Financial Innovation

RESEARCH Open Access

Deep learning for Bitcoin price direction prediction: models and trading strategies empirically compared

Oluwadamilare Omole[1] and David Enke[2*]

*Correspondence:
enke@mst.edu

[1] Laboratory for Investment and Financial Engineering, Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, 203 Engineering Management, 600 W. 14th Street, Rolla, MO 65409-0370, USA
[2] Laboratory for Investment and Financial Engineering, Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, 221 Engineering Management, 600 W. 14th Street, Rolla, MO 65409-0370, USA

Abstract
This paper applies deep learning models to predict Bitcoin price directions and the subsequent profitability of trading strategies based on these predictions. The study compares the performance of the convolutional neural network–long short-term memory (CNN–LSTM), long- and short-term time-series network, temporal convolutional network, and ARIMA (benchmark) models for predicting Bitcoin prices using on-chain data. Feature-selection methods—i.e., Boruta, genetic algorithm, and light gradient boosting machine—are applied to address the curse of dimensionality that could result from a large feature set. Results indicate that combining Boruta feature selection with the CNN–LSTM model consistently outperforms other combinations, achieving an accuracy of 82.44%. Three trading strategies and three investment positions are examined through backtesting. The long-and-short buy-and-sell investment approach generated an extraordinary annual return of 6654% when informed by higher-accuracy price-direction predictions. This study provides evidence of the potential profitability of predictive models in Bitcoin trading.

Keywords: Backtesting, Bitcoin, Cryptocurrency, Deep learning, Feature selection, On-chain data

Introduction

One of the earliest mentions of digital money was in the 1980s by Chaum (1981). Later, in 1990, he launched the first cryptocurrency, called DigiCash, but did not find much success. In 2008, a pseudonymous person/persons by the name Satoshi Nakamoto published the Bitcoin white paper (Nakamoto 2008), which marked the inception of the cryptocurrency industry as known today. The Bitcoin protocol is built on a distributed ledger technology called blockchain. Blockchain is a system for recording information that makes changing, hacking, or cheating the system difficult. Since the launch of Bitcoin—with Bitcoin trading and price data tracked through the BTC ticker symbol—other protocols have been created, including Ethereum, Ripple, Cardano, Polkadot, Cosmos, and Solana. This new technology has presented a plethora of promises that include payments, decentralized storage, digital voting, digital identity, and supply chain tracking.

The fast growth of the cryptocurrency market has attracted worldwide attention from institutions and governments. However, this emerging financial market is notorious for high volatility and is seen by many as a speculative phenomenon. Cryptocurrency's volatility is caused by several factors, including investors' irrationality, market immaturity, and exogenous shocks. Smales (2019) observed that Bitcoin exhibits greater volatility than traditional assets like gold, even under normal market conditions. The 24/7 trading nature of cryptocurrencies could also result in greater price fluctuations. This is unlike stocks, which typically do not trade on weekends (De Leon et al. 2022). Cryptocurrency's high volatility could result in uncertainty for investors and users. Therefore, predictive models are needed that can help guide investment decisions.

Predicting financial markets is a difficult task (Chen et al. 2022). This can be attributed to noise, uncertainty, and hidden relationships within market data (Huang et al. 2005; Alonso-Monsalve et al. 2020). Predicting the cryptocurrency market can be even more difficult and less reliable due to high price volatility and a lack of historical data that are typically available for the stock market (Park and Seo 2022). Therefore, predictive models capable of predicting the behavior of the cryptocurrency market are a topic of interest in the literature. High-performing predictive models are sought after due to the potential gains that can be achieved. Furthermore, the high intraday volatility observed in most cryptocurrencies makes them suitable for high-frequency trading.

The main policy-level problem considered by this work revolves around using cryptocurrencies and distributed ledger technology (blockchain). There are advantages of such as system, yet cryptocurrencies that utilize blockchain technology are often volatile in price and therefore difficult to transact or use as a store of value. Often, this is due to a lack of price understanding among market participants. This has resulted in new regulations or failures to approve new financial securities that could be used to hedge cryptocurrency risk. To provide better forecasting ability for cryptocurrencies and reduce price/return volatility, the same combinations of inputs, outputs, and traditional modeling approaches cannot simply be used. Instead, features unique to the distributed ledger technology (blockchain) network must be used. Therefore, this work considers the unique on-chain data available from the blockchain to better understand the pricing dynamics for one widely traded cryptocurrency—Bitcoin. As price transparency becomes more known and efficient, the approval of securities like a spot Bitcoin ETF should occur, providing more access to these cryptocurrencies, as well as more adoption and hopefully greater efficiency within these markets.

The models explored in the literature range from simple statistical models to more computationally expensive machine learning (ML) models. ML has often shown performance superior to that of statistical models. This may be attributed to the ability of ML algorithms to uncover complex, nonlinear relationships within the input data, which lets them make accurate predictions without relying on the statistical properties of the input data (Alonso-Monsalve et al. 2020). The development of ML models addresses three major considerations: input types, such as historical prices, technical analysis metrics, and sentiment, or on-chain, data; predictive model choice; and chosen output type, which may involve predicting the actual magnitude of prices (i.e., a regression problem) or determining the direction of price movements (i.e., a classification problem).

The novelty in this research lies in combining deep learning (DL) models, input types, and output types that have not been sufficiently explored in the literature. The inputs for the model were Bitcoin price and a large set of on-chain data. Studies have shown that a large feature size could result in the curse of dimensionality problem (Zhong and Enke 2017a, 2017b; Aras 2021). Therefore, feature selection was carried out to obtain a smaller subset containing the most relevant data. Boruta, genetic algorithm (GA), and light gradient boosting machine (LightGBM) methods were used for this research. The performance of the convolutional neural network–long short-term memory (CNN–LSTM), long- and short-term time-series network (LSTNet), and temporal convolutional network (TCN) models were then compared. These models were chosen because studies have suggested using DL models over traditional ML models. In addition, the selected models have been understudied within the context of Bitcoin price prediction. To further solidify the credibility of the performance of DL models, a benchmark analysis was conducted against a popular statistical model, ARIMA. For the type of output, the prediction of the Bitcoin price direction was utilized. This decision was informed by the literature discussed in the next section. Finally, trading strategies were implemented to investigate the profitability of predictions. To the authors' knowledge, this unique combination has not been explored in the existing literature.

The objectives of the study include (1) investigating predictive improvements that can be achieved through feature-selection methods; (2) comprehensively comparing CNN–LSTM, LSTNet, and TCN models, an under researched area; and (3) assessing the profitability of the selected model using different trading strategies. This approach is designed to address the knowledge gaps in the current literature and offer actionable insights for researchers and cryptocurrency traders. Through this study, a clearer understanding of the potential and limitations of modern predictive techniques for the cryptocurrency market will be achieved, with the ultimate goal of providing a robust foundation for more intelligent investment decisions in this vibrant and evolving market.

The structure of this paper is as follows. The literature review is outlined next and explains the methodology employed for the experiments. The results derived from these experiments are then thoroughly discussed. Key findings are summarized and directions for future work are presented.

Literature review

Historically, there has been growing interest in employing ML models to predict Bitcoin prices. Scholars have approached this challenge by focusing on one or an integration of three aspects: the nature of the input data, the structure of the model architecture, and the characteristics of the output.

Type of inputs used in Bitcoin price prediction

The nature of the input data employed is critical to crafting predictive models for Bitcoin prices. In the scholarly literature, the predominant input types investigated alongside Bitcoin price include technical indicators, sentiment data, and more recently, on-chain data. Twitter has emerged as a primary repository of textual information for sentiment analysis (Valencia et al. 2019; Huynh 2021, 2022; Critien et al. 2022; Shahzad et al. 2022). Additional platforms used in this analytical process include Google Trends, Reddit, and

Bitcointalk (Wołk 2020; Loginova et al. 2021). Passalis et al. (2022) have further demonstrated the feasibility of garnering sentiment data from news articles and financial documents. While the integration of sentiment data has led to reported enhancements in model performance, identifiable drawbacks exist. Wołk (2020) has noted that sentiment related to cryptocurrencies can be highly speculative. Documented challenges and limitations include expenses and restrictions tied to procuring textual data from social platforms like Twitter and the pervasiveness of bots on these platforms, potentially degrading the integrity of sentiment data (Kraaijeveld and De Smedt 2020). Technical indicators are another input that has been rigorously examined in the literature. A plethora of investigations have conveyed that incorporating technical indicators significantly boosts model performance (Huang and Huang 2019; Resta et al. 2020; Gyamerah 2021; Yang and Fantazzini 2022; Ye et al. 2022). For instance, Resta et al. (2020) used technical indicators, including oscillators, trend-following metrics, and moving averages. Yang and Fantazzini (2022) proposed a novel approach that formulated oscillators from pricing models. Other studies have implemented a wide-ranging collection of technical indicators, such as research by Huang and Huang (2019), whose tree-based model used 124 features and observed improved subsequent performance. Furthermore, the fusion of technical indicators with other types of inputs, like sentiment data, has been pursued by researchers (Cavalli and Amoretti 2021; Ortu et al. 2022; Tripathi and Sharma 2022; Ye et al. 2022). Ranjan et al. (2022) emphasized the importance of high-dimensional feature sets and their ability to compensate for the simplicity of ML models.

