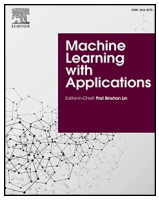
Machine Learning with Applications 20 (2025) 100674

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa



Bitcoin price direction prediction using on-chain data and feature selection

Ritwik Dubey [a], David Enke [b,*]

[a] Laboratory for Investment and Financial Engineering, Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, 223 Engineering Management, 600 W. 14th Street, Rolla, MO 65409-0370, USA
[b] 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

ARTICLE INFO

Keywords:
Bitcoin
On-chain data
Feature selection
CNN-LSTM
TCN

ABSTRACT

Bitcoin is the most traded cryptocurrency by volume and market cap. A number of scholars have directed their research towards characterizing Bitcoin's speculative behavior using a myriad of techniques such as technical analysis, price regression, and direction classification. For this work, research is conducted using the relatively nascent technique of on-chain data analysis. The goal of this research is to evaluate Bitcoin's on-chain data in predicting future price direction. First, a classification process of on-chain data features that helps the reader understand their relevance is proposed. To address the curse of dimensionality, feature selection algorithms such as L1 regression, Boruta, and the dimensionality reduction algorithm Principal Component Analysis (PCA) are utilized. The research then explores advanced neural networks for next day price direction prediction, including the Convolutional Neural Network-Long-Short Term Memory (CNN-LSTM) and the Temporal Convolutional Network (TCN). Neural network models and trading strategies are then compared based on their return statistics. A comparative analysis of feature selection, learning model performance, and trading strategy performance is also conducted. Results from the research show that the Boruta feature selection algorithm combined with the CNN-LSTM model performs best compared to other combinations with a prediction accuracy of 82.03 % over the testing period. In addition, the on-chain features within the category, realized value, and unrealized value classifications have higher predictive powers for next day price direction prediction. Finally, during trade simulations, the CNN-LSTM model with a Long-Short strategy had an annualized return of 1682.7 % and a Sharpe Ratio of 6.47.

1. Introduction

Bitcoin is a popular cryptocurrency that uses blockchain technology. Since its inception in 2008 by Satoshi Nakamoto (Nakamoto, 2008), it has seen gains in social acceptance among the general population and financial institutions. Bitcoin users aspire to achieve complete decentralization of transactions. Bitcoin participants establish trust by recording transactions on blocks using hash functions. Bitcoin's distributed ledger system records all transactions and therefore allows an observer to infer the transaction history. Information such as coins exchanged, time, wallet address, fees, etc., can be extracted from the blocks. Both blockchain technology and Bitcoin have spurred new fields of research in speculative trading, economics, cryptography, and machine learning (Fang et al., 2022).

Bitcoin and other cryptocurrencies give an impression of being assets that investors speculate with and trade on exchanges. However, the question of Bitcoin being an asset in the traditional sense is not a settled debate. (Marthinsen & Gordon, 2022) identified factors such as inelastic supply/demand and the lack of an independent price index as potential reasons for high price volatility. (Gourieroux et al., 2020) demonstrated the usefulness of bivariate predictive density analysis in predicting outset and proximity of bubble burst in Bitcoin price, drawing further parallels with S&P 500 and NASDAQ price series. A speculative asset with high volatility can potentially translate into a profitable trading strategy with the aid of sophisticated predictive modeling techniques.

To understand how on-chain data can help in building a profitable trading strategy, it is important to first understand what they represent. Bitcoin and other cryptocurrencies have a common goal of achieving decentralization of financial transactions. Decentralization itself is a two-fold challenge; how transactions will be validated and how the transaction ledger is maintained in the absence of a central system. Bitcoin offers a solution for these challenges by using a combination of

* Corresponding author.
E-mail addresses: rdqhk@mst.edu (R. Dubey), enke@mst.edu (D. Enke).

https://doi.org/10.1016/j.mlwa.2025.100674

Available online 20 May 2025

flexible hashing rate and controlling the supply of new coins.

A peer-to-peer network of Bitcoin miners have the dual role of validating blockchain transactions and generating Bitcoins. Blockchains by design record transaction history, with the miners validating all transactions on the blockchain. Every new and valid transaction on the Bitcoin network is added to the blockchain as a unit called a "block" - an irreversible data record using a hashing function. It is a Bitcoin miner's work that validates the transaction using a proof of work validation approach. By design, such proof of work requires processing blocks in chronological order. The validity of this new block is established by miner consensus. On-chain data captures quantitative attributes of the transaction history embedded in the Bitcoin blockchain.

