refine its parameters and align with the intricacies of EGARCH-informed predictions.

$$W_{ss} \leftarrow W_{ss} - \eta \frac{\partial Loss}{\partial W_{ss}} \tag{49}$$

$$W_{sx} \leftarrow W_{sx} - \eta \frac{\partial Loss}{\partial W_{sx}} \tag{50}$$

$$W_{ys} \leftarrow W_{ys} - \eta \frac{\partial Loss}{\partial W_{ys}} \tag{51}$$

$$b_s \leftarrow b_s - \eta \frac{\partial Loss}{\partial b_s} \tag{52}$$

$$b_y \leftarrow b_y - \eta \frac{\partial Loss}{\partial b_y} \tag{53}$$

The introduction of the unitary operator U(Δθ), governed by a rotation matrix, adds a quantum-inspired dimension to this financial modeling landscape, opening avenues for novel approaches to volatility modeling and risk assessment.

$$U(\Delta\theta) = \begin{pmatrix} cos\,\Delta\theta & sin\,\Delta\theta \\ -sin\,\Delta\theta & cos\,\Delta\theta \end{pmatrix} \tag{54}$$

Quantum amplitude amplification is a technique used to enhance the probability of obtaining correct outcomes in a quantum system. In the context of QRNN training, this involves adjusting parameters and selecting the desired outcomes. A generic form of amplitude amplification can be represented as follows:

$$U_{amplify} = 2(|\psi\rangle \left\langle \psi \right| - \frac{1}{N} I) - I \tag{55}$$

where $|\psi\rangle$ is the quantum state, I is the identity matrix, and N is a normalization factor.

Repetition-to-Success (RUS) Circuit and Correction. The RUS circuit is used to check if a quantum neuron has been executed correctly. The correction circuit is applied when the outcome is not as expected. The operations can be expressed as:

$$U_{RUS} = |0\rangle \langle 0| + e^{i\emptyset}|1\rangle)\langle 1| \tag{56}$$

$$U_{correction} = -I \tag{57}$$

being ϕ is a phase factor that depends on the success or failure of the quantum neuron.

Post-selection techniques aim to fine-tune the training process based on measured outcomes. In the case of fixed-point oblique amplitude amplification, additional quantum operations are performed. The operations can be represented as:

$$U_{post\text{-}select} = diag(1, e^{i\theta}) \tag{58}$$

where θ is a phase factor that influences the post-selection based on measured outcomes.

3.2. Genetic algorithms

3.2.1. Adaptive Boosting and Genetic Algorithm (AdaBoost-GA) with EGARCH Integration

In the traditional AdaBoost algorithm, the weight of each base classifier is determined after computation, without considering the adaptivity of each classifier individually (Wang et al., 2011). Therefore, in this study, Genetic Algorithm (GA) is employed in the adaptive integration procedures of the base classifiers. Both the probability of crossover and the probability of mutation play a crucial role in influencing the optimization effectiveness of the algorithm (Cheng et al., 2019). In selecting the appropriate crossover probability and mutation probability, we refer to the literature (Drezner and Misevičius, 2013). The crossover probability and mutation probability are defined as follows:

$$P_c = \gamma \tag{59}$$

$$P_m = 0.1\ (1 - \gamma) \tag{60}$$

being γ a regulatory factor. The function evaluating fitness was defined as:

$$fit = \frac{\sum_{i=1}^{N} I(y(X_i)=y_i)}{N} \tag{61}$$

Given $\{\omega_{m,j} | i = 1,2, \dots, N; m = 1 \dots, M\}$, which represents the weight of each sample in the base classifier. Here, $\omega_{m,j}$ denotes the weight of the ith sample in m_{th} base classifier. Consider $y_m(x)$ as a base classifier and Y(x) as a strong classifier. The term α_m indicates the weight assigned to the m_{th} classifier, while ε_m signifies the error function of the m_{th} base classifier. The parameter M corresponds to the total number of base classifiers. The AdaBoost-GA suggested algorithm can be elaborated as follows:

$$input_{i,t} = [x_t, h_i] \tag{62}$$

where EGARCH conditional volatility (h_i) is appended to each input sequence (x_i) at each time step.

Output:

Y(.): The final strong classifier:

$$Y(x) = sign\left(\sum_{j=1}^{M} \alpha_j y_j\ (x)\right) \tag{63}$$

1. Initialize $\omega_{1,i} = 1/N$.
2. For i=1 to N do
3.

$$\varepsilon_m = \sum_{i=1}^{N} \omega_{m,i}\, I(y_m(X_i) \neq y_i) \tag{64}$$

4.

$$\alpha_m = 1n\left\{\frac{1-\varepsilon_m}{\varepsilon_m}\right\} \tag{65}$$

if $\alpha_m \geq 0$ then α_m increases with the decrement of ε_m. End if

5.

$$\omega_{m+1,i} = \frac{\omega_{m,i}}{Z_m} exp\left(-\alpha_m y_i y_m([x_t, h_i])\right) \tag{66}$$

where

$$Z_m = \sum_{i=1}^{N} \omega_{m,i} \; exp\left(-\alpha_m y_i y_g([x_t, h_i])\right) \tag{67}$$

6. end for

$$\alpha_m = GA\left(\alpha_{m,1}, \; \alpha_{m,2}, \ldots, \alpha_{m,k}, \beta_{m,1}, \; \beta_{m,2}, \ldots, \beta_{m,l}\right) \tag{68}$$

Utilize a Genetic Algorithm to optimize the weights ($\alpha_{m,i}$) of the base classifiers, considering both EGARCH parameters and base classifier weights.

$$Y_x = sign\left(\sum_{j=1}^{M} \alpha_j y_j \left([x_t, h_i]\right)\right) \tag{69}$$

Combine the base classifiers with their optimized weights, considering EGARCH-informed input, to form the final strong classifier. These equations represent the integration of EGARCH with AdaBoost-GA, where EGARCH information is embedded in the input, influencing both the training of base classifiers and the optimization process through Genetic Algorithms.

3.2.2. Adaptive Neuro-Fuzzy Inference System (ANFIS)-Quantum Genetic Algorithm with EGARCH Integration

In this strategy, each qubit's probability amplitudes are treated as two genes, resulting in two gene strings within every chromosome. Each gene string serves as a representation of an optimization solution, with the quantity of genes determined by the number of optimization parameters. Quantum rotation gates are applied to bring individual qubits to fruition in the optimal chromosome, while non-quantum gates induce mutations to bolster population diversity. The mutation process utilizes non-quantum gates, and quantum rotation gates facilitate crossover and selection operations. The design of the rotation angle calculation function is shaped by the fluctuating trend of the fitness function at the search point. Should the change rate of the fitness function at a specific search point exceed that at other points, the rotation angle undergoes adjustment accordingly (Cao and Shang, 2010). The essential stages of the proposed model are outlined in Algorithm 1.

In Algorithm 1, the proposed model unfolds in several steps. The process begins with Data Pre-processing in Phase one, followed by the generation of a random population in Step 2. Subsequently, radii values are calculated in Step 3, and the ANFIS model is initialized in Step 4. Moving on to Phase two, Step 5 involves executing the optimization algorithm using DCQGA. The algorithm then progresses to Step 6, where the optimal accuracy and performance are identified. Finally, Step 7 signals the termination of the entire process. This systematic approach outlines the key stages of the proposed model, combining data pre-processing, population generation, model initialization, optimization, and

performance evaluation. The implementation of the proposed model involves two major phases.

In Phase 1, referred to as Data Pre-processing, a systematic procedure is employed to convert raw inputs and outputs into a suitable format before the training process. The primary objective of this phase is to diminish the dimensionality of the input data and improve the overall generalization performance, as articulated by Bishop (1995).

The initial data is normalized to the range [0,1] through min-max normalization:

$$X(i) = \frac{x(i)-m}{M-m} \tag{70}$$

for the time series data x, m=min{x}, M=max(x). Four categories of time series, such as the opening price, closing price, highest price, and lowest price, are individually standardized in the experiment.

In the initial step, we enrich the input representation for each time step within the ANFIS-Quantum GA model. The EGARCH conditional volatility (h_j) is integrated into the original time series (x_j) as part of the input sequence:

$$input_{i,t} = [x_t, h_i] \tag{71}$$

This step ensures that the model considers both the raw time series data and the corresponding EGARCH volatility information during the training process, enhancing its ability to capture both temporal dependencies and conditional volatility patterns.

Phase 2. Optimisation algorithm: The algorithm applied in this stage draws inspiration from the double-stranded quantum genetic algorithm (Cao and Shang, 2010) and is formulated as follows:

1.Generate the initial angle to form the double strand from it

$$P_{i,1} = \left(cos\left(y_{i,1}\right), cos\left(y_{i,2}\right), \dots, cos\left(y_{i,n}\right)\right) \tag{72}$$

$$P_{i,2} = \left(sin\left(y_{i,1}\right), sin\left(y_{i,2}\right), \dots, sin\left(y_{i,n}\right)\right) \tag{73}$$

being $y_{i,n}$ a random number between 0 and 2π, $P_{i,1}$ is cosine solution and $P_{i,2}$ represents sine solution.

2. Perform transformation in the solution space

$$X_{(i,j)c} = 0.5^{*}\left[b_i\left(1+\alpha_{i,j}\right) + a_i\left(1-\alpha_{i,j}\right)\right] \tag{74}$$

$$X_{(i,j)s} = 0.5^{*}\left[b_i\left(1+\beta_{i,j}\right) + a_i\left(1-\beta_{i,j}\right)\right] \tag{75}$$

We denote i = 1: m, j = 1: n, m as the number of qubits, and n stands for the population size

3. Calculate the fitness value, which is equivalent to

$$Fitness = \frac{1}{1+MSE} \tag{76}$$

4. Update the optimal composition.

$$\Delta\theta_{i,j} = -sgn(A)\Delta\theta_0 exp\left(-\frac{|\nabla f(X_{i,j})|-\nabla f_i min}{\nabla f_i max-\nabla f_i min}\right) \tag{77}$$

where $\Delta\theta_0$ represents the initial value of the rotation angle,

5. Perform quantum rotation gates to update the relevant qubits on each actual chromosome for every chromosome.

$$A = \begin{vmatrix} \alpha_0 & \alpha_1 \\ \beta_0 & \beta_1 \end{vmatrix} \tag{78}$$

Hence, the rotation angle Δθ can be determined based on rules such as the following: If A ≠ 0, then the sign of (Δθ) is set to be opposite to the sign of A; otherwise, if A = 0, the direction of Δθ is arbitrary,

$$\nabla f(X_{i,j}) = \frac{f(X_{i,p}) - f(X_{i,c})}{(X_{i,j})^p - (X_{i,j})^c} \tag{79}$$

$$\nabla f_i max = max\left(\left|\frac{f(X_{1,p}) - f(X_{1,c})}{(X_{1,j})^p - (X_{1,j})^c}\right|, \left|\frac{f(X_{2,p}) - f(X_{2,c})}{(X_{2,j})^p - (X_{2,j})^c}\right|, \ldots, \left|\frac{f(X_{n,p}) - f(X_{n,c})}{(X_{n,j})^p - (X_{n,j})^c}\right|\right) \tag{80}$$

$$\nabla f_i min = min\left(\left|\frac{f(X_{1,p}) - f(X_{1,c})}{(X_{1,j})^p - (X_{1,j})^c}\right|, \left|\frac{f(X_{2,p}) - f(X_{2,c})}{(X_{2,j})^p - (X_{2,j})^c}\right|, \ldots, \left|\frac{f(X_{n,p}) - f(X_{n,c})}{(X_{n,j})^p - (X_{n,j})^c}\right|\right) \tag{81}$$

being $X_{i,p}$and $X_{i,c}$ ith vector in solution space like the parent colony and child colony, and $(X_{1,j})^p$, and $(X_{1,j})^c$, denotes the jth variable of the vector $X_{i,p}$ and $X_{i,c}$, correspondingly.

6. Alter qubits according to the mutation probability (Pm = 0.1) equivalent to 1 over the population size (pop size = 10). These values have been empirically determined to ensure a satisfactory balance of speed and efficiency for the model.

In the final step, the EGARCH-enhanced ANFIS-Quantum GA model is encapsulated as a function *Y(x,h) = ANFIS-QGA (x,h)*. This function integrates the optimized ANFIS model with the additional EGARCH information embedded in the input. The combination of ANFIS structure and quantum optimization, guided by EGARCH, results in a powerful model capable of capturing both temporal dependencies and conditional volatility patterns for improved forecasting.

3.2.3. Adaptive Genetic Algorithm with Fuzzy Logic (AGAFL) with EGARCH Integration

The genetic algorithm commences with a collection of solutions, referred to as chromosomes, known as a population. These solutions give rise to additional solutions, termed offspring, based on their fitness value – the higher the fitness value, the more probable their reproduction. The fundamental genetic algorithm is outlined in the next paragraph.

The Fundamental Genetic Algorithm begins by creating a random population with m chromosomes, denoted as f(y). Fitness is computed for each chromosome and f(y) within the population. The algorithm iterates through steps, including selection based on fitness, crossover of parents to produce new offspring, mutation, and acceptance of the offspring into a fresh population. This process is repeated until a termination condition is met, concluding with the return of the best solution from the current population. The algorithm then loops back to its initial steps for further iterations. This structured approach aims to iteratively refine populations, striving for optimal solutions.

The adaptive genetic algorithm (AGA) is an enhanced version of the GA, employing adaptive mutations to reach the targeted optimisation results. A GA uses mutations on each parent chromosome, with random gene swapping occurring. In the suggested adaptive mutation, the mutation computation rate is related to the fitness of the chromosome. The mutation yield is based on the mutation rate. For the operation of the AGA, chromosomes have to be created from the set of solutions. Each chromosome undergoes many AGA steps.