In recent studies, transactional records found on the Bitcoin blockchain have been probed as a form of input for Bitcoin price prediction. A significant observation by Kukacka and Kristoufek (2023) highlighted that the dynamics of Bitcoin's price are greatly influenced by on-chain activities. Various studies have integrated on-chain data into ML models, consequently observing improvements in predictions (Jagannath et al. 2021; Kim et al. 2022; Casella and Paletto 2023). Although the current body of work has proven the potential efficacy of on-chain data in the context of Bitcoin price prediction, avenues for improvement still exist, particularly regarding the scope of on-chain data and the feature engineering methods used. Therefore, more research is needed within this domain.

Feature-selection methods

Using many technical indicators or on-chain data creates a large feature set, which can lead to overfitting and consequently poor prediction performance (Zhong and Enke 2017a, 2017b; Aras 2021). Various feature-selection methods have been used in the literature to identify the most relevant features. For instance, Ranjan et al. (2022) used a wrapper method to select 12 relevant features from 14 total, including price data, sentiment data, technical indicators, and on-chain data. Tripathi and Sharma (2022) used the Boruta method and correlation filtering to select relevant features from a dataset comprising price data, sentiment data, technical indicators, on-chain data, and macroeconomic variables. Rafi et al. (2023) utilized the Boruta method to select 8 relevant features from a list of 21, significantly improving model performance. Zhu et al. (2023) compared XGBoost and random forest (RF) for feature selection. They observed that XGBoost-selected features resulted in greater improvement. Cho et al. (2021) applied

GA for feature selection and reported an improvement in Gaussian process (GP) regression performance for Bitcoin price forecasting. Erfanian et al. (2022) comprehensively compared five feature-selection methods—best first search, principal component analysis, particle swarm optimization, variance inflation factor, and evolutionary algorithm. They observed that using no feature-selection method provided the best performance for their support vector regression (SVR) model. A similar observation was reported by Jang and Lee (2018). Although feature selection has not always resulted in improvements, Tripathi and Sharma (2022) suggested that a well-implemented feature-selection method can significantly improve the generalization capability of ML models. This calls for more research to explore the impact of various feature-selection methods on ML models.

Bitcoin price-prediction methods

Several ML models have been explored for Bitcoin price prediction. Zhu et al. (2023) compared the performance of three ML models: SVR, least squares SVR, and twin SVR (TWSVR). They reported that TWSVR outperformed the other models for calculation speed and prediction accuracy, achieving an explained variance score of 0.9547. The model also showed significant improvement over the benchmark ARIMA model. Cho et al. (2021) compared SVR, extra trees, and GP regression models. Their results showed that the GP regression model performed best in both regression and classification tasks. Ranjan et al. (2022) compared logistic regression, linear discriminant analysis, quadratic discriminant analysis, RF, support vector machine, XGBoost, decision trees, and K-nearest neighbors across two price intervals. They reported that logistic regression excelled in daily predictions, while XGBoost was best for 5-min interval predictions. They suggested that leveraging DL models could yield even better results. Valencia et al. (2019) echoed this sentiment, recommending DL models like LSTM.

The existing literature offers varying opinions about the effectiveness of DL models versus traditional ML models. Derbentsev et al. (2020) compared the performance of the artificial neural network (ANN), binary autoregressive tree (BART), and RF models. They observed that while the BART and ANN models performed best for predicting price magnitude, the BART model outperformed the others for predicting price movements. Erfanian et al. (2022) compared linear regression, ensemble models, SVR, and MLP. They found that the SVR model outperformed the others, including the ANN model. Jang and Lee (2018) employed Bayesian neural networks for Bitcoin price and volatility predictions and reported that their method outperformed benchmark models like linear regression and SVR.

Some studies have highlighted the potential of neural network variants like recurrent neural networks and convolutional neural networks (CNNs). Research by Wu et al. (2019) incorporated LSTM to predict Bitcoin prices. They introduced two LSTM model variants: conventional LSTM and LSTM coupled with autoregression (AR). The hybrid LSTM outperformed the traditional LSTM model. Cavalli and Amoretti (2021) underscored the potential of one-dimensional convolutional neural networks (1D CNNs) to predict Bitcoin price movements. Their 1D CNN model demonstrated higher accuracy than that of LSTM models. Ye et al. (2022) proposed an ensemble model that stacked LSTM and gated recurrent unit (GRU) models for price predictions. They observed that their stacking ensemble model provided better prediction performance than the individual LSTM and GRU models and the average and blending ensemble methods. Ortu et al. (2022) comprehensively compared the performance of MLP, CNN, LSTM, and MALSTM-FCN models for Bitcoin and Ethereum predictions. They determined that no single model was universally best, as different models excelled under varying conditions. Nonetheless, they emphasized the criticality of appropriately tuning these models. This outcome is commonplace in the literature, with models showing varied performance under varying conditions.

Nature of the output

The nature of the output often presents researchers with a choice between predicting price magnitudes (regression) or predicting price directions (classification). This decision affects which predictive model is chosen and the metrics used in evaluating its performance. For example, regression-based models have been used in studies by Jang and Lee (2018); Wu et al. (2019); Erfanian et al. (2022). However, some studies have adopted a classification-based approach (Ortu et al. 2022; Ranjan et al. 2022). Moreover, other researchers have chosen to compare both approaches to leverage their respective benefits, as seen in the works of Ji et al. (2019) and Derbentsev et al. (2020). Ji et al. (2019) contrasted the profitability of regression-based and classification-based predictions. They observed that prediction with the classification approach was more profitable than using the regression approach. Similarly, Livieris et al. (2021) suggested that knowing the direction of price movements holds greater significance for investors and researchers than knowing the exact price magnitude.

Research gap

Despite the advancements that have been made in applying ML models for Bitcoin price prediction, a research gap persists when using DL models with on-chain data. The existing literature has marginally explored using CNN–LSTM, while TCN and LSTNet have not been explored for Bitcoin price prediction. Furthermore, using feature-selection methods—Boruta, GA, and LightGBM—for feature engineering of the large feature sets of on-chain data has been under researched. This paper explores this gap by combining the identified features of selected on-chain data and stated DL models to predict directional Bitcoin price movements. This combination presents a novel approach and contributes to the understanding of the dynamics of Bitcoin prices.

Methodology

The methodology employed in this study is shown in Fig. 1. The framework begins with data preparation, where the raw data is cleaned and transformed into a suitable format for models. The next stage involves feature engineering to select the most relevant features. The models are then trained, evaluated, compared, and benchmarked to select the best-performing combination of feature selection and model. Finally, trading strategies are implemented to investigate the profitability of the predictions.

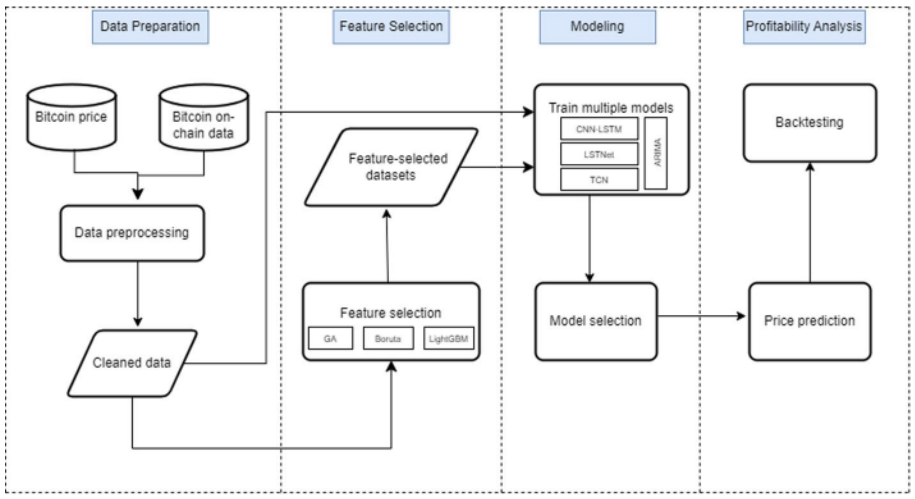

Fig. 1 Methodological framework

Data preprocessing

Preprocessing is usually the first stage in an ML pipeline. It involves cleaning and transforming data into a form that can be efficiently utilized by models. Preprocessing addresses issues within the data, including missing data, outliers, inconsistent scale, and noise. These activities are crucial for improving the performance of predictive models.

Directional movements in price

As suggested by Livieris et al. (2021), this study considers directional movements and not actual price magnitude. Thus, the prediction problem becomes a classification problem using binary classification, as described below:

- Class 1: This represents the scenario where the price of Bitcoin shows an increase from day t to day t + 1 ($y_{t+1} > y_t$) and is encoded as 1.
- Class 0: This denotes a scenario where the Bitcoin price either decreases or stays the same from day t to day t + 1 ($y_{t+1} \leq y_t$). This is encoded as 0.

Handling missing data

The strategy for handling missing data depended on whether the data was assessed to be missing completely at random (MCAR) or missing not at random (MNAR). In cases where the data was determined to be MCAR, the listwise deletion method was employed. This approach removes any row containing missing values and was specifically used for data records unavailable before a certain date. For instance, transaction volumes for cryptocurrency exchanges are naturally unavailable before the date the exchanges were launched. Conversely, when the data was found to be MNAR, regression imputation was used to estimate the missing values, leveraging existing relationships among the variables.

