Performance evaluation of deep learning and boosted trees for cryptocurrency closing price prediction

Authors

Azeez A. Oyedele
Department of Law and Finance
University of Bedfordshire, Luton Campus, United Kingdom
Emails: abiolaoyedele@yahoo.com and Azeez.Oyedele@beds.ac.uk

Anuoluwapo O. Ajayi (Corresponding Author)
Big Data, Enterprise and Artificial Intelligence Laboratory (Big-DEAL)
University of the West of England
Frenchay Campus, Coldhabour Lane, Bristol, United Kingdom
E-mail: anuoluwapo.ajayi@uwe.ac.uk

Lukumon O. Oyedele
University of the West of England
Frenchay Campus, Coldhabour Lane, Bristol, United Kingdom
ayolook2001@yahoo.co.uk

Sururah A. Bello
University of the West of England
Frenchay Campus, Coldhabour Lane, Bristol, United Kingdom
Sururah.Bello@uwe.ac.uk

Kudirat O. Jimoh
Department of Information and Communication Technology,
Osun State University, Nigeria.
kudirat.jimoh@uniosun.edu.ng

Corresponding author: Anuoluwapo O. Ajayi
CA's email: anuoluwapo.ajayi@uwe.ac.uk

Abstract

The emergence of cryptocurrencies has drawn significant investment capital in recent years with an exponential increase in market capitalization and trade volume. However, the cryptocurrency market is highly volatile and burdened with substantial heterogeneous datasets characterized by complex interactions between predictors, which may be difficult for conventional techniques to achieve optimal results. In addition, volatility significantly impacts investment decisions; thus, investors are confronted with how to determine the price and assess their financial investment risks reasonably. This study investigates the performance evaluation of a genetic algorithm tuned Deep Learning (DL) and boosted tree-based techniques to predict several cryptocurrencies' closing prices. The DL models include Convolutional Neural Networks (CNN), Deep Forward Neural Networks, and Gated Recurrent Units. The study assesses the performance of the DL models with boosted tree-based models on six cryptocurrency datasets from multiple data sources using relevant performance metrics. The results reveal that the CNN model has the least mean average percentage error of 0.08 and produces a consistent and highest explained variance score of 0.96 (on average) compared to other models. Hence, CNN is more reliable with limited training data and easily generalizable for predicting several cryptocurrencies' daily closing prices. Also, the results will help practitioners obtain a better understanding of crypto market challenges and offer practical strategies to lower risks.

Keywords: Artificial intelligence, Deep learning, Boosted trees, Optimization, Forecasting, Cryptocurrencies

1. Introduction

Cryptocurrencies have become a global phenomenon attracting a significant number of users due to their decentralization, immutability, and security. They are based on trust in technological infrastructure, allowing financial resources to be sent from anywhere with almost zero latency while network users provide the necessary authentication mechanisms. This new concept thus combines the advantages of transaction anonymity with the speed and convenience of electronic transactions without a central management institution. Over a few years, their increased transaction frequency, turnover, number of participants, and their structural self-organization have resulted in nearly indistinguishable complexity characteristics experienced in traditional financial markets, i.e., the foreign currency market (Forex), at the level of individual time series (Watorek et al., 2021).

Consequently, due to their increasing growth and popularity, they are now being used in official cash flows and the exchange of goods (Chowdhury et al., 2020). Similarly, due to the rapid flow of information and the availability of high-frequency data, machine learning (ML) techniques have gained popularity in the crypto market, especially price prediction, a critical step in financial decision-making related to portfolio optimization, risk evaluation, and trading. However, the cryptocurrency market is highly volatile and complex (Choo, 2015; Watorek et al., 2021; Zoumpekas et al., 2020), with substantial heterogeneous datasets characterized by complex interactions between predictors, which may be difficult for conventional ML techniques to achieve optimal results. Moreover, as a measure of price fluctuations, volatility significantly impacts trade strategies and investment decisions (Guo et al., 2018). Thus, it is essential to have models that can predict the crypto market accurately at par with the stock market. Furthermore, instant knowledge of price movements can lead to higher profits and lower investment risks for investors. Therefore, investigating the possibility of predicting several cryptocurrencies' closing prices using an optimal model configuration on training sets with significant peaks and drops in price missing and evaluating prediction accuracies on datasets from multiple data sources; is the motivation for this study.

Thus, this paper develops a decision support tool and contributes to the findings on comparing prediction models by making a focused comparison of Deep Learning (DL) and boosted tree-based techniques on six cryptocurrency datasets collected from three different data sources. More specifically, using the same optimal model configuration on different cryptocurrencies to investigate their robustness and resistance across imperfect training and testing datasets. A situation where training data is limited or covers only some of the phenomena in the training set has received relatively little attention in the literature. Few studies that used boosted tree-based techniques for modeling the crypto market either predict a famous and single cryptocurrency platform or use a single data source for training and testing the developed models (Sun et al., 2020). The DL techniques are selected because they are good at discovering intricate structures in high-dimensional data (LeCun et al., 2015) and their remarkable problem-solving success in several domains. Furthermore, the predictive performance of DL techniques is benchmarked with three powerful boosted tree-based techniques that are scalable and robust for modeling complex data (Hastie et al., 2009; Sheridan et al., 2016). These attributes have resulted in them being incorporated into the Spark library for large-scale ML applications.

The rest of the paper is organized as follows: Section 2 presents the applications and benchmarking of Artificial intelligence (AI) and ML techniques for cryptocurrency price prediction. Then, Section 3 discusses the methodology, particularly the description of crypto datasets and data pre-processing techniques, genetic algorithms, DL, and boosted tree-based techniques. Finally, section 4 discusses prediction results, and Section 5 concludes the study.

2. Related work

Several quantitative research on financial market modeling has been carried out. Watorek et al. (2021) gave a detailed and comprehensive review of these studies and the statistical properties of the financial market price fluctuations. Based on its high trading activity by investors, many scholars are interested in modeling the crypto market or studying its linear and nonlinear dynamics. A few examples of such studies include using a time-scale multifractal approach to investigate the high frequency of Bitcoin prices and volume (Lahmiri & Bekiros, 2020a), detecting analysis of structural breaks and volatility spillovers (Canh et al., 2019), and modeling large abrupt price swings and long memory in volatility (Chaim & Laurini, 2019). Others involve examining long-range memory, distributional variation, and randomness of bitcoin volatility (Lahmiri et al., 2018) and analyzing the nonlinear correlations and multiscale characteristics of the cryptocurrency market (Watorek et al., 2021). Other interesting studies closely related to the present study are interested in forecasting cryptocurrency prices using artificial intelligence and advanced machine learning algorithms (Dutta et al., 2019; Kwon et al., 2019; Lahmiri & Bekiros, 2019; Lahmiri & Bekiros, 2020b; Mallqui & Fernandes, 2019; Miura et al., 2019; Zoumpekas et al., 2020).

Accordingly, in the last few years, the AI/ML community has deeply explored ML techniques (Table 1., i.e., classification, regression, time series forecasting) to automatically generate profitable trading signals for the cryptocurrency market (Kristjanpoller & Minutolo, 2018). Most studies on quantitative cryptocurrency trading under classification aim to forecast the price trend of Bitcoin (BTC), a protocol based on a peer-to-peer network and public and private key cryptographic techniques. Bitcoin is also the natural base for other cryptocurrencies (Watorek et al., 2021), the leading and most capitalized ($434 Billion) as of July 2022. For example, studies such as those from Alonso-Monsalve et al. (2020), Atsalakis et al. (2019), Ibrahim et al. (2021), Lahmiri and Bekiros (2020b), Mallqui and Fernandes (2019), Mudassir et al. (2020), Nakano et al. (2018), and Sun et al. (2020) addressed the prediction of the next-day direction (up or down) of Bitcoin (BTC) using classification models trained on historical data. These studies considered, amongst others, statistical and ML techniques such as Autoregressive Integrated Moving Average (ARIMA) (Ibrahim et al., 2021), k-nearest neighbor (Chowdhury *et al*., 2020; Lahmiri & Bekiros, 2020), Artificial Neural Networks (ANN) (Chowdhury et al., 2020; Ibrahim et al., 2021; Lahmiri & Bekiros, 2020b; Mallqui & Fernandes, 2019; Mudassir et al., 2020), Logistic Regression (LR) (Borges & Neves, 2020; Chen et al., 2020), Random Forest RF) (Borges & Neves, 2020; Chen et al., 2020; Ibrahim et al., 2021; Sun et al., 2020), and Support Vector Machines (SVM) (Borges & Neves, 2020; Chen et al., 2020; Lahmiri & Bekiros, 2020b; Mallqui & Fernandes, 2019; Sun et al., 2020). Others include Bayesian Neural Networks (BNN) (Shah & Zhang, 2014), Gradient Boosting Machines (GBM) (Borges & Neves, 2020; Lahmiri & Bekiros, 2020; Sun et al., 2020), Extreme Gradient Boosting (XGB), neuro-fuzzy (Atsalakis et al., 2019), Long Short-Term Memories (LSTM) and Recurrent Neural Networks (Cherati et al., 20021; Mallqui & Fernandes, 2019; Mudassir et al., 2020), and Convolution Neural Networks (Alonso-Monsalve et al., 2020). For instance, Atsalakis et al. (2019) adopted neuro-fuzzy techniques to forecast the change in the direction of the BTC price and reported an increase of 71.21% in investment returns by the proposed model compared to the naive buy-and-hold strategy. Also, Alonso-Monclave et al. (2020) used hybrid Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) neural networks, CNN, ANN, and Radial Basis Neural

Networks (RBFNN) for intraday trend classification of BTC, Dash, Ether, Litecoin (LTC), Monero (XMR), and Ripple (XRP), based on technical indicators.

Table 1. Previous studies using AI/ML approaches for cryptocurrency modeling

Reference	**Algorithms**	**Data source**	**Remarks**
Alonso-Monsalve *et al*. (2020)	CNN, hybrid CNN-LSTM, ANN, and RBFNN	Cryptocompare	intraday trend classification for BTC, DASH, Ether, LTC, XMR and XRP, based on technical indicators.
Atsalakis *et al.* (2019)	Neuro-Fuzzy	Bitcoincharts	to forecast the direction in the change of the BTC price.
Borges and Neves (2020)	LR, RF, SVM and GBM	Binance	BNB price trend predicting
Chen *et al.* (2020)	LR, RF, XGB, QDA, SVM and LSTM	CoinMarketCap Binance	A classification problem to predict the sign change of BTC price
Cherati et al 2021	LSTM	Not indicated	forecast daily closing price direction of BTC
Chowdhury *et al*. (2020)	Ensemble learning, GBM, ANNs, and K-NN	Coinmarketcap	forecast the close (closing) price of the cryptocurrency index 30 and nine constituents of cryptocurrencies
Dutta *et al*. (2019)	ANN, LSTM, and GRU	Bitcoincharts	daily BTC price prediction
Guo et al. (2018)	GARCH, RF, Gaussian process, XGT, ElasticNet, LSTM, Temporal mixture models	Not indicated	short-term volatility prediction of BTC price.
Ibrahim *et al*. (2021)	ARIMA, Prophet, RF, RF Lagged-Auto-Regression, and FFDNN	Coinmarketcap	predict market movement direction of BTC
Jang and Lee (2018)	BNN and linear models	Bitcoincharts	analyzing BTC processes
Kristjanpoller and Minutolo (2018)	GARCH and ANN	Not indicated	predict the price volatility of BTC
Kwon *et al.* (2019)	LSTM and GBM	Bithumb	a classification problem for the price trend (price-up or price-down) of BTC, ETH, XRP, BCH, LTC, DASH, and Ethereum Classic.
Lahmiri and Bekiros (2019)	LSTM and GRNN	Not indicated	for price prediction in Dash, XRP, and BTC.
Lahmiri and Bekiros (2020b)	SVR, GRP, RT, kNN, ANN, BRNN and RBFNN	Bitcoin intraday price data	comparatively evaluate ML techniques in forecasting high-frequency price level of BTC.
Mallqui and Fernandes (2019)	Ensembles, RNNs, ANN, SVM	Bitcoincharts and Quandl	compare different ensembles and neural networks to classify BTC price direction and predict closing price.
Huang *et al.* (2019)	Decision trees	Investing	BTC returns prediction
Miura *et al*. (2019)	LSTM, ANN, Ridge, SVM, and GRU	Bitstamp	to predict BTC price volatility
Mudassir *et al.* (2020)	ANN, Stacked ANN, LSTM, SVM	Bitinfocharts	for predicting BTC price movements and prices in short and medium terms
Nakano *et al*. (2018)	DNN	Poloniex	to predict price direction on BTC 15-min time intervals using prices and technical indicators
Peng *et al*. (2018)	GARCH with SVR	Altcoin Charts	predict volatilities of BTC, ETH, and DASH
Poongodi *et al.* (2020)	Linear regression and SVM	Etherchain.org	Ether coin close price prediction.
Shah and Zhang (2014)	BNN	Okcoin	to predict the BTC price variation.
Sun *et al.* (2020)	Light GBM, SVM, RF	Investing	forecast the price trend (falling, or not falling) of cryptocurrency markets
Zoumpekas *et al*. (2020)	CNN, LSTM, Stacked LSTM, Bidirectional LSTM, and GRU	Poloniex	predict the ETH closing price in a short period

They reported that the hybrid CNN-LSTM architecture outperformed other methods considered. Similarly, Ibrahim et al. (2021) compared ARIMA, Prophet, Random Forest, Random Forest Lagged-Auto-Regression, and feed-forward deep neural networks (FFDNN); they reported that FFDNN achieved the highest accuracy of 54% compared to other predictive models. Also, Borges and Neves (2020) compared LR, RF, SVM, and GBM with ensemble voting for the BNB coin market and risk value prediction. They reported that the ensemble voting method, on average, outperformed other learning algorithms with an accuracy of 56.28%. Finally, Chen et al. (2020) compared LR, SVM, LSTM, XGB, Linear discriminant analysis, Quadratic discriminate analysis, and RF for the Bitcoin price trend prediction. The LSTM model obtained the highest accuracy of 67.2%. Also, Cherati et al. (2021) used an LSTM model to forecast the daily closing price direction of the BTC/US and obtained an accuracy of 76.83% on the testing data.