The steps in AGA are the creation of chromosomes, estimating fitness function, crossover, adaptive mutation, and selection. The chromosome stands for the information contained in a predefined form of the solution. Chromosome information is usually encoded in a binary string. Each bit of the chain could maintain a match with the feature of the solution. To put it another way, a number may be expressed by a whole string. There are many encoding techniques for encrypting solutions; it mainly pertains to the problem to be solved.

Step 1: Chromosome Generation. Generating chromosomes is the starting step of this AGA algorithm. Since the chromosomes here are only the produced rules using fuzzy; the genes are only the parameters of the rules. In the solution space, several random 'C' chromosomes are produced given in the next formula.

$$Ch_k = \left[G_0^k G_1^k \ldots G_{C_L-1}^k\right] 0 \le k \le M-1; 0 \le i \le C_L - 1 \tag{82}$$

being G_i^k the jth gene of the chromosome, M represents the total population and CL denotes the chromosome´s length.

Step 2 involves calculating the fitness function, as depicted in formula (83). The primary aim of this function is to optimize the rules by maximizing their effectiveness as solutions are chosen.

$$f_t = \sum_{K=1}^{M} R_s \,/M \tag{83}$$

with s represents $\frac{m}{k}$ to be inserted into the sum expression, it serves as a variable that embodies the ratio of 'm' to 'k'. . R_s denotes the selected rule, with M representing the total count of rules. The fitness value, denoted as f_t, is calculated for each chromosome based on the adopted rules. Every chromosome undergoes evaluation using the fitness function. Those chromosomes that meet the requirements of the fitness function, with the assistance of mutation, will be selected for inclusion in the breeding process.

Step 3 involves crossover. This process entails generating a new chromosome by crossing two parent chromosomes. The resulting chromosome is termed the offspring. Crossover is executed based on targeted genes, and the success rate of producing offspring relies on the crossover rate (CO rate). The formula for determining the crossover point is presented below.

$$CP_{rate} = \frac{CG}{CL} \tag{84}$$

where CP_{rate} represents the Crossover Rate, CG denotes the number of Genes Generated and the CL shows the length of the chromosome. Based on the calculated crossover (CO) rate, the parental

chromosomes undergo crossover, generating a set of new chromosomes referred to as offspring. During crossover, we determine the crossover point and swap the genes at these points between the parental chromosomes, leading to offspring that inherit characteristics from both parents. The generated chromosomes typically exhibit higher fitness compared to the previous generation, making them better suited for further processing.

Step 4: Adaptive Mutation. The mutation is done according to a rate of mutation, which is estimated as:

$$MU_r = \frac{P_m}{C_L} \tag{85}$$

being MU_r the Mutation rate, P_m represents the Mutation Point and C_L symbols the length of the chromosome.

Mutation rate selection relies on the predicted fitness value. Due to the rules determined by fuzzy logic, the fitness value serves as the basis for this method. A mutation rate compared with the given fitness values is performed according to the threshold and the resulting values are chosen as the final mutation rate. The vector depicting the mutation points follows:

$$MU_r = \{mp_{1,}mp_{2,\dots}\, mp_{l,}\} \tag{86}$$

being l length of the chromosome. The rate of mutation r identification is done according to the fitness f_t

$$MU_r = \begin{cases} 1; if\ f_t \le T \\ 0; else \end{cases} \tag{87}$$

where the computation of T is performed according to the resulting fuzzy rule. The mutation rate varies for each chromosome over each iteration and relies on the fitness value.

Step: 5 Selection. Given the fitness value achieved, the new chromosomes (N_p) are placed in a sorting pool. In the selection pool, the chromosomes with the best fitness value will be kept at the top. The highest N_p chromosomes reserved in the selection pool are selected as the next generation among the $2N_p$ chromosomes. There are several steps in the prediction model: Rough set-based attribute reduction, normalisation, and then AGAFL classification. Each step of the described model is detailed below:

1. Standardization

Consider the dataset, which consists of attributes and entities. Normalization is applied to simplify the numerical complexity of the data by transforming it into a specific range. The common min-max method is employed for this normalization. The initial dataset is transformed into a range through min-max normalization.

$$D^n = \frac{D - D_{min}}{D_{max} - D_{min}} X[new_{min} - new_{max}] + new_{min} \tag{88}$$

where the range of transform datasets is denoted by new_{min}, new_{max} ; where new_{min} = 0 and new_{max}= 1.

The core framework of the proposed model involves processing inputs such as Decentraland, LuckyBlock, SafeMoon, Avalanche, SeeSaw Protocol, Binamon, Kasta, and X2Y2 to yield an Optimized Fuzzy Logic Classifier with refined classification rules as the output. The model's progression includes the registration of rough set theory to extract valuable information from the input datasets. Mined features from this process are then fed into the fuzzy logic classifier for model training and the generation of classification rules. This encompasses key steps like fuzzification, fuzzy rule generation, and defuzzification. Following this, an Adaptive Genetic Algorithm (AGA) is employed to optimize the classification rules derived in the previous step. The model's validity is established through cross-validation using test data, and its performance is evaluated using accuracy, specificity, and sensitivity metrics. Additionally, a thorough statistical analysis is conducted to affirm the robustness of the results obtained from the model. This comprehensive approach ensures the effectiveness and reliability of the Optimized Fuzzy Logic Classifier in handling the specified inputs.

2. Reduced attributes through Rough Sets

Reducing the attributes using Rough sets is the primary task. Moreover, the number of attributes is minimized and details that are not relevant, disjoint, noisy, or even redundant are removed.

3. Depiction of the solution

The solution is expressed in a binary system. In each bit 1 displays the selection as 0, meaning the non-selection of the equivalence attribute. To quote a scenario, the data set comprising 10 attributes $(a_1, a_2, a_3, \dots a_{10})$ plus a solution Y = 1010110010, then the attributes selected will become $(a_1, a_3, a_5, a_6, a_9)$.

4. Fitness Function

The fitness value of each solution is established by the fitness function and the best solution is selected according to the fitness value. The set of fuzzy logic classifier rules will constitute the population where the fitness function will be implemented. The rule chosen to engage in reproduction to create the following generation will be the superset of S_f. Let $R = \{r_1, r_2, r_3, \dots r_m\}$ be the set of rules to be considered to spawn the next population. Let $R_f \subseteq R \; where \; R_f$ is the set of rules holding super sets of S_f . The merit of each solution is evaluated using the fitness function S_f.

5. Completion standards

Only if the maximum number of iterations is achieved will the algorithm cease its implementation. The solution containing the highest fitness value is chosen using RS, and the AGAFL is employed to rank the datasets. The best attributes are provided as input to the fuzzy classifier.

6. Forecasting with the fuzzy logic system

Having performed feature reduction on the input dataset, the hybrid ADAFL classifier forecasts the model. The fuzzy logic classifier involves three stages: Fuzzification, Fuzzy inference engine, and De-Fuzzification. Incoming data is transformed into a membership value ranging from 0 to 1 through the application of the membership function (MF). The triangular membership approach has been selected to convert the input data into a fuzzy value. The underlying principle governing the assessment of membership values is detailed below:

$$f_x = \begin{cases} 0 \; if \; x \leq = i \; and \; if \; x \geq k \\ \frac{x-i}{j-i} \; if \; i \leq x \leq j \\ \frac{k-x}{k-j} \; if \; j \leq x \leq k \end{cases} \tag{89}$$

Formulating fuzzy rules is essential for linking inputs to their respective outputs. Given attributes represented by $A_{1,}A_{2,}\dots,A_N$ and class labels symbolized by $C_{1,}C_{2,}$, these rules can be constructed using linguistic values like high, medium, and low.

The AGAFL system receives test data with narrowed attributes, which are subsequently transformed into fuzzy values using a fuzzy membership function. These fuzzy inputs are compared with the fuzzy rules established in the rule base. The rule inference technique is then utilized to derive linguistic values, which are further converted into fuzzy ratings through a weighted average approach. Finally, rankings are determined based on the obtained fuzzy scores.

The integration of EGARCH with AGAFL involves enhancing the optimization process with EGARCH-based conditional volatility. The fitness function in AGAFL is designed to maximize the rules as solutions are selected. To incorporate EGARCH, we introduce conditional volatility (σ_t) into the fitness function. The modified fitness function (F_t) now considers both the fuzzy rules and EGARCH-based volatility:

$$F_t = \sum_{K=1}^{M} \frac{R_S}{M} \times \sigma_t^2 \tag{90}$$

being R_s represents the chosen rule, M is the total rule count, and σ_t is the conditional volatility at time t obtained from EGARCH.

The mutation rate in AGAFL is adaptive and depends on the fitness of the chromosome. To integrate EGARCH, we modify the mutation rate computation to consider EGARCH-based volatility:

$$MU_r = P_m \times \frac{\sigma_t}{C_L} \tag{91}$$

where MU_r is the mutation rate, P_m represents the mutation point, σ_t is the EGARCH-based conditional volatility, and C_L is the length of the chromosome.

The selection process can be modified to favor chromosomes that not only provide good rule solutions but also align with the predicted volatility. One way to achieve this is by introducing a selection probability (P_s):

$$P_S = \frac{F_t}{\sum_{i=1}^{N} F_i} \tag{92}$$

where F_t is the weighted fitness of the current chromosome and N is the total number of chromosomes.

The AGAFL algorithm is enhanced by integrating EGARCH-based volatility computation in every iteration. The calculation of EGARCH volatility is incorporated, and the mutation rate is adjusted accordingly. This modification ensures that the optimization process takes into account not just the fuzzy logic rules but also factors in the volatility of the financial market.

3.2.4. Quantum Genetic Algorithm (QGA) with EGARCH Integration

The quantum evolutionary algorithm (QEA) is an evolutionary algorithm rooted in quantum computing principles. It incorporates concepts such as superposition states in quantum computing and utilizes a single encoding form to achieve enhanced experimental outcomes in combinatorial

optimization problems. We aim to improve the global optimization capability of the genetic algorithm by incorporating a local search ability based on the quantum probabilistic model. To address limitations in the Quantum Evolutionary Algorithm (QEA), we introduce a new variant called the "Quantum Genetic Algorithm." This algorithm utilizes a quantum probabilistic vector encoding mechanism, combining the genetic algorithm's crossover operator with the updating strategy of quantum computation. The goal is to enhance the overall global search capacity of the quantum algorithm. The economic dispatching process using the quantum genetic algorithm is outlined in the following steps:

Step 1: Population Initialization in the Quantum Genetic Algorithm (QGA), the fundamental unit of information is a quantum bit. The state of a quantum bit can be represented as either 0 or 1.

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{93}$$

where α and β are two complex numbers corresponding to the likelihood of the respective state: $(|\alpha|^2 + |\beta|^2 = 1), |\alpha|^2, |\beta|^2$symbolize the likelihood of the quantum bit being in the 0 and 1 states, respectively. Quantum Genetic Algorithms (QGA) present an innovative coding method by utilizing quantum bits, where a quantum bit is described using a pair of complex numbers. The representation of a system with m quantum bits is expressed as:

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \cdots & \beta_m \end{bmatrix} \tag{94}$$

In the equation, $|\alpha_i|^2 + |\beta_i|^2 = 1$ (i = 1, 2, ..., m). This display method can be employed to represent any linear superposition of states. For example, consider a system of three quantum bits with the following probability amplitudes:

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix} \tag{95}$$

The state of the system can be identified as

$$\frac{\sqrt{3}}{4\sqrt{2}}|000\rangle + \frac{3}{4\sqrt{2}}|001\rangle + \frac{1}{4\sqrt{2}}|010\rangle + \frac{\sqrt{3}}{4\sqrt{2}}|011\rangle + \frac{\sqrt{3}}{4\sqrt{2}}|100\rangle + \frac{\sqrt{3}}{4\sqrt{2}}|101\rangle + \frac{1}{4\sqrt{2}}|110\rangle + \frac{\sqrt{3}}{4\sqrt{2}}|111\rangle \tag{96}$$

Step 2: Carry out individual coding and measurement of the population generating units. QGA is a probabilistic algorithm similar to EA. The algorithm is denoted as $H(t) = \{Q_1^t, Q_2^t, \ldots Q_h^t, \ldots Q_l^t\}$ h=1,2,…l) being h the size of the population, $Q_l(t) = \{q_1^t, q_2^t, \ldots q_j^t, \ldots q_n^t\}$, where n stands for the number of generator units, t is the evolution generation, q_j^t represents the binary coding of the generation volume of the jth generator unit. Its chromosome is depicted below:

$$q_j^t = \begin{bmatrix} {\alpha_1}^t & {\alpha_2}^t & \cdots & {\alpha_m}^t \\ {\beta_1}^t & {\beta_2}^t & \cdots & {\beta_m}^t \end{bmatrix} \tag{97}$$

(j = 1, 2, ..., n) (m is the length of the quantum chromosome).

In the ''initialization of H(t)," if ${\alpha_1}^t$, ${\beta_1}^t$ (i = 1, 2, ..., m) in $Q_l(t)$ and all the q_j^t are initialized, it indicates that all possible linear superposition states will occur with equal likelihood. During the step of "generating S(t) from H(t)", a collective solution set S(t) is formed by observing the state of H(t), where in the t_{th} generation, $S(t) = \{P_1^t, P_2^t, \ldots, P_h^t, \ldots, P_l^t\}, P_l = \{x_1^t, x_2^t, \ldots, x_j^t, \ldots, x_n^t\}$. Every x_j^t (j =

1, 2, ..., n) is a series, $(x_1, x_2, \ldots, x_i, \ldots, x_m)$, of length m, obtained from the amplitude of the quantum bit $|\alpha_i^t|^2$ or $|\beta_i^t|^2$ (i = 1, 2, ..., m). In the binary scenario, a random number in the range [0, 1] is chosen. If it is greater than $|\alpha_i^t|^2$; '1' is chosen; otherwise, '0' is selected.