Data splitting and transformation

The dataset was categorized into training and testing sets using an 80–20 split, with the first 80% used for training and the remaining 20% used for testing. To preserve the temporal nature of the data, no shuffling was applied during this split. The input features were then standardized to ensure they have the same scale. This step was necessary to ensure that no feature dominates the learning process because of its larger scale. The decision to standardize the data after splitting was made to avoid any potential data leakage and maintain the integrity of the experiment. Data leakage could lead to overly optimistic performance metrics, as the model may have access to information from the test set during training.

Feature selection

High-dimensional data often result in the curse of dimensionality and high computational costs for running ML algorithms. Feature selection is a common preprocessing method used to find the most relevant feature subset. Feature selection is broadly grouped into filter, wrapper, and embedded methods. Filter methods use statistical characteristics inherent in the data to select the most relevant features, whereas embedded and wrapper methods use ML algorithms to perform their selections. In this research, Boruta, GA (wrapper method), and LightGBM (embedded method) are used to perform feature selection. The feature-selection process created three additional datasets, one for each method. A univariate dataset containing just the price was also created, thereby resulting in five datasets (including the original set) for each DL method.

Boruta feature selection

Boruta is a wrapper feature-selection method built on the RF algorithm (Kursa and Rudnicki 2010). It aims to capture all relevant features, including those that may exhibit weak associations yet still augment the predictive capacity of the model. The procedure of the Boruta algorithm is as follows:

Step 1: The original dataset is duplicated, creating a "shadow dataset." Each feature in the original dataset has a corresponding feature in the shadow dataset.

Step 2: The shadow features are then shuffled randomly to eliminate their correlation with the target variable. This shuffling generates a set of random variables that have no predictive capability.

Step 3: The original and shadow datasets are concatenated, and an RF classifier is trained on this combined dataset. The Z-scores are then calculated and recorded.

Step 4: The maximum Z-score among the shadow features, known as MZSA (Maximum Z-score among shadow attributes), is determined. Each feature's score is ranked and any feature scoring higher than this MZSA is labeled as important, with other features labeled as unimportant.

Step 5: To evaluate the importance of each feature, a two-sided test on MZSA is performed. Features that are found to be significantly less critical than MZSA are considered unimportant and permanently removed from the feature set. Features found to be significantly more important than MZSA are considered important and retained in the feature set.

Step 6: All shadow features are subsequently discarded, and steps are reiterated until all features are classified as important or unimportant, or until a predefined stopping criterion is reached.

Genetic algorithm feature selection

GAs can be used as a wrapper method, wherein the population consists of solutions (Chen and Zhou 2020). Each potential solution, referred to as an "individual," represents a subset of features derived from the dataset. The process involves steps such as initializing the population, evaluating the fitness function, conducting selection, performing crossover and mutation, and ultimately reaching termination. In this particular context, the fitness function involves training an RF classifier on the selected features and assessing its performance. Compared with other feature-selection methods, the GA method is less likely to become trapped within local optima and is therefore more capable of finding a global optimum. The drawback of this method is its computational expense, particularly for high-dimensional datasets, as it requires training a model for each subset of features in the population during every iteration.

Light gradient boosting machine feature selection

LightGBM is an embedded method that uses a gradient-boosting framework to perform feature selection. It calculates a feature's importance by considering the number of times it is used during model training and the improvement it provides when used. Once the model has been trained, features can be ranked based on their importance scores. Features with higher scores are considered more important. A threshold can be set to select the top n features or a recursive procedure can be implemented that iteratively eliminates the least important features, retrains the model, and continues this process until a stopping criterion is met. LightGBM has the advantage of considering interactions between features because feature selection is part of its training process.

Deep learning models

Convolutional neural network–long short-term memory

The CNN–LSTM model combines the strengths of CNN and LSTM models. It captures spatial features through CNN units and temporal dependencies through LSTM units (Lu et al. 2020). The CNN component starts with a 1D convolutional layer, a filter size of 64, and a kernel size of 3. Rectified linear unit (ReLU) was used as the activation function. Next, a batch normalization layer standardizes inputs for stability and performance. An average pooling layer with a pool size of 1 is then employed to reduce dimensions while minimizing parameters and computational requirements. To prevent overfitting, a dropout layer with a rate of 0.5 was applied. The LSTM component consists of two layers; the first layer has 128 units, and the second has 80 units, with both employing the TanH activation function. After each LSTM layer, batch normalization and dropout layers with a rate of 0.5 are added. The information extracted by the LSTM component is then fed into a dense layer that consists of one unit and uses the sigmoid function for activation. This final layer maps the features learned by the previous layers to the final output. The architecture of the CNN–LSTM model is shown in Fig. 2.

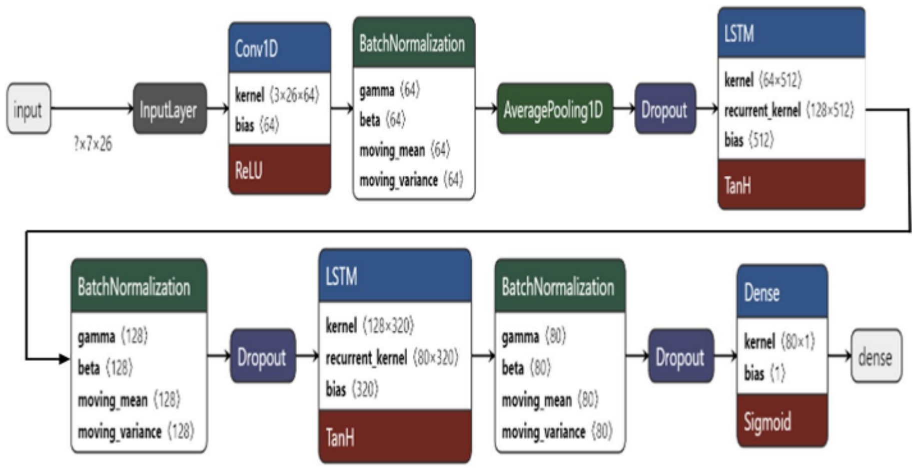

Fig. 2 Architecture of convolutional neural network–long short-term memory model

Long- and short-term time-series network

The LSTNet is a DL model specifically designed to capture both term and long-term patterns in time series data (Lai et al. 2018; Ouyang et al. 2022). This model consists of a CNN, an LSTM, and an AR component. The CNN component is responsible for capturing dependencies and short-term patterns within the data. It includes two layers of 1D CNN with filter sizes of 128 and 64, along with kernel sizes of 3. These layers are followed by three time-distributed dense layers with units set to 64, 32, and 1. Afterward, a global average pooling layer is applied. The LSTM component focuses on capturing long-term dependencies within the data. It utilizes two LSTM layers with units set to 256 and 128. Following these layers are three time-distributed dense layers with units set to 64, 32, and 1. An additional global average pooling layer is performed after the LSTM layers. The AR component consists of two layers with units set to 64 and 32, followed by a dense layer with one unit. After these AR layers, a global average pooling layer is applied. The AR component captures any relationships between the target sequence and its lag values, allowing for the modeling of patterns that might not have been accounted for by the other components. A combination of ReLU, TanH, and linear activation functions was used in the model architecture, as shown in Fig. 3.

The outputs from these three components are combined using the addition operation. The combined output then goes through two dense layers with 64 and 32 units, followed by dropout layers with a rate of 0.5. Finally, an output-dense layer is present with one unit and a sigmoid activation function.

Temporal convolutional network

The TCN is suitable for tasks involving sequence modeling (Bai et al. 2018). Its unique architecture allows it to handle inputs of varying lengths and to capture long-range dependencies. The initial layer employs 64 filters, a kernel size of 3, and dilations of 1, 2 4, and 8. The ReLU activation function was used. This layer produces 3D output to ensure compatibility with the next TCN layer. A dropout layer with a rate of 0.5 follows is then applied. The second TCN layer is similar in structure to the first but includes