Regarding regression modeling, studies such as those from Chowdhury *et al*. (2020), Dutta *et al*. (2019), Lahmiri and Bekiros (2019), Poongodi *et al.* (2020), and Zoumpekas *et al*. (2020) developed regression models for cryptocurrency price prediction. Such studies employed ML techniques, i.e., linear regression and SVM for Ether coin price prediction (Poongodi *et al.*, 2020), ANN for cryptocurrencies, i.e., BTC, ETH, Dash, price prediction (Chowdhury *et al*., 2020; Dutta *et al*., 2019), GBM (Chowdhury *et al*., 2020), k-NN (Chowdhury *et al*., 2020), and deep learning LSTM and GRU (Dutta *et al*., 2019; Kwon *et al.*, 2019; Lahmiri & Bekiros, 2019; Zoumpekas *et al*., 2020) for predictive model development. For instance, Dutta et al. (2019) compared Recurrent Neural Networks (RNN) and ANNs to predict daily BTC prices, using daily data from January 2010 to June 2019. In the study, feature selection was based on the Variance Inflation Factor (VIF), and the authors reported the performance of RNNs over ANNs on this task. Similarly, Lahmiri and Bekiros (2019) benchmarked LSTM with Generalized Regression Neural Networks (GRNN) for BTC price prediction and reported that LSTM performed better with a smaller Root Mean Square Error (RMSE: 2.75 × 103) compared to 8.80 × 103 for GRNN. Also, Poongodi et al. (2020) adopted linear regression and SVM models for Ethereum (ETH) closing price prediction and concluded that the SVM method had higher accuracy (96.06%) than the LR method (85.46%). Similarly, Zoumpekas (2020) utilized deep learning algorithms to predict the closing price of the Ethereum cryptocurrency and reported a Squared-R of predicted versus the actual ETH/USD data to a degree of more than 60%. Finally, Chowdhury *et al*. (2020) used ANN, GBM, KNN, and ensemble learning methods to forecast the closing price of the cryptocurrency index 30 and nine constituents of cryptocurrencies and reported the highest RMSE obtained for BTC as 32.863 with the GBM model.

In terms of time series prediction, Jang and Lee (2018) used Bayesian Neural Networks to predict the log price and the log volatility of Bitcoin price and obtained MAPE values equal to 0.0198 and 0.6302 for log price and log volatility, respectively. The authors also compared the predictive performance of BNN with a Support Vector Regression (SVR) and linear models. Kristjanpoller and Minutolo (2018) integrated Generalized Auto-regressive Conditional Heteroskedasticity (GARCH) and ANN with Principal Component Analysis (PCA) to predict Bitcoin's price volatility. They reported that the proposed model could capture price volatility to mitigate exposure to financial risk. Also, Peng et al. (2018) evaluated the predictive performance of a hybrid GARCH and Support Vector Regression model in estimating the volatility of three cryptocurrencies and three currencies. Similarly, Guo et al. (2018) formulated probabilistic temporal mixture models to capture autoregressive dependence in price volatility history. They benchmarked the predictive performance of the proposed models with some conventional techniques and concluded that the proposed model had the lowest RMSE in price volatility prediction. Also, Miura et al. (2019) predicted the future volatility values based on past samples using ANN, GRU, LSTM, SVM, and Ridge Regression techniques. They concluded that the Ridge Regression had the overall best performance.

Consequently, previous studies employing AI/ML techniques have aimed to model the cryptocurrency market for improved decision-making regarding investments with higher returns and lower risk (Borge & Neves, 2020; Kristjanpoller & Minutolo, 2018). This interest is associated with increasing efforts being expended by researchers and financial organizations to minimize financial risks. However, the predictive performance of current frameworks still needs improvements, as evidenced by several cryptocurrency price modeling studies aimed at improving forecasting methods for profitable investment decisions (Ibrahim et al., 2021; Huang et al., 2019; Jang & Lee, 2018;

Kristjanpoller & Minutolo, 2018; Peng et al., 2018; Watorek et al., 2021). Furthermore, financial investors need efficient strategies to reduce financial risk resulting from the increased complexity characteristics observed across most financial markets (Watorek et al., 2021). Traditional ML techniques require manual feature extraction from massive datasets to transform data into internal forms to enhance ML models' predictive ability (LeCun et al., 2015) for guaranteed optimal results. This limitation, in addition to the specific issue each ML model has. For instance, the logistic regression model has difficulty capturing nonlinear and local relationships among dependent and independent variables. Similarly, despite their ability to learn from data and fault tolerances, ANNs can suffer from uncontrolled convergence speed and local optima. Also, Bayesian Neural Networks and SVMs have computational complexity issues. Also, decision trees can have high variance across samples, making predictions and probabilities unstable for new cases. Besides these challenges, modern financial markets are characterized by a rapid flow of information, high-frequency data, nonlinear interactions, and complex characteristics (Wątorek et al., 2021), which may be difficult for conventional ML techniques to achieve optimal results.

Also, in modeling cryptocurrencies and benchmarking different ML models' performance, most existing studies considered a single data source of historical data for training, validating, and testing their models. In addition, most used ML models to predict famous and single cryptocurrency platforms, i.e., BTC price (Atsalakis et al., 2019; Chen et al., 2020; Lahmiri & Bekiros, 2020; Shah & Zhang, 2014), ETH (Zoumpekas et al., 2020), and BNB (Borges & Neves, 2020). However, other cryptocurrencies, i.e., LTC, Dogecoin (DOGE), and Stellar (XLM), are among the top 10 currencies with the potential to be adopted in financial institutions. Akyildirum et al. (2021) also opined that these other cryptocurrencies had attracted relatively less attention. Therefore, besides the established cryptocurrencies, it is worth investigating ML models' robustness on less famous ones to offer a suitable strategy for their price prediction and understand their overall price dynamics. Similarly, the robustness of an optimal model configuration on several cryptocurrencies and performance on the different testing datasets require an assessment, especially models' sensitivity to training sets, where peaks and drops in prices are not adequately represented, has hitherto received little academic attention. Therefore, constructing robust predictive models to accurately forecast prices for multiple cryptocurrencies is a significant business challenge for probable investors and government agencies. Accordingly, a robust technique is desirable to improve prediction ability to enhance the prediction of cryptocurrencies' closing prices for improved financial investments.

Therefore, deep learning techniques are adopted in this study because of their advantage in discovering intricate structures in high-dimensional data, their ability to represent complex data, and their remarkable problem-solving successes in several domains (LeCun et al., 2015). Though, deep learning architectures, i.e., CNN, LSTM, GRU, DFNN, have been actively used for cryptocurrency price, volatility, and return predictions in recent years (Alonso-Monsalve et al., 2020; Dutta et al., 2019; Guo et al., 2018; Ibrahim et al., 2021; Mallqui & Fernandes, 2019; Nakano et al., 2018; Zoumpekas et al., 2020). However, it is noted that in studies such as those from Alonso-Monsalve et al. (2020) and Nakano et al. (2018), CNN and DNN techniques were used primarily for trend (price direction) classification problems. Instead, this present study utilizes both techniques for estimating a real-valued variable (closing price). Also, it is observed that Dutta et al. (2019) and Zoumpekas et al. (2020) adopted deep learning techniques, i.e., CNN and GRU, to predict the closing price of either BTC or ETH. In contrast, this current study adopts the same configurations of CNN, DFNN, and GRU architectures to predict the daily closing prices of multiple cryptocurrencies from multiple data sources. Also, the predictive abilities of the proposed all-inclusive and optimal deep learning models are benchmarked with a few key powerful boosted tree techniques in GBM, Adaboost, and XGB using standard metrics from the literature.

3. Methodology

3.1. Dataset and Data preprocessing

The study collected datasets from Yahoo finance, UK Investing, and Bitfinex to investigate the robustness of prediction models in terms of how they respond to patterns in multiple data sources. For example, the dataset from Yahoo finance is for training and validating the models, while other datasets are for testing the prediction models. The Yahoo finance dataset for the six cryptocurrencies (BTC-USD, ETH-USD, BNB-USD, LTC-USD, XLM-USD, and DOGE-USD) covers the duration between January 1, 2018, to December 31, 2021 (1442 observations). UK Investing dataset covers from July 1, 2021, to March 2, 2022 (245 observations). Also, the Bitfinex dataset for the six cryptocurrencies is from January 1, 2021, to July 6, 2021 (187 observations). More importantly, validating models on different datasets helps guarantee that they do not fit data-specific features. Each dataset has five features, namely, the closing price (Close), highest price (High), lowest price (Low), opening price (Open), and the daily cryptocurrency volume (Volume). In addition, additional features are created, i.e., weighted average (using "Price" as values and "year" as weights) and technical indicators that may impact prices, which include simple moving average (SMA) and exponential moving average (EMA). The rationale for selecting SMA is to allow a model to recognize trends by smoothing the data more efficiently. At the same time, EMA facilitates the dampening of the effects of short-term oscillations. The significant difference between SMA and EMA is that EMA assigns a greater weight to recent data and reacts faster to recent price variations. Furthermore, these technical indicators are calculated using different periods, i.e., day 3, day 10, and day 20.

Fig 1a depicts the price exhibiting nonlinear dynamical traits for the six cryptocurrencies versus time from January 1, 2018, to December 31, 2021. For instance, in Fig. 1a, BTC indicates that the price significantly rose in late 2020 (around 13/11/2020, approximately at index 1048 on the graph). Similarly, for ETH, a significant increase in closing price becomes noticeable around 02/02/2021, approximately at index 1129 on the graph. Also, a significant increase in the BNB price is at index 1136 on the graph, i.e., 09/02/2021. Similarly, Figs 1a and Table 2 present the summary statistics of cryptocurrencies (Yahoo finance) from 2018 to 2021. Though world financial markets do not work simultaneously due to different time zones, most work around the clock, and only temporary price fluctuations caused by the human factor are noticeable. Thus, to study this temporary price fluctuation, the percentage changes in cryptocurrency prices during the day were calculated, and moderate changes ranging from 0.05% to 1% were discovered. Thus, the time zones have little or no effect on the cryptocurrency market. Furthermore, these cryptocurrencies from the three datasets (Yahoo finance, UK Investing, Bitfinex) are similar and have identical distributions.

Again, there are no missing values in the datasets; however, there are noticeable outliers, as depicted in the box plots containing the information to identify their distribution. For instance, the closing prices for BTC in 2018 and 2020 have outliers (Fig. 1b). Also, LTC in 2020 and 2021 (Fig. 1c) has outliers, and XLM has outliers in 2018, 2020 and 2021 (Fig. 1d). Nevertheless, outliers are kept in the predictive modeling phase since they carry meaningful information and deleting them could cause substantial information loss. However, the normalization technique is adopted to transform raw data into a form where the features are all uniformly distributed, i.e., standardizing the features with their mean and standard deviation to address the dominant features and outliers. Nevertheless, understanding the effect of the training data size on the model performance is critical to advancing the knowledge about its generalization ability, specifically in investigating the robustness of models where specific insights are not in training sets. Thus, two scenarios are presented in training the prediction models and tuning their parameters. The purpose is to investigate the training set size dimensions' effects on prediction quality. The first (Scenario A) divides each cryptocurrency dataset from Yahoo Finance into training (1154 days, i.e., from 01/01/2018 - 28/02/2021) and test (i.e., 305 days: 01/03/2021 - 31/12/2021). Here prominent peaks in cryptocurrency prices are missing from the training set. Scenario B divides each cryptocurrency dataset from Yahoo finance into training (1240 days, i.e., from 01/01/2018 to 25/05/2021) and test set (i.e., 219 days: 26/05/2021-31/12/2021). Thus, scenario B has sufficient peaks and lows in prices captured in the training set. For instance, Fig 2. depicts the two scenarios illustrated with BNB/USD, where higher spikes are not part of the training set (Scenario A), and some of these spikes are incorporated in the training set (Scenario B).