Like any other publicly traded asset, Bitcoin price prediction becomes a time-series forecasting problem. Two categories can be used to describe the published literature in this field of prediction: model-based and feature-based (Jaquart et al., 2020).

Model-based prediction: Bitcoin is an asset that is known for its high volatility and potential asymmetric returns when compared against more traditional assets such as stocks, bonds, or commodities. Like many time series, the Bitcoin price series is non-stationary, such that its statistical properties evolve, often displaying a skewed, non-Gaussian, heavy tail distribution, as well as some level of autocorrelation over both short-term and long-term growth and retracement periods. While the Bitcoin price does not display traditional time series seasonality like agricultural commodities, there are some seasonal effects that have been displayed as a result of its four-year halving cycle, where the amount of Bitcoin provided to miners is cut in half, thereby slowing the annual supply of new Bitcoin; volatility clustering often increases before and after these structural breaks, causing Bitcoin to have a heavy tail distribution, and even experience periods of exponential growth and decline. Other regulatory and market sentiment-driven moves, as well as the adoption and use of new vehicles for owning and trading Bitcoin - such as Bitcoin futures and ETF products, have been less predictable, but have also impacted its liquidity and behavioral-based price moves. Finally, the correlation of Bitcoin with other assets, in particular the U.S. Dollar, stocks, and traded stock index products, has varied over time, resulting in Bitcoin acting at times as a highly volatile version of the asset it is correlated with. These changing periods of cointegration have made it difficult to model Bitcoin prices based on the movements of other assets.

Initial modeling of Bitcoin often began using a regression-based modeling technique. For instance, (Kaiser, 2019) used popularly known time-series analysis tools like autoregressive (AR) processes, the univariate moving average (MA), simple exponential smoothing (SES), and autoregressive integrated moving average (ARIMA) to investigate seasonality properties of the Bitcoin price data. The author did not find strong calendar effects in Bitcoin returns. Although some effects of seasonality were discovered for volume, volatility, and spreads, they were not significant enough to trade on. Mixed causal-noncausal autoregressive (MAR) models for point and density forecasting can take a variable's lead and lag state into account for forecasting. (Hencic & Gourieroux, 2015) detected non-causal components in Bitcoin's pricing bubbles. (Jasiak & Gourieroux, 2015) developed closed form estimators of predictive density. This forecasting method provides simulation-based prediction intervals at multiple finite horizons. (Gourieroux et al., 2018; Gourieroux and Jasiak, 2016) then evaluated this approach in terms of forecast accuracy, error behavior, and goodness of fit of estimated predictive density. (Ranjan et al., 2023) evaluated Logistic Regression and XGBoost for Bitcoin price prediction. They found 64.84 % accuracy using Logistic Regression for daily price prediction and 59.40 % accuracy from XGBoost on 5-minute interval price prediction.

Artificial neural networks (ANN), specifically Recurrent Neural Networks (RNN) and deep learning models – such as Long Short-Term Memory (LSTM) networks, have shown potential for addressing common time series forecasting challenges such as non-stationarity, heavy tails, autocorrelation, seasonality, and non-Gaussian characteristics. LSTM models do not require the time series to be stationary (unlike ARIMA models) but instead attempt to capture longer-term dependencies in the data. Heavy tails and extreme values can also be captured without explicitly modeling them. RNN and LSTM models are also designed to help capture temporal dependencies, including autocorrelation, as well as capture seasonality in the data. Finally, ANNs do not need to assume a specific distribution, often allowing them to be more successful when modeling non-Gaussian time series.

(McNally et al., 2018) concluded in their work that RNN and LSTM neural networks are effective in Bitcoin price prediction because of their ability to identify temporal features. In their experiments, they found LSTM achieved an accuracy of 52.78 % for predicting price direction. (Koo & Kim, 2021) used LSTM with enhancements like Flattening Distributions Strategy (FDS) observed an accuracy of 56.27 %. (Dutta et al., 2020) proposed a Gated Recurrent Unit (GRU) neural network architecture for Bitcoin price prediction and provided empirical evidence of better model performance compared to LSTM. Their experiments demonstrated RMSE of 0.019 for GRU network compared to 0.024 for LSTM network.