Step 3: Make an individual measurement for every item in S(t). Employ a fitness assessment function to evaluate each object in S(t) and retain the best object in the generation. If a satisfactory solution is achieved, the algorithm stops; if not, proceed to the fourth step.

Step 4: Apply a suitable quantum rotation gate U(t)U(t) to update S(t)S(t). The traditional GA employs mating and mutation operations to uphold diversity within the population. In contrast, the quantum genetic algorithm utilizes a logic gate to modulate the likelihood amplitude of the quantum state, ensuring the preservation of population diversity. Therefore, the essence of the quantum genetic algorithm lies in the update facilitated by a quantum gate.

$$U = \begin{bmatrix} cos\,\theta & sin\,\theta \\ sin\,\theta & cos\,\theta \end{bmatrix} \tag{98}$$

with θ representing the rotation angle of the quantum gate. Its numerical value is expressed as

$$\theta = k.f(\alpha_i, \beta_i) \tag{99}$$

$$k = \pi.exp\left(-\frac{t}{iter_{max}}\right) \tag{100}$$

We treat k as a variable associated with the evolutionary generation to dynamically adjust the mesh size. t denotes the evolutionary generation, π be an angle, and $iter_{max}$ be a constant dependent on the complexity of the optimization problem. The purpose of the function (α_i, β_i) is to guide the algorithm in searching for the optimal direction.

Thus, the process of utilizing the quantum rotation gate to modify the probability amplitude for each individual object in the population, specifically by employing the quantum rotation gate U(t) to upgrade S(t) within the quantum genetic algorithm, can be articulated as:

$$S(t+1) = U(t) \times S(t) \tag{101}$$

where t denotes the evolutionary generation, U(t) represents the quantum rotation gate for the t_{th} generation, S(t) means the probability amplitude of a specific object in the t-h generation, and S(t+1) indicates the probability amplitude of the same object in the $t + 1^{th}$generation.

Step 5: Perturbation is introduced as a countermeasure. Given QGA's tendency to be ensnared by superior local extreme values, we introduce population disturbance. QGA investigations reveal that if the leading individual of the current generation represents a local extreme value, liberating the algorithm becomes highly challenging. Consequently, the algorithm risks being confined to local extrema if the foremost individual remains unaltered in successive generations.

Incorporating EGARCH into the Quantum Genetic Algorithm (QGA) enhances its capacity to optimize financial models by introducing volatility dynamics into the evolutionary process. EGARCH, a powerful tool for modeling financial time series, is seamlessly integrated into the QGA framework to improve the global optimization and local search abilities of the algorithm.

The quantum bit representation is extended to include EGARCH parameters in addition to QGA parameters. For a system with mm quantum bits, the quantum chromosome matrix Ψ is defined as:

$$\Psi = \begin{bmatrix} \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \\ \vdots & \vdots \\ \alpha_m & \beta_m \end{bmatrix} \tag{102}$$

where α_i and β_i are complex numbers corresponding to the likelihood of the respective state (0 or 1).

The quantum chromosome for EGARCH at time t is denoted as:

$$q_j^t = \begin{bmatrix} \alpha_1^t & \beta_1^t \\ \alpha_2^t & \beta_2^t \\ \vdots & \vdots \\ \alpha_m^t & \beta_m^t \end{bmatrix} \tag{103}$$

Then, we update the fitness function to include the volatility from EGARCH:

$$F_t = \sum_{K=1}^{M} \frac{\omega \cdot R_s}{M} \cdot \sigma_t^2 \tag{104}$$

being σ_t^2 is the conditional volatility obtained from EGARCH at time t.

The quantum rotation gate is applied to update the quantum chromosome for EGARCH:

$$U(t) = \begin{bmatrix} cos\,(\theta) & sin\,(\theta) \\ -sin\,(\theta) & cos\,(\theta) \end{bmatrix} \tag{105}$$

The rotation angle θ is determined as:

$$\theta = k \cdot f(\alpha_i, \beta_i) \tag{106}$$

$$k = \pi \cdot (-\frac{t}{inter_{max}}) \tag{107}$$

being $iter_{max}$ is a constant, and $f(\alpha_i,\beta_i)$ guides the algorithm towards the optimal solution.

The application of the quantum rotation gate to the entire probability amplitude is represented as:

$$S(t+1) = U(t) \times S(t) \tag{108}$$

where S(t) is the probability amplitude of the current generation at time t, and S(t+1) is the updated probability amplitude for the next generation.

To introduce perturbation in the population and prevent the algorithm from getting stuck in local optima, a perturbation function can be defined. One common approach is to add a small random perturbation to the quantum chromosome elements. Let ΔΨ represent the perturbation matrix:

$$\Delta\Psi = \begin{bmatrix} \Delta\alpha_1 & \Delta\beta_1 \\ \Delta\alpha_2 & \Delta\beta_2 \\ \vdots & \vdots \\ \Delta\alpha_m & \Delta\beta_m \end{bmatrix} \tag{109}$$

The perturbation is then applied to the quantum chromosome Ψ as follows:

$$\Psi_{perturbed} = \Psi + \Delta\Psi \tag{110}$$

being $\Delta\alpha_i$ and $\Delta\beta_i$ are small random perturbations added to the corresponding elements of α_i and β_i. The perturbation function ensures exploration of the solution space, promoting diversity in the population and preventing convergence to local extrema. The amount of perturbation can be controlled by adjusting the magnitude of $\Delta\alpha_i$ and $\Delta\beta_i$ based on the desired level of exploration.

3.2.5. Support Vector Machine- Genetic Algorithm (SVM-GA) with EGARCH Integration

Addressing the parameterization challenge in Support Vector Machines (SVM) involves various approaches, from straightforward methods to more advanced metaheuristics. This method considers a trio of parameters, including the kernel type and a set of kernel-specific constants. The process initiates with the generation of an initial population of chromosomes using a pseudo-random or chaotic distribution. Each chromosome consists of both an integer-valued and a real-valued component. Individuals for the mating pool in each generation can be selected through elitism, roulette, or the recommended Boltzmann choice method. The remaining slots in the mating pool are filled using an n-point crossover operator and a boundary mutation technique (Ping-Feng et al., 2006; Chih-Hung et al., 2009), employing pseudorandom and chaotic sequences.

SVR_{GBC} is an amalgamation of an integer-valued and a real-valued genetic algorithm, employing various genetic operators like selection, crossover, and mutation to generate offspring from the population of real solutions. The chosen individuals are then gathered into a breeding pool, where crossover and mutation operators are applied. An issue with the GA crossover operation in the SVR parameter setting problem is chromosomal heterogeneity. A solution involves implementing a dominance scheme, as suggested by Lewis et al. (1998). Each gene's value is determined by its kernel, considering upper and lower limits of the allele. The subsequent stage is mutation, where several chromosomes are targeted for mutation to produce altered clones introduced into the new population. The current work employs a uniform mutation, as follows:

$$\begin{aligned} c^{old} &= \{c_1, c_2, \dots, c_i, \dots c_n\} \\ c_i^{new} &= LB_i + r * (UB_i - LB_i) \\ c^{new} &= \{c_1, c_2, \dots, c_i{}^{new}, \dots c_n\} \end{aligned} \tag{111}$$

being c^{old} and c^{new} a new selected chromosome before and after mutation accordingly; n symbols the number of genes in the chromosome structure; c_i^{new} stands for the new value for the i allele after mutation; r represents a random number in the range [0,1], generated by one of the available probability distributions; LB_i and UB_i mean the lower and upper bounds of the i allele.

The former is applied to manage the kernel type K_T due to its integer encoding, while the latter alters values like P_1 or P_2 To achieve this, a slight modification of the presented mutation operator is required: A rounding function is introduced immediately after the perturbation of allele i to ensure accurate values. Building on Goldberg's work (1990), we recommend Boltzmann selection, which is used to choose the surviving set of mates based on the system's temperature determined by a cooling schedule. Each solution is either accepted or rejected following the Boltzmann distribution, as defined by (112).

$$P_{(x_i)} = e^{\left(\frac{-\Delta E}{KT}\right)} \tag{112}$$

With k as the Boltzmann constant, ΔE indicating the energy difference between the best and the current solution, and T representing the current system temperature. In simulated annealing (SA), the system temperature is determined by employing a cooling function, selected from the classical exponential or linear options of annealing schemes (Kirkpatrick et al., 1983). The optimization process of Support Vector Regression (SVR) parameters through the Genetic Algorithm (GA) requires a fitness function to evaluate and select chromosomes for the mating set. In this study, x MSE is utilised to assess the quality of each solution to maintain a robust genetic material. This is determined by (113), where σ_t s the observed volatility for period t, $\hat{\sigma}_t$ represents the forecasted volatility for period t, and n denotes the total forecasted time frame.

$$MSE = \frac{\sum_{i=1}^{n}(\sigma_t - \hat{\sigma}_t)^2}{n} \tag{113}$$

Furthermore, the Mean Squared Error (MSE) is determined through a statistical estimation of the model's generalization error, known as k-fold cross-validation (CV). This involves a technique for deriving the parameter values of a model from a training sample (Kohavi, 1995; Refaeilzadeh et al., 2009).

The integration of EGARCH into SVM-GA marks a sophisticated synergy of financial modeling techniques. EGARCH's prowess in capturing time-varying volatility is seamlessly combined with SVM-GA, a robust optimization algorithm for SVM parameters. This fusion enhances the SVM-GA's capacity to optimize not only SVM parameters but also to consider dynamic volatility through EGARCH. The approach involves quantum-inspired chromosome representation, fitness function modification for EGARCH, and the adoption of Boltzmann selection in the genetic algorithm. This synergistic integration offers an advanced solution for refining support vector models, addressing both market trends and volatility dynamics. Now, we delve into the key steps that define this powerful integration.

First, we extend the chromosome representation to include both SVM and EGARCH parameters. For a system with m genes:

$$Chromosome = [C \quad \alpha_1 \quad \beta_1 \quad \alpha_2 \quad \beta_2 \quad \ldots \quad \alpha_m \quad \beta_m] \tag{114}$$

being C represents SVM parameters, and α_i, β_i represent EGARCH parameters.

In the integration of EGARCH into SVM-GA, generating a chromosome using a pseudo-random distribution involves assigning random values to the genes corresponding to SVM parameters. Here is the equation for generating a chromosome using a pseudo-random distribution for integer-valued SVM parameters:

$$c_i = RandomInteger(LBi, UBi) \tag{115}$$

being c_i represents the i-th gene in the chromosome, RandomInteger(LB_i,UB_i) generates a pseudo-random integer within the specified lower (LB_i) and upper (UB_i) bounds for the i-th SVM parameter. We update the fitness function to include both SVM and EGARCH components. Consider the volatility obtained from EGARCH as well as the SVM performance:

$$Fitness = \lambda \times SVM\_Fitness + (1-\lambda) \times EGARCH\_Fitness \tag{116}$$

where λ is a weight parameter to balance the influence of SVM and EGARCH on the overall fitness.

After, we choose individuals for the mating pool using the selection method Boltzmann selection. The Boltzmann selection can be implemented using a probabilistic approach based on the fitness of individuals. The probability of selecting an individual x_i for the mating pool is determined by the Boltzmann distribution. The formula for Boltzmann selection probability (Px_i) is given by:

$$P_{(x_i)} = \frac{e^{-\left(\frac{-\Delta E}{KT}\right)}}{\sum_{j=1}^{N} e^{-\left(\frac{-\Delta E_j}{KT}\right)}} \tag{117}$$

where ΔE is the difference in fitness between the best solution and the current solution x_i, k is the Boltzmann constant, T is the temperature of the system, which may decrease over generations, and N is the total number of individuals in the population.

This probability distribution ensures that individuals with better fitness have a higher probability of being selected but allows for exploration by considering the temperature T. Then, we apply crossover and mutation operators to generate offspring. The crossover should ensure the integration of both SVM and EGARCH parameters. For mutation, modify both SVM and EGARCH parameters.

$$Mutation: \alpha_i^{new} = LB_i + r \times (UB_i - LB_i) \tag{118}$$

$$Mutation: \beta_i^{new} = LB_i + r \times (UB_i - LB_i) \tag{119}$$

$$Mutation: C_i^{new} = LB_C + r \times (UB_C - LB_C) \tag{120}$$

being r is a random number in the range [0,1], and LB, UB denote lower and upper bounds for the corresponding parameters.

4. Data and Indicators

We obtain trading data for the emerging cryptocurrencies under study in our article based on their exchange rate with the U.S. dollar through the websites of CoinMarketCap and CoinGecko. The results of Vidal-Tomás (2022) show that cryptocurrency ranking platforms, such as Coinmarketcap, Coingecko, and BraveNewCoin, cover the majority of trading activity in the cryptocurrency market. They determine that Coinmarketcap and Coingecko are suitable options due to the consistency of its results with major exchange platforms and databases in US dollars. The inputs used in this study are expressed in the prices of the cryptocurrencies, usually applied in the algorithmic trading (Vo and Yost-Bremm, 2020; Fang et al.,2022), and with data in daily frequency. In addressing concerns about potential variations in training intervals across cryptocurrencies, we systematically tackled the issue by implementing a unified training period from 2022/02/15 to 2022/07/18. Despite variations in specific dates for each currency, this standardized timeframe ensures a consistent and comparable duration for the training data. Table 1 summarizes the metadata of the datasets and descriptive statistics.