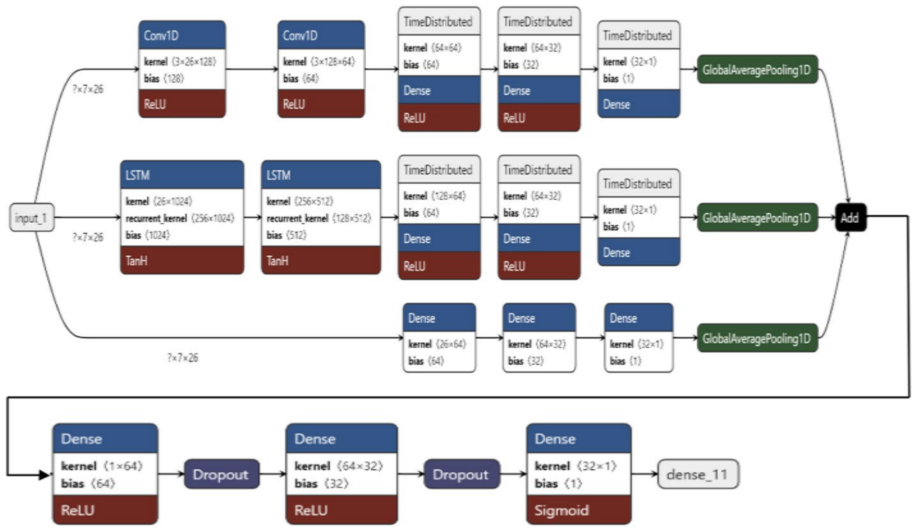
Fig. 3 Architecture of the LSTNet model

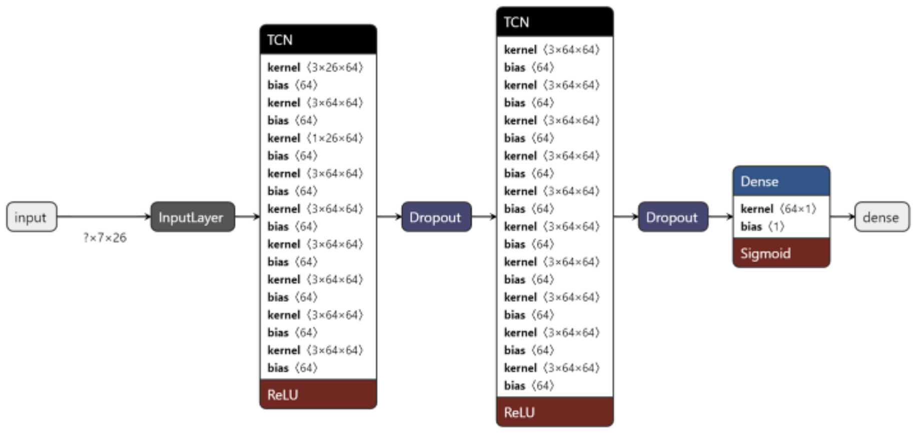
Fig. 4 Architecture of TCN model

an additional dilation of 16. This layer produces 2D output and is followed by a dropout layer with a rate of 0.5. Finally, the TCN layer output is passed to a dense layer that consists of one unit and employs a sigmoid activation function. This final layer maps the features learned to generate predictions for Bitcoin price movement. The architecture of the TCN model is shown in Fig. 4.

Hyperparameter tuning

For hyperparameter optimization, this study adopts a random search method. Contrary to grid search, which comprehensively investigates each feasible permutation of hyperparameters, random search randomly explores a subset of the hyperparameter space. This enables a more diverse exploration of configurations and often yields satisfactory solutions faster than grid search techniques. The optimization process is initiated by defining a range for each hyperparameter. These encompass the learning rate, batch size, epoch count, dropout coefficient, LSTM units, CNN filter dimensions, convolutional kernel dimensions, and activation functions. Subsequently, random combinations of these hyperparameters are curated, and the model undergoes training and validation via cross-validation techniques. The ultimate selection of hyperparameters is contingent on the model's performance metrics.

Evaluation metrics

Since this is a binary classification problem, the following evaluation metrics were used to assess the performance of the models discussed above.

Accuracy

This measures the ratio of correctly classified observations to total observations,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

where TP is the count of true positives, TN is the count of true negatives, FP is the count of false positives, and FN is the count of false negatives.

Precision

This measures the proportion of true positive predictions to total positive predictions:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall

This measures the ratio of true positive predictions to the number of actual positive observations:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F1-score

This is the harmonic mean of precision and recall and calculates a balance between the two:

$$F1 - score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{4}$$

Area under the receiver operating characteristic curve score

This measures the model's performance in a binary classification problem and is particularly useful for imbalanced datasets. It is obtained by calculating the area under the plot of the true positive (TP) rate against the false positive (FP) rate for different classification thresholds.

Matthews correlation coefficient (MCC)

This metric considers true and false positives and negatives. The MCC addresses the limitations of other performance metrics by providing a balanced measure of the quality of binary classifications:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \quad (5)$$

Backtesting approach

To ascertain not only the predictive accuracy but also the financial viability of the model's forecasts, this study proposes a backtesting framework. Backtesting was used to simulate trades predicated on three strategies for understanding how the model's predictions would have performed over a defined historical period (Piravechsakul et al. 2021; Sebastião and Godinho 2021; Kalariya et al. 2022). The trading strategies developed for backtesting include.

1. Buy-and-sell: This strategy capitalizes on the basic principle of buying low and selling high. Specifically, the trading algorithm is programmed to buy when the model forecasts an increase in price for the subsequent day, and conversely, to sell when a price decrease is forecasted for the next day.
2. Buy-and-sell with price protection: This approach is an extension of the first, with an additional constraint. While the trading algorithm still buys and sells based on the model's predictions, it is restricted from selling at a price lower than the entry price. This strategy attempts to minimize losses by holding onto the asset during short-term price dips.
3. Baseline strategy: The third strategy uses moving average convergence divergence (MACD) as a baseline strategy. MACD is a momentum indicator that shows the relationship between two moving averages of an asset's price. Mathematically, the MACD line is the difference between the 12-day and 26-day exponential moving averages (EMAs), while the signal line is the 9-day EMA of the MACD line. A buy order is initiated when the MACD line crosses above the signal line, while a sell order is triggered when the MACD line crosses below the signal line. The former indicates a bullish momentum, while the latter indicates a bearish momentum:

$$\text{MACDline} = EMA_{12-day} - EMA_{26-day} \quad (6)$$

$$\text{Signalline} = EMA_{9-day}(MACDline) \quad (7)$$

For testing these strategies, three potential investment positions were considered: long only, short only, and both long and short. The performance and risk of each strategy were evaluated using the Rate of Return (ROR), Sharpe ratio, maximum drawdown (MDD), market exposure, and volatility metrics calculated from Eqs. (8)–(12).

$$ROR = \frac{End\ Portfolio - Starting\ Portfolio}{Starting\ Portfolio} x100 \tag{8}$$

$$MDD = \frac{Trough\ Value - Peak\ Value}{Peak\ Value} \tag{9}$$

$$Sharpe\ Ratio = \frac{Expected\ Return - Risk\ Free\ Rate}{\sigma_{Excess\ Return}} \tag{10}$$

$$Market\ Exposure = \frac{\sum Trade\ Duration}{Total\ Period} \tag{11}$$

$$Volatility = \sigma_{Return\%} \tag{12}$$

Results and discussion

This study uses an extensive set of 87 distinct on-chain metrics alongside price data sourced from Glassnode (2023). The data were collected from February 6, 2013, to February 18, 2023, thus comprising 3665 days. Figure 5 plots the price of Bitcoin over the study period.

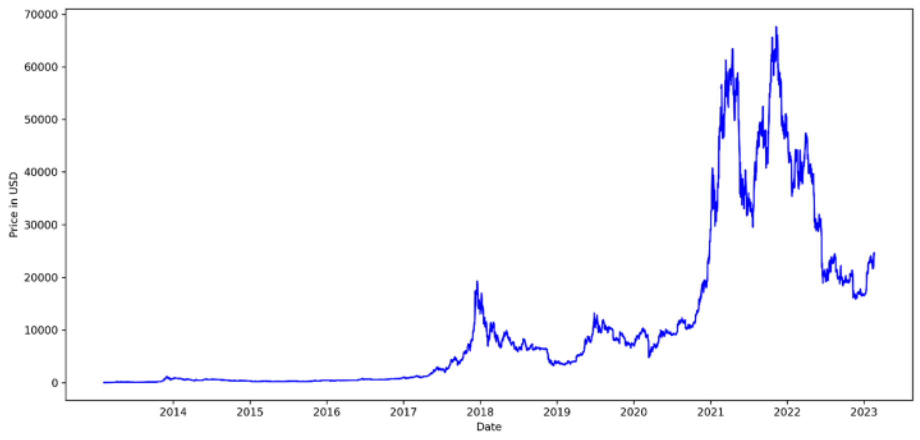

Fig. 5 Plot of Bitcoin price over study duration

Table 1 Original and feature-selected datasets

Feature selection method	Number of features
Boruta	26
GA	42
LightGBM	43
None	87 (all features)
Univariate (price)	1

Experimental datasets

The original dataset for this study comprises 87 on-chain data metrics, including price. To address the potential for the curse of dimensionality, three feature-selection methods, namely Boruta, GA, and LightGBM, were applied to the original dataset to create three subsets as additional datasets. In addition, a univariate dataset consisting solely of the Bitcoin price was created to serve as a benchmark. Table 1 shows the results for the feature-selection methods used. A comprehensive list of the features and resulting datasets can be found in Table 07 in the Appendix.

The experiments were carried out using the CNN–LSTM, LSTNet, TCN, and ARIMA models to predict the five datasets. Trading strategies were then used to assess the profitability of the best model, as presented later in this section.

Parameter study

To assess the sensitivity and robustness of the four models, a parameter study was carried out. Unlike hyperparameter tuning, which focuses on model-specific settings, the focus here is on the influence of external parameters such as seed values and window sizes. This study identifies how these variables influence each model's performance accuracy, thereby guiding model reproducibility and reliability.

Setting the seed value

A predetermined seed value was set for Python's random number generator so that whenever the model runs, the initial values for its parameters, such as weights and biases, are always the same. This ensures that the results can be reproduced and that all models use the same starting point. Starting values have been shown to influence model accuracy (Madhyastha and Jain 2019; Dusenberry et al. 2020). To better understand how the seed value affects model performance, an experiment was conducted that varied the seed value and compared how well the model performed in each case. The findings are presented in Fig. 6.