(Mudassir et al., 2020) applied a machine learning (ML) model for Bitcoin price prediction over multiple day periods: 1-day, 7-days, 30-days and 90-days. They used a variety of models, such as ANN, Stacked Artificial Neural Network (SANN), Support Vector Machines (SVM), and LSTM for price prediction and discovered that LSTM had the best performance for a next-day forecast with 65 % accuracy. (Ortu et al., 2022) experimented with a Multi-Layered Perceptron (MLP), Convolutional Neural Network (CNN), LSTM, and Attention Long Short Term Memory-Full Convolutional Network (ALSTM-FCN) models on a feature set comprised of technical, trading, and social indicators for Bitcoin and Ethereum (another popular cryptocurrency). For daily classification, (Ortu et al., 2022) observed 67 %-84 % accuracy in prediction. (G. Kim et al., 2022) used Bitcoin on-chain data and a Self-Attention Based Multiple Long Short Term Memory (SAM-LSTM) model to perform Bitcoin price prediction. The prediction pipeline used a Pruned Exact Linear Time (PELT) algorithm for identifying change points in the dataset and used it to normalize the dataset. The published results show, among other metrics, Mean Absolute Error (MAE) of 0.3462 for the PELT normalized SAM-LSTM model. (Li, et al., 2022) used features generated by variational mode decomposition (VMD) and a hybrid bidirectional deep-learning model for forecasting the daily price change of Bitcoin. VMD was used to decompose price data in low, medium, and high frequency modes. This new model, named VMD-LMH-BiLSTM, produced a price direction prediction accuracy of 81.7 %.

Feature-based prediction: A group of publications have explored signals from alternative data on Bitcoin prices. (Jaquart et al., 2021) tried to expand on the feature sets in search for better predictors for Bitcoin price direction prediction. They included features representative of different categories. As an example, features such as Bitcoin price returns were used for the technical category, the MSCI World and S&P 500 index returns were used for the asset-based category, and Twitter sentiment was used for the sentiment-based category. Their search has reinforced the predictive nature of features in the technical category as the alternatives had little or no effect on prediction. (Philippas et al., 2019) examined the effect of social media on Bitcoin prices, specifically signals derived from Twitter and Google Trends. They identified a partial relationship between Bitcoin prices and the momentum of media attention on social media. (Shen et al., 2019) studied the effect of the number of tweets with the word "Bitcoin" on Bitcoin prices. A causal relation was discovered between the number of such tweets and the next day Bitcoin volume traded. (Kim et al., 2016) studied the effect of user comments on online platforms and discovered Bitcoin price correlated with number of positive comments on social media. (Critien et al., 2022) used Twitter sentiment to predict both price direction and price change magnitude with an accuracy of 64.18 % for a 1-day lag closing price

target. They also discovered that a 2-layer Bidirectional LSTM (BiLSTM) outperformed CNN and LSTM for price direction prediction. For next day closing price change prediction, they found a 2-layer CNN outperformed other models with accuracy of 57 % over 10 classes of price change (classes 1-5 show downtrend and classes 6-10 show uptrend).

The following section provides an overview of the proposed prediction pipeline, along with further supporting background and descriptions for the on-chain data and data preprocessing, methods for feature selection, machine learning models, and trading strategies deployed to test the models and provide for a comparative analysis.

2. Prediction Pipeline

The key modules of the proposed prediction pipeline are shown in Fig. 1. This section describes the functionality of each module.

2.1. On-Chain data

Multiple data vendors are in the business of analyzing the raw information on the Bitcoin blockchain and providing a curated dataset of on-chain metrics for public use. For this research, the daily Bitcoin on-chain dataset was obtained from Glassnode[1]. The dataset ranges from 12-13-2012 to 05-14-2023, with 196 features. The time period from 12-13-2012 to 04-13-2021 was used for in-sample training and testing of the model. In-sample training and testing data was split into 80 % training and 20 % testing with randomization turned off. Turning off randomization is important for time data so that sequential information is preserved and training is not polluted by future information. The time period from 04-19-2021 to 05-14-2023 was used for the out-sample validation. A list of all features and their categories is provided in Appendix A.

Understanding the intuition behind these features might not be easily given the large number of features. Therefore, a classification system is devised to help understand the value these features bring to Bitcoin price prediction. This classification system is generic enough so that it could be applied to on-chain metrics for other cryptocurrencies. The five classes of on-chain data, or factors, are classified as (1) Mining, (2) Realized Value, (3) Unrealized Value, (4) Stationarity, and (5) Activity. The underlying idea of creating these categories is to identify the key attributes that offer predictive value to a cryptocurrency.

The Mining category group has features that quantify a miners influence on the state of cryptocurrency network. Miners perform validation of blockchain transaction and are rewarded with cryptocurrency that is termed as miner fees. Features like “miner revenue fees” and “fees total“ show revenue generated from mining activity.