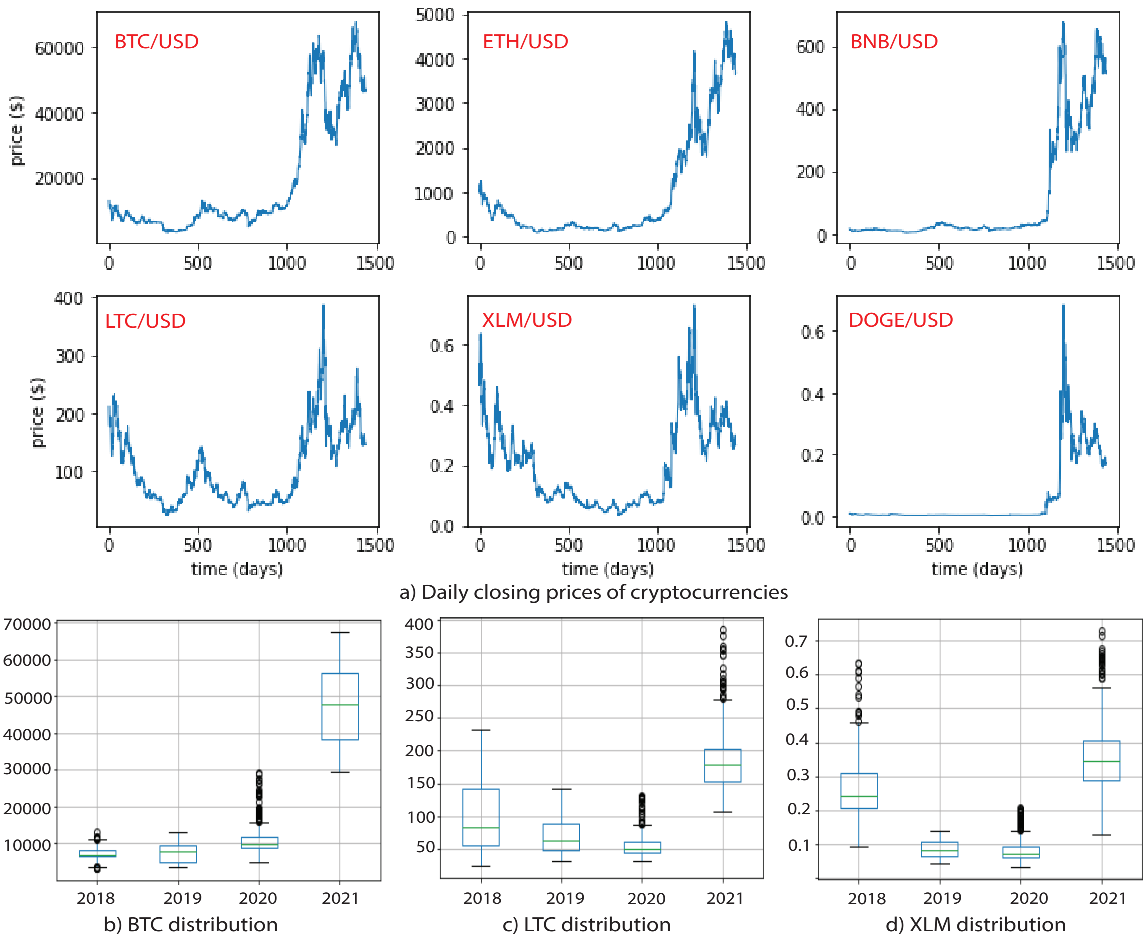

Fig. 1. Cryptocurrency trends and distributions

Table 2. Dataset description- Yahoo Finance

		Open	High	Low	Volume	WP	SMA3	SMA10	SMA20	EMA10	Price
BTC	min	3236.3	3275.4	3191.3	2.9e+9	8.9	3244.0	3397.0	3524.7	3425.1	3236.8
	mean	18408.8	16054.0	15160.1	2.6e+10	50.5	18405.3	18317.8	18199.8	18319.5	18429.6
	max	67549.7	68789.6	66382.1	3.5e+11	185.1	66511.3	64698.9	63149.3	64411.4	67566.8
LTC	min	23.5	23.8	22.8	1.9e+8	0.1	23.6	24.6	27.4	25.3	23.5
	mean	102.7	106.6	98.3	2.8e+9	0.2	102.6	102.8	103.1	102.8	96.34
	max	387.9	413.0	345.3	1.7e+10	1.1	374.4	347.2	316.0	342.2	386.5
ETH	min	84.3	85.3	82.8	9.5e+8	0.2	84.7	89.2	97.3	90.5	84.3
	mean	933.5	965.8	897.0	1.3e+10	2.6	933. 2	926.7	917.0	926.5	935.1
	Max	4174.6	4891.7	4718.0	8.4e+10	13.2	4727.8	4655.8	4531.5	4621.4	4812.1
BNB	min	4.5	4.6	4.2	9.3e+3	0.0	4.6	4.7	5.0	4.8	4.5
	mean	108.6	113.0	103.9	8.8e+8	0.3	108.5	107.3	105.5	107.3	108.9
	max	676.3	690.9	634.5	1.7e+10	1.9	655.3	643.1	614.7	637.0	675.7
DOGE	min	0.0	0.0	0.0	2.1e+6	0.0	0.0	0.0	0.0	0.0	0.0
	mean	0.1	0.1	0.1	1.0e+9	0.0	0.1	0.1	0.1	0.1	0.1
	max	0.7	0.7	0.6	6.9e+10	0.0	0.6	0.6	0.5	0.5	0.7
XLM	min	0.0	0.0	0.0	1.9e+7	0.0	0.0	0.0	0.0	0.0	0.0
	mean	0.2	0.2	0.2	5.1e+8	0.0	0.2	0.2	0.2	0.2	0.2
	max	0.7	0.8	0.7	1.0e+10	0.0	0.7	0.7	0.6	0.7	0.7

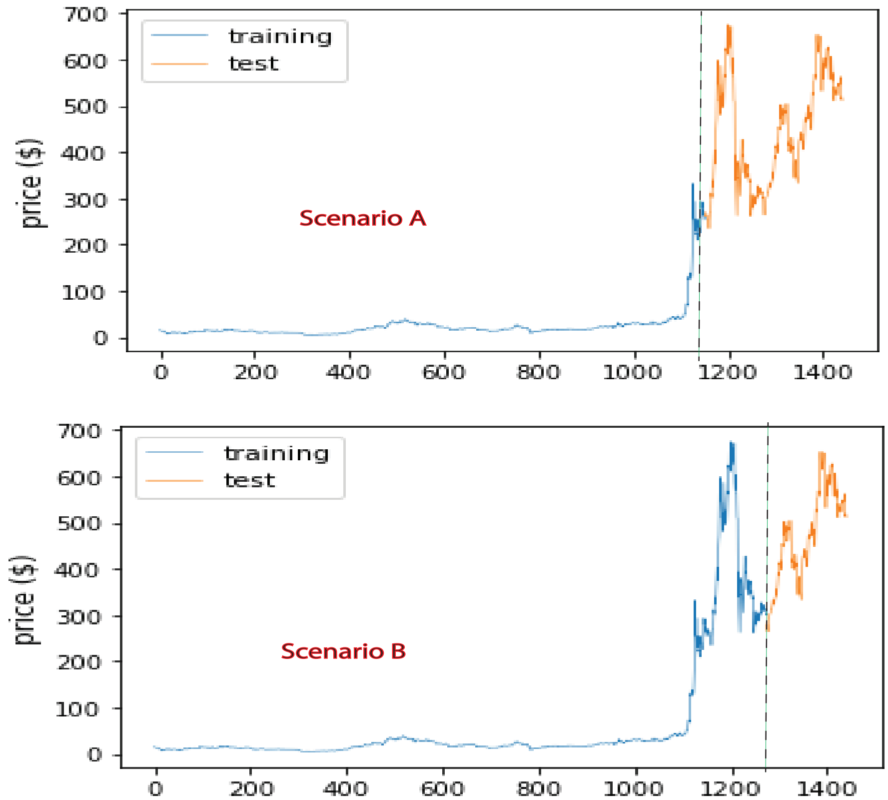

Fig 2. Illustrating the two scenarios (A and B) with respect to training data sizes

3.2. Boosted tree-based technique

This technique represents ensembles of multiple weak trees to improve robustness over a single predictive model. The three boosted tree-based techniques considered are briefly described.

3.2.1. Adaptive boosting

AdaBoost or ADAB is generally less susceptible to overfitting problems and works by fitting a series of weak learners (i.e., decision trees) on repeatedly modified versions of data. Predictions from the weak learners are then combined through a weighted majority vote to produce the final prediction. The data modifications at each boosting iteration apply weights $w_1, w_2, \ldots, w_N$ to training samples, with initial weights at $w_j=1/N$. Thus, the first step merely trains a weak learner on the original data. Then, the sample weights are individually modified for each successive iteration, and the learning algorithm is reapplied to the reweighted data. At a given step, the training examples with incorrect predictions at the previous step will have their weights increased. In contrast, those predicted correctly will have their weights decreased. As iterations progress, the difficult to predict examples will receive more attention. Thus, each subsequent weak learner is forced to concentrate on examples previously missed in the sequence (Hastie et al., 2009). Mathematically, given a training set with m samples. Let t(j) be the actual cryptocurrency price of sample j for j=1, 2, ⋯, k. ADAB generates L sub-regressors (l_p) and trains each regressor on a sampled sub-dataset D_p, p=1, 2, ..., L of the same size as the original training set. For each regressor l_p, the normalized estimation error for sample j, j=1,2, ⋯, k, is denoted as $e_p^j = \frac{|t^j - l_p(x^j)|}{max_{j=1}^{k}|t^j - l_p(x^j)|}$, and the estimation error of l_p computed using $\beta_p = \sum_{j=1}^{k} \omega_p^j e_p^j$. Then, the weight of sample j is updated as

$$\omega_p^j = \frac{\omega_{p-1}^j}{Z_{p-1}} \left(\frac{\beta_{p-1}}{1-\beta_{p-1}} \right)^{1-e_{p-1}^j} \qquad (1)$$

Where Z_{p-1} is a normalizing constant, intuitively, by Equation (1), the samples with a significant estimation error in the last iteration are assigned a significant sampling weight in the next iteration. Thus, during the training process, ADAB reduces estimation errors by paying attention to samples that are difficult to predict accurately. The final trained AdaBoost regressor is a weighted regressor l(x) overall L sub-regressors defined as $l(x) = \sum_{p=1}^{L} ln\left(\frac{1-\beta_p}{\beta_p}\right) l_p(x)$, where the weight $ln\left(\left(1-\beta_p/\beta_p\right)\right)$of regressor l_p decreases with estimation errors, β_p, i.e., regressors with smaller estimation errors contribute more to the final regressor l(x). |The genetic algorithm (Algorithm 1) was used to tune three ADAB parameters: the number of estimators (*n_estimators*), loss, and the learning rate (*learning_rate*). A slower learning rate takes much time and has more probability of converging or being stuck in an undesirable local minimum. At the same time, a higher one makes the learning jump over minima. Thus, this study considers the range (1.0 to 1.3) for the learning rate since this range was observed to be more appropriate during validation. The *number of estimators* is another parameter affecting the model's accuracy, as a larger value may lead to overfitting; hence, the hyperparameter sample space was limited to between 65 to 75. Additionally, the *"loss*" parameter was set to 'square' since it gave the best results during validation. Finally, the optimal configuration for ADAB was derived by computing the average value of these parameters for the repeated trials (6) representing the number of cryptocurrencies as depicted in Algorithm 1.

3.2.2. Gradient Boosting Machines

GBMs (Friedman, 2001) derive a strong learner by combining an ensemble of weak learners (i.e., decision trees) interactively. For a training dataset, S defined as $S = \{x_j, y_j\}_1^n$, the goal of the algorithm is to approximate function F∗(x) to give F^(x), i.e., mapping instances of x to output values y by minimizing the expected loss function, L (y, F(x)). Thus, the algorithm builds an additive approximation of F∗(x) as a weighted sum of functions defined as $F_k(x) = F_{k-1}(x) + \omega_k h_k(x)$where ω_k represents the weight of the kth function, $h_k(x)$. In constructing the approximation, a constant approximation of F ∗(x) is first derived as

$$F_0(x) = \underset{\alpha}{argmin} \sum_{j=1}^{n} L(y_j, \alpha) \qquad (2)$$

With subsequent models minimizing

$$\omega_k h_k(x) = \underset{\omega, h}{argmin} \sum_{j=1}^{n} L\left(y_j, F_{k-1}(x_j) + \omega h(x_j)\right) \qquad (3)$$

Where each model, h_k, is seen as a greedy step in a gradient descent optimization for F∗, and each h_k is trained on a new dataset $S = \{x_j \gamma_{kj}\}_{j=1}^n$with residuals, γ_{kj}, derived as

$$\gamma_{kj} = \left\{ \frac{\partial L\left(y_j F(x)\right)}{\partial F(x)} \right\}_{F(x)=F_{k-1}(x)} \qquad (4)$$

The value of ω_k is consequently computed by solving a linear search optimization problem, which suffers from overfitting if the iterative process is not correctly regularized (Friedman, 2001). Nevertheless, when controlling the additive process of the gradient boosting algorithm, several regularization parameters are often considered. One way to regularize the algorithm is to apply a shrinkage factor, ϑ, to reduce each gradient descent step $F_k(x) = F_{k-1}(x) + \vartheta \omega_k h_k(x), \vartheta \in [0,1.0]$. Also, regularization can be achieved by limiting the complexity of the trained models, i.e., by limiting the depth of the trees or the minimum number of instances necessary for node splitting.

The genetic algorithm was used to tune three GBM parameters: the number of estimators (*n_estimators*), tree depth (max_depth), and the learning rate (*learning_rate*). Boosting may potentially overfit when large estimators are used; hence, this range was limited to between 65 to 75, a very conservative value compared to examples provided in the literature (Hastie et al., 2009). The tree depths between 3 and 8 are known to give the best results (Hastie et al., 2009). Moreover, stumps with only one split allow for no variable interaction effects. Thus, a tree depth range of between 3 to 6 was used to allow a reasonable interaction. Also, the learning rate (α) represents the speed of learning achieved by the model. This study considers $0.80 \leq \alpha \leq 1.20$, due to the small number of trees used and to derive a computationally feasible model. Using a genetic algorithm (GA) specified in Algorithm 1, a reasonable number of estimators (70), tree depth (4), and learning rate (0.999) were similarly derived (Table 3).

3.2.3. Extreme Gradient Boosting

The XGB (Chen & Guestrin, 2016) represents an ensemble tree model utilizing the gradient boosting framework designed to be highly scalable and improve gradient boosting. This algorithm also exhibits better capability and higher computation efficiency when dealing with overfitting. XGB constructs an additive expansion of the objective function by decreasing a variation of the loss function, L', used to control the complexity of the trees and defined as Equation (5):

$$L' = \sum_{j=1}^{n} L\left(y_j, F(x_j)\right) + \sum_{k=1}^{m} \theta(h_k) \quad (5)$$

Where $\theta(h) = \gamma Z + {}^{1}/_{2}\,\lambda\|\eta\|^2$, Z represents the number of leaves in the tree, and η represents the output scores of leaves. This loss function is incorporated into the split criterion of decision trees leading to a pre-pruning procedure. The value of γ controls the minimum loss reduction gain required to split the internal node; however, higher values of γ result in simpler trees. An additional regularization parameter, known as shrinkage, λ, can be employed to reduce step size in the additive expansion. Also, the complexity of trees can be controlled using other approaches, i.e., the depth of the trees. Tree complexity reduction ensures models are trained faster with less storage space requirement. Furthermore, randomization techniques (random subsamples and column subsampling) are available to reduce overfitting and training speed. Also, three XGB parameters: the learning rate, number of estimators (n_estimators), and the maximum depth of the tree (*max_depth*), were tuned using the GA method while setting other parameters at their default values. The optimal configurations (Table 3) obtained after averaging the best configurations from the six cryptocurrencies are learning_rate (1.135), n_estimators (63), and max_depth (5).

3.3. Deep Learning techniques

These are a new branch of ML techniques that have gained widespread recognition and are successfully deployed in various applications. A brief discussion on DL architectures considered is as follows.

3.3.1. Deep feedforward neural networks

The deep feedforward neural network (DFNN) is the typical DL model for hierarchically learning complex and abstract data representations. This learning process transmits data through multiple transformation layers (Ajayi et al., 2020). The typical architecture of DFNN has three layers, namely, input layer, hidden layer, and output layer, in which each layer has several interconnected processing units. In DFNN, each layer utilizes a nonlinear transformation on its input to produce its output. The neural network is assumed to consist of *N* layers. The output signal of the l^{th} layer is expressed as in Equation (6).

$$d_j^l = f\left(w_j^T a_j^{l-1} + b_j\right), \quad l = 1,2,3,\dots,N \quad (6)$$

Where f is the activation function; w_j^T is the weight vector which indicates the effect of all units in the same hidden layer; a_j^{l-1} defines the output signal of the $l-1^{th}$ layer; b_j represents the bias parameter of the j^{th} unit in the l^{th} layer.