Figure 6 shows no clear performance trend for the DL models for all datasets as the seed value is increased from 0 to 10. The CNN–LSTM model generally outperforms the others for feature-selected datasets regardless of the seed value. The TCN model mostly surpasses the other models when no feature selection is applied. The LSTNet model consistently performs worst of the three DL models for all feature-selected datasets. The benchmark model's performance was the same for all seed values in each dataset; this is because the ARIMA model is set in a deterministic manner and therefore does not rely on random seed values for parameter estimation. Interestingly, ARIMA outperformed

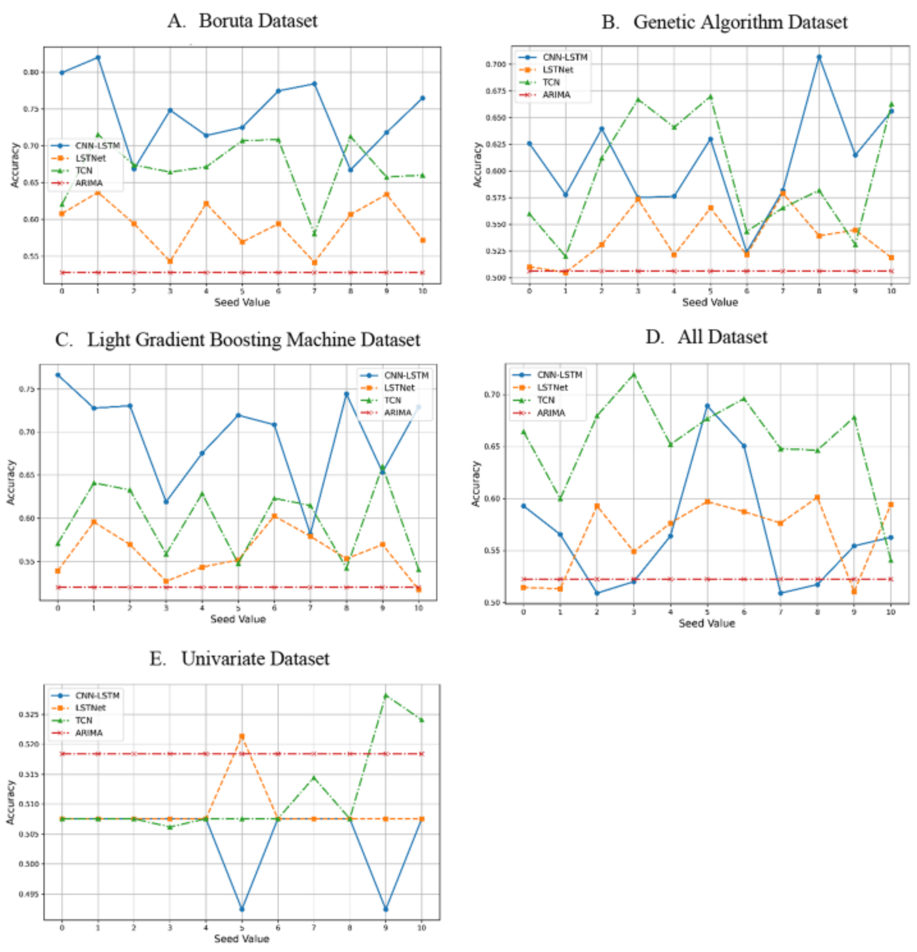

Fig. 6 Impact on model accuracy from varying the seed values of each dataset (**A** Boruta Dataset; **B** Genetic Algorithm Dataset; **C** Light Gradient Boosting Machine Dataset; **D** All Dataset; **E** Univariate Dataset)

the DL models for most scenarios involving univariate input. This superior performance can be attributed to the DL models' propensity to underfit when provided with insufficient input features.

Window sizing

The impact of changing window size on model performance was also investigated for a constant seed value. Window size refers to the number of lag days used to create input features for the models. The explored window sizes are 3, 5 7, 14, and 30 days, with the results presented in Fig. 7.

Figure 7 illustrates that model performance varies significantly with window size. For the CNN–LSTM model, no trend is observed as window size increases across datasets. Conversely, the LSTNet model shows a clear pattern, with improved performance through a window size of 5 that declines with further increases in the window size. Similarly, the TCN model initially shows improved performance through varying window sizes for the five datasets, but its performance eventually declines. In contrast to the DL models, the ARIMA model consistently underperformed across all scenarios. ARIMA

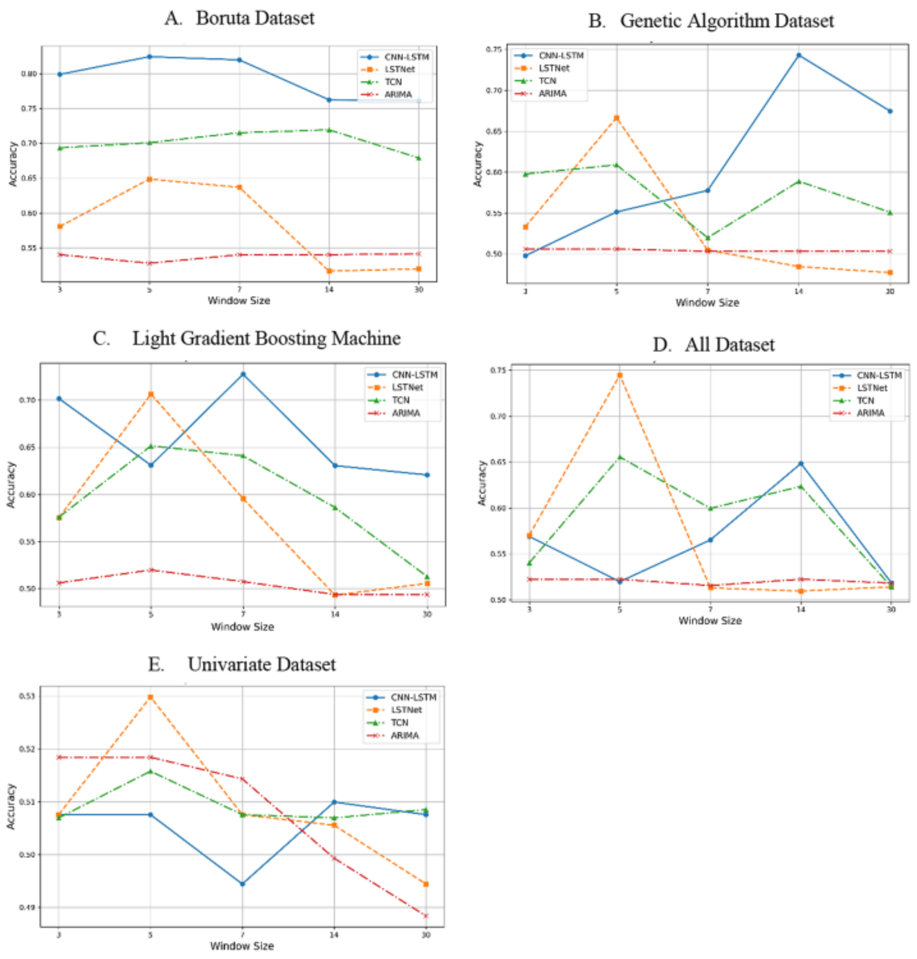

Fig. 7 Impact of changing window size on model accuracy for each dataset (**A** Boruta Dataset; **B** Genetic Algorithm Dataset; **C** Light Gradient Boosting Machine Dataset; **D** All Dataset; **E** Univariate Dataset)

showed more stable performance for multivariate scenarios but a downward trend for the univariate scenario.

Table 2 summarizes model performance for variations in the seed values and window sizes of the datasets. Interestingly, the CNN–LSTM model consistently achieved higher accuracy across all datasets except when all the data were used as input features. The Boruta + CNN–LSTM combination shows the best performance, with an overall max of 0.8244. It also demonstrates good consistency, with means of 0.7439 (±0.050) for the seed value experiment and 0.7934 (±0.030) for the window size experiment. The weakest performer was the GA + ARIMA combination model, with an overall max of 0.5061 and means of 0.5061 (±0.000) and 0.5045 (±0.001) for the seed value and window size experiments, respectively. Interestingly, poor ARIMA performance was observed across all datasets, indicating that the model's simplicity is insufficient for predicting Bitcoin price movements.

The CNN–LSTM model's performance suggests that it struggles to accurately differentiate between noise and signal, so feature selection becomes necessary. The

Table 2 Summary of deep learning performance on datasets when seed value and window size are varied

Task	Seed	Window size	Overall max
Boruta + CNN–LSTM	0.7439 (±0.050)	0.7934 (±0.030)	0.8244
GA + CNN–LSTM	0.6099 (±0.050)	0.6090 (±0.099)	0.7431
LightGBM + CNN–LSTM	0.6959 (±0.057)	0.6623 (±0.049)	0.7662
AllData + CNN–LSTM	0.5667 (±0.058)	0.5642 (±0.053)	0.6891
Univariate + CNN–LSTM	0.5048 (±0.006)	0.5059 (±0.007)	0.5099
Boruta + LSTNet	0.5930 (±0.033)	0.5806 (±0.062)	0.6488
GA + LSTNet	0.5372 (±0.026)	0.5334 (±0.078)	0.6667
LightGBM + LSTNet	0.5588 (±0.027)	0.5752 (±0.085)	0.7064
AllData + LSTNet	0.5647 (±0.036)	0.5705 (±0.101)	0.7449
Univariate + LSTNet	0.5088 (±0.004)	0.5090 (±0.013)	0.5298
Boruta + TCN	0.6701 (±0.042)	0.7016 (±0.016)	0.7194
GA + TCN	0.5958 (±0.057)	0.5734 (±0.037)	0.6699
LightGBM + TCN	0.5963 (±0.045)	0.5935 (±0.056)	0.6602
AllData + TCN	0.6546 (±0.049)	0.5867 (±0.059)	0.7194
Univariate + TCN	0.5114 (±0.008)	0.5093 (±0.004)	0.5282
Boruta + ARIMA	0.5280 (±0.000)	0. 5381 (±0.006)	0.5416
GA + ARIMA	0.5061 (±0.000)	0.5045 (±0.001)	0.5061
LightGBM + ARIMA	0.5198 (±0.000)	0.5042 (±0.011)	0.5198
AllData + ARIMA	0.5225 (±0.000)	0.5203 (±0.003)	0.5225
Univariate + ARIMA	0.5184 (±0.000)	0.5078 (±0.013)	0.5184

LSTNet model consistently underperformed on test data despite exhibiting excellent training performance. This indicates that the model is overly complex and prone to overfitting. Of the three models, TCN appears most effective at distinguishing noises and signals. It performed best when all data were used as input features. However, it suffers from underfitting when feature selection reduces the number of features.