The Realized Value category group has features that represent the change in realized value as a result of trading cryptocurrencies. When cryptocurrencies are bought or sold, the holders incur profit or loss. Realized profit, realized loss, and the spent output profit ratio quantify if holders collectively made transactions at a profit or loss.

The Unrealized Value category group has features that show the inherent value of the cryptocurrency network. Features derived from market cap take in account the theoretical market capitalization based on coins in circulation and their current value. For example, features derived from unspent transaction output (UTXO) are considered unrealized value based on the current price.

The Stationarity category group captures the holding behavior of coin owners. Owners may feel incentivized to hold on to coins in hope for a higher future value. Features derived from coin days destroyed, such as liveliness and dormancy, capture the owner’s intent to hold on to coins. The HODL wave, on the other hand, derives a comparative holding pattern among accounts of different lifespans.

The Activity category group quantifies the transactional nature of cryptocurrency trading among addresses. Exchange related features like exchange balance show the change in value to and from addresses that are deemed exchange addresses.

Table 1 describes how features within these five categories correlate with the target. Collectively, the Realized Value, Unrealized Value, and Stationarity categories have the highest correlation with the target. This is evident from the mean, max, and min correlation numbers that show higher magnitude compared to other categories. Table 2 provides a list of the top 10 features with the highest target correlation; the highlighted values hold a high correlation value with the target. Section 3.1 discusses the results from feature selection algorithms used to reduce feature set dimension. This table will later be useful in understanding why feature selection techniques chose certain features over others.

2.2. Data preprocessing

In the data processing stage, the dataset is processed to make it suitable for training the machine learning model. Key steps in preprocessing include encoding contextual information for generating new features, impute missing data, and normalization. The on-chain dataset consists of a few features that came in existence in recent years or implied large time periods, such as one year or more. This implies a short history for the purpose of the training dataset. Therefore, such features were dropped from the dataset. Appropriate data imputations were made for missing data. For data normalization, the features were transformed to a zero mean and unit standard deviation scale.

2.2.1. Target definition

The goal of the prediction model is to find the direction of the close price for the next 24 hour period. One way of defining the target using an OHLC (open, high, low, close) dataset is as follows: the target is +1 if the close price on the next day is higher than the close price of the current date, otherwise -1. This rule can be represented mathematically as:

$$T_d = sgn(C_{d+1} - C_d) \tag{1}$$

where T_d is the prediction target on date d, C_d is the close price on date d, and C_{d+1} is the close price on date d+1. This target definition is widely used in modeling traditional asset classes like equities where historical price information is available in an OHLC format.

2.3. Feature Selection

The on-chain dataset suffers from the curse of dimensionality since it contains 196 features and close to 5000 data points. For better model training and performance it is important to reduce the feature set. To achieve this, three dimension reduction techniques were experimented with for feature selection[2]: LASSO regression, Boruta, and Principal Component Analysis (PCA).

2.3.1. LASSO regression

LASSO or L1 regularization and feature selection can be expressed using a linear model as

$$Yi = \beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 \ldots + x_{ik}\beta_k + e \tag{2}$$

where $i = 1, 2, \ldots n$ is the number of samples. $\beta_0, \beta_1, \beta_2 \ldots \beta_k$ are the

[1] https://glassnode.com/ crypto currency analytics platform provides variety of on-chain metrics for multiple cryptocurrencies.

[2] Two other dimension reduction algorithms were considered: mutual information and Recursive Feature Elimination (RFE). These algorithms were considered unsuitable for the purpose of this work. Mutual information does not consider interactions between features. RFE is typically useful for datasets with a moderate number of features, unlike the dataset used for this prediction. Also, RFE requires choosing a predefined number of features which introduces human bias in the process.

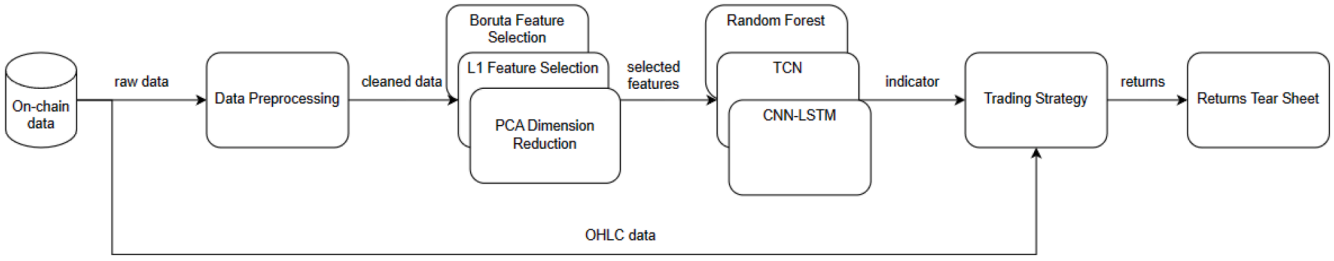


Fig. 1. Prediction Pipeline.