Since this problem is a regression problem, the rectified linear unit (ReLU) was selected as the activation function in the DFNN architecture to transfer input signals to the output because it is computationally efficient and will ensure better performance of models. Also, ReLU is nearly linear and easy to optimize with gradient-based methods. In addition, it can learn faster with networks consisting of many layers, thus allowing the training of deep neural networks without an unsupervised pre-training (Glorot et al., 2011). Mathematically, ReLU is expressed as $f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$. Furthermore, the Mean square error (MSE) was used to evaluate the model's prediction accuracy while training the DFNN model. MSE is normally expressed as $MSE = \frac{1}{n}\sum_{j=1}^{n}\left(o_j - y_j\right)^2$,where n is the number of samples in a training set; o_j represent the measured values and yj represent predicted values. In constructing the DFNN model, the number of hidden layers was first derived. Generally, increasing the number of hidden layers imply longer computational time and larger storage of training parameters. However, considering the datasets and computational cost, it was observed that a neuron network architecture with only two hidden layers could reasonably model the cryptocurrency problem in this study. Therefore, the architecture consisted of an input layer, two hidden layers, and an output layer. The Root Mean Square Propagation (RMSprop) optimizer was adopted for the MSE loss minimization since its combination with ReLU attained the lowest training MSE at 100 epochs. In addition, the RMSProp optimizer made the entire network converge faster. Also, the dropout method (Srivastava et al., 2014) was used to deal with the overfitting problem. The drop rate used is 0.1%. Choosing an optimal number of neurons for each hidden layer is critical to the performance of a neural network. Thus, the GA method was used to tune the parameters: the number of neurons in each Dense layer (two layers were used), the number of epochs, and the training batch size (Table 3), to derive an optimal model configuration for the cryptocurrencies.

3.3.2. Gated recurrent units

The gated recurrent units (GRUs) by Cho et al. (2014) use gates to control information flow, and they are introduced to solve the vanishing gradient problem with the standard RNNs. Though GRU is similar to LSTM, however GRU network has an update gate that combines the forget and input gates of LSTM into a single update gate. In addition, the cell state and the hidden state are further merged in GRU, thus, making its structure simpler, more efficient in the training phase, and, in general, train faster than LSTM. Furthermore, GRUs are known to outperform LSTMs in tasks with a limited number of data instances. The linking of the writes and forget gates in the GRU update gate imposes a restraint on the cell state to coordinate the writes and forgets. Alternatively, rather than doing selective writes and selective forgets, a selective overwrites, i.e., setting the forget gate equal to 1 minus the write gate, is done using Equation (7):

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h_t} \quad (7)$$

Where z_t denotes the update gate, and h_t represents the memory content. An element-wise multiplication, $\odot$, is applied to $(1-z_t)$ and the preceding memory content h_{t-1} in (6), followed by an element-wise multiplication on z_t and the current memory content $\widetilde{h_t}$, thus resulting in a summation of two element-wise multiplications. Usually, the GRU unit structure consists of the update gate (z_t), reset gate (r_t), and the current memory content ($\widetilde{h_t}$). These gates permit the storage of values in the GRU unit memory for a certain amount of time and then carry these values forward, when required, to the current state to update at a future date. The update gate multiplies and adds the input x_t and the output from the previous unit h_{t-1} and is used to tackle the vanishing gradient problem when training models. A sigmoid function is used to obtain outputs between 0 and 1. The reset gate regulates how much of the past information to disregard. The current memory content is where x_t is multiplied by W and r_t is multiplied by h_{t-1} elementwise, with a tanh activation function applied to the final summation. The final GRU unit memory, h_t, holds the information for the current unit, which is passed on to the network. The GRU architecture consists of a single layer of GRU unit driven by the input sequence and the activation function, set as ReLU. Also, RMSprop was used to optimize the training and GA (Algorithm 1) to tune its parameters (number of neurons in GRU and Dense layers, epochs, and training batch size). As a result, the optimal values (Table 3) obtained are the number of neurons (13 each) for GRU and Dense layers, epochs (80), and training batch size (43), respectively.

3.3.3. Convolutional neural networks (1-D)

CNNs typically consist of a set of successive convolutional and subsampling layers, one or more hidden layers, and an output layer. The first two types of layers are combined to extract high-level feature vectors in one dimension. The feature vectors are later handled by the fully connected multilayer perceptron and output layers. Also, an activation function is usually applied to the resulting field following the convolution operation. The ReLU activation function is computationally efficient for CNNs (Dahl et al., 2013), in addition, it is favored because it preserves the magnitude of positive signals as they travel forward and backward through the network (LeCun et al., 2015). Finally, convolution filters are applied across all inputs simultaneously, which allows them to identify correlated patterns across multiple input variables or the results of previous convolutions. The advantage of CNN is that the training is relatively easy because its number of weights is less than that of a fully connected architecture, thus, facilitating the easy extraction of essential features. Formally, the 1D forward propagation from convolution layer l-1 to the input of a neuron in layer l is expressed as in Equation (8):

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} conv1D\left(w_{ik}^{l-1}, s_i^{l-1}\right) \qquad (8)$$

Where the scalar bias of the k^th^ neuron b_k^l, the output of the i^th^ neuron at layer l-1 s_i^{l-1}, and the kernel from the i^th^ neuron at layer l-1 to the k^th^ neuron at layer l w_{ik}^{l-1} are used to determine the input x_k^lat layer l. Also, the conv1D (.,.) function represents a 1-D convolution without zero padding on the boundaries. Finally, the intermediate output of the neuron, y_k^l, which is a function of the input, x_k^l, and the output of the neuron s_k^lat layer l (a subsampled version of y_k^l) is as defined in Equation (9):

$$y_k^l = f\left(x_k^l\right) \; and \; s_k^l = y_k^l \downarrow ss \qquad (9)$$

Where s_k^lstands for the output of the k^{th} neuron of the layer, *l*, and "$\downarrow ss$" represents the down-sampling operation with a scalar factor, ss. In achieving the utmost computational efficiency, the study adopts a simple 1-D CNN with only one CNN layer and one MLP layer. Moreover, most recent studies employing 1D CNN applications use compact (with 1–2 hidden CNN layers) configurations. Also, since CNN models learn very quickly, a dropout layer (Srivastava et al., 2014) was used to help slow down the learning process and facilitate better generalization. Furthermore, ReLU, a computationally efficient activation function for CNNs (Dahl et al., 2013), and RMSprop were used in learning optimization. Finally, four parameters: the number of filers, the number of neurons in the dense layer, the number of epochs, and the batch size for the six cryptocurrencies, were tuned using Algorithm 1. The optimal values of these parameters are filters (10), units (13), epochs (82), and batch size (52).

3.4. Genetic algorithms

Genetic algorithms (GA) (Goldberg, 2006) provide the opportunity to randomly search the hyper-parameter space while utilizing the previous results to direct the search. Each hyperparameter to optimize is encoded as a single gene for each individual. A range is then defined for each gene to eliminate searching for disinterested areas in the hyperparameter space. Initially, the population is generated by selecting each gene from a uniform random distribution, then each individual's fitness is evaluated. Each generation is then formed using selection, crossover, and mutation predicated on individuals having the highest fitness scores from the previous generation. This procedure represents a single generation of random search followed by a result-driven search based on the best previous individuals. A selection operation is performed by removing individuals from the population with a fitness value smaller than their generation's average fitness. Then, the next generation is created by performing the crossover and mutation operations on the remaining individuals. The *GASearchCV* function in Python's sklearn-genetic-opt is used to optimize the hyperparameters by minimizing the RMSE of the prediction models. The algorithm used to derive the optimal hyperparameters for the deep learning and tree-based methods is depicted in Algorithm 1, and the function code *genetic_parameters_tune* defined in the Code Ocean platform https://codeocean.com/capsule/0499275/tree/v1

After deriving the best parameters of the models on each cryptocurrency dataset, the average value of these parameters was then determined as the optimal model configuration. Table 3 presents the optimal configurations of predictive models' hyperparameters for cryptocurrencies considered in this study. Some important *GASearchCV* arguments used are:

1) population: This represents the initial amount of hyperparameters candidates to generate randomly, thus was set to 10 in this study.
2) generations: The argument represents the number of iterations the algorithm will make and creates a new population every generation. It was set to 5 in this study.
3) crossover_probability: The probability that a crossover occurs in a particular mating. A crossover probability of 0.9 was used in this study.
4) mutation_probability: The probability that an already fixed individual suffers a random change in some of its hyperparameters values. A mutation probability of 0.05 was used to limit the search radius for faster convergence.
5) param_grid: a dictionary with keys as names of hyperparameters and their values, i.e., a list of parameters for a typical GBM model can be expressed as:
param_grid = {'learning_rate': Continuous (0.8, 1.3),'max_depth': Integer (3, 6), 'n_estimators': Integer (65, 75)}

Table 3. Parameter bounds, optimal parameters for each cryptocurrency, and the average value for models

			Cryptocurrency						
Model	Parameter	Range	BTC	ETH	BNB	LTC	XLM	DOGE	Average
ADA	learning_rate	[1.00 – 1.30]	1.283	1.286	1.246	1.218	1.030	1.223	1.214
	n_estimators	[65-75]	65	73	70	74	75	75	72
GBM	loss	Square	-	-	-	-	-	-	
	max_depth	[3-6]	3	3	6	4	3	6	4
	learning_rate	[0.80 - 1.30]	1.192	1.001	0.868	1.048	1.016	0.864	0.999
	n_estimators	[65 - 75]	70	71	65	70	70	74	70
XGB	max_depth	[3 - 6]	6	6	4	6	3	5	5
	learning_rate	[0.80 - 1.30]	1.271	1.137	0.839	1.115	1.149	1.298	1.135
	n_estimators	[60 - 70]	63	60	67	63	62	65	63
MLP	units (each of the 2 layers)	[14 - 19]	15	18	14	15	15	17	16
	batch_size	[40 - 50]	43	41	43	47	50	47	45
	epochs	[80 - 90]	85	84	84	87	80	80	83
GRU	units (each of the 2 layers)	[11 - 14]	11	13	13	14	13	11	13
	batch_size	[42 - 45]	42	43	42	44	43	42	43
	epochs	[76 - 84]	80	84	76	76	79	82	80
CNN	filters	[9 - 12]	10	12	10	9	9	9	10
	units	[10 - 14]	10	13	14	13	14	12	13
	batch_size	[44 - 54]	54	52	51	49	54	50	52
	epochs	[75-85]	83	85	77	81	81	84	82

3.5. Performance evaluation

Consequently, to finally evaluate the performance of prediction models on testing datasets (yahoo finance; validating sets -UK Investing and Bitfinex), statistical analysis involving standard metrics is conducted to quantify the extent to which the predicted closing prices are close to the corresponding true values. These metrics are briefly described:

1) Nash-Sutcliffe coefficient of Efficiency (NSE) provides a more direct measure of the agreement between the observed closing price and predicted values, and it is expressed as in Equation (10).

$$NSE = 1 - \left[\frac{\sum_{j=1}^{n}\left(o_j - y_j\right)^2}{\sum_{j=1}^{n}\left(o_J - \bar{o}\right)^2}\right] \qquad (10)$$

Where y_j represents forecasts, o_j represents corresponding measured outputs, and $\bar{o}$ represents the mean of the measured output. A value of NSE closer to 1 implies that the model can satisfactorily reproduce the observed cryptocurrency closing price. NSE = 1.0 indicates a perfect match of the model predictions to the observed values.

ALGORITHM 1

```
INPUT: X1,6 // cryptocurrency datasets (BTC, ETH, BNB, LTC, XLM, DOGE)
OUTPUT: xgb_opt, gbm_opt, ada_opt, gru_opt, mlp_opt, cnn_opt // optimal parameter values

Xtrain1,6, Xtest1,6 =split_data (X1,6, proportions)
ytrain1,6, ytest1,6=split_data (y1,6, proportions)

# p1, p2, p3, etc represents parameters, i.e., max_depth, learning rate, filters, units, batch size, epochs,

#    Set bounds for parameters
#LBound – lower bound of a parameter, UBound – maximum value of a parameter
xgb_grid = {'p1': LBound, UBound,'p2': (LBound, UBound),'p3': LBound, UBound }
gbm_grid = {'p1': LBound, UBound,'p2': (LBound, UBound),'p3': LBound, UBound }
ada_grid = {'p1': LBound, UBound,'p2': (LBound, UBound),'p3': LBound, UBound }
gru_grid = {'p1': LBound, UBound,'p2': (LBound, UBound),'p3': LBound, UBound }
mlp_grid = {'p1': LBound, UBound,'p2': (LBound, UBound),'p3': LBound, UBound }
cnn_grid = {'p1': LBound, UBound,'p2': (LBound, UBound),'p3': LBound, UBound,'p4': LBound, UBound,}

# Set up the genetic solver for each of the predictive model, i.e., XGB, GBM, ADA, GRU, MLP & CNN
xgb_ = GASearchCV (solver =xgb, cv=3, scoring='mse', population=10, generations=5, param_grid=xgb_grid, ….,)
gbm_ = GASearchCV (solver =gbm, cv=3, scoring='mse', population=10, generations=5, param_grid=gbm_grid, ….,)
ada_ = GASearchCV (solver =ada, cv=3, scoring='mse', population=10, generations=5, param_grid=ada_grid, ….,)
gru_ = GASearchCV (solver =gru, cv=3, scoring='mse', population=10, generations=5, param_grid=gru_grid, ….,)
mlp_ = GASearchCV (solver =mlp, cv=3, scoring='mse', population=10, generations=5, param_grid=mlp_grid, ….,)
cnn_ = GASearchCV (solver =cnn, cv=3, scoring='mse', population=10, generations=5, param_grid=cnn_grid, ….,)

# Fit the generic solver on the data to find optimum parameters
FOR i=1 TO 6 DO
      xgb_.fit (Xtrain[i], ytrain[i])
      gbm_.fit (Xtrain[i], ytrain[i])
      ada_.fit (Xtrain[i], ytrain[i])
      gru_.fit (Xtrain[i], ytrain[i])
      mlp_.fit (Xtrain[i], ytrain[i])
      cnn_.fit (Xtrain[i], ytrain[i])

     # Store the best parameters used to fit models for each dataset Xtrain
     xgb_dict[(xgb_.best['p1'],xgb_.best['p2'],xgb_.best['p3'])] = {mse(ytest[i], xgb_.predict(Xtest[i])}   # append dictionary
     gbm_dict[(gbm_.best['p1'],gbm_.best['p2'],gbm_.best['p3'])]= {mse(ytest[i], gbm_.predict(Xtest[i])}
     ada_dict[(ada_.best['p1'],ada_.best['p2'],ada_.best['p3'])]= {mse(ytest[i], ada_.predict(Xtest[i])}
     gru_dict[(gru_.best['p1'],gru_.best['p2'],gru_.best['p3'])] = {mse(ytest[i], gru_.predict(Xtest[i])}
     mlp_dict[(mlp_.best['p1'],mlp_.best['p2'],mlp_.best['p3'])] = {mse(ytest[i], mlp_.predict(Xtest[i])}
     cnn_dict[(cnn_.best['p1'],cnn_.best['p2'],cnn_.best['p3'],cnn_.best['p4'])] = {mse(ytest[i], cnn_.predict(Xtest [i])}
END DO

# Compute average values of these parameters as the optimal parameter configurations of models for all datasets
xgb_opt= mean(DataFrame(xgb_dict)) # three parameters tuned
gbm_opt= mean(DataFrame(gbm_dict)) # three parameters tuned
ada_opt= mean(DataFrame(ada_dict)) # compute the means of two numerical parameters tuned
gru_opt=  mean(DataFrame(gru_dict)) # Averages of three parameters tuned
mlp_opt= mean(DataFrame(mlp_dict)) # Averages of the three parameters tuned
cnn_opt= mean(DataFrame(cnn_dict)) # Averages of four tuned parameters
```

2) Explained Variance Score (EVS) compares the variance within the expected outcomes to the variance in the model error. This metric essentially represents the amount of variation (dispersion) in the original dataset that a model can explain, and it is estimated as follows.

$$EVS(o, y) = 1 - \frac{var(o - y)}{var(o)} \tag{11}$$

Where y is the estimated target output, o represents the corresponding target output, and var is the variance (i.e., the square of the standard deviation). The best possible score is 1.0, and lower values are worse for prediction models.