To assess the significance of performance variations between models, Wilcoxon signed-rank and Friedman tests were conducted. The Wilcoxon signed-rank test allows for pairwise comparisons, whereas the Friedman test makes multiple comparisons (Derrac et al. 2011; Li et al. 2022). In both tests, the null hypothesis assumes no performance differences among the models. A p-value less than 0.05 indicates significant performance differences, leading to rejection of the null hypothesis. Summaries of these test results are provided in Tables 3 and 4.

Table 3 compares the outcomes of the Boruta + CNN–LSTM model with the others. The Boruta + CNN–LSTM model outperforms all others except the LightGBM + CNN–LSTM model. The two models cannot be assumed to have similar performance, however, because the value of W = 13 is not sufficiently large to be conclusive. Furthermore, the results of the Friedman test strongly suggest that one model performs differently from the rest. This implies that Boruta + CNN–LSTM is significantly better than the other models. These findings highlight how feature selection can potentially enhance ML model performance. The performance metrics of the Boruta + CNN–LSTM model are presented in Table 5.

Table 3 Wilcoxon signed-rank test comparison of model performance

Pair of models compared	Statistic	*P*-value	Conclusion
Boruta + CNN–LSTM vs GA + CNN–LSTM	2	0.0029	Reject null
Boruta + CNN–LSTM vs LightGBM + CNN–LSTM	13	0.0830	Accept null
Boruta + CNN–LSTM vs AllData + CNN–LSTM	0	0.0010	Reject null
Boruta + CNN–LSTM vs Univariate + CNN–LSTM	0	0.0010	Reject null
Boruta + CNN–LSTM vs Boruta + LSTNet	0	0.0010	Reject null
Boruta + CNN–LSTM vs GA + LSTNet	0	0.0010	Reject null
Boruta + CNN–LSTM vs LightGBM + LSTNet	0	0.0010	Reject null
Boruta + CNN–LSTM vs AllData + LSTNet	0	0.0010	Reject null
Boruta + CNN–LSTM vs Univariate + LSTNet	0	0.0010	Reject null
Boruta + CNN–LSTM vs Boruta + TCN	5	0.0098	Reject null
Boruta + CNN–LSTM vs GA + TCN	0	0.0010	Reject null
Boruta + CNN–LSTM vs LightGBM + TCN	0	0.0010	Reject null
Boruta + CNN–LSTM vs AllData + TCN	1	0.0019	Reject null
Boruta + CNN–LSTM vs Univariate + TCN	0	0.0010	Reject null
Boruta + CNN–LSTM vs Boruta + ARIMA	0	0.0010	Reject null
Boruta + CNN–LSTM vs GA + ARIMA	0	0.0010	Reject null
Boruta + CNN–LSTM vs LightGBM + ARIMA	0	0.0010	Reject null
Boruta + CNN–LSTM vs AllData + ARIMA	0	0.0010	Reject null
Boruta + CNN–LSTM vs Univariate + ARIMA	0	0.0010	Reject null

Table 4 Friedman test for overall comparison

Test	Statistic	*P*-value	Conclusion
All models	178	7.202e-28	Reject null

Table 5 Performance metrics of Boruta + convolutional neural network–long short-term memory model

Metrics	Values
Accuracy	0.8244
Recall	0.8078
Precision	0.8309
F-1 Score	0.8192
AUC-ROC	0.8242
MCC	0.6489

Backtesting results

This section discusses the outcome of the backtesting process conducted to assess the profitability of the best model, Boruta + CNN–LSTM. For simplicity, it is assumed that slippages are negligible, with no liquidity constraints. These assumptions help to focus the evaluation on the capability of the model within a controlled environment that is removed from real-world trading complexities. The MACD model was used as a benchmark for comparison. As described previously, a buy signal was triggered when the MACD line crossed above the signal line, whereas a sell signal was triggered when it fell below. The trading simulation involved the test data and covered the last

Table 6 Summary of results of trading simulation

Strategy	Annual returns (%)	Sharpe ratio	Market exposure	Volatility	MDD	Trading days
Long-only buy-and-sell	437.2461	0.1312	0.4787	3.1488	− 0.1131	203
Short-only buy-and-sell	1084.1600	0.1814	0.5213	3.1562	− 0.0582	204
Long and short	6653.7497	1.8583	0.5583	3.1759	− 0.0704	406
Long-only with price protection	− 8.6255	-0.0003	0.9369	12.3345	− 0.5761*	25
Short-only with price protection	107.7501	0.0075	0.7901	7.2039	− 0.5035*	84
Long and short with price protection	35.5278	0.0022	0.0658	9.3113	− 0.5761*	48
Long-only buy-and-sell (baseline)	− 45.0594	− 0.0023	0.5199	10.2983	− 0.7116	33
Short-only buy-and-sell (baseline)	− 29.3954	− 0.0023	0.4801	8.0177	− 0.4840	32
Long and short (baseline)	− 55.5873	− 0.0032	0.0892	9.2961	− 0.7968	65

* It might be anticipated that the MDD for a strategy involving price protection should be zero, as price protection prevents the closing of positions at a loss. However, the strategies resulted in a negative MDD because they were at a loss at the end of the simulation period (day 729)

729 days in the Boruta feature-selected dataset. The starting capital was $1000, and the simulation was designed to reinvest the compounded capital after each trade. A 30% tax rate was applied to each realized profit. Additionally, transaction costs were set at a constant rate of 0.5% to ensure the consistency of the simulation parameters. Table 6 shows the results of the backtesting process.

Table 6 shows that the "long-only buy-and-sell" strategy generated an annual return of 437%, indicating that it capitalized on predicted upward price movements. However, this strategy had a significantly negative MDD and would thus experience losses during market downturns. Similarly, the "short-only buy-and-sell" strategy generated a yearly return of 1084%, suggesting it is particularly effective during bearish markets. Interestingly, it also had a marginally better Sharpe ratio than the "long-only buy-and-sell" strategy, which suggests a better risk-adjusted return. The standout, however, was the "long-and-short" strategy. It eclipsed both previous strategies, with a staggering annual return of over 6600%. It also had the highest Sharpe ratio, suggesting that the "long-and-short" strategy effectively balanced risk and return. Additionally, this strategy had the lowest MDD, indicating resilience during market downturns. To solidify the credibility of these outcomes, a separate simulation was carried out using actual price movement values ("ground truth") and theoretically achieved 100% accuracy. The annual returns obtained were 918%, 1803%, and 20,453% for the "long-only buy-and-sell," "short-only buy-and-sell," and "long and short" strategies, respectively. The backtesting returns appear believable given the model accuracy score of 82.44%.

The Boruta + CNN–LSTM model demonstrates a precision of 0.8309, as reflected in returns achieved through the "long-only buy-and-sell," "short-only buy-and-sell," and "long-and-short" strategies. Essentially, this model excels at accurately identifying trades. Moreover, the model's recall score of 0.8078 likely influenced the number of trading days. A higher recall score suggests missed profit opportunities, thereby increasing trading frequency. Alternatively, strategies that incorporate price-protection mechanisms, such as "long-only with price protection," "short-only with price protection," and "long-and-short with price protection" resulted in lower annual returns. This implies that while these mechanisms help avoid selling at a loss during short-term price dips, they may limit opportunities for profit during market rebounds. Compared with the baseline strategies that recorded returns, the DL-based strategies show a clear advantage. However, this superior performance comes with increased risk, as indicated by volatility and MDD. The observations from this simulation align with the findings of Piravechsakul et al. (2021) and emphasize the importance of predictive models in guiding trading decisions.

Conclusions and future work

In this work, extensive experiments were conducted to investigate the performance of DL models, namely CNN–LSTM, LSTNet, and TCN, as well as an ARIMA benchmark model, in predicting short-term Bitcoin price movements. A multivariate approach incorporating on-chain data was employed for model training, with a univariate approach serving as the performance benchmark. Furthermore, the influences of various feature-selection techniques—i.e., Boruta, GA, and LightGBM—on model accuracy were scrutinized. Finally, the profitability of the best-performing model was evaluated through trading strategies.