According to the trading literature (Vo and Yost-Bremm, 2020), we created the following indicators, which are conventionally used by finance professionals: Relative Strength Index, Stochastic Oscillator, Williams %R, Moving average convergence divergence, On Balance Volume, Bollinger bands, Trend elimination oscillator, and Klinger oscillator. They are incorporated into a random walk

model such as a benchmark for the introduction of the inputs in the methodologies used in this study.

Table 1. Datasets' metadata and descriptive statistics.

Dataset	Date Range	Min Open Price	Max Open Price	Mean (Price)	Median (Price)	Standard Deviation (Price)	Kurtosis (Price)	Market Cap
Decentraland	2021/09/24–2022/07/18	$0.503	$5.90	$2.18	2.97515	2.74	3.86	$1,695,267,577
LuckyBlock	2022/01/28–2022/07/18	$0.0006616	$0.009617	$0.00349	0.0051393	1.15	2.72	$38,783,443
SafeMoon	2021/03/11–2022/07/18	$0.000000003257	$0.00000654	$0.000000958	0.00003272	0.073	3.19	$2,557,257
Avalanche	2020/09/24–2022/07/18	$9.48	$146.22	$47.34	77.85	59.22	4.57	$6,692,455,244
SeeSaw Protocol	2022/02/14–2022/07/18	$0.0005877	$0.4603	$0.00831	0.23044385	3.64	4.03	$1,849,274
Binamon	2021/07/01–2022/07/18	$0.01029	$0.8615	$0.1793	0.435895	4.07	3.67	$1,616,528
Kasta	2022/02/05–2022/07/18	$0.05889	$1.49	$0.2081	0.774445	1.37	3.11	$2,183,656
X2Y2	2022/02/15–2022/07/18	$0.1254	$4.17	$1.29	2.1477	1.92	2.83	$32,712,428,183,656

Relative Strength Index (RSI)

RSI is a trend indicator that seeks to capture overbought or oversold conditions. It compares the mean gain of bullish periods to the mean loss of bearish periods over a specified time frame. Furthermore, we use a 14-day period (Vargas et al., 2018). Moreover, RSI assesses the pace of price movement changes and is calculated using:

$$RSI = 100 - \frac{100}{1+RS_p} \tag{121}$$

where RS represents the ratio between the average gain and the average loss over a specific period p. RSI is applied to determine a global trend, allowing for extrapolation of commercial momentum (Wilder, 1978). RSI's default time framework is 14 (p = 14), and a lower time frame denotes an improved sensitivity to movement. RSI range is between 0 and 100, and a score less than 30 points to an over-sold situation, whereas a score greater than 70 signals an overbought scene (Wilder, 1978). RSI allows us to observe price momentum changes over time.

Stochastic Oscillator (SO)

The stochastic oscillator is a well-known momentum indicator. This indicator, credited to Lane (1984), values the momentum in the movement of price changes relative to price or volume. The equation for the stochastic oscillator is:

$$SO = 100 * \frac{C - L_p}{H_p - L_p} \tag{122}$$

where C is the present closing price; H and L represent, respectively, the highest and lowest prices over a specified period p. Similar to the RSI, the default time frame is 14 (p = 14). The stochastic oscillator is measured on a scale from 0 to 100. A reading of 20 or below indicate oversold conditions, while a reading of 80 or above signifies overbought conditions.

Williams %R (WR)

Williams %R is another momentum indicator that evaluates the stock price movement. It gauges the closing price by considering the difference between the maximum and minimum prices. The calculation for Williams %R is performed at regular intervals as follows:

$$\%R = \frac{H_p - C}{H_p - L_p} * -100 \tag{123}$$

where C is the present closing price; H and L denote, respectively, the highest and lowest prices over a specified period p. Typically, the period is set at 14. The RSI scale ranges from 0 to -100. If the Williams %R falls below -80, it is deemed oversold; if it goes above -20, it is considered overbought. While the Williams %R might be viewed as the inverse of the stochastic oscillator, it offers a precise indication of a potential market reversal in upcoming trading periods (Murphy, 1999).

Moving average convergence divergence (MACD)

MACD serves as a trend-following indicator, calculated by deducting the 26-day Exponential Moving Average (EMA) from the 12-day EMA. Subsequently, a 9-day EMA of the MACD is utilised as a signal line (Vargas et al., 2018). MACD stands as a valuable momentum indicator in trading. It is widely acknowledged among traders for its effectiveness and simplicity, providing both trend and momentum signals (Appel, 2005). MACD reveals a difference between two exponential smoothing moving averages—one over a longer period and the other over a shorter period. This highlights the convergence and divergence of these two moving averages. The general understanding is that when the short-term moving average surpasses the long-term moving average (whether in an upward or downward direction), it signals a noise reduction insight into the asset's short-term future trajectory. The MACD equation can be interpreted as a signal line. If the MACD falls below the signal line, it indicates a sell signal, and conversely, if it rises above the signal line, it suggests a buy signal. The equations for both MACD and the signal line are presented as follows:

$$MACD = EMA_p(C) - EMA_q(C) \tag{124}$$

and

$$Signal\ Line = EMA_r(MACD) \tag{125}$$

being $EMA_p(C)$ and $EMA_q(C)$ the exponential smoothing averages of the closing price C over periods p and q, respectively. The period p is consistently smaller than the period q. Similarly, $EMA_r(MACD)$ is the exponential smoothing average of the MACD calculated over a period r. The

usual configurations for p, q, and r are commonly set to 12, 26, and 9, respectively. However, these values might be adjusted depending on the measurement interval.

On Balance Volume (OBV)

OBV stands for an impulse indicator that employs volume flow to forecast changes in the stock price. It is computed based on the addition of volume when the closing price is higher than the previous day's price or the subtraction of volume when the closing price is lower than the previous day's price. We calculate a 20-day EMA of the OBV below

To complete the definition, we set the OBV at the reference day p=0 to be zero, i.e., we let OBV_0 =0. At p=1, OBV_1 =V_1 or - V_1, according to if the price at p=1 increases or decreases in relation to the price at p=0. A high value of OBV denotes good market sentiment. The OBV may also be applied to forecast market reversal. If OBV moves upwards and price moves downwards, and vice versa, it signals upcoming market sentiment reversal (Tsang, and Chong, 2009).

Unlike prior indicators that only use price, OBV is a cumulative indicator that is centred on volume traded. Because the volume movement predates the price movement, a change in OBV indicates a subsequent price change (Granville, 1976). An increase in OBV suggests that the price will rise, while a fall in OBV implies a decline. The formula for estimating the OBV is

$$If\ C_p = C_{(p-1)} \rightarrow OBV_{(p)} = OBV_{(p-1)} \tag{126}$$

$$If\ C_p < C_{(p-1)} \rightarrow OBV_{(p)} = OBV_{(p-1)} - V_p \tag{127}$$

$$If\ C_p > C_{(p-1)} \rightarrow OBV_{(p)} = OBV_{(p-1)} + V_p \tag{128}$$

being C_p the closing price at time p, V_p exhibits the volume at time p. When the closing price of a share equals its closing price of the preceding period, OBV stays constant. Whenever the closing price rises ($C_p > C_{(p-1)}$), the trading volume becomes an addition to the prior OBV. Similarly, if the closing price declines, the trading volume is reduced from the prior period's OBV.

Bollinger bands (BB)

BB is a volatility indicator that captures prices by constructing an upward and a downward band around two standard deviations over a simple 21-day moving average (Vargas et al., 2018). Bollinger Bands are an indicator that contemplates feasible buy/sell points when prices are close to the bands (top/bottom) and takes profits when prices move towards the middle band. Bollinger bands trace two standard deviations up and down from a simple moving average. Moving standard deviations are employed to draw bands around a moving average.

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N}(X_j - \bar{X})^2}{N}} \tag{129}$$

$$\bar{X} = MA = \frac{\sum_{j=1}^{N} X_j}{N} \tag{130}$$

$$Upper\ band = \bar{X} + 2\,\sigma$$

$$Middle\ band = \bar{X} \tag{131}$$

$$Lower\ band = \bar{X} - 2\,\sigma$$

Trend elimination oscillator (TEO)

Trend oscillators are indicators that swing above and below a central line or between predetermined levels, with their values fluctuating over time. Oscillators have the ability to stay at extreme levels, indicating overbought or oversold conditions, for extended durations. However, they are not capable of indicating a trend continuously. In contrast, the On-Balance Volume (OBV) has the ability to follow a trend indefinitely, as it consistently increases or decreases in value over a sustained period (Fernández-Blanco et al., 2008).

TEO recognizes cycles and overbought and oversold areas independently of price movements. Cycles of longer duration are in turn formed by smaller cycles. Moreover, TEO focuses more attention on short trading periods regardless of longer-term trends. An appropriate period must be chosen as shorter cycles are best monitored. Short-term traders can buy when the oscillator turns positive and sell when it turns negative.

$$Med_i = \frac{1}{p}\sum_{i=1}^{1+\zeta} Cie_i \tag{132}$$

$$TEO_i = Cie_i - Med_{i-\left(\frac{\zeta}{2}+1\right)} \tag{133}$$

being Cie_i the current bar closure.

Klinger Volumen Oscillator (KVO)

KVO relates prices and volume by comparing the average of today's prices with yesterday's and, according to the results, applies a multiplier coefficient to the volume. Each day, therefore, it adds or subtracts a figure relating to volume and prices. It then applies two exponential averages to absorb the movements and applies a moment technique to establish the underlying trend of the resulting curve. Finally, KVO applies another exponential moving average to the result of the previous calculation. The result is a curve that moves by trends stably. The different interpretations of this indicator are:

-From a charting point of view, interpreting divergences, breakouts, etc. about the price curve.

-As a filter for signals from other oscillators, indicating the most favorable situations for market trading.

-As a buy point, if it crosses the zero line upwards, and sell if it crosses it in the opposite direction.

$$if\ \frac{Max_i+Min_i+Cie_i}{3} > \frac{Max_{i-1}+Min_{i-1}+Cie_{i-1}}{3} \tag{134}$$

so Data1=1; otherwise Data1=0

$$Data2 = Max_i - Min_i \tag{135}$$

If $Data1_i = Data1_{i-1}$; so Data3=$Data3_{i-1} + Data2_i$; otherwise Data3=$Data2_{i-1} + Data2_i$

$$Data4 = Vol_i * \left|2 * \frac{Data2}{Data3} - 1\right| * Data1 * 100 \tag{136}$$

$$Data5 = Med.Exp\ (p2).Data4 \tag{137}$$

$$Data6 = Med.Exp\ (p3).Data4 \tag{138}$$

$$Data7 = Data5 - Data6 \tag{139}$$

$$KVO = Med.Exp\ (p1).Data7 \tag{140}$$

being Max_i the volume on the current bar, Max_{i-1} the volume in the previous bar, Cie_i the closure of the current bar, Cie_{i-1} the closure of the previous bar, Min_i the minimum on the current bar, and Min_{i-1} the minimum on the previous bar.

5. Results

Tables 2 and 3 show the accuracy results to predict a buy or sell decision in neural networks and genetic algorithms respectively with three metrics: F1, Precision, and Recall. For computing these metrics, we first calculate the values of the next parameters for every classifier:

-TP: An output where the model classifies properly to each cryptocurrency

-FP: An output where the model classifies wrongly to each cryptocurrency

-FN: An output where the model classifies wrongly denied to each cryptocurrency

-TN: An output where the model classifies properly denied to each cryptocurrency

We then estimate the precision and recall for individual classifiers according to the below equations:

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{141}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{142}$$

To summarize the results more compactly, we utilize the micro-means of every metric. We calculate the micro-means of precision and recall using the below formulas:

$$Precision^{\mu} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FP_i} \tag{143}$$

$$Recall^{\mu} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FN_i} \tag{144}$$

being n the number of classifiers.

Also, we calculate the F1 score, which merges the micro-averaged precision and recall into a single value. The F1 score is the harmonic mean of precision and recall.

$$F_1 = \frac{2 \times Precision^{\mu} \times Recall^{\mu}}{Precision^{\mu} + Recall^{\mu}} \tag{145}$$

Table 2. Position accuracy (buy and sell) in Neural Networks with EGARCH integration.