3) The t-test illustrates the overestimation or underestimation of the data at a 95% significance level. The t-test is calculated (Equation 12) as the ratio of SS_1 and SS_2

$$t = \frac{SS_1}{SS_2} \qquad (12)$$

where $SS_1 = \frac{\sum_{j=1}^{n}(o_j - y_j)}{n}$, is the average of the differences between the measured, o_j, and the estimated, y_j, cryptocurrency price values, and $SS_2 = \sqrt{\frac{\sum_{j=1}^{n}|(o_j - y_j) - SS_1|^2}{n-1}}$. Where a population of $n > 120$ and the absolute value of $t <= 1.96$, there is no statistically significant difference between the observed and calculated data at a 95% confidence level. Values of t close to zero indicate a higher accuracy. For a positive t-test value, the measured value is not statistically greater than the estimated one (at 95% confidence level). Conversely, for a negative t-test value, the calculated value is not significantly greater than the measured value at the confidence level of 95%.

4) The Mean Absolute Percentage Error (MAPE) measure determines the percentage of error per prediction and is defined in Equation (13):

$$MAPE = \frac{1}{n} * \sum_{j=1}^{n} \left| \frac{y_j - o_j}{o_j} \right| * 100 \qquad (13)$$

The smaller the MAPE, the better the performance of the model. MAPE is relevant in finance as gains and losses are often measured in relative values. In addition, it is valuable for calibrating products' prices since customers are sometimes more sensitive to relative variations than absolute variations.

To further investigate the performance of the prediction models, graphical techniques were employed to determine the degree of agreement between forecasts and measured closing price values. Also, the graphical methods facilitate qualitative and subjective evaluation. Finally, all the models were developed using the Keras high-level DL library and TensorFlow as the low-level backend. All experimental work was carried out on a personal computer (2.9 GHz 6- Core Intel with 32 GB of RAM and a hard disk memory of 1TB). Also, sample outputs presented in the following sections can also be simulated by interested readers using the source code available on Code Ocean (https://doi.org/10.24433/CO.2359079.v1).

4. Results and discussion

The performance evaluation of models using statistical indicators: NSE, EVS, t-test, and MAPE on each cryptocurrency for the two scenarios is summarized in Tables 4 and 5. The results (Table 4A: Scenario A) indicate that the average EVS for the six cryptocurrencies ranges between 0.60 and 0.94. However, DL techniques obtained a more significant percentage (between 88% to 94% on average), indicating that they have accounted for the total variance in the observed data. This result contrasts those from boosted tree-based models having average EVS values ranging from 0.60 to 0.62, thus, struggling in their predictions, especially for ETH, BNB, and DOGE, due to insights not captured in training sets. Similarly, in comparing the robustness of models with different data sources, DL models, especially CNN and GRU, produce consistent and higher EVS values ($0.76 \leq EVS \leq 0.99$) compared to boosted tree-based models, which are extremely low in some cases ($0.03 \leq EVS \leq 0.50$) for ETH and DOGE. Thus, boosted tree-based models exhibit unreliable predictions for these cryptocurrencies when certain information is missing from the training set. Nonetheless, in Table 4B (Scenario B), there is an improvement in predictions from most models as the EVS for predictions ranges between 0.75 and 1. A high value of EVS indicates more significant similarities between the measured and predicted values. A perfect model has EVS = 1. Thus, predictions (on average) from CNN and GRU models (Scenarios A and B) for most cryptocurrencies are considered ideal since $0.95 \leq EVS \leq 0.96$ and acceptable

$0.91 \leq EVS < 0.93$ (for DFNN). However, predictions (on average) from the boosted tree-based models (Scenarios A and B) can be considered very good ($0.78 \leq EVS \leq 0.80$).

The MAPE metric quantifies how close the models' predictions are to the actual (closing price) values. The smaller the MAPE value (closer to zero), the closer the predictions are to the true values and the better the predictive models' performance. As shown in Tables 4 and 5, the predictive models yield smaller MAPE values (Scenario A). For example, the CNN model has the smallest average MAPE of 9%, followed by GRU with an average MAPE of 11%, GBM (average MAPE of 17%), DFNN (average MAPE of 18%), XGB (average MAPE of 18%), and ADAB (average MAPE of 18%). However, the MAPE indexes of the closing prices estimated by the models for all cryptocurrencies on the three testing sets are within 3% to 12% (Table 4: Scenario B), implying a high prediction accuracy of models. In addition, all the predictive models obtain the absolute (t-test) value, $t<=1.96$, for all cryptocurrencies, except DFNN ($1.99 \leq$ t-test ≤ 2.02) for DOGE (Scenario A). Thus, for most predictive models, there is no statistically significant difference between the observed and predicted closing price at a 95% confidence level, as depicted in Table 4. Also**,** the predictive models' fit expressed as NSE obtained in Scenario A ranged from -3.09 to 0.99 (Boosted tree-based models) and -1.35 to 0.99 (DL techniques). Similarly, NSE (Scenario B) ranged from 0.16 to 1.00 (Boosted tree-based techniques) and -0.37 to 0.99 (DL techniques). The NSE higher values indicate better model performance. Thus, the mean Nash–Sutcliffe coefficient obtained by combining both scenarios for all models ranges between 0.45 and 0.88 (Table 5), with the highest mean Nash–Sutcliffe coefficient value obtained by CNN (0.88) followed by GRU (0.85). The least overall mean of the Nash–Sutcliffe coefficient was obtained by ADAB (0.45).

Furthermore, as a means of visual inspection, the fitness of the prediction models was evaluated using residual plots to examine the prediction bias of models (Scenario A) on the UK Investing datasets. For the DL techniques, most cryptocurrencies have residuals randomly distributed around the zero horizontal lines, except CNN (BTC and DOGE), DFNN (DOGE), and GRU (BTC, ETH, and BNB). Thus, the residuals (Fig. 3), for most cryptocurrencies, exhibited no defined patterns and satisfied the assumption that the residuals have a constant variance. However, residuals from the boosted tree-based counterparts (Fig. 4) were neither symmetric to the origin nor randomly distributed. Hence, their inability to learn from limited examples and generalize some degree of knowledge in predicting closing prices. Figs. 5-8 further demonstrate the daily variations of the observed and predicted closing prices for the UK investing and Yahoo finance datasets (Scenarios A and B). As shown in Fig. 5. (Scenario A), the DL-based models' predictions correspond well with the observed values, i.e., the overall trend or pattern is entirely consistent, showing a good correlation, especially for CNN models. Thus, CNN produces more accurate and robust results for the different cryptocurrency datasets. However, by outputting high error margins, the boosted tree-based models underfit some cryptocurrency datasets (i.e., ETH, BNB, and DOGE). Thus, they do not present a representative picture of the relationship between predictions and measured values for these cryptocurrencies. Consequently, the results confirm the boosted trees-based limitations when vital information is missing from training sets. However, all six predictors performed well and produced more accurate results on the validation datasets as more training data, capturing peaks and drops in prices, were used. This realistic prediction performance is captured in Table 4: Scenario B (all validation sets- Yahoo finance, UK investing, and Bitfinex) and Figs 7 and 8 for the UK investing datasets.

The performance of DL and boosted tree-based models is also graphically evaluated using Taylor's diagram (Fig. 9 and 10 - Scenario A). The diagram graphically displays a statistical summary of how well the predictions from the models correspond to the observed values in terms of their correlation coefficient, center RMSE, and standard deviation. From Fig. 9, the position of colored numbers (i.e., 1, 2, 3, 4, 5 and 6) quantifies how close the predictions from the models (i.e., ADAB, GBM, XGB, GRU, DFNN, and CNN) are to the observed closing prices for each cryptocurrency. The red dotted arc in the diagram represents the observed standard deviation at the point marked ***"observed"*** on the x-axis. Predictions from boosted trees are farther from the point marked "observed" for ETH-USD, BNB-USD, LTC-USD, and DOGE-USD, compared to predictions from the DL techniques.

Table 4. Statistical performance of prediction models

Scenario A (Table 4A)

Model	Statistical Index	BTC			ETH			BNB			LTC			XLM			DOGE			Mean
		Data1	Data2	Data3	Data1	Data2	Data3	Data1	Data2	Data3	Data1	Data2	Data3	Data1	Data2	Data3	Data1	Data2	Data3	
XGB	EVS	0.97	0.98	0.99	0.03	0.50	0.50	0.08	0.53	0.55	0.79	0.92	0.92	0.97	0.98	0.98	0.03	0.10	0.10	0.61
	MAPE	1%	2%	2%	25%	17%	17%	24%	21%	20%	2%	2%	2%	2%	3%	3%	65%	56%	56%	18%
	t-test	0.00	-0.04	-0.08	1.43	0.88	0.89	1.14	0.88	0.93	0.32	0.21	0.21	0.18	-0.06	-0.12	1.79	1.27	1.27	0.62
	NSE	0.97	0.98	0.99	-1.96	0.11	0.09	-1.11	0.16	0.16	0.77	0.92	0.92	0.97	0.98	0.98	-3.09	-1.37	-1.37	0.01
GBM	EVS	0.97	0.95	0.95	0.03	0.50	0.50	0.19	0.59	0.59	0.80	0.92	0.92	0.94	0.97	0.97	0.04	0.13	0.13	0.62
	MAPE	1%	3%	3%	24%	16%	16%	21%	18%	18%	2%	2%	1%	2%	3%	3%	65%	56%	56%	17%
	t-test	-0.07	0.05	-0.01	1.38	0.85	0.86	1.13	0.88	0.88	0.31	0.20	0.20	0.39	0.08	-0.01	1.78	1.30	1.30	0.64
	NSE	0.97	0.95	0.95	-1.83	0.14	0.14	-0.85	0.27	0.27	0.78	0.92	0.92	0.93	0.97	0.97	-3.02	-1.34	-1.34	0.04
ADAB	EVS	0.97	0.99	0.99	0.03	0.49	0.49	0.06	0.47	0.47	0.76	0.91	0.91	0.98	0.99	0.99	0.04	0.13	0.13	0.60
	MAPE	1%	1%	1%	25%	16%	16%	28%	23%	23%	2%	1%	1%	1%	2%	2%	64%	55%	55%	18%
	t-test	-0.03	0.06	0.08	1.42	0.85	0.86	1.36	1.00	1.00	0.34	0.23	0.23	0.15	0.06	-0.01	1.77	1.29	1.29	0.66
	NSE	0.97	0.99	0.99	-1.93	0.11	0.11	-1.70	-0.07	-0.07	0.74	0.91	0.91	0.98	0.99	0.99	-2.98	-1.32	-1.32	-0.04
GRU	EVS	0.98	0.98	0.98	0.76	0.89	0.89	0.81	0.84	0.86	0.99	0.99	0.99	0.98	0.98	0.98	0.91	0.91	0.91	0.92
	MAPE	4%	2%	2%	14%	12%	12%	20%	29%	28%	1%	3%	2%	2%	4%	4%	6%	24%	24%	11%
	t-test	-2.53	-0.5	-0.5	1.57	1.19	1.19	2.01	1.90	2.10	-0.42	0.54	0.57	-0.49	-0.93	-0.78	0.49	1.28	1.28	0.44
	NSE	0.87	0.98	0.98	0.17	0.73	0.73	0.03	0.26	0.27	0.99	0.98	0.98	0.98	0.96	0.96	0.89	0.75	0.75	0.74
DFFN	EVS	0.99	0.98	0.99	0.99	0.98	0.99	0.83	0.68	0.70	0.98	0.95	0.96	0.98	0.98	0.98	0.75	0.54	0.54	0.88
	MAPE	1%	2%	2%	3%	8%	5%	9%	42%	43%	2%	5%	4%	3%	3%	3%	39%	79%	79%	18%
	t-test	0.46	0.38	0.4	1.38	1.29	0.98	0.99	2.07	2.13	0.53	0.73	0.60	1.00	0.17	0.18	1.99	2.02	2.02	1.07
	NSE	0.99	0.98	0.98	0.96	0.95	0.98	0.67	-0.69	-0.69	0.97	0.92	0.95	0.96	0.98	0.98	-0.25	-1.35	-1.35	0.44
CNN	EVS	**0.99**	**0.99**	**0.99**	**0.97**	**0.99**	**0.99**	**0.90**	**0.88**	**0.90**	**0.97**	**0.98**	**0.98**	**0.98**	**0.98**	**0.97**	**0.82**	**0.82**	**0.82**	**0.94**
	MAPE	3%	2%	2%	3%	3%	3%	6%	20%	19%	3%	4%	3%	3%	3%	3%	22%	34%	34%	9%
	t-test	-3.17	0.50	0.40	0.00	0.50	0.59	0.38	1.51	1.68	-0.88	-0.75	-0.6	1.06	0.09	0.18	1.31	1.46	1.46	0.32
	NSE	0.92	0.98	0.98	0.97	0.99	0.99	0.89	0.61	0.62	0.95	0.97	0.97	0.96	0.98	0.97	0.52	0.43	0.43	0.84

Scenario B (Table 4B)