Table 1
Correlation statistics with target for feature categories.

	Mining	Activity	Stationarity	Realized Value	Unrealized Value
count	17	98	27	29	24
mean	0.0138	0.0108	0.0082	0.0191	0.0254
std	0.0353	0.0247	0.0406	0.0844	0.0658
min	-0.0735	-0.0539	-0.0558	-0.1792	-0.0972
25 %	-0.0103	-0.0043	-0.0179	-0.0139	-0.0121
50 %	0.0164	0.0101	0.0003	-0.0030	0.0142
75 %	0.0478	0.0270	0.0207	0.0494	0.0703
max	0.0529	0.0788	0.1076	0.2175	0.1523

Table 2
Features with highest correlation with the target. Refer to Appendix A for feature descriptions.

Feature Name	Target Correlation	Category
realized_profit_loss_ratio	0.217461	realized
profit_relative	0.187239	realized
loss_sum	-0.17916	realized
sopr_adjusted	0.157824	realized
utxo_profit_relative	0.15228	unrealized
profit_sum	0.146239	realized
sopr	0.143034	realized
price_drawdown_relative	0.119369	unrealized
mvrv	0.118377	unrealized
dormancy_flow	0.107622	stationarity

coefficients of the linear regression model and k being the number of explanatory variables. e is the error with zero mean, unit variance and normal distribution. The equation can be represented in vector form as:

$$\boldsymbol{Y} = \boldsymbol{X}\beta + e \tag{3}$$

Using the representation from (Ng, 2004; Tibshirani, 1996), the L1 optimization problem can be defined as minimizing

$$\left(\frac{||Y - X\beta||^2}{n}\right) \tag{4}$$

subject to:

$$\sum_{j=1}^{k} ||\beta|| < \boldsymbol{t} \tag{5}$$

where t is upper bound for sum of coefficients. This problem is equivalent to parameter estimation in following form:

$$\beta(\lambda) = \underset{\beta}{\text{argmin}}\left(\frac{||Y - X\beta||^2}{n} + \lambda||\beta||\right) \tag{6}$$

This equation can be expanded to the following form:

$$\beta(\lambda) = argmin\left\{\frac{1}{2}\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{k}|\beta|\right\} \tag{7}$$

The parameter $\lambda \geq 0$ controls the regularization penalty, the larger the value, the larger the parameter shrinkage. Depending on the value of λ, $\beta(\lambda) = 0$ for some value of j (a specific feature). As shown in Fig. 2 below, both LASSO and Ridge regression methods find the point on the constraint region where the least square error function touches it. LASSO has a diamond shaped constraint region based on $|\beta_1| + |\beta_2| \leq t$, whereas Ridge regression has a circle as its constraint region based on the ${\beta_1}^2 + {\beta_2}^2 \leq t$ constraint equation. The ellipses represent the border of the least square error function.

2.3.2. Boruta feature selection

Boruta is a feature selection algorithm built upon a Random Forest classification (Kursa & Rudnicki, 2010). Boruta tests individual features iteratively and eliminates them from the selected feature list if they are less relevant compared to random probes. (Kursa & Rudnicki, 2010) described the Boruta algorithm in following steps:

1. Add "shadow attributes" to the dataset that are copies of existing attributes.
2. Shuffle the shadow attributes to ensure they are not correlated to the target. These attributes act as the random probes referred earlier.
3. Random Forest classification is performed on this dataset which has both original and shadow attributes. Then z-score is computed for the attributes.
4. Maximum z score is computed for shadow attributes and termed as MZSA. Each feature's z score is compared against MZSA. Features with higher z score are labeled important and features with lower z score are labeled unimportant.
5. Each feature's importance is evaluated using two-sided MZSA test. For features with significantly less MZSA are eliminated from selected feature set. Features with significantly high MZSA are retained in selected feature set.
6. All shadow attributes are dropped and the process is repeated again until all features in feature set are labeled important or unimportant or a predefined stop criteria is achieved.

2.3.3. Principal component analysis

Principal Component Analysis (PCA) is a well-known technique to reduce feature dimensions. PCA attempts to find new features that are a linear transformation of the original dataset and then capture the variance in the dataset in lower dimension. The "Principal Components" in PCA are the eigenvectors and eigenvalues that express the properties of linear transformation and variance. Readers are encouraged to refer to publications on PCA, such as (Shlens, 2014), that are focused on building the intuition behind this popular technique.