	Dataset	Decentraland	LuckyBlock	SafeMoon	Avalanche	SeeSaw Protocol	Binamon	Kasta	X2Y2
CNN-LSTM	F1 Buy	94.51	94.54	94.56	94.57	94.61	94.67	94.69	94.72
GRU-CNN	F1 Buy	96.24	96.26	96.26	96.28	96.32	96.37	96.38	96.39
QNN	F1 Buy	97.47	97.50	97.50	97.51	97.54	97.60	97.60	97.64
DRCNN	F1 Buy	95.70	95.71	95.75	95.75	95.77	95.82	95.85	95.87
QRNN	F1 Buy	96.83	96.86	96.89	96.90	96.91	96.97	96.99	97.00
CNN-LSTM	F1 Sell	93.10	93.13	93.13	93.16	93.20	93.24	93.26	93.29
GRU-CNN	F1 Sell	94.81	94.81	94.81	94.82	94.85	94.93	94.97	94.99
QNN	F1 Sell	96.02	96.03	96.05	96.08	96.11	96.17	96.18	96.18
DRCNN	F1 Sell	94.28	94.31	94.35	94.37	94.40	94.45	94.45	94.47
QRNN	F1 Sell	95.39	95.42	95.44	95.48	95.50	95.55	95.57	95.60
CNN-LSTM	Precision Buy	93.83	93.85	93.87	93.87	93.88	93.93	93.93	93.94
GRU-CNN	Precision Buy	95.54	95.55	95.57	95.59	95.63	95.72	95.74	95.77
QNN	Precision Buy	96.77	96.80	96.80	96.85	96.86	96.89	96.89	96.92
DRCNN	Precision Buy	95.01	95.03	95.03	95.03	95.06	95.12	95.13	95.16
QRNN	Precision Buy	96.13	96.13	96.15	96.16	96.17	96.23	96.27	96.31
CNN-LSTM	Precision Sell	93.25	93.25	93.28	93.29	93.31	93.35	93.37	93.39
GRU-CNN	Precision Sell	94.95	94.99	95.01	95.02	95.04	95.07	95.11	95.14
QNN	Precision Sell	96.17	96.20	96.24	96.25	96.28	96.32	96.35	96.36
DRCNN	Precision Sell	94.42	94.44	94.47	94.51	94.51	94.56	94.59	94.63
QRNN	Precision Sell	95.54	95.55	95.60	95.60	95.64	95.70	95.71	95.73
CNN-LSTM	Recall Buy	92.45	92.46	92.46	92.46	92.47	92.52	92.56	92.58
GRU-CNN	Recall Buy	94.14	94.15	94.17	94.17	94.21	94.24	94.27	94.29
QNN	Recall Buy	95.35	95.37	95.39	95.41	95.44	95.49	95.49	95.50
DRCNN	Recall Buy	93.62	93.62	93.65	93.70	93.71	93.74	93.77	93.79
QRNN	Recall Buy	94.72	94.72	94.74	94.77	94.79	94.83	94.87	94.89
CNN-LSTM	Recall Sell	92.91	92.95	92.96	92.99	93.01	93.05	93.05	93.08
GRU-CNN	Recall Sell	94.61	94.63	94.64	94.67	94.71	94.77	94.79	94.80
QNN	Recall Sell	95.82	95.84	95.87	95.89	95.90	95.95	95.95	95.98
DRCNN	Recall Sell	94.08	94.10	94.11	94.13	94.16	94.18	94.20	94.22
QRNN	Recall Sell	95.19	95.23	95.27	95.31	95.34	95.37	95.40	95.41

Table 3. Position accuracy (buy and sell) in Genetic Algorithms with EGARCH integration.

	Dataset	Decentraland	LuckyBlock	SafeMoon	Avalanche	SeeSaw Protocol	Binamon	Kasta	X2Y2
AdaBoost-GA	F1 Buy	93.91	93.91	93.95	93.95	93.95	93.99	94.03	94.07
ANFIS-QGA	F1 Buy	95.63	95.66	95.67	95.71	95.73	95.79	95.83	95.86
AGAFL	F1 Buy	98.62	98.66	98.68	98.69	98.73	98.80	98.84	98.88
QGA	F1 Buy	96.83	96.88	96.92	96.94	96.94	96.99	97.02	97.03
SVM-GA	F1 Buy	95.88	95.92	95.94	95.96	95.99	96.03	96.05	96.06
AdaBoost-GA	F1 Sell	92.51	92.53	92.54	92.57	92.61	92.65	92.67	92.69
ANFIS-QGA	F1 Sell	94.20	94.23	94.27	94.32	94.33	94.37	94.40	94.44
AGAFL	F1 Sell	97.16	97.16	97.19	97.22	97.22	97.27	97.31	97.34
QGA	F1 Sell	95.39	95.41	95.45	95.46	95.49	95.51	95.51	95.54
SVM-GA	F1 Sell	94.46	94.48	94.51	94.51	94.53	94.57	94.58	94.60
AdaBoost-GA	Precision Buy	93.23	93.25	93.28	93.33	93.33	93.36	93.38	93.40
ANFIS-QGA	Precision Buy	94.94	94.96	95.00	95.02	95.06	95.13	95.16	95.18
AGAFL	Precision Buy	97.91	97.95	97.98	97.99	98.01	98.10	98.12	98.14
QGA	Precision Buy	96.13	96.16	96.18	96.20	96.25	96.33	96.36	96.40
SVM-GA	Precision Buy	95.19	95.21	95.24	95.26	95.29	95.33	95.37	95.41
AdaBoost-GA	Precision Sell	92.65	92.67	92.68	92.70	92.71	92.77	92.80	92.84
ANFIS-QGA	Precision Sell	94.35	94.36	94.37	94.38	94.42	94.47	94.52	94.56
AGAFL	Precision Sell	97.31	97.33	97.35	97.36	97.38	97.46	97.46	97.48
QGA	Precision Sell	95.54	95.55	95.58	95.58	95.62	95.68	95.69	95.72
SVM-GA	Precision Sell	94.60	94.62	94.66	94.67	94.70	94.74	94.77	94.80
AdaBoost-GA	Recall Buy	91.86	91.90	91.94	91.96	91.97	92.01	92.02	92.04
ANFIS-QGA	Recall Buy	93.54	93.59	93.62	93.65	93.66	93.71	93.72	93.73
AGAFL	Recall Buy	96.48	96.48	96.48	96.49	96.50	96.58	96.60	96.61
QGA	Recall Buy	94.73	94.76	94.80	94.82	94.86	94.88	94.90	94.94
SVM-GA	Recall Buy	93.80	93.83	93.86	93.88	93.90	93.94	93.96	93.99
AdaBoost-GA	Recall Sell	92.32	92.35	92.35	92.36	92.41	92.45	92.50	92.54
ANFIS-QGA	Recall Sell	94.01	94.03	94.04	94.08	94.10	94.16	94.17	94.17
AGAFL	Recall Sell	96.96	96.97	97.00	97.02	97.03	97.05	97.08	97.09
QGA	Recall Sell	95.19	95.21	95.22	95.25	95.29	95.32	95.34	95.38
SVM-GA	Recall Sell	94.26	94.27	94.29	94.31	94.36	94.41	94.44	94.45

On average, Genetic Algorithms perform better than Neural Networks. In Neural Networks methodology, the technique QNN has the highest accuracy and precision with the three metrics, with X2Y2 cryptocurrency having the greatest level of accuracy in this technique (97.64% in F1 Buy, 96.92% in Precision Buy, and 95.50% in Recall Buy). In the genetic algorithm method, the technique AGAFL has the most accurate and precise value with all three metrics, being again the X2Y2 cryptocurrency is the one with the highest accuracy level in this approach (98.88% in F1 Buy, 98.14% in Precision Buy and 96.61% in Recall Buy). X2Y2 is a cryptocurrency where crypto investors may interface with various cryptocurrencies like Coinbase Wallet, WalletConnect, and imToken among others. It was

launched in February 2022 to face the bigger and older NFT trading platform namely OpenSea (Sinha, 2022). In both methodologies, accuracy rates for "buy" decisions are higher than for "sell" decisions.

We have also assessed the cumulative gains for each cryptocurrency by assigning a potential dollar value through the execution of a rolling, out-of-sample simulation that aims to accurately mimic a real live algorithmic trading system. The cumulative profits are presented in Table 4. Initially, we allocate $1,000 of capital. In the first instance, we use one week of currency and signal data to train the methods. Subsequently, we apply the parameters of this model for the following week and evaluate the gains or losses of the trades. We then update the data for the second week to train a new model, and use the parameters of that new model as trading signals for the third week, repeating this process across the entire sample. The timeframes considered are 5, 15, and 360 minutes. Profits or losses for each trade are cumulative (not accrued). This approach can be viewed as a simple **"**long-term only**"** strategy. In this Table 4 we have also applied the Gibbson Ross Shanken (GRS) test from our benchmark model. GRS test serves to examine the ex ante efficiency of portfolios with a variety of different interpretations. However, the most usual interpretation is the **"**geometric mean-standard deviation**"**. From practical examples, the multivariate approach of the GRS test produces more reliable results than the univariate dependent approach (García, 2004). GRS calculation is based on the formula below:

$$F_{GRS} = \frac{T-K-N}{N} \frac{\hat{\alpha}^T \sum^{-1} \hat{\alpha}}{1+\hat{\mu}_K^T \sum_K^{-1} \hat{\mu}_K} \tag{146}$$

being $\hat{\boldsymbol{\alpha}}$ the intercept of the asset pricing model, T represents the total number of observations of each risk factor, K exhibits the total number of test assets, $\sum_{\boldsymbol{K}}^{-\mathbf{1}}$ means the covariance matrix of the idiosyncratic component of asset returns, and $\boldsymbol{\mu_K}$ stands for the average of each risk factor.

Most of the p-values are greater than 0.05 which means that the model successfully passes the GRS test, as, at the 5% significance level, the p-value shows that the results are significant.

Across all cryptocurrencies, the economic benefits of applying a 360-minute trading frequency are smaller, but it is a meaningful enhancement over the "long-only" strategy. Our model yields sizable economic benefits. The highest results occur at the 15- min interval, yielding the highest result for the cryptocurrency Kasta with 14,055.51, in second place Binamo (13,312.66) and followed by SXY2 with 12,643.75. Kasta allows crypto investors to send cryptocurrencies for free right away. Each crypto transfer in crypto wallets becomes easy and convenient just by scanning QR codes. KASTA Token plays a significant part in driving the adoption growth of cryptocurrencies (Sinha, 2022).

Moreover, Figure 1 reveals the average importance of each indicator in the sample for each cryptocurrency in our model.

Table 4. Cryptocurrencies log returns with a $1,000 investment.

Decentraland	Period	Cumulative Profits	Annualized Average Returns	Sharpe Ratio	GRS Test	p-value (GRS Test)	2020	2021	2022
	5 min	7,355.25	529.08%	791.741.538	3.94	0.043	790.40%	13.21%	60.33%
	15 min	8,376.76	314.14%	803.836.676	4.17	0.067	1024.44%	51.20%	186.30%
	360 min	1.344.018.317	91.31%	144.621.766	3.43	0.029	107.02%	28.52%	187.19%
	Long only	6.870.166.014	42.03%	137.548.728	3.24	0.048	34.60%	73.13%	88.11%
LuckyBlock	Period	Cumulative Profits	Annualized Average Returns	Sharpe Ratio	GRS Test	p-value (GRS Test)	2022		
	5 min	5,435.97	456.84%	7.415.007.158	5.16	0.033	634.50%		
	15 min	6,190.93	241.48%	7.542.707.625	4.92	0.051	881.60%		
	360 min	9.933.101.775	82.42%	1.854.816.605	4.696	0.027	112.99%		
	Long only	5.077.464.895	32.59%	134.665.947	4.48	0.019	36.53%		
SafeMoon	Period	Cumulative Profits	Annualized Average Returns	Sharpe Ratio	GRS Test	p-value (GRS Test)	2021	2022	
	5 min	5,195.70	856.84%	7.998.863.842	5.82	0.063	789.09%	13.18%	
	15 min	5,917.29	700.48%	8.120.919.948	5.55	0.047	922.74%	51.11%	
	360 min	9.494.058.676	122.42%	1.963.993.711	5.34	0.014	106.84%	28.48%	
	Long only	4.853.040.946	62.59%	1.287.137.122	5.05	0.017	34.55%	73.01%	
Avalanche	Period	Cumulative Profits	Annualized Average Returns	Sharpe Ratio	GRS Test	p-value (GRS Test)	2020	2021	2022
	5 min	9,932.10	810.21%	8.509.145.672	3.79	0.032	746.14%	12.47%	56.95%
	15 min	11,311.49	662.36%	8.624.559.485	4.01	0.061	967.08%	48.33%	180.83%
	360 min	1.814.884.257	115.76%	1.857.113.174	3.62	0.020	101.03%	26.93%	176.71%
	Long only	9.277.073.073	59.19%	1.217.091.119	3.12	0.012	32.67%	69.04%	83.17%
SeeSaw Protocol	Period	Cumulative Profits	Annualized Average Returns	Sharpe Ratio	GRS Test	p-value (GRS Test)	2022		
	5 min	8,389.05	855.42%	8.983.956	4.97	0.072	687.78%		
	15 min	9,554.14	699.32%	9.105.809.904	4.74	0.036	821.04%		
	360 min	1.532.923.838	122.22%	1.960.740.089	4.35	0.029	96.67%		
	Long only	7.835.787	62.49%	1.285.004.804	4.15	0.025	34.49%		

Binamon	Period	Cumulative Profits	Annualized Average Returns	Sharpe Ratio	GRS Test	p-value (GRS Test)	2021	2022
	5 min	11,689.23	953.55%	1.001.453.829	5.61	0.028	731.79%	29.34%
	15 min	13,312.66	779.54%	101.503.705	5.35	0.045	948.48%	93.76%
	360 min	213.596.389	136.24%	2.185.663.721	4.9	0.039	99.08%	63.38%
	Long only	1.091.832.331	69.66%	1.432.412.382	4.68	0.042	32.44%	112.50%
Kasta	Period	Cumulative Profits	Annualized Average Returns	Sharpe Ratio	GRS Test	p-value (GRS Test)	2022	
	5 min	12,341.49	1006.76%	1.057.334.953	3.65	0.047	927.15%	
	15 min	14,055.51	823.03%	1.071.676.118	3.86	0.058	1201.68%	
	360 min	2.255.150.675	143.84%	2.307.623.757	3.06	0.064	125.54%	
	Long only	1.152.756.575	73.55%	1.512.340.993	2.89	0.052	40.59%	
X2Y2	Period	Cumulative Profits	Annualized Average Returns	Sharpe Ratio	GRS Test	p-value (GRS Test)	2022	
	5 min	11,101.89	1062.93%	1.116.334.244	4.78	0.071	898.88%	
	15 min	12,643.75	868.96%	1.131.475.645	4.56	0.052	1048.74%	
	360 min	2.028.638.831	151.86%	2.436.389.163	4.02	0.068	112.54%	
	Long only	1.036.971.399	77.65%	1.596.729.621	3.84	0.046	39.85%	

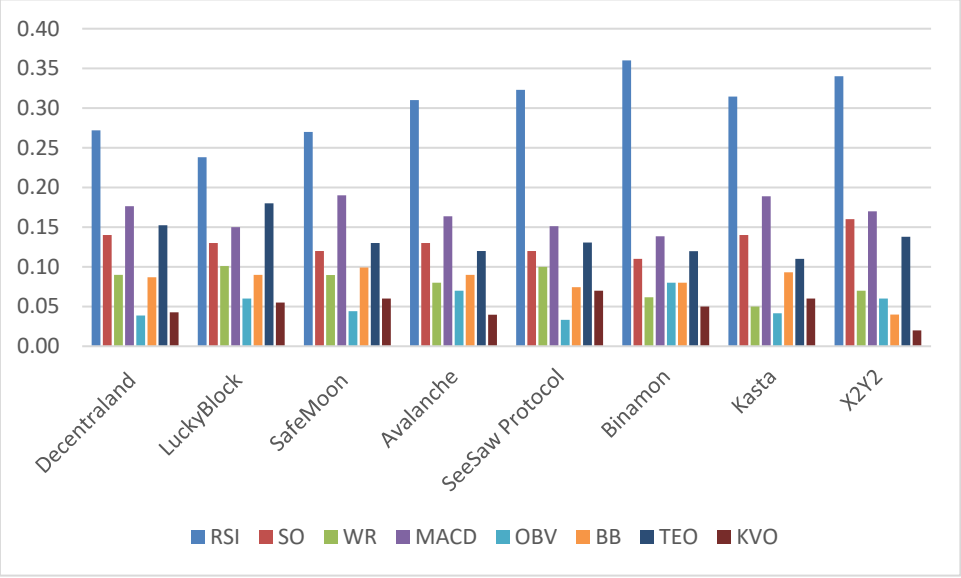

Figure 1. Average financial indicators feature importance.