A dataset of 87 on-chain data metrics was used in this study. Feature-selection methods were then applied to generate three additional datasets. Additionally, a univariate dataset consisting solely of price was created. Therefore, five datasets were used in total. Extensive experimentation revealed that the combination of Boruta feature selection and the CNN–LSTM model consistently outperformed other configurations, attaining an accuracy of 82.44%. Meanwhile, the weakest performer was the combination model of GA with ARIMA, which exhibited a max performance of 0.5061. This result underscored the importance of feature selection in enhancing DL model performance, which aligns with the findings of Li et al. (2020). To further solidify the credibility of CNN–LSTM model performance, Wilcoxon signed-rank and Friedman tests were conducted. The outcomes showed that the CNN–LSTM model significantly outperformed the others, including the benchmark ARIMA model. A backtesting process was then conducted to ascertain the financial viability of the model's forecasts. The results showed that a "long-and-short" trading strategy, which takes advantage of both upward and downward price movements, had better performance than all other strategies, including the benchmark strategy of momentum trading using MACD. The "long-and-short" strategy generated a 6653% return while maintaining a manageable MDD of 0.0704.

This study establishes an approach for making intelligent investment choices in the highly unpredictable cryptocurrency market. It can serve as a tool for traders and portfolio managers. Predictive models can effectively manage traders' risks by accurately forecasting price trends. Portfolio managers could use these findings to improve their investment decisions, ultimately maximizing returns.

Unique on-chain data can provide greater predictability in cryptocurrency price forecasting. With more cryptocurrency price modeling (in this case, Bitcoin pricing), price transparency should increase along with efficiency in these historically volatile markets. Thus, proper modeling and better price transparency should make it possible for securities exchanges and regulators to offer and approve new security offerings, such as a spot Bitcoin ETF, for greater adoption and liquidity while offering a vehicle for both speculation and risk management. Providing more access to various cryptocurrencies should increase adoption and use, hopefully providing greater market price efficiency within these markets.

This study's limitations should be acknowledged along with potential areas for future research. This study focused on using on-chain and price data as inputs without considering factors such as sentiment data and technical analysis indicators. Future studies could benefit from exploring the advantages of incorporating these data types for improved prediction accuracy. Additionally, the study's prediction time frame was relatively short. Future research could investigate how predictive models perform under varying market conditions to provide a comprehensive understanding of their effectiveness at capturing long-term trends.

Appendix

See Table 7.

Table 7 List of on-chain features and resulting feature selected datasets

Features	All dataset	GA dataset	LightGBM dataset	Boruta dataset	Univariate dataset
90-day coin days destroyed	✓	✓			
Adjusted circulating supply	✓				
Adjusted SOPR	✓		✓	✓	
Aggregate security spend thermocap (USD)	✓				
Average coin dormancy	✓	✓	✓		
Average spent output lifespan	✓		✓		
All exchange balance (BTC)	✓				
Balanced price (USD)	✓	✓			
Circulating supply	✓				
Coin days destroyed	✓		✓	✓	
Coin years destroyed	✓	✓			
Cumulative value days destroyed	✓	✓			
Delta cap (USD)	✓				
Difficulty ribbon compression	✓	✓			
Entity-adjusted dormancy flow	✓	✓	✓		
All exchange net position change (BTC)	✓	✓	✓		
HODL waves (1D-1W)	✓	✓	✓		
HODL waves (1 M-3 M)	✓				
HODL waves (1W-1 M)	✓	✓	✓		
HODL waves (1Y-2Y)	✓		✓		
HODL Waves (24H)	✓	✓	✓		
HODL waves (2Y-3Y)	✓		✓		
HODL waves (3 M-6 M)	✓	✓			
HODL waves (3Y-5Y)	✓				

Table 7 (continued)

Features	All dataset	GA dataset	LightGBM dataset	Boruta dataset	Univariate dataset
HODL waves (5Y-7Y)	✓				
HODL waves (6 M-12 M)	✓	✓			
HODL waves (7Y-10Y)	✓				
HODL waves (over 10Y)	✓	✓			
Inflation rate	✓		✓		
Investor capitalization	✓				
Issuance (BTC)	✓	✓			
Liveliness	✓				
Market cap to thermocap ratio	✓				
Market cap (USD)	✓	✓			
MVRV ratio	✓			✓	
Median Spent output lifespan	✓	✓	✓		
MVRV Z-score	✓	✓		✓	
Net realized profit/loss (USD)	✓	✓	✓	✓	
All exchanges net transfer volume (BTC)	✓	✓	✓		
Net unrealized profit/loss	✓	✓		✓	
Network value to transactions ratio	✓	✓	✓		
NVT signal	✓		✓		
Active addresses	✓		✓		
Non-zero balance addresses	✓	✓			
Addresses with balance over 10 K	✓				
Total addresses	✓				
New addresses	✓	✓			
Number of transfers from all exchanges	✓	✓			
UTXOs in loss	✓		✓	✓	
UTXOs in profit	✓	✓		✓	
Percent balance on all exchanges	✓				
Percent miner revenue from fees	✓				
Percent of supply last active 1 year ago	✓				
Percent of UTXOs in profit	✓	✓	✓	✓	
Percent of supply in profit	✓	✓	✓	✓	
PI cycle top indicator (MA111)	✓				
PI cycle top indicator (MA350 × 2)	✓	✓			
Drawdown from ATH	✓		✓	✓	
OHLC close (USD)	✓		✓	✓	
OHLC high (USD)	✓		✓	✓	
OHLC low (USD)	✓		✓	✓	
OHLC open (USD)	✓	✓	✓	✓	
Price (USD)	✓		✓		✓
Puell multiple	✓		✓	✓	
Realized Cap (USD)	✓				
Realized HODL ratio	✓		✓		
Realized loss (USD)	✓	✓	✓	✓	
Realized price (USD)	✓				
Realized profit/loss ratio	✓	✓	✓	✓	
Realized profit (USD)	✓		✓	✓	
RPV ratio	✓	✓	✓	✓	
Relative unrealized loss	✓	✓	✓	✓	
Relative unrealized profit	✓	✓		✓	

Table 7 (continued)

Features	All dataset	GA dataset	LightGBM dataset	Boruta dataset	Univariate dataset
Reserve risk	✓	✓	✓	✓	
Seller exhaustion constant	✓		✓		
Stock to flow deflection	✓				
Stock to flow ratio (days till halving)	✓	✓			
Stock to flow ratio (USD)	✓	✓			
Supply-adjusted CDD	✓	✓	✓	✓	
Supply-adjusted CYD	✓				
Total supply in loss (BTC)	✓	✓	✓	✓	
Total supply in profit (BTC)	✓		✓	✓	
Transaction fees (BTC)	✓				
Transfer volume between exchanges (BTC)	✓	✓	✓		
Transfer volume from all exchanges (BTC)	✓		✓		
Transfer volume to all exchanges (BTC)	✓	✓	✓		
Top exchange trading volume	✓		✓		

Abbreviations

ARIMA	AutoRegressive integrated moving average
BNN	Bayesian neural networks
CNN–LSTM	Convolutional neural network–long short-term memory
EVS	Explained variance score
GA	Genetic algorithm
GRU	Gated recurrent unit
LDA	Linear discriminant analysis
LightGBM	Light gradient boosting machine
LSTM	Long short-term memory
LSSVR	Least squares support vector regression
LSTNet	Long and short-term time-series network
MACD	Moving average convergence divergence
MALSTM-FCN	Multi-scale attention LSTM-fully convolutional network
MCAR	Missing completely at random
MCC	Matthews correlation coefficient
MDD	Maximum drawdown
MNAR	Missing not at random
ML	Machine learning
MLP	Multilayer perceptron
MZSA	Maximum Z-score among shadow attributes
QDA	Quadratic discriminant analysis
ReLU	Rectified linear unit
RF	Random forest
RNN	Recurrent neural network
ROR	Rate of Return
SVR	Support vector regression
TCN	Temporal convolutional network
TWSVR	Twin support vector regression

Acknowledgements
The authors would like to acknowledge the Laboratory for Investment and Financial Engineering and the Department of Engineering Management and Systems Engineering at the Missouri University of Science and Technology for their financial support and for the use of their facilities.

Author contributions
Conceptualization: OO and DE. Methodology: OO and DE. Data Acquisition: DE. Data Curation: OO. Model Development and Coding: OO. Validation: OO and DE. Writing – Original Draft Preparation: OO and DE. Writing and Revising: OO and DE. All authors read and approved the final manuscript.

Funding
None.