2.4. Machine learning models

To generate indicators based on selected features, multiple modeling techniques were evaluated. Autoregressive techniques like ARIMA were not used since the Bitcoin time-series is non-stationary and heteroskedastic. Neural network models were considered suitable given that they have a large number of tunable parameters and therefore have the

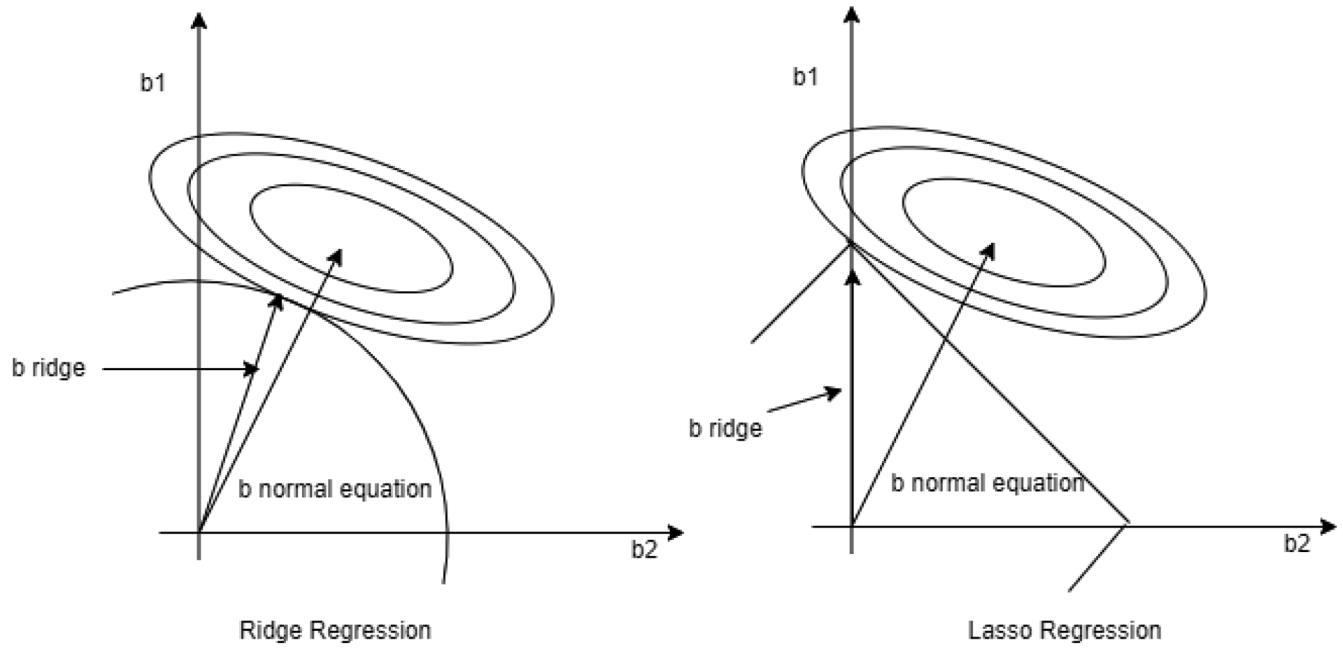


Fig. 2. Ridge regression and LASSO regression error profiles and constraint regions (Tibshirani, 1996).

ability to adapt to the properties of the time series. For a financial time series prediction problem the model should have the capability to pick temporal as well as dimensional attributes. For these reasons two neural network models were selected: Temporal Convolutional Network (TCN) and Convolutional Neural Network-Long Short Term Memory (CNN-LSTM). Both TCN and CNN-LSTM have convolutional network units and sequential network units that capture dimensional and temporal attributes. To prevent overfitting the model employs a dropout layer with dropout rate defined in Sections 2.4.2 and 2.4.3. The training process for both neural networks use an early stopping mechanism with validation loss monitoring and patience as 100. For a model performance baseline, the Random Forest model is chosen. The following subsections describe the key specifics for each model.

2.4.1. Random forest

Random Forest, first proposed by (Ho, 1995), is a machine learning algorithm that is widely used for classification and regression tasks. It is an ensemble learning method that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forests are known for their high accuracy, robustness to noise and overfitting, and ability to handle high-dimensional data. They are also relatively easy to use and require little tuning of hyperparameters.