RSI is the most important in all cryptocurrency's studies. The MACD indicator holds great importance as well, with a peak for SafeMoon currency. Last, SO and TEO indicators are the third and fourth in order of importance; however, in the LuckyBlock currency TEO indicator outperforms MACD. In comparison with other works, Nakano et al. (2018) demonstrate the relative effectiveness of technical indicators in Bitcoin as ANN input to simple time series of returns. These indicators are MACD, SO, and OBV. They conclude that the employment of various technical indicators potentially avoids overfitting in sorting non-stationary financial time series data, improving trading performance. Vo and Yost-Bremm (2020) create financial indicators and used an ML algorithm to devise a high-frequency minute-level trading strategy for Bitcoin, we employ six exchanges as our information technology framework. The financial indicators utilised include RSI, SO, WR, MACD, and OBV. The findings reveal that RSI holds the utmost significance, surpassing other indicators by a significant margin. The remaining indicators carry similar importance, with a particular emphasis on WR and SO.

In summary, our study marks a notable advancement in algorithmic trading analyses by achieving good precision and surpassing the accuracy benchmarks established in prior research. The integration of advanced methodologies, notably the EGARCH model, significantly contributes to this precision, offering a refined understanding of market dynamics and risk factors. This heightened precision becomes especially crucial in the volatile realm of cryptocurrency markets, where rapid fluctuations demand sophisticated modeling approaches.

6. Discusion

Our research confirms the use of Genetic algorithms and the techniques of Neural Networks as suitable strategies in algorithmic trading of the youngest cryptocurrencies. We achieve goods results when compared to previous literature. Madan et al. (2015) center on Bitcoin price data and leverage data at 10-minute and 10-second intervals. They model the price prediction issue like a binomial classification assignment, testing with a custom algorithm that leverages both Random Forest (RF) and generalized linear models. Their results are 50–55% accurate in forecasting the 10-minute forward price sign of future prices. McNally et al. (2018) predict the price of Bitcoin with machine learning algorithms and its results showed that SVM achieved the highest accuracy (62.31%). Zhengyang et al. (2019) analyze the accuracy of Bitcoin price in USD through the implementation of a Bayesian optimized recurrent neural network (RNN) and a LSTM. The LSTM achieves the highest classification accuracy of 52% and an RMSE of 8%. They concluded that RNN and LSTM are efficient for Bitcoin prediction, with LSTM being better able to recognize long-term dependencies. The LSTM surpassed the RNN slightly, but not significantly. Nevertheless, the LSTM takes noticeably more time to train.

Nakano et al. (2018) explore Bitcoin intraday technical trading based on Artificial Neural Networks (ANN) to derive significant trading signals from technical input metrics computed from time series performance data at 15-minute timeslots. They achieved an average accuracy of about 75%. Vo and Yost-Bremm (2020) create an HFT strategy for Bitcoin using RF, reaching an average precision of 94%. They used Design Science Research, a method of inquiry providing guidance on how to construct and assess an information technology device and how the device resolves a formulated problem. Nielsen, and Chuang (2001) implement two machine learning models, ANN and LSTM to forecast the model the price of various cryptocurrencies, like Bitcoin, Ethereum, Ripple, Stellar Lumens, Litecoin,

and Monero. They considered RMSE in joint prediction, obtaining an average of 7%. Their results revealed that ANN, overall, beats LSTM even though theoretically, LSTM is better suited than ANN in terms of time-series dynamics modeling. Their results show that the prospective future state of a cryptocurrency time series depends to a large extent on its historical development.

In 2020, Othman et al. (2020) conducted an analysis of the volatility structure of Bitcoin currency. They utilized a Rapid-Miner program implementing an artificial neural network (ANN) algorithm. The results reveal that ANN proves to be a suitable model, achieving accuracy level of 92.15% in predicting Bitcoin market prices compared to actual prices. Notably, symmetric volatility attributes are employed in this analysis. The findings underscore the significant role of the low price attribute, which emerges as the primary factor influencing Bitcoin price trends, accounting for 63% of the variation. Following this, the close price attribute holds a percentage of 49%, trailed by high price at 46%, and open price at 37%. Alonso-Monsalve et al. (2020) explored the use of convolutional neural networks (CNNs) for intraday trend classification of cryptocurrency exchange rates against the USD. Specifically, they assessed 1-minute exchange rates spanning a year-long period from July 1st, 2018 to June 30th, 2019. They analyzed six major cryptocurrencies over a year, finding CNNs, especially Convolutional LSTM, to be effective predictors of price trends, notably for Bitcoin, Ether, and Litecoin. Convolutional LSTM consistently outperformed other models, including for Dash and Ripple, though with a slight margin. However, the performance of other models was limited for Dash and Ripple, possibly due to inherent noise or unaccounted temporal behavior in the data generation process. Patel et al. in 2020 introduce a hybrid cryptocurrency prediction scheme based on LSTM and GRU, focusing on Litecoin and Monero. Their findings demonstrate good accuracy in price prediction, indicating the scheme's potential applicability across various cryptocurrencies. In this study, a hybrid model combining GRU and LSTM was proposed, outperforming the LSTM network as evidenced by prediction errors.

In 2021, Akyildirim, Goncu, and Sensoy examined twelve major cryptocurrencies, assessing their predictability across daily and various intraday time scales. They employ machine learning classification algorithms such as support vector machines, logistic regression, artificial neural networks, and random forests. Notably, support vector machines emerged as the most reliable and robust models, consistently achieving over 50% accuracy in predicting next day returns across all cryptocurrencies and time scales. While performance varied among coins, many achieved predictive accuracies surpassing 69% without extensive feature tuning. This suggests that with further refinement in model selection and feature exploration, machine learning algorithms could easily surpass 70% predictive accuracy. In 2022, Ortu et al. explore the impact of integrating social and trading indicators alongside traditional technical variables to analyze cryptocurrency price fluctuations at hourly and daily frequencies. They employ various deep learning techniques, including Multi-Layer Perceptron (MLP), Multivariate Attention Long Short Term Memory Fully Convolutional Network (MALSTM-FCN), Convolutional Neural Network (CNN), and Long Short Term Memory (LTMS) neural networks. Focusing on Bitcoin and Ethereum, they evaluate two models: a restricted model considering only technical indicators, and an unrestricted model incorporating social and trading indicators. In the restricted analysis, MALSTM-FCN achieved the highest performance, with an average f1-score of 54% for Bitcoin and CNN for Ethereum at hourly frequency. In the unrestricted case, LSTM yielded the best results for both Bitcoin and Ethereum, boasting average accuracies of 83% and 84%, respectively.

In brief, previous literature has focused mainly on Bitcoin and Ethereum. However, there is an increasing acceptance and popularity of new emerging cryptocurrency markets and some authors have proposed future lines of research to build a multi-asset portfolio of several new cryptocurrencies (Nakano, Takahashi, and akahashi, 2018; Vo, and Yost-Bremm, 2020).

7. Conclusions

We unfold a thorough comparison of methodologies for analyzing algorithmic trading in 8 new cryptocurrencies of 2022, with a unique focus on integrating the EGARCH model. Diverse methods, such as CNN-LSTM, GRU-CNN, QNN, DRCNN, and QRNN in Neural Networks, and AdaBoost-GA, ANFIS-QGA, AGAFL, QGA, and SVM-GA in Genetic Algorithms, are applied to establish a robust model. Notably, the AGAFL technique, enriched by the integration of EGARCH, stands out with the highest accuracy and precision values across all metrics. The X2Y2 cryptocurrency, underpinned by EGARCH, attains acceptable accuracy levels (98.88% in F1 Buy, 98.14% in Precision Buy, and 96.61% in Recall Buy). Furthermore, the Neural Networks methods, particularly the QNN model, also exhibit notable accuracy (97.64% in F1 Buy, 96.92% in Precision Buy, and 95.50% in Recall Buy) through the incorporation of EGARCH.

In contrast to prior research, our study not only achieves superior accuracy results but also underscores the advantages of integrating EGARCH in enhancing machine learning models for algorithmic trading. The EGARCH model, known for its proficiency in capturing volatility dynamics, contributes significantly to the robustness of the applied methodologies. This integration allows for a more nuanced understanding of market behavior and risk factors, providing a comprehensive foundation for improved predictive capabilities.

The innovative integration of EGARCH into our methodologies plays a pivotal role in addressing the complexities of cryptocurrency markets, particularly in managing and forecasting volatility. The EGARCH-enhanced AGAFL technique emerges as a standout performer, showcasing the model's ability to refine precision and accuracy metrics. This underscores the importance of considering volatility patterns, an area where EGARCH excels, when constructing robust machine learning models for algorithmic trading.

7.1. Implications

Our study can serve as a resource for investors, regulators, or even developers. First, regarding investors, accuracy in predicting cryptocurrency movements can significantly influence their investment decisions. High accuracy can lead to more profitable trades in asset buying or selling, and reduce losses. The findings of our study would be a valuable and significant contribution for commercial purposes among cryptocurrency market participants. Investors will proactively predict cryptocurrency price trends and make the correct investment decision, whether its buying, holding, or selling to achieve normal market returns. According to Othman et al. (2020), financial markets have evolved significantly in the last decade, driven by new technologies in automated trading and financial infrastructure. Our finding is particularly important for algorithmic trading purposes, as deciding when to buy and sell cryptocurrencies intraday can be successfully automated with machine learning algorithms.

Second, although cryptocurrency markets are far from being regulated, policymakers must continue to pay attention to these markets. The reason is that Chicago Board Options Exchange (CBOE) and Chicago Mercantile Exchange (CME) introduced Bitcoin futures in December 2017, so it would be realistic to assume that Bitcoin options and those of other emerging cryptocurrencies could also be introduced in the near future (Akyildirim et al., 2021). Regulators could, therefore, consider implementing guidelines or regulations to ensure transparency, fairness, and accountability in the use of neural networks for cryptocurrency trading. Policies could impose disclosure requirements on companies or individuals using neural networks in trading, including information about the accuracy of models, data sources, and potential conflicts of interest. Additionally, as indicated by Wang et al. (2023), a forecast analysis model is also a concern for policymakers and central banks, who are increasingly interested in cryptocurrency volatility. Some central banks are considering launching their own cryptocurrencies.

Third, developers play a crucial role in designing, implementing, and refining neural network models for cryptocurrency analysis. They need to continuously improve the accuracy of their models by incorporating new data, refining algorithms, and addressing potential biases or limitations.

The implications of our analysis, enriched by the integration of EGARCH, extend to regulators and trading platform designers, influencing monetary policy appropriateness and the capacity of central banks to enforce it successfully. The EGARCH model contributes to a more accurate evaluation of risk, providing valuable insights for decision-makers in navigating the dynamic cryptocurrency landscape. The increasing acceptance of new emerging cryptocurrency markets is further underscored by the advantages brought about by EGARCH integration. EGARCH's ability to capture volatility nuances contributes significantly to understanding the characteristics of the cryptocurrency field, providing a solid foundation for strategic decision-making.

In summary, our research, accentuated by the integration of EGARCH, makes an indispensable contribution to the field of High-Frequency Trading. The advantages of incorporating EGARCH are evident in the improved accuracy and precision of machine learning models, emphasizing its role in refining predictions and risk management. Stakeholders gain valuable insights into the application of EGARCH-infused methodologies for trading optimization, asset allocation, and portfolio construction in a globally oriented context.