Model	Statistical Index	BTC Data1	BTC Data2	BTC Data3	ETH Data1	ETH Data2	ETH Data3	BNB Data1	BNB Data2	BNB Data3	LTC Data1	LTC Data2	LTC Data3	XLM Data1	XLM Data2	XLM Data3	DOGE Data1	DOGE Data2	DOGE Data3	Mean
XGB	EVS	0.99	0.99	0.99	0.95	0.99	0.99	0.98	0.99	0.99	0.98	0.99	0.99	0.96	0.98	0.98	0.81	0.98	0.98	0.97
	MAPE	1%	1%	1%	3%	2%	2%	3%	3%	3%	1%	1%	1%	2%	3%	3%	8%	8%	8%	3%
	t-test	0.10	0.14	0.16	0.44	0.25	0.27	-0.17	-0.14	-0.12	-0.25	-0.05	-0.02	0.24	0.08	-0.03	-0.5	-0.44	-0.44	-0.03
	NSE	0.99	0.99	0.99	0.94	0.99	0.99	0.97	0.99	0.99	0.98	0.99	0.99	0.95	0.98	0.98	0.76	0.98	0.98	0.97
GBM	EVS	0.98	0.98	0.98	0.94	0.98	0.98	0.95	0.98	0.98	0.99	0.99	0.99	0.95	0.98	0.98	0.69	0.97	0.97	0.96
	MAPE	1%	2%	2%	3%	3%	3%	5%	5%	5%	1%	1%	1%	2%	3%	3%	10%	9%	9%	4%
	t-test	-0.34	-0.02	-0.02	-0.06	-0.11	-0.16	-0.32	-0.19	-0.19	0.2	-0.03	-0.09	0.16	-0.07	-0.14	-0.57	-0.46	-0.46	-0.16
	NSE	0.98	0.98	0.98	0.94	0.98	0.98	0.95	0.98	0.98	0.99	0.99	0.99	0.95	0.98	0.98	0.58	0.97	0.97	0.95
ADAB	EVS	0.99	0.99	0.99	0.96	0.99	0.99	0.99	1.00	1.00	0.99	1.00	1.00	0.96	0.99	0.99	0.61	0.95	0.95	0.96
	MAPE	1%	1%	1%	2%	2%	2%	1%	3%	3%	1%	1%	1%	1%	2%	2%	15%	16%	16%	4%
	t-test	0.08	0.05	0.06	0.34	0.29	0.31	-0.44	-0.36	-0.32	0.38	0.16	0.15	0.22	0.08	-0.02	-1.07	-0.73	-0.73	-0.09
	NSE	0.99	0.99	0.99	0.96	0.99	0.99	0.99	1.00	1.00	0.99	1.00	1.00	0.96	0.99	0.99	0.16	0.93	0.93	0.94
GRU	EVS	0.96	0.94	0.94	0.99	0.99	0.99	0.99	1.00	1.00	0.98	0.98	0.98	0.98	0.98	0.98	0.97	0.98	0.98	**0.98**
	MAPE	2%	5%	5%	3%	2%	2%	3%	3%	4%	1%	3%	2%	2%	3%	3%	0.06	8%	8%	4%
	t-test	0.79	0.93	0.97	-0.98	-0.37	-0.4	1.8	0.87	0.9	0.16	0.51	0.43	1.21	0.29	0.39	1.78	-0.2	-0.2	0.49
	NSE	0.93	0.89	0.89	0.98	0.99	0.99	0.97	0.99	0.99	0.98	0.97	0.98	0.94	0.98	0.98	0.87	0.98	0.98	0.96
DFFN	EVS	0.99	0.98	0.98	0.95	0.97	0.98	1.00	0.97	0.97	0.98	0.97	0.98	0.98	0.98	0.98	0.97	0.75	0.75	0.95
	MAPE	3%	4%	4%	6%	6%	6%	3%	14%	13%	2%	3%	2%	1%	3%	3%	16%	67%	67%	12%
	t-test	2.27	1.49	1.52	1.52	1.16	1.47	2.09	-2.23	-2.12	1.10	-0.22	-0.04	0.54	-0.33	-0.24	5.44	2.13	2.13	0.98
	NSE	0.92	0.95	0.95	0.83	0.94	0.94	0.98	0.84	0.85	0.95	0.97	0.98	0.97	0.98	0.98	0.19	-0.37	-0.37	0.75
CNN	EVS	**0.99**	**0.99**	**0.99**	**0.99**	**0.97**	**0.97**	**0.99**	**0.98**	**0.99**	**0.99**	**0.98**	**0.99**	**0.98**	**0.98**	**0.98**	**0.94**	**0.96**	**0.96**	**0.98**
	MAPE	1%	3%	3%	4%	11%	11%	7%	15%	16%	3%	4%	3%	5%	3%	3%	7%	13%	13%	7%
	t-test	0.65	1.08	1.00	2.26	2.00	1.87	3.73	2.94	3.68	-2.5	-1.13	-1.04	-3.37	-0.43	-0.49	-1.23	1.07	1.07	0.62
	NSE	0.98	0.97	0.97	0.94	0.86	0.86	0.88	0.85	0.85	0.92	0.96	0.98	0.69	0.98	0.98	0.84	0.91	0.91	0.91

Data1 -Yahoo finance, **Data2**- UK Investing, **Data3** - Bitfinex

Table 5. Summary of models' performance in the two scenarios

Model	Metric	Scenario A (mean)	Scenario B (mean)	Overall mean
XGB	EVS	0.61	0.97	0.79
	MAPE	0.18	0.03	0.11
	t-test	0.62	-0.03	0.30
	NSE	0.01	0.97	0.49
GBM	EVS	0.62	0.96	0.79
	MAPE	0.17	0.04	0.11
	t-test	0.64	-0.16	0.24
	NSE	0.04	0.95	0.50
ADAB	EVS	0.60	0.96	0.78
	MAPE	0.18	0.04	0.11
	t-test	0.66	-0.09	0.29
	NSE	-0.04	0.94	0.45
GRU	EVS	0.92	0.98	0.95
	MAPE	0.11	0.04	0.08
	t-test	0.44	0.49	0.47
	NSE	0.74	0.96	0.85
DFNN	EVS	0.88	0.95	0.92
	MAPE	0.18	0.12	0.15
	t-test	1.07	0.98	1.03
	NSE	0.44	0.75	0.60
CNN	EVS	0.94	0.98	0.96
	MAPE	0.09	0.07	0.08
	t-test	0.32	0.62	0.47
	NSE	0.84	0.91	0.88

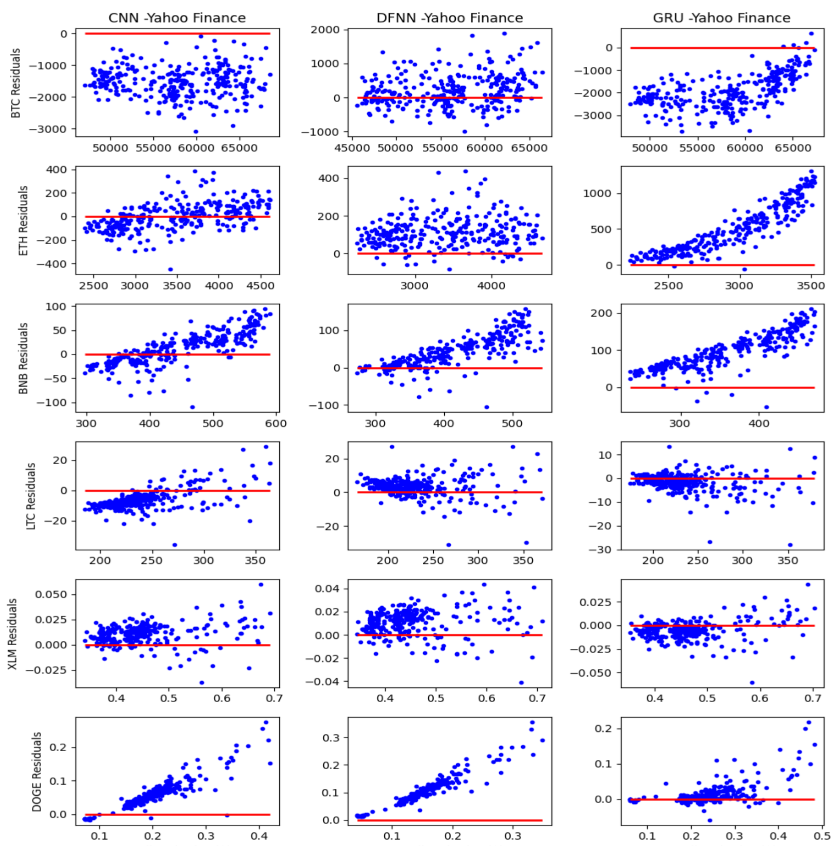

Fig. 3. The plot of residuals against predicted closing prices (Scenario A- deep learning models) for the cryptocurrencies. The models' predictions are on the x-axis, and the residuals are on the y-axis.

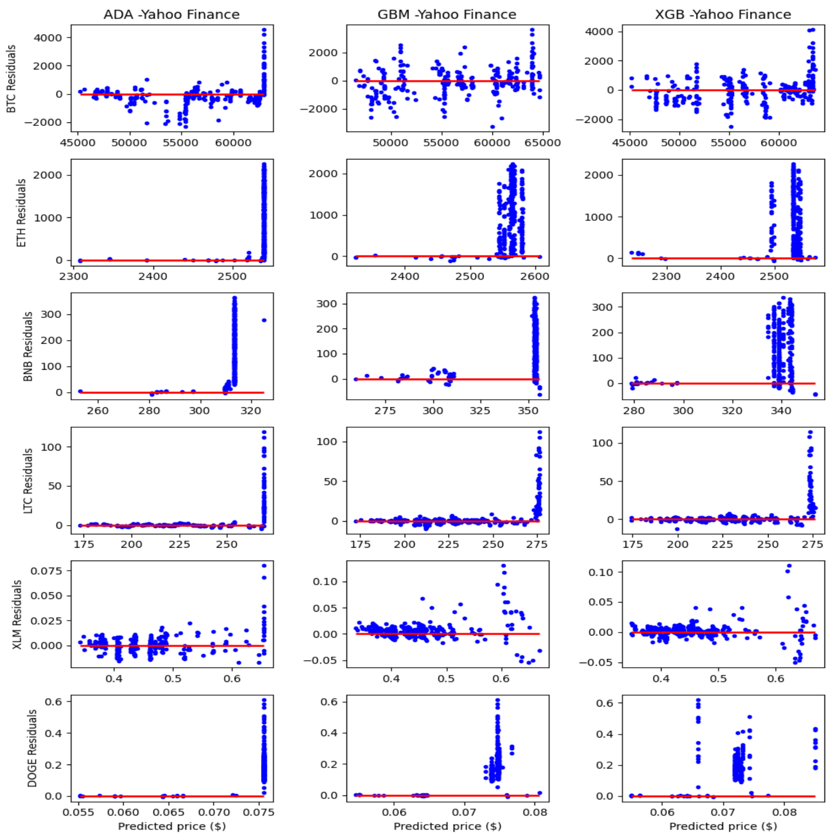

Fig. 4. The plot of residuals (tree-based models) against predicted closing prices (Scenario A) of cryptocurrencies

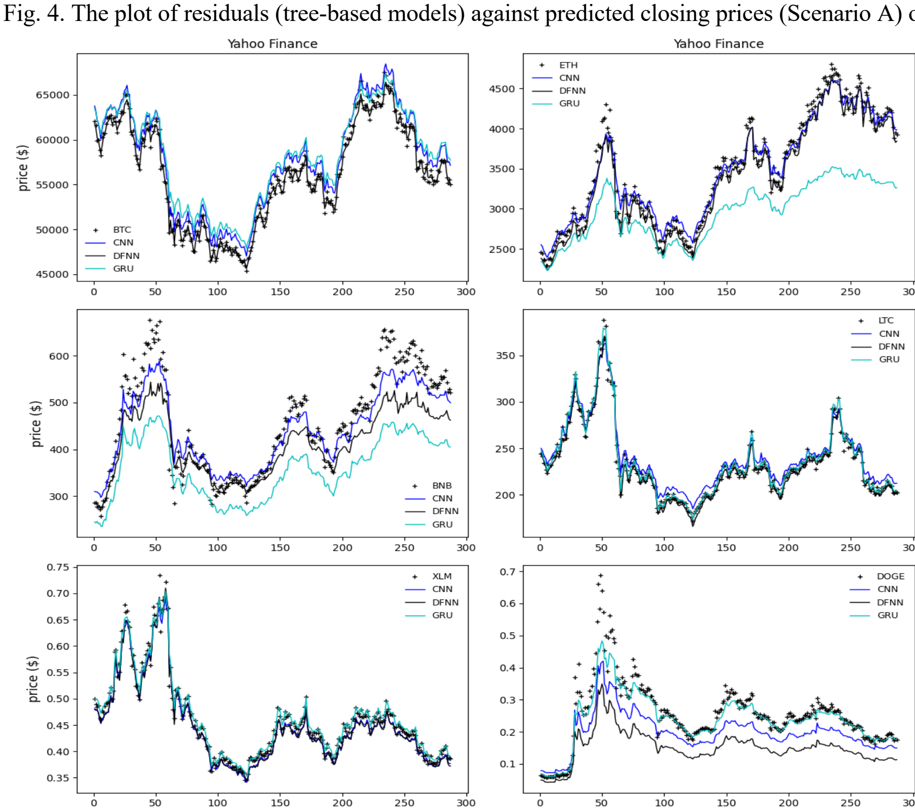

Fig. 5. Scenario A: Daily variation of estimated closing prices (Yahoo finance) of DL models compared with measured values

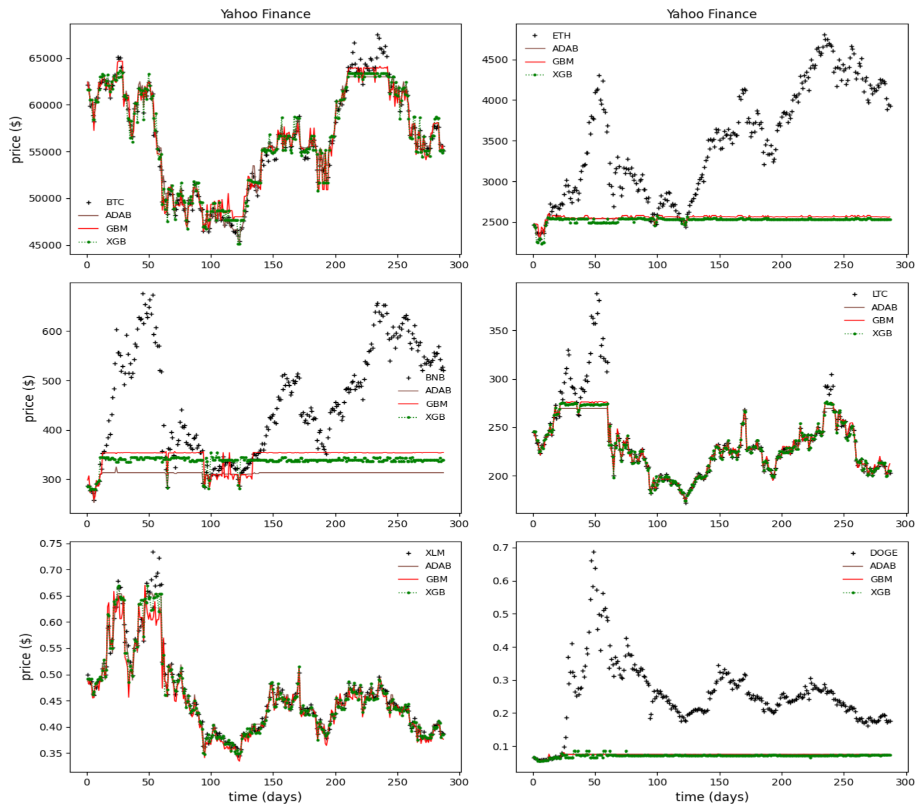

Fig. 6. Scenario A: Predicted daily closing prices (Yahoo finance) from tree-based models