The decision trees are recursive partition algorithm that partition dataset in two groups based on certain criteria. The predictor variables selected for partitioning can be considered as an optimization problem. A commonly used criterion for optimization is entropy. For each decision node, the split is made such that entropy is maximized to gain maximum information.

The decision trees suffer from the problem of overfitting. (Ho, 1995) introduced the idea of random subspace method to increase generalization accuracy. The random forest model is an ensemble learning process that is built on bootstrapped samples and average predictions over individual decision trees. (Hastie et al., 2009) describes the Random Forest algorithm using the following pseudo code:

1. For b = 1 to B (number of trees):

a. Draw a bootstrap sample Z* of size N from training data.
b. Grow a random forest tree T_b to the bootstrap data for each terminal node of the tree until minimum node size n_{min} is reached.
   i. Select m variables at random from all p variables.
   ii. Pick the best variable/split-point among the m variables.
   iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$

To make predictions at a new point x:

Regression:

$$f_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{16}$$

where $f_{rf}^B(x)$ is average value of all decision trees for input value x.

Classification: Let $C_b(x)$ be the class prediction of the bth random forest tree.

Then:

$$C_{rf}^B(x) = majority\ vote\ \{C_b(x)\}_1^B \tag{17}$$

where $C_{rf}^B(x)$ is the class determined for input x by majority vote across all decision trees.

2.4.2. CNN-LSTM

The CNN-LSTM model brings together the best features of both CNN and LSTM models. The CNN network was proposed by (Lecun et al., 1998) and consists of two parts, the convolutional layer and the pooling layer. The convolutional layer helps to capture highly dimensional feature information. The mathematical representation of convolutional layer operation is:

$$l_t = \tanh(x_t * k_t + b_t) \tag{18}$$

where x_t in input vector, b_t is bias of the kernel, k_t is the weight vector of the kernel, tanh is the activation function and l_t is the output vector after convolution operation. The pooling layer helps reduce feature dimensions by combining the convolutional layer output.

LSTM model was proposed by (Hochreiter & Schmidhuber, 1997) to solve the exploding or vanishing gradients in Recurrent Neural Networks (RNN). LSTM is widely used in sequential prediction problems like speech recognition, text analysis, and sentiment analysis. (Lu et al., 2020) explained the workings of the LSTM cell. Fig. 3 shows a LSTM cell

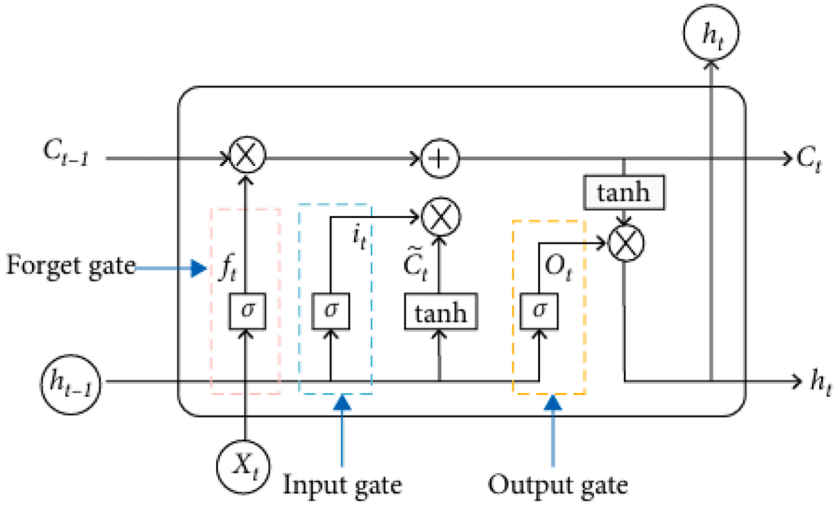


Fig. 3. LSTM memory cell (Lu et al., 2020).

that has three units: the forget gate, the input gate, and the output gate. The LSTM output calculation is described in following steps:

1. The output value of the last time step and input value of the current time step is sent to the forget gate. The output value of the forget gate is found after applying the following equation:

$$f_t = \sigma\left(W_f \,.\, [h_{t-1}, x_t] + b_f\right) \tag{19}$$

where the value range of f_t is (0,1), W_f is the weight of the forget gate, x_t is the input value at current time, σ is the activation function, and h_{t-1} is the output value at previous time.

2. The output value of last time step and input value of current time step are sent to the input gate. The output value and candidate cell state of the input gate is obtained using the following expressions:

$$i_t = \sigma\ (W_i \,.\, [h_{t-1},\ x_t] + b_i) \tag{20}$$

$$C'_t = \tanh\ (W_c \,.\, [h_{t-1}, x_t] + b_c) \tag{21}$$

The value range of i_t is (0,1), W_i is the weight of the input gate, b_i is the bias of the input gate, W_c is the weight of the candidate input gate, and b_c is the bias of the candidate input gate.