7.2. Future research

Future research can continue exploring and refining neural network models for cryptocurrency trading, aiming to enhance their reliability. Moreover, consideration can be given to analysing more complex models, incorporating sentiment data that could enrich cryptocurrency predictions. This sentiment data may include indicators extracted from social media platforms, such as the sentiment and emotions expressed in comments on GitHub and Reddit, as demonstrated by Ortu et al. (2022) in their study, considering emotions such as love, joy, anger, sadness, excitement, and similar sentiments. Besides, future research directions involve delving deeper into the advantages of EGARCH integration, exploring its impact on diverse cryptocurrency trading strategies, and extending its application to less-explored assets like derivatives and fixed-income assets. Additionally, ongoing exploration of models

describing agent behavior in crypto asset markets should consider the added value that EGARCH brings to accurate estimations and simulations.

Use of AI tools declaration

The authors affirm that no artificial intelligence (AI) tools were used in the creation of this work.

Funding

This research was funded by the Universitat de Barcelona, under the grant UB-AE-AS017634.

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

Abakah EJA, Gil-Alana LA, Madigu G, et al. (2020) Volatility persistence in cryptocurrency markets under structural breaks. *Int Rev Econ Financ* 69: 680–691. https://doi.org/10.1016/j.iref.2020.06.035.

Abdallah A, Maarof MA, Zainal A (2016) Fraud detection system: A survey. *J Netw Comput Appl* 68: 90–113. https://doi.org/10.1016/j.jnca.2016.04.007

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

Akyildirim E, Goncu A, Sensoy A (2021) Prediction of cryptocurrency returns using machine learning. *Ann Oper Res* 297: 3–36. https://doi.org/10.1007/s10479-020-03575-y

Alameer Z, Elaziz MA, Ewees AA, et al. (2019) Forecasting copper prices using hybrid adaptive neuro-fuzzy inference system and genetic algorithms. *Nat Resour Res* 28: 1385–1401. https://doi.org/10.1007/s11053-019-09473-w

Alaminos D, Esteban I, Salas MB (2023) Neural networks for estimating Macro Asset Pricing model in football clubs. *Intell Syst Account Financ Manage* 30: 57–75. https://doi.org/10.1002/isaf.1532

Alaminos D, Esteban I, Salas MB, et al. (2020) Quantum Neural Networks for Forecasting Inflation Dynamics. *J Sci Ind Res* 79: 103–106. https://doi.org/10.56042/jsir.v79i2.68439

Alaminos D, Salas MB, Fernandez-Gámez MA (2022) Forecasting Stock Market Crashes via Real-Time Recession Probabilities: A Quantum Computing Approach. *Fractals* 30: 1–16. https://doi.org/10.1142/S0218348X22401624

Alonso-Monsalve S, Suárez-Cetrulo AL, Cervantes A, et al. (2020) Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. *Expert Syst Appl* 149*:* 113250. https://doi.org/10.1016/j.eswa.2020.113250

Aloud ME, Alkhamees N (2021) Intelligent Algorithmic Trading Strategy Using Reinforcement Learning and Directional Change. *IEEE Access* 9: 114659–114671. https://doi.org/10.1109/ACCESS.2021.3105259

Appel G (2005) *Technical analysis: power tools for active investors*, FT Press.

Arévalo A, Niño J, Hernández G, et al. (2016) High-frequency trading strategy based on deep neural networks. In International conference on intelligent computing, 424–436. Springer, Cham. https://doi.org/10.1007/978-3-319-42297-8_40

Atsalakis GS, Atsalaki IG, Pasiouras F, et al. (2019) Bitcoin price forecasting with neuro-fuzzy techniques. *Eur J Oper Res* 276: 770–780. https://doi.org/10.1016/j.ejor.2019.01.040

Bhattacharyya S, Jha S, Tharakunnel K, et al. (2011) Data mining for credit card fraud: A comparative study. *Decis support syst* 50: 602–613. https://doi.org/10.1016/j.dss.2010.08.008

Bianchi D, Babiak M, Dickerson A (2022) Trading volume and liquidity provision in cryptocurrency markets. *J Bank Financ* 142: 106547. https://doi.org/10.1016/j.jbankfin.2022.106547

Binns R (2018) Fairness in machine learning: Lessons from political philosophy. In Conference on Fairness, Accountability and Transparency, 149–159. PMLR.

Bishop CM (1995) *Neural networks for pattern recognition,* Oxford university press. https://doi.org/10.1093/oso/9780198538493.001.0001

Borovkova S, Tsiamas I (2019) An ensemble of LSTM neural networks for high-frequency stock market classification. *J Forecasting* 38: 600–619. https://doi.org/10.1002/for.2585

Bouri E, Gupta R, Roubaud D (2019) Herding behaviour in cryptocurrencies. *Financ Res Lett* 29: 216–221. https://doi.org/10.1016/j.frl.2018.07.008.

Bouri E, Lau CKM, Lucey BM, et al. (2019) Trading volume and the predictability of return and volatility in the cryptocurrency market. *Financ Res Lett* 29: 340–346. https://doi.org/10.1016/j.frl.2018.08.015

Bouri E, Lucey B, Roubaud D (2020) The volatility surprise of leading cryptocurrencies: Transitory and permanent linkages. *Financ Res Lett* 33: 101188. https://doi.org/10.1016/j.frl.2019.05.006

Bouri E, Shahzad S, Roubaud D (2019) Co-explosivity in the cryptocurrency market. *Financ Res Lett* 29: 178–183. https://doi.org/10.1016/j.frl.2018.07.005

Briola A, Turiel J, Marcaccioli R, et al. (2021) Deep reinforcement learning for active high frequency trading.

Bustos O, Pomares-Quimbaya A (2020) Stock market movement forecast: A systematic review. *Expert Syst Appl* 156: 113464. https://doi.org/10.1016/j.eswa.2020.113464

Cao M, Shang F (2010) Double chains quantum genetic algorithm with application in training of process neural networks. In 2010 Second International Workshop on Education Technology and Computer Science, 19–22, IEEE. https://doi.org/10.1109/ETCS.2010.88

Cheng Y, Zheng Z, Wang J, et al. (2019) Attribute reduction based on genetic algorithm for the coevolution of meteorological data in the industrial internet of things. *Wirel Commun Mob Com* 2019: 1–8. https://doi.org/10.1155/2019/3525347

Chih-Hung W, Gwo-Hshiung T, Rong-Ho L (2009) A Novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression. *Expert Syst Appl* 36: 4725–4735. https://doi.org/10.1016/j.eswa.2008.06.046

Corbet S, Eraslan V, Lucey B, et al. (2019) The effectiveness of technical trading rules in cryptocurrency markets. *Financ Res Lett* 31: 32–37. https://doi.org/10.1016/j.frl.2019.04.027

Corbet S, Katsiampa P (2018). Asymmetric mean reversion of Bitcoin price returns. *Int Rev Financ Anal.* https://doi.org/10.1016/j.irfa.2018.10.004.

Corbet S, Larkin CJ, Lucey BM, et al. (2020) Kodakcoin: a blockchain revolution or exploiting a potential cryptocurrency bubble? *Appl Econ Lett* 27: 518–524. https://doi.org/10.1080/13504851.2019.1637512

Demir E, Gozgor G, Lau CKM, et al. (2018) Does economic policy uncertainty predict the Bitcoin returns? An empirical investigation. *Financ Res Lett* 26: 145–149. https://doi.org/10.1016/j.frl.2018.01.005

Demir S, Mincev K, Kok K, et al. (2019) Introducing technical indicators to electricity price forecasting: A feature engineering study for linear, ensemble, and deep machine learning models. *Appl Sci* 10: 255. https://doi.org/10.3390/app10010255

Drachal K, Pawłowski M (2021) A review of the applications of genetic algorithms to forecasting prices of commodities. *Economies* 9: 6. https://doi.org/10.3390/economies9010006

Drezner Z, Misevičius A (2013) Enhancing the performance of hybrid genetic algorithms by differential improvement. *Com Oper Res* 40: 1038–1046. https://doi.org/10.1016/j.cor.2012.10.014

Egger DJ, Gambella C, Marecek J, et al. (2020) Quantum computing for finance: State-of-the-art and future prospects. *IEEE Trans Quantum Eng* 1: 1–24. https://doi.org/ 10.1109/TQE.2020.3030314

Fang F, Ventre C, Basios M, et al. (2022) Cryptocurrency trading: a comprehensive survey. *Financial Innovation* 8: 1–59. https://doi.org/10.1186/s40854-021-00321-6

Feng W, Wang Y, Zhang Z (2018) Informed trading in the Bitcoin market. *Financ Res Lett* 26: 63–70. https://doi.org/10.1016/j.frl.2017.11.009.

Fernández-Blanco P, Bodas-Sagi DJ, Soltero FJ, et al. (2008) Technical market indicators optimization using evolutionary algorithms. In Proceedings of the 10th annual conference companion on Genetic and evolutionary computation. 1851–1858. https://doi.org/10.1145/1388969.1388989

Frino A, Garcia M, Zhou Z (2020) Impact of algorithmic trading on speed of adjustment to new information: Evidence from interest rate derivatives. *J Futures Mark* 40: 749–760. https://doi.org/10.1002/fut.22104

Gandal N, Hamrick J, Moore T, et al. (2018) Price manipulation in the Bitcoin ecosystem. *J Monetary Econ* 95: 86–96. https://doi.org/10.1016/j.jmoneco.2017.12.004

Gao X, Li X, Zhao B, et al. (2019) Short-term electricity load forecasting model based on EMD-GRU with feature selection. *Energies* 12: 1140. https://doi.org/10.3390/en12061140

García EAC (2004) An Application of Gibbons-Ross-Shanken'S Test of The Efficiency of A Given Portfolio. Revista Mexicana de Economía y Finanzas Nueva Época REMEF (*Mexican J Econ Financ*) 3. https://doi.org/10.21919/remef.v3i1.161

Gerritsen DF, Bouri E, Ramezanifar E, et al. (2020) The profitability of technical trading rules in the bitcoin market. *Financ Res Lett* 34: 101263. https://doi.org/10.1016/j.frl.2019.08.011

Giudici P, Abu-Hashish I (2019). What determines Bitcoin exchange prices? a network var approach. *Financ Res Lett* 28: 309–318. https://doi.org/10.1016/j.frl.2018.05.013.

Goldberg DE (1990) A note on Boltzmann tournament selection for genetic algorithms and populationoriented simulated annealing. *Complex Syst* 4: 44

Goldblum M, Schwarzschild A, Patel A, et al. (2021) Adversarial attacks on machine learning systems for high-frequency trading. In Proceedings of the Second ACM International Conference on AI in Finance, 1–9. https://doi.org/10.1145/3490354.3494367

Gonçalves DSCP (2019) Quantum neural machine learning: Theory and experiments. *Machine Learning in Medicine and Biology*, 95–115. IntechOpen. https://doi.org/10.5772/intechopen.84149

Goodell JW, Kumar S, Lim WM, et al. (2021) Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis. *J Behav Expl Financ* 32: 100577. https://doi.org/10.1016/j.jbef.2021.100577

Gradojevic N, Kukolj D, Adcock R, et al. (2023) Forecasting Bitcoin with technical analysis: A not-so-random forest? *Int J Forecast* 39: 1–17. https://doi.org/10.1016/j.ijforecast.2021.08.001

Granville JE (1976). Granville's new strategy of daily stock market timing for maximum profit. Prentice-Hall. Hoboken, New Jersey, U.S.

Greaves A, Au B (2015) Using the Bitcoin Transaction graph to predict the price of bitcoin. Available from: htpp://snap.standord.edu/class/cs224w-2015/projects_2015/.

Griffin JM, Shams A (2018) Is Bitcoin really un-tethered? SSRN. Available from: https://ssrn.com/abstract=3195066.

Grobys K, Sapkota N (2020) Predicting cryptocurrency defaults. *Appl Econ* 52: 5060–5076. https://doi.org/10.1080/00036846.2020.1752903

Grover LK (2005) Fixed-point quantum search. *Phys Rev Lett* 95: 150501. https://doi.org/10.1103/PhysRevLett.113.210501

Guerreschi GG (2019) Repeat-until-success circuits with fixed-point oblivious amplitude amplification. *Phys Rev A* 99: 022306. https://doi.org/10.1103/PhysRevA.99.022306

Guidi B, Michienzi A (2022) Social games and blockchain: exploring the metaverse of decentraland. In 2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops, 199–204. https://doi.org/10.1109/ICDCSW56584.2022.00045

Gupta N, Jalal AS (2020) Integration of textual cues for fine-grained image captioning using deep CNN and LSTM. *Neural Comput Appl* 32: 17899–17908. https://doi.org/10.1007/s00521-019-04515-z

Hasan M, Naeem MA, Arif M, et al. (2022) Liquidity connectedness in cryptocurrency market. *Financial Innovation* 8: 1–25. https://doi.org/10.1186/s40854-021-00308-3

Hassan MK, Hudaefi FA, Caraka RE (2022) Mining netizen's opinion on cryptocurrency: sentiment analysis of Twitter data. *Stud Econ Financ* 39: 365–385. https://doi.org/ 10.1108/SEF-06-2021-0237

Hendershott T, Jones CM, Menkveld AJ (2011) Does algorithmic trading improve liquidity? *J Financ* 66: 1–33. https://doi.org/10.1111/j.1540-6261.2010.01624.x

Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9: 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huang B, Huan Y, Xu LD, et al. (2019) Automated trading systems statistical and machine learning methods and hardware implementation: a survey. *Enterp Inf Syst* 13: 132–144. https://doi.org/10.1080/17517575.2018.1493145

Huang CW, Narayanan SS (2017) Deep convolutional recurrent neural network with attention mechanism for robust speech emotion recognition. In 2017 IEEE international conference on multimedia and expo (ICME), 583–588. IEEE. https://doi.org/10.1109/ICME.2017.8019296

Huang JZ, Huang W, Ni J (2019). Predicting Bitcoin returns using high-dimensional technical indicators. *J Financ Data Sci* 5:140–155. https://doi.org/10.1016/j.jfds.2018.10.001

Jia WU, Chen WANG, Xiong L, et al. (2019) Quantitative trading on stock market based on deep reinforcement learning. In 2019 International Joint Conference on Neural Networks (IJCNN), 1–8. IEEE. https://doi.org/ 10.1109/IJCNN.2019.8851831

Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. *Multimedia Tools Appl* 80: 8091–8126. https://doi.org/10.1007/s11042-020-10139-6

Katsiampa P, Corbet S, Lucey B (2019). Volatility spillover effects in leading cryptocurrencies: a BEKK-MGARCH analysis. *Financ Res Lett* 29: 68–74. https://doi.org/10.1016/j.frl.2019.03.009.