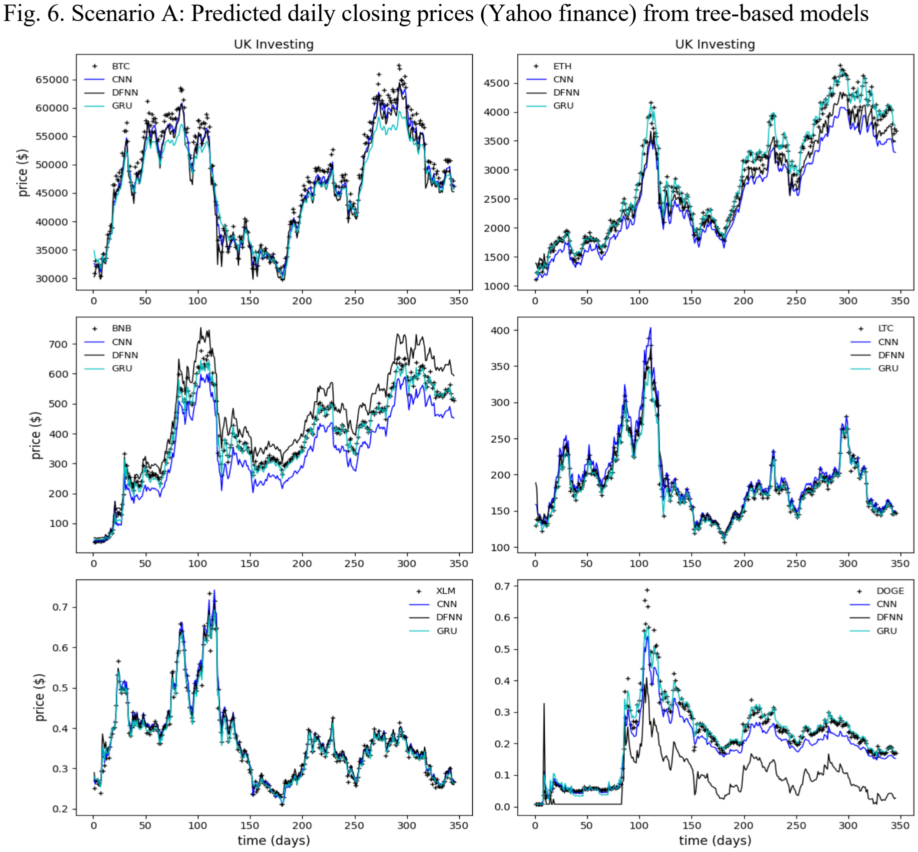

Fig. 7. Estimated daily closing price predictions of DL models (UK Investing- Scenario B)

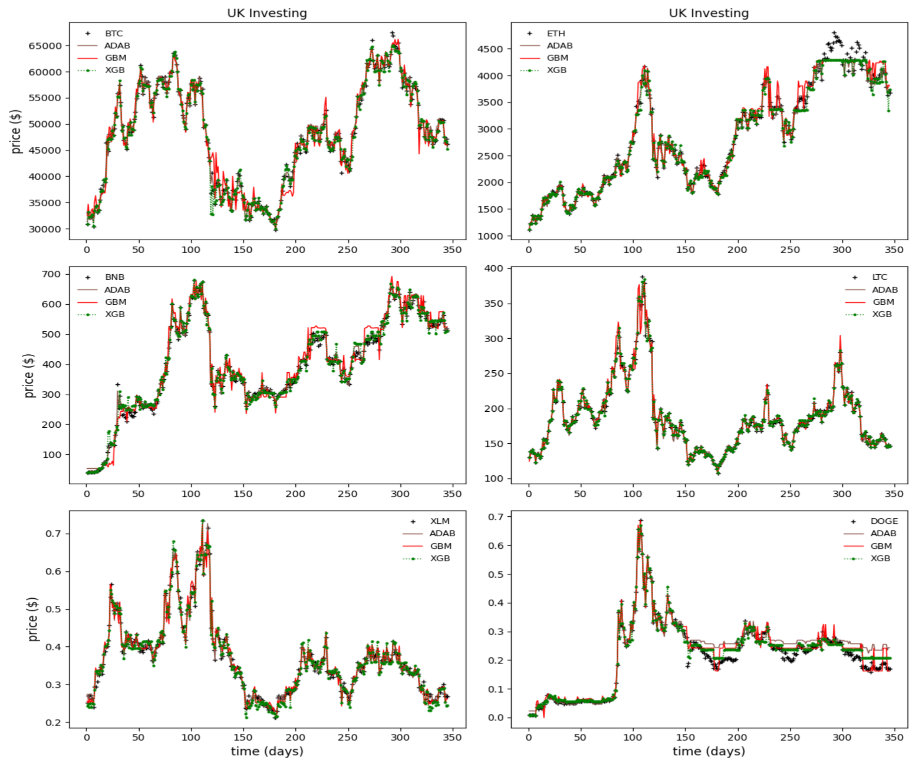

Fig. 8. Estimated daily closing price predictions of boosted tree models (UK Investing- Scenario B)

Also, in Fig. 10, black contours indicate the centered RMSE between the predictions and observed values, and this RMSE is proportional to the marked point ***"observed"*** on the x-axis. Predictions (Fig. 10) correspond well with observed values (lying nearest to the red arc marked ***"observed"***) and have high correlation and low RMSEs. It can be deduced from Fig. 10 that predictions of models agree best with measured closing prices.

Thus, for most cryptocurrencies (Scenario A), the CNN, GRU, and DFNN models produce high correlation coefficients, low RMSE, and standard deviations from the measured observations, compared to tree-based models that did not work effectively for some cryptocurrencies due to the presence of noisy random features and extreme volatility. Hence, DL techniques are more reliable when the training data is limited or when peaks and drops in crypto prices are inadequately captured.

4.1. Comparison with results in the literature

A comparison of the result obtained by the optimal configuration in this paper and a few other studies (Chowdhury *et al*., 2020; Dutta *et al*., 2019; Lahmiri & Bekiros, 2019; Mudassir *et al*., 2020) listed in Table 1, especially those related to prediction/regression problems regarding the daily Bitcoin price prediction is presented. Furthermore, a Root Mean Square Error metric, $RMSE = \sqrt{\sum_{j=1}^{n}\left(o_j - y_j\right)^2 / n}$, is adopted in comparing the results since all these studies utilized RMSE to measure differences between predicted crypto prices and actual observations. The result summary is presented in Table 5. For example, Chowdhury *et al*. (2020) adopted GBM and ensemble techniques to forecast the BTC/USD closing price and reported an RMSE of 32.86 for the GBM method. Similarly, Dutta et al. (2019) obtained RMSE values of 0.03 and 0.02, respectively, for neural networks and LSTM for the BTC closing price prediction.

Also, Lahmiri and Bekiros (2019) and Mudassir et al. (2020) used deep learning techniques (i.e., GRNN, LSTM, Stacked ANN) to forecast the BTC/USD price. In Lahmiri and Bekiros (2019), the LSTM and GRNN models obtained RMSE values of 2750 and 8800, respectively, while in Mudassir et al. (2020), with the data collection period from

April 1, 2013, to December 31, 2019, the stacked ANN and LSTM obtained RMSE values of 156.30 and 219.59 (for 30th-day forecast) respectively. However, comparing the result from this study with previous studies such as those from Chowdhury *et al*. (2020), Dutta *et al*. (2019), Lahmiri and Bekiros (2019), and Mudassir et al. (2020), the RMSE values obtained in these studies were higher than an RMSE of 0.01 obtained by the best model, optimal CNN, in this study.

Thus, the proposed optimal architecture could efficiently model the trends and patterns in these cryptocurrencies and produce a more reliable result, especially for the BTC closing price prediction. Consequently, the optimal deep learning models, especially the optimal CNN architecture, exhibit an inclusive and exemplary performance in the overall prediction of cryptocurrencies' closing prices, an important attribute helpful for the older and well-established financial markets (stock, forex, commodities) with same complexity characteristics, i.e., volatility clustering, non-linear correlations, effects resembling fractality and multifractality (Wątorek et al., 2021).

Table 6. Comparison with existing studies

Existing study	Model	RMSE	Cryptocurrency
Chowdhury *et al*. (2020)	GBM	32.86	BTC
Dutta *et al*. (2019)	Neural networks	0.03	BTC
	LSTM	0.02	
Lahmiri & Bekiros (2019)	LSTM	2750.00	BTC
	GRNN	8800.00	
Mudassir *et al*. (2020)	Stacked ANN	156.30	BTC
	LSTM	219.59	
Present study	optimized CNN	**0.03**	BTC
	Optimized GRU	**0.02**	BTC

4.2. Implication for study

The results of this research are two-fold: (1) in creating predictive models based on advanced ML methods for modeling the crypto market to lower investment risks, and (2) suggesting an optimal configuration for the predictive analytics models to forecast the daily closing price of any cryptocurrency efficiently. The previous studies in the area mainly focus on a single historical data source for training, validating, and testing their models. Also, they use ML models to predict famous and single cryptocurrency platforms. To the best of our knowledge, this study is the first to forecast daily closing prices by benchmarking the robustness of DL and boosted tree techniques in terms of using an optimal model's configuration across several cryptocurrencies. Also, to guarantee the effectiveness of the models, a genetic algorithm is utilized to determine their optimal configurations, including the number of neurons in the hidden layers, batch size, and the learning rate. In addition, the performance of prediction models on three different testing sets was investigated, and their sensitivity to the training data, where peaks and drops in prices are not adequately captured in training sets, was evaluated. Unlike previous studies, a report detailing the conservative estimate of the explained variances using four statistical metrics (EVS, MAPE, t-tests, and NSV) and graphical plots (residuals, Taylor diagram, time variation plots) was presented. Thus, this study's results can help make futuristic plans to minimize risks and uncertainties and increase investment returns.

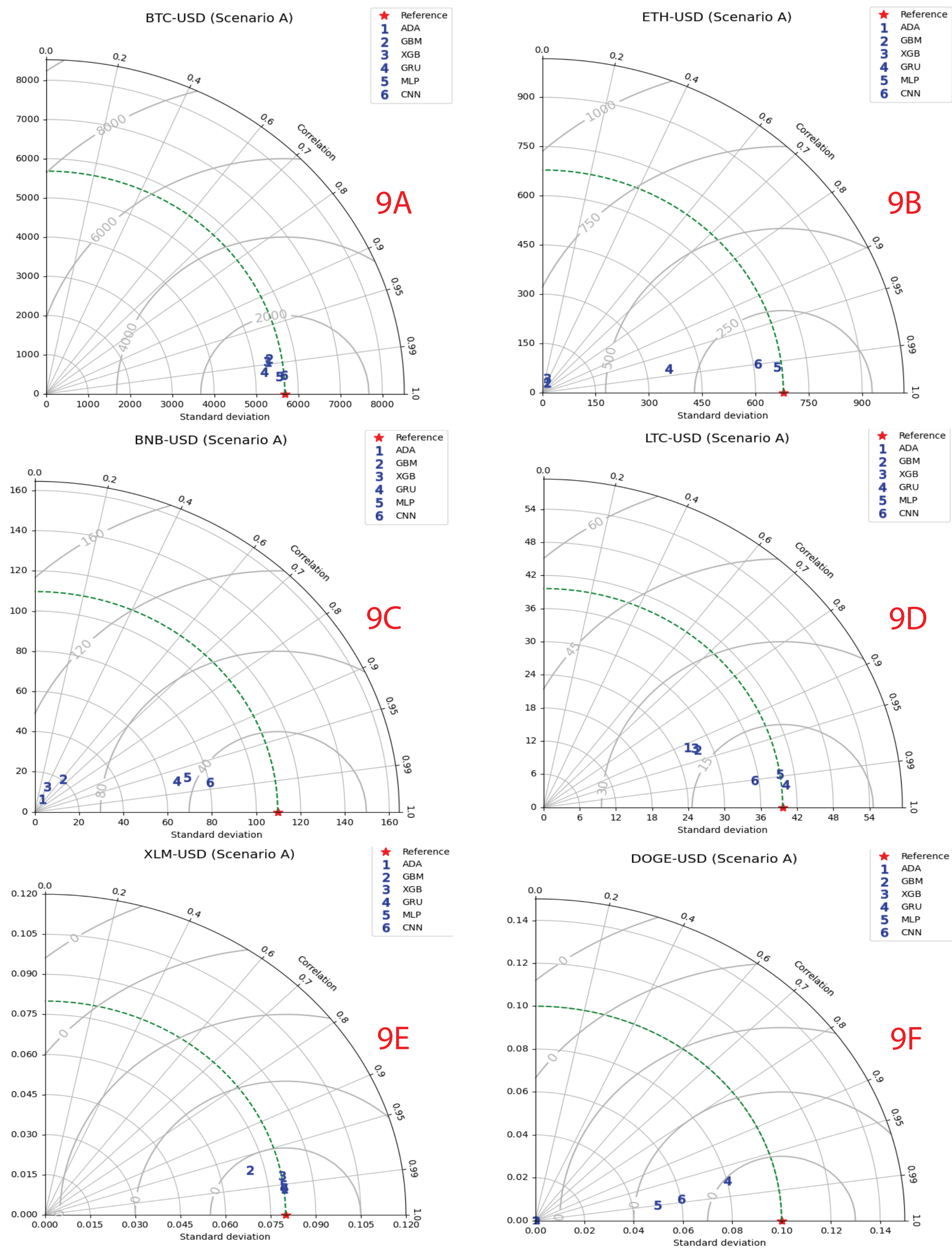

Fig. 9. Taylor's diagram showing a statistical comparison between forecasts and measured values (Yahoo finance)