3. Update the current cell state using the following expression:

$$C_t = f_t * C_{t-1} + i_t * C'_t \tag{22}$$

4. The output $h_{t-1}$and input x_t are received as input values of the output gate at time t and the output o_t of the output gate is obtained using the following expression:

$$o_t = \sigma(W_o \,.\, [h_{t-1}, x_t] + b_o) \tag{23}$$

where W_o is the weight of the output gate and $b_o$is the bias of output gate.

5. The output value of the LSTM is obtained by calculating following expression:

$$h_t = o_t * \tanh\ (C_t) \tag{24}$$

(Ortu et al., 2022) has proved the utility of CNN and LSTM in timeseries forecasting. (Lu et al., 2020) experimented with CNN-LSTM for financial time series forecasting where CNN layers are stacked with LSTM layers. This research extends the usage of CNN-LSTM to Bitcoin timeseries prediction.

The CNN-LSTM model parameters are shown in Fig. 4. The first layer is a 1-dimensional convolutional layer with filter size 8 and kernel size 3, with a Rectilinear Linear Unit (ReLU) activation function. The first convolutional layer is followed by a batch normalization layer and a dropout layer with dropping probability of 0.5. The second layer is a LSTM layer with 32 units and ReLU activation. Batch normalization and a dropout layer with dropping probability of 0.5 also follows this layer. The third layer is also a LSTM layer with 64 units and ReLU activation. The penultimate layer is a dense layer with 16 units and ReLU activation. The network is terminated with a single unit Dense layer with sigmoid activation.

2.4.3. TCN

The TCN network proposed by (Lea et al., 2017) uses a hierarchy of temporal convolutions to perform spatiotemporal feature extraction. (Bai et al., 2018; Lea et al., 2017) demonstrated that TCN models are capable of capturing long-range dependencies and train faster compared to LSTM or other recurrent network models.

The distinguishing characteristics of TCN are: 1) convolutions in the architecture are causal, i.e., no future information leaks to the past, and 2) the architecture can take sequence of any length and map it to the output of the same length, like an RNN. There are four key discussion points for TCN network: sequence modeling, causal convolution, dilated convolution, and residual connections. These points are described below:

Sequence Modeling: The nature of the sequence modeling task is such that for an input sequence $x_0, \ldots x_t$, the corresponding output sequence is $y_0, \ldots y_t$ at each time. The key constraint is that for output y_t at some time t, only those inputs used are those previously observed, i.e. $x_0, \ldots x_t$, and not future observations like x_{t+1}.

Causal Convolutions: The first distinguishing characteristic of the TCN is that no future information leaks to the past. This is accomplished using causal convolution, where output at time t is convolved with elements at time t and earlier in the previous layer. The second distinguishing characteristic of the TCN is that the architecture can take a sequence of any length and map it to an output of the same length. To accomplish this, the TCN uses a 1D fully convolutional network (FCN) where each hidden layer is the same length as the input layer, and zero padding of length (kernel size -1) is added to keep subsequent layers the same length as the previous one. In other words, TCN = 1D FCN + causal convolution. Fig. 5 shows an example of causal convolution with kernel size 2.

Dilated Convolutions: A simple causal convolution is only able to perform lookback in history proportional to the depth of the network. Applying causal convolution on sequence task becomes challenging because of this restriction, especially for the tasks that require a longer history. This problem is solved using dilated convolutions that enable exponentially large receptive field. To put it formally, for a 1D sequence input x and filter $f : \{0, \ldots k-1\}$, the dilated convolution operation F on

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 4, 8)	408
batch_normalization (BatchNormalization)	(None, 4, 8)	32
dropout_2 (Dropout)	(None, 4, 8)	0
lstm (LSTM)	(None, 4, 32)	5248
batch_normalization_1 (BatchNormalization)	(None, 4, 32)	128
dropout_3 (Dropout)	(None, 4, 32)	0
lstm_1 (LSTM)	(None, 4, 64)	24832
flatten (Flatten)	(None, 256)	0
dense_1 (Dense)	(None, 16)	4112
dropout_4 (Dropout)	(None, 16)	0
dense_2 (Dense)	(None, 1)	17

Total params: 34,777
Trainable params: 34,697
Non-trainable params: 80

Fig. 4. CNN-LSTM model summary.