Kim BS, Kim TG (2019). Cooperation of simulation and data model for performance analysis of complex systems. *Int J Simulation Model* 18: 608–619. https://doi.org/10.2507/IJSIMM18(4)491

King T, Koutmos D (2021) Herding and feedback trading in cryptocurrency markets. *Ann Oper Res* 300: 79–96. https://doi.org/10.1007/s10479-020-03874-4

Kirkpatrick S, Gelatt CDJ, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220: 671–680. https://doi.org/10.1126/science.220.4598.671

Klaus T, Elzweig B (2017) The market impact of high-frequency trading systems and potential regulation. *Law Financ Mark Rev* 11: 13–19. https://doi.org/10.1080/17521440.2017.1336397

Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95) 2: 1137–1143.

Kristoufek L (2015) What are the main drivers of the Bitcoin price? Evidence from wavelet coherence analysis. *PLoS One* 10: e0123923. https://doi.org/10.1371/journal.pone.0123923

Lahmiri S, Bekiros S (2019) Cryptocurrency forecasting with deep learning chaotic neural networks. *Chaos Soliton Fract* 118: 35–40. https://doi.org/10.1016/j.chaos.2018.11.014

Lane GC (1984) Lane's stochastics. *Tech Anal Stocks Commodities* 2: 80.

LeCun Y, Bottou L, Bengio Y, et al. (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86: 2278–2324. https://doi.org/ 10.1109/5.726791

Lewis J, Hart E, Ritchie G (1998) A comparison of dominance mechanisms and simple mutation on non-stationary problems. *Parallel Problem Solving from Nature (PPSN V)* 1498: 139–148. https://doi.org/10.1007/BFb0056857

Livieris IE, Pintelas E, Stavroyiannis S, et al. (2020) Ensemble deep learning models for forecasting cryptocurrency time-series. *Algorithms* 13: 121. https://doi.org/10.3390/a13050121

Lu W, Li J, Li Y, et al. (2020) A CNN-LSTM-based model to forecast stock prices. *Complexity* https://doi.org/10.1155/2020/6622927

Ma M, Mao Z (2019) Deep recurrent convolutional neural network for remaining useful life prediction. In 2019 IEEE International Conference on Prognostics and Health Management (ICPHM), 1–4. IEEE. https://doi.org/10.1109/ICPHM.2019.8819440

Madan I, Saluja S, Zhao A (2015) Automated Bitcoin trading via machine learning algorithms. Available from: http://cs229.stanford.edu/projects2014.html.

Maghsoodi AI (2023) Cryptocurrency portfolio allocation using a novel hybrid and predictive big data decision support system. *Omega* 115: 102787. https://doi.org/10.1016/j.omega.2022.102787

Mahajan RP (2011) Hybrid quantum inspired neural model for commodity price prediction. In 13th International Conference on Advanced Communication Technology (ICACT2011), 1353–1357. IEEE.

Makarov I, Schoar A (2022) Cryptocurrencies and Decentralized Finance (DeFi). *Brookings Pap Econ Ac* 2022: 141–215. https://doi.org/10.1353/eca.2022.0014

Makrichoriti P, Moratis G (2016) BitCoin's roller coaster: systemic risk and market sentiment. http://dx.doi.org/10.2139/ssrn.2808096.

Makridis CA, Fröwis M, Sridhar K, et al. (2023) The rise of decentralized cryptocurrency exchanges: evaluating the role of airdrops and governance tokens. *J Corp Financ* 79: 102358. https://doi.org/10.1016/j.jcorpfin.2023.102358

Marzo GD, Pandolfelli F, Servedio VD (2022) Modeling innovation in the cryptocurrency ecosystem. *Sci Reports* 12: 1–12. https://doi.org/10.1038/s4159802216924-7

McNally S, Roche J, 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). IEEE, 339–343. https://doi.org/10.1109/PDP2018.2018.00060

Mensi W, Al-Yahyaee K, Kang S (2019) Structural breaks and double long memory of cryptocurrency prices: a comparative analysis from Bitcoin and ethereum. *Financ Res Lett* 29: 222–230. https://doi.org/10.1016/j.frl.2018.07.011.

Mensi W, Lee YJ, Al-Yahyaee KH, et al. (2019). Intraday downward/upward multifractality and long memory in Bitcoin and ethereum markets: an asymmetric multifractal detrended fluctuation analysis. *Financ Res Lett.* https://doi.org/10.1016/j.frl.2019.03.029.

Milana C, Ashta A (2021) Artificial intelligence techniques in finance and financial markets: a survey of the literature. *Strat Change* 30: 189–209. https://doi.org/10.1002/jsc.2403

Mirkamol S, Mansur E (2023) Cryptocurrencies as the Money of the Future, In: Koucheryavy, Y., Aziz, A. (eds) *Internet of Things, Smart Spaces, and Next Generation Networks and Systems,* NEW2AN 2022, Lecture Notes in Computer Science, 13772. https://doi.org/10.1007/978-3-031-30258-9_21

Murray-Rust D, Elsden C, Nissen B, et al. (2023) Blockchain and Beyond: Understanding Blockchains through Prototypes and Public Engagement. *ACM T Comput Hum Int* 29: 1–73. https://doi.org/10.1145/3503462

Murphy JJ (1999) *Technical analysis of the financial markets: A comprehensive guide to trading methods and application,* Penguin. London, U.K.

Nakano M, Takahashi A, Takahashi S (2018) Bitcoin technical trading with artificial neural network. *Physica A* 510: 587–609. https://doi.org/10.1016/j.physa.2018.07.017

Nica O, Piotrowska K, Schenk-Hoppé KR (2022) Cryptocurrencies: Concept and Current Market Structure. In Cryptofinance: A New Currency for a New Economy, 1–28. https://doi.org/10.1142/12353

Nielsen MA, Chuang IL (2001) Quantum computation and quantum information. *Phys Today* 54: 60. https://doi.org/10.1063/1.1428442

Ortu M, Uras N, Conversano C, et al. (2022) On technical trading and social media indicators for cryptocurrency price classification through deep learning. *Expert Syst Appl* 198: 116804. https://doi.org/10.1016/j.eswa.2022.116804

Othman AHA, Kassim S, Rosman RB, et al. (2020) Prediction accuracy improvement for Bitcoin market prices based on symmetric volatility information using artificial neural network approach. *J Revenue Pricing Ma* 19: 314–330. https://doi.org/10.1057/s41272-020-00229-3

Panagiotidis T, Stengos T, Vravosinos O (2018) On the determinants of Bitcoin returns: a lasso approach. *Financ Res Lett* 27: 235–240. https://doi.org/10.1016/j.frl.2018.03.016.

Patel MM, Tanwar S, Gupta R, et al. (2020) A deep learning-based cryptocurrency price prediction scheme for financial institutions. *J Inf Security Appl* 55: 102583. https://doi.org/10.1016/j.jisa.2020.102583

Paule-Vianez J, Prado-Román C, Gómez-Martínez R(2020) Economic policy uncertainty and Bitcoin. Is Bitcoin a safe-haven asset? European Journal of Management and Business Economics 29 (3): 347-363. https://doi.org/10.1108/EJMBE-07-2019-0116

Petukhina AA, Reule RC, Härdle WK (2021) Rise of the machines? Intraday high-frequency trading patterns of cryptocurrencies. *Eur J Financ* 27: 8–30. https://doi.org/10.1080/1351847X.2020.1789684

Ping-Feng P, Chih-Shen L, Wei-Chiang H, et al. (2006) A hybrid support vector machine regression for exchange rate prediction. *Int J Inf Manage Sci* 17: 19–32.

Polasik M, Piotrowska AI, Wisniewski TP, et al. (2015) Price fluctuations and the use of Bitcoin: An empirical inquiry. International *J Electron Comm* 20: 9–49. https://doi.org/10.1080/10864415.2016.1061413

Qin L, Yu N, Zhao D (2018) Applying the convolutional neural network deep learning technology to behavioural recognition in intelligent video. *Tehnički vjesnik* 25: 528–535. https://doi.org/10.17559/TV-20171229024444

Refaeilzadeh P, Tang L, Liu H (2009) Cross-validation. *Encyclopedia Database Syst,* 532–538. https://doi.org/10.1007/978-0-387-39940-9_565.

Ren YS, Ma CQ, Kong XL, et al. (2022) Past, present, and future of the application of machine learning in cryptocurrency research. *Res Int Bus Financ* 63: 101799. https://doi.org/10.1016/j.ribaf.2022.101799

Saad M, Choi J, Nyang D, et al. (2019) Toward characterizing blockchain-based cryptocurrencies for highly accurate predictions. *IEEE Syst J* 14: 321–332. https://doi.org/10.1109/INFCOMW.2018.8406859

Sensoy A (2018) The inefficiency of Bitcoin revisited: a high-frequency analysis with alternative currencies. *Finance Res Lett.* https://doi.org/10.1016/j.frl.2018.04.002.

Sezer OB, Gudelek MU, Ozbayoglu AM (2020) Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Appl Soft Comput* 90: 106181. https://doi.org/10.1016/j.asoc.2020.106181

Sinha D (2022) https://www.analyticsinsight.net/top-10-new-cryptocurrencies-of-2022-to-buy-for-good-returns/.

Ta VD, Liu CM, Tadesse DA (2020) Portfolio optimization-based stock prediction using long-short term memory network in quantitative trading. *Appl Sci* 10: 437. https://doi.org/10.3390/app10020437

Tacchino F, Macchiavello C, Gerace D, et al. (2019) An artificial neuron implemented on an actual quantum processor. *npj Quantum Inf* 5: 1–8. https://doi.org/10.1038/s41534-019-0140-4

Trimborn S, Li M, Härdle WK (2020) Investing with cryptocurrencies—A liquidity constrained investment approach. *J Financ Econometrics* 18: 280–306. https://doi.org/10.1093/jjfinec/nbz016

Tsang WW H, Chong TTL (2009) Profitability of the on-balance volume indicator. *Econ Bull* 29: 2424–2431.

Vargas MR, Dos Anjos CE, Bichara GL, et al. (2018) Deep leaming for stock market prediction using technical indicators and financial news articles. In 2018 international joint conference on neural networks (IJCNN), 1–8. IEEE. https://doi.org/ 10.1109/IJCNN.2018.8489208

Vidal-Tomas D, Ibanez A, Farinos J (2018) Herding in the cryptocurrency market: cssd and csad approaches. *Financ Res Lett*. https://doi.org/10.1016/j.frl.2018.09.008

Vidal-Tomás D (2022) Which cryptocurrency data sources should scholars use? *Int Rev Financ Anal* 81: 102061. https://doi.org/10.1016/j.irfa.2022.102061

Vo A, Yost-Bremm C (2020) A high-frequency algorithmic trading strategy for cryptocurrency. *J Comput Inf Syst* 60: 555–568. https://doi.org/10.1080/08874417.2018.1552090

Wan KH, Dahlsten O, Kristjánsson H, et al. (2017) Quantum generalisation of feedforward neural networks. *npj Quantum Inf* 3: 1–8. https://doi.org/10.1038/s41534-017-0032-4

Wang J, Gao L, Zhang H, et al. (2011) Adaboost with SVM-based classifier for the classification of brain motor imagery tasks. In International Conference on Universal Access in Human-Computer Interaction, 629–634, Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21663-3_68

Wang J, Ma F, Bouri E, et al. (2023) Which factors drive Bitcoin volatility: Macroeconomic, technical, or both? *J Forecast* 42: 970–988. https://doi.org/10.1002/for.2930

Weng F, Hou M, Zhang T, et al. (2018) Application of regularized extreme learning machine based on BIC criterion and genetic algorithm in iron ore price forecasting. In 2018 3rd International Conference on Modelling, Simulation and Applied Mathematics (MSAM 2018), 212–217. Atlantis Press. https://doi.org/10.2991/msam-18.2018.45

Westland JC (2023) Determinants of liquidity in cryptocurrency markets. *Digital Financ* 5: 261–293. https://doi.org/10.1007/s42521-022-00073-7

Wilder JW (1978) *New concepts in technical trading systems,* Bloomington, IN: Trend Research.

Yang B, Sun Y, Wang S (2020) A novel two-stage approach for cryptocurrency analysis. *Int Rev Financ Anal*. https://doi.org/10.1016/j.irfa.2020.101567.

Zhengyang W, Xingzhou L, Jinjin R, et al. (2019) Prediction of cryptocurrency price dynamics with multiple machine learning techniques. In Proceedings of the 2019 4th International Conference on Machine Learning Technologies, 15–19. https://doi.org/10.1145/3340997.3341008

Zhu Y, Dickinson D, Li J (2017) Analysis on the influence factors of Bitcoin's price based on VEC model. *Financial Innovation* 3: 3. https://doi.org/10.1186/ s40854-017-0054-0.

AIMS Press

© 2024 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0)