Availability of data and materials
Data and codes used in this paper are readily available from the corresponding author upon request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 19 June 2023 Accepted: 22 February 2024
Published online: 05 August 2024

References
Alonso-Monsalve S, Suárez-Cetrulo AL, Cervantes A, Quintana D (2020) Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. Expert Syst Appl 149:113250
Aras S (2021) Stacking hybrid GARCH models for forecasting Bitcoin volatility. Expert Syst Appl 174:114747
Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:180301271
Cavalli S, Amoretti M (2021) CNN-based multivariate data analysis for bitcoin trend prediction. Appl Soft Comput. https://doi.org/10.1016/j.asoc.2020.107065
Chaum DL (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24:84–90
Chen Y, Wu J, Wu Z (2022) China's commercial bank stock price prediction using a novel K-means-LSTM hybrid approach. Expert Syst Appl 202:117370. https://doi.org/10.1016/j.eswa.2022.117370
Chen S, Zhou C (2020) Stock prediction based on genetic algorithm feature selection and long short-term memory neural network. IEEE Access 9:9066–9072
Cho D-H, Moon S-H, Kim Y-H (2021) Genetic feature selection applied to KOSPI and cryptocurrency price prediction. Mathematics 9:2574
Critien JV, Gatt A, Ellul J (2022) Bitcoin price change and trend prediction through twitter sentiment and data volume. Financ Innov 8:45. https://doi.org/10.1186/s40854-022-00352-7
De Leon LGN, Gomez RC, Tacal MLG, Taylar J V, Nojor V V, Villanueva AR (2022) Bitcoin Price Forecasting using Time-series Architectures. In: 2022 International conference on ICT for smart society (ICISS). IEEE, pp. 1–6
Derbentsev V, Matviychuk A, Soloviev VN (2020) Forecasting of cryptocurrency prices using machine learning. Advanced studies of financial technologies and cryptocurrency markets. Springer, Cham, pp 211–231
Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1:3–18
Dusenberry MW, Tran D, Choi E, Kemp J, Nixon J, Jerfel G, Heller K, Dai AM (2020) Analyzing the role of model uncertainty for electronic health records. In: Proceedings of the ACM conference on health, inference, and learning. pp. 204–213
Erfanian S, Zhou Y, Razzaq A, Abbas A, Safeer AA, Li T (2022) Predicting bitcoin (BTC) price in the context of economic theories: a machine learning approach. Entropy 24(10):1487. https://doi.org/10.3390/e24101487
Glassnode (2023) Glassnode studio—on-chain market intelligence. https://studio.glassnode.com/metrics
Gyamerah SA (2021) Two-stage hybrid machine learning model for high-frequency intraday bitcoin price prediction based on technical indicators, variational mode decomposition, and support vector regression. Complexity. https://doi.org/10.1155/2021/1767708
Huang J-Z, Huang WC (2019) Predicting bitcoin returns using high-dimensional technical indicators. J Financ Data Sci. https://doi.org/10.1016/j.jfds.2018.10.001
Huang W, Nakamori Y, Wang S-Y (2005) Forecasting stock market movement direction with support vector machine. Comput Oper Res 32:2513–2522
Huynh TLD (2021) Does bitcoin react to Trump's tweets? J Behav Exp Financ 31:100546. https://doi.org/10.1016/j.jbef.2021.100546
Huynh TL (2023) When Elon Musk changes his tone, does bitcoin adjust its tune? Comput Econ 62(2):639–661
Jagannath N, Barbulescu T, Sallam KM, Elgendi I, McGrath B, Jamalipour A, Abdel-Basset M, Munasinghe K (2021) An on-chain analysis-based approach to predict ethereum prices. IEEE Access 9:167972–167989
Jang H, Lee J (2017) An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information. IEEE Access 4(6):5427–5437
Ji S, Kim J, Im H (2019) A comparative study of bitcoin price prediction using deep learning. Mathematics 7(10):898. https://doi.org/10.3390/math7100898
Kalariya V, Parmar P, Jay P, Tanwar S, Raboaca MS, Alqahtani F, Tolba A, Neagu B-C (2022) Stochastic neural networks-based algorithmic trading for the cryptocurrency market. Mathematics 10:1456
Kim G, Shin D-H, Choi JG, Lim S (2022) A deep learning-based cryptocurrency price prediction model that uses on-chain data. IEEE Access 10:56232–56248
Kraaijeveld O, De Smedt J (2020) The predictive power of public Twitter sentiment for forecasting cryptocurrency prices. J Int Finan Markets Inst Money 65:101188
Kukacka J, Kristoufek L (2023) Fundamental and speculative components of the cryptocurrency pricing dynamics. Financ Innov. https://doi.org/10.1186/s40854-023-00465-7
Kursa MB, Rudnicki WR (2010) Feature selection with the Boruta package. J Stat Softw 36:1–13
Lai G, Chang W-C, Yang Y, Liu H (2018) Modeling long-and short-term temporal patterns with deep neural networks. In: The 41st international ACM SIGIR conference on research & development in information retrieval. pp. 95–104

Li H, Hua J, Li J, Li G (2020) Stock forecasting model FS-LSTM based on the 5G Internet of things. Wirel Commun Mob Comput 2020:1–7
Li M-W, Xu D-Y, Geng J, Hong W-C (2022) A hybrid approach for forecasting ship motion using CNN–GRU–AM and GCWOA. Appl Soft Comput 114:108084
Livieris IE, Kiriakidou N, Stavroyiannis S, Pintelas P (2021) An advanced CNN–LSTM model for cryptocurrency forecasting. Electronics 10:287
Loginova E, Tsang WK, van Heijningen G, Kerkhove L-P, Benoit DF (2021) Forecasting directional Bitcoin price returns using aspect-based sentiment analysis on online text data. Mach Learn. https://doi.org/10.1007/s10994-021-06095-3
Lu W, Li J, Li Y, Sun A, Wang J (2020) A CNN–LSTM-based model to forecast stock prices. Complexity 2020:1–10
Madhyastha P, Jain R (2019) On Model Stability as a Function of Random Seed. In: Proceedings of the 23rd conference on computational natural language learning (CoNLL). Association for computational linguistics, Hong Kong, China, pp. 929–939
Nakamoto S (2008) Bitcoin: a peer-to-peer electronic cash system. Bitcoin–URL: https://bitcoin.org/bitcoin.pdf
Ortu M, Uras N, Conversano C, Bartolucci S, Destefanis G (2022) On technical trading and social media indicators for cryptocurrency price classification through deep learning. Expert Syst Appl 198:116804
Ouyang Z, Ravier P, Jabloun M (2022) Are deep learning models practically good as promised? A strategic comparison of deep learning models for time series forecasting. In: 2022 30th European signal processing conference (EUSIPCO). IEEE, pp. 1477–1481
Park J, Seo Y-S (2022) A deep learning-based action recommendation model for cryptocurrency profit maximization. Electronics 11:1466
Passalis N, Avramelou L, Seficha S, Tsantekidis A, Doropoulos S, Makris G, Tefas A (2022) Multisource financial sentiment analysis for detecting Bitcoin price change indications using deep learning. Neural Comput Appl 34:19441–19452. https://doi.org/10.1007/s00521-022-07509-6
Piravechsakul P, Kasetkasem T, Marukatat S, Kumazawa I (2021) Combining technical indicators and deep learning by using lstm stock price predictor. In: 2021 18th International conference on electrical engineering/electronics, computer, telecommunications and information technology (ECTI-CON). IEEE, pp. 1155–1158
Rafi M, Mirza QAK, Sohail MI, Aliasghar M, Aziz A, Hameed S (2023) Enhancing cryptocurrency price forecasting accuracy: a feature selection and weighting approach with Bi-directional LSTM and trend-preserving model bias correction. IEEE Access 11:65700–65710. https://doi.org/10.1109/ACCESS.2023.3287888
Ranjan S, Kayal P, Saraf M (2023) Bitcoin price prediction: a machine learning sample dimension approach. Comput Econ 61(4):1617–1636
Resta M, Pagnottoni P, Giuli MED (2020) Technical analysis on the Bitcoin market: trading opportunities or investors' pitfall? Risks. https://doi.org/10.3390/risks8020044
Sebastião H, Godinho P (2021) Forecasting and trading cryptocurrencies with machine learning under changing market conditions. Financ Innov 7:1–30
Shahzad SJ, Anas M, Bouri E (2022) Price explosiveness in cryptocurrencies and Elon Musk's tweets. Financ Res Lett 1(47):102695
Smales LA (2019) Bitcoin as a safe haven: Is it even worth considering? Financ Res Lett 30:385–393
Tripathi B, Sharma RK (2023) Modeling Bitcoin prices using signal processing methods, bayesian optimization, and deep neural networks. Comput Econ 62(4):1919–1945. https://doi.org/10.1007/s10614-022-10325-8
Valencia F, Gómez-Espinosa A, Valdés-Aguirre B (2019) Price movement prediction of cryptocurrencies using sentiment analysis and machine learning. Entropy 21:589
Wołk K (2020) Advanced social media sentiment analysis for short-term cryptocurrency price prediction. Expert Syst 37:e12493
Wu C-H, Lu C-C, Ma Y-F, Lu R-S (2019) A new forecasting framework for Bitcoin price with LSTM. In: IEEE International conference on data mining workshops, ICDMW. pp. 168–175
Yang Z, Fantazzini D (2022) Using crypto-asset pricing methods to build technical oscillators for short-term Bitcoin trading. Information. https://doi.org/10.3390/info13120560
Ye Z, Wu Y, Chen H, Pan Y, Jiang Q (2022) A stacking ensemble deep learning model for Bitcoin price prediction using twitter comments on Bitcoin. Mathematics 10:1307
Zhong X, Enke D (2017a) A comprehensive cluster and classification mining procedure for daily stock market return forecasting. Neurocomputing 267:152–168
Zhong X, Enke D (2017b) Forecasting daily stock market return using dimensionality reduction. Expert Syst Appl 67:126–139
Zhu Y, Ma J, Gu F, Wang J, Li Z, Zhang Y, Xu J, Li Y, Wang Y, Yang X (2023) Price prediction of Bitcoin based on adaptive feature selection and model optimization. Mathematics 11:1335