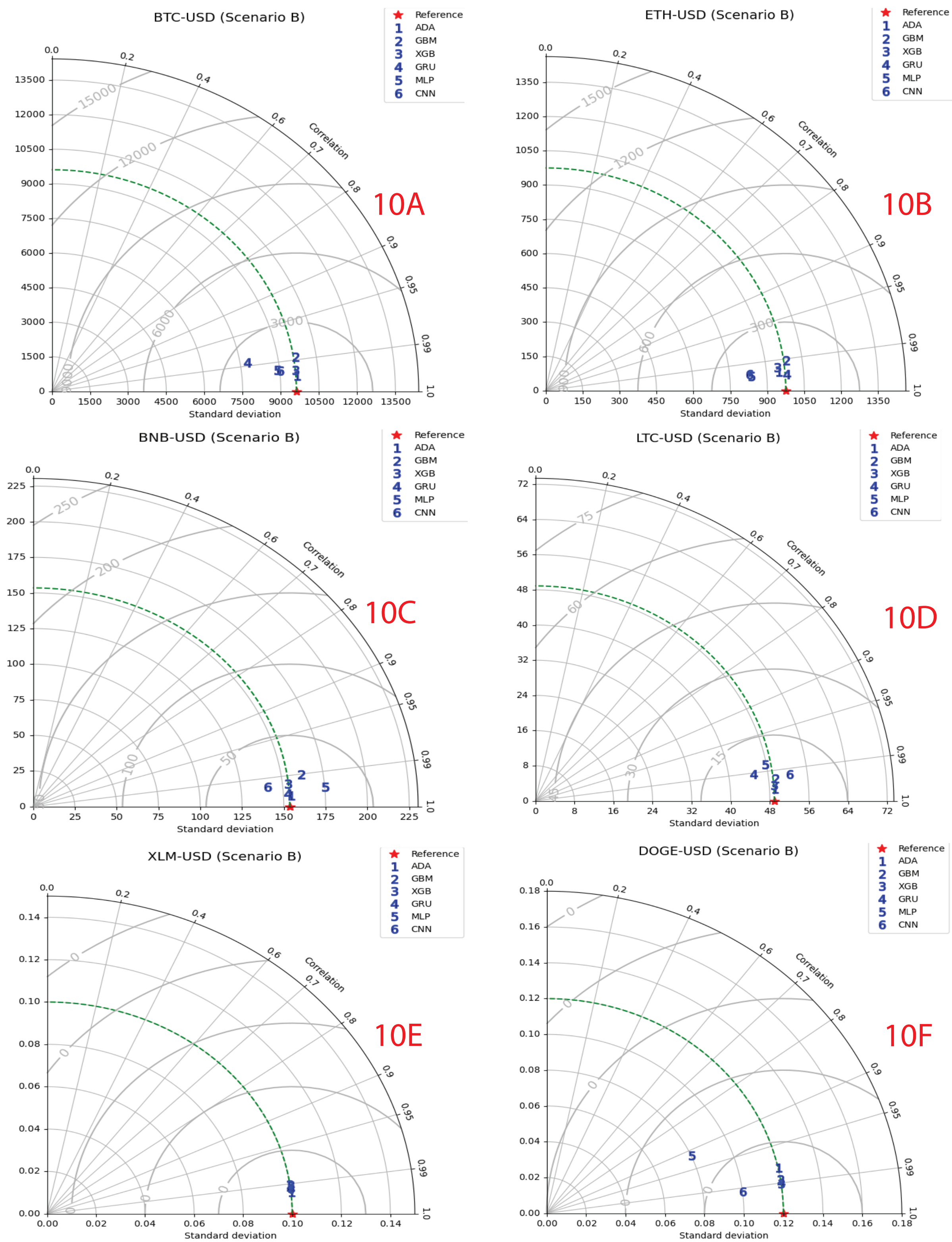

Fig. 10. Taylor's diagram showing a statistical comparison between forecasts and measured values (UK Investing)

5. Conclusion

Forecasting the cryptocurrencies market is a challenging task in finance and a concern to investors due to its high volatile behavior. Therefore, this paper proposes a robust and optimal predictive models' configuration that can predict cryptocurrency closing prices with a training set having significant peaks and drops in prices not captured. Thus, the study benchmarks DL and boosted tree-based techniques, using the models' optimal configurations to predict the daily closing prices of six different cryptocurrencies datasets collected from more than one data source. Based on the prediction results achieved in the present study, the following conclusions can be drawn, given a limited training data sample:

1 The DL techniques obtain a more significant EVS percentage (between 88% to 98%), indicating that they have accounted for the total variance in the observed data. However, the boosting trees techniques struggle to predict daily closing prices for ETH, BNB, and DOGE cryptocurrencies due to the missing insights from the training set. However, the CNN model produces more accurate results than other DL techniques.

2 In comparing the robustness of the models on the different test datasets (Yahoo finance, UK investing, and Bitfinex), DL models (CNN and GRU) on average, produce consistent and higher EVS values ($0.92 \leq$ EVS ≤ 0.98) compared to boosted tree-based models, which are low in some cases ($0.03 \leq$ EVS ≤ 0.50) for ETH and DOGE, thus showing the unreliability of the predicted regression for these group of cryptocurrencies when certain information is missing from the training set.

3 For Scenario A, the CNN model has the smallest average MAPE of 9%, followed by GRU with an average MAPE of 11%, GBM (an average MAPE of 17%), DFNN (18%), XGB (average MAPE of 18%), and ADAB (average MAPE index of 18%).

4 The residuals from the DL techniques are randomly distributed around the zero horizontal lines, thus exhibiting no defined patterns, and satisfying the assumption of residuals having a constant variance. However, residuals from the boosted tree-based counterparts are either not symmetric to the origin or randomly distributed due to peaks and falls of crypto prices not adequately captured in the training sets. Hence, their inability to learn from limited examples and generalize some degree of knowledge in predicting closing prices in this scenario.

5 The CNN optimal model configuration produces high correlation coefficients, low RMSE, and standard deviations from the measured observations for most cryptocurrencies. Hence, it is more reliable for limited training data or when peaks and drops in crypto prices are inadequately captured in the training data. Hence, CNN is efficient and readily generalizable to predict any cryptocurrency's daily closing price.

6 This study has revealed the possibility of a single and optimal model's architecture for predicting the prices of multiple cryptocurrencies. Though, predicting the financial market is difficult due to its complex systems dynamics. However, deep learning techniques have been the modern approaches to modeling this market. For instance, deep learning architectures have been applied for stock market prediction (Nelson et al., 2017; Ticknor, 2013), forex market prediction (Hadizadeh Moghaddam & Momtazi, 2021; Ni et al., 2019), and commodity market volatility prediction (Kamdem et al., 2020). In the same vein, the computationally efficient deep learning models developed in this study, especially the optimized CNN model, can be adapted with little modifications to other financial markets (i.e., stock, bond rates, forex, commodities).

References

Ajayi, A., Oyedele, L., Akinade, O., Bilal, M., Owolabi, H., Akanbi, L., & Davila-Delgado, J. (2020). Optimised big data analytics for health and safety hazards prediction in power infrastructure operations. Safety Science, 125. https://doi.org/https://www.sciencedirect.com/science/article/pii/S0925753520300539

Alonso-Monsalve, S., Suárez-Cetrulo, A., Cervantes, A., & Quintana, D. (2020). Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. Expert Systems with Applications, 149(2020.113250).

Atsalakis, G. S., Atsalaki, I. G., Pasiouras, F., & Zopounidis, C. (2019). Bitcoin price forecasting with neuro-fuzzy techniques. European Journal of Operational Research, 276(2), 770–780.

Borges, T., & Neves, R. (2020). Ensemble of machine learning algorithms for cryptocurrency investment with different data resampling methods. Applied Soft Computing Journal, 90(106187). 10.1016/j.asoc.2020.106187

Canh, N. P., Wongchoti, U., Thanh, S. D., & Thong, N. T. (2019). Systematic risk in cryptocurrency market: Evidence from DCC-MGARCH model. Finance Research Letters, 29, 90–100. https://doi.org/10.1016/J.FRL.2019.03.011

Chaim, P., & Laurini, M. P. (2019). Nonlinear dependence in cryptocurrency markets. The North American Journal of Economics and Finance, 48, 32–47.

Chen, Z., Li, C., & Sun, W. (2020). Bitcoin price prediction using machine learning: An approach to sample dimension engineering. Journal of Computational and Applied Mathematics, 365, 112395. 0.1016/j.cam.2019.112395

Cherati, M. R., Haeri, A., & Ghannadpour, S. F. (2021). Cryptocurrency direction forecasting using deep learning algorithms. Journal of Statistical Computation and Simulation, 91(12), 2475–2489.

Choo, K. K. R. (2015). Cryptocurrency and Virtual Currency: Corruption and Money Laundering/Terrorism Financing Risks? Handbook of Digital Currency: Bitcoin, Innovation, Financial Instruments, and Big Data, 283–307

Chowdhury, R., Rahman, M., Rahman, M., & Mahdy, M. (2020). An approach to predict and forecast the price of constituents and index of cryptocurrency using machine learning. Physica A. Statistical Mechanics and Its Applications, 551, 124569. 10.1016/j.physa.2020.124569

Dutta, A., Kumar, S., & Basu, M. (2019). A gated recurrent unit approach to Bitcoin price prediction. Journal of Risk and Financial Management, 13(23), 1–16.

Guo, T., Bifet, A., & Antulov-Fantulin, N. (2018). Bitcoin Volatility Forecasting with a Glimpse into Buy and Sell Orders. IEEE International Conference on Data Mining (ICDM), 989–994.

Hadizadeh Moghaddam, A., & Momtazi, S. (2021). Image processing meets time series analysis: Predicting Forex profitable technical pattern positions. Applied Soft Computing, 108, 107460.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: data mining, inference, and prediction. Springer.

Huang, J. Z., Huang, W., & Ni, J. (2019). Predicting Bitcoin returns using high-dimensional technical indicators. The Journal of Finance and Data Science, 5(3), 140–155.

Ibrahim, A., Kashef, R., & Corrigan, L. (2021). Predicting market movement direction for bitcoin: A comparison of time series modeling methods. Computers & Electrical Engineering, 89, 106905. https://doi.org/10.1016/J.COMPELECENG.2020.106905

Jang, H., & Lee, J. (2018). An empirical study on modeling and prediction of Bitcoin prices with Bayesian neural networks based on blockchain information. IEEE Access, 6, 5427–5437.

Kristjanpoller, W., & Minutolo, M. C. (2018). A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. Expert Systems with Applications, 109, 1–11.

Kwon, D., Kim, J., Heo, J., Kim, C., & Han, Y. (2019). Time series classification of cryptocurrency price trend based on a recurrent LSTM neural network. Journal of Information Processing Systems, 15(3), 694–706.

Lahmiri, S., Bekiros, S., & Salvi, A. (2018). Long-range memory, distributional variation and randomness of bitcoin volatility. Chaos, Solitons & Fractals, 107, 43–48.

Lahmiri, S., & Bekiros, S. (2019). Cryptocurrency forecasting with deep learning chaotic neural networks Chaos. Chaos, Solitons & Fractals, 118, 35–40.

Lahmiri, S., & Bekiros, S. (2020a). Big data analytics using multi-fractal wavelet leaders in high-frequency Bitcoin markets. Chaos, Solitons & Fractals, 131, 109472. https://doi.org/10.1016/J.CHAOS.2019.109472

Lahmiri, S., & Bekiros, S. (2020b). Intelligent forecasting with machine learning trading systems in chaotic intraday Bitcoin market Chaos. Chaos, Solitons & Fractals, 133, 109641. 10.1016/j.chaos.2020.109641

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.

Li, T., Chamrajnagar, A., Fong, X., Rizik, N., & Fu, F. (2018). Sentiment-based prediction of alternative cryptocurrency price fluctuations using gradient boosting tree model. Frontiers in Physics, 7(98), 1–8.

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

Miura, R., Pichl, L., & Kaizoji, T. (2019). Artificial neural networks for realized volatility prediction in cryptocurrency time series. International Symposium on Neural Networks, 165–172.

Mudassir, M., Bennbaia, S., Unal, D., & Hammoudeh, M. (2020). Time-series forecasting of Bitcoin prices using high-dimensional features: a machine learning approach. Neural Computing and Applications, 1–15. https://doi.org/10.1007/S00521-020-05129-6/FIGURES/10

Nakano, M., Takahashi, A., & Takahashi, S. (2018). Bitcoin technical trading with artificial neural network. Physica A. Statistical Mechanics and Its Applications, 510, 587–609.

Nelson, D., Pereira, A., & de Oliveira, R. (2017). Stock market's price movement prediction with LSTM neural networks. International Joint Conference on Neural Networks, 1419–1426.

Ni, L., Li, Y., Wang, X., Zhang, J., Yu, J., & Qi, C. (2019). Forecasting of Forex Time Series Data Based on Deep Learning. Procedia Computer Science, 147, 647–652.

Peng, Y., Albuquerque, P. H. M., Camboim de Sá, J. M., Padula, A. J. A., & Montenegro, M. R. (2018). The best of two worlds: Forecasting high frequency volatility for cryptocurrencies and traditional currencies with Support Vector Regression. Expert Systems with Applications, 97, 177–192.

Poongodi, M., Sharma, A., Vijayakumar, V., Bhardwaj, V., Sharma, A. P., Iqbal, R., & Kumar, R. (2020). Prediction of the price of Ethereum blockchain cryptocurrency in an industrial finance system. Computers & Electrical Engineering, 81, 106527. https://doi.org/10.1016/J.COMPELECENG.2019.106527

Kamdem, S.J., Essomba, R.B, & Berinyuy, J.N. (2020). Deep learning models for forecasting and analyzing the implications of COVID-19 spread on some commodities markets volatilities. Chaos, Solitons & Fractals, 140, 110215. https://doi.org/10.1016/J.CHAOS.2020.110215

Shah, D., & Zhang, K. (2014). Bayesian regression and Bitcoin. 52nd Annual Allerton Conference on Communication, Control, and Computing, 409–414.

Sheridan, R., Wang, W., Liaw, A., Ma, J., & Gifford, E. (2016). Extreme Gradient Boosting as a Method for Quantitative Structure-Activity Relationships. Journal of Chemical Information and Modeling, 56(12), 2353–2360.

Ticknor, J. L. (2013). A Bayesian regularized artificial neural network for stock market forecasting. Expert Systems with Applications, 40(14), 5501–5506.

Wątorek, M., Drożdż, S., Kwapień, J., Minati, L., Oświęcimka, P., & Stanuszek, M. (2021b). Multiscale characteristics of the emerging global cryptocurrency market. Physics Reports, 901, 1–82.

Sun, X, Liu, M., & Zeqian, S. (2020). A novel cryptocurrency price trend forecasting model based on LightGBM. Finance Research Letters, 32, 101084. https://doi.org/10.1016/J.FRL.2018.12.032

Zoumpekas, T., Houstis, E., & Vavalis, M. (2020). ETH analysis and predictions utilizing deep learning. Expert Systems with Applications, 162(30), 113866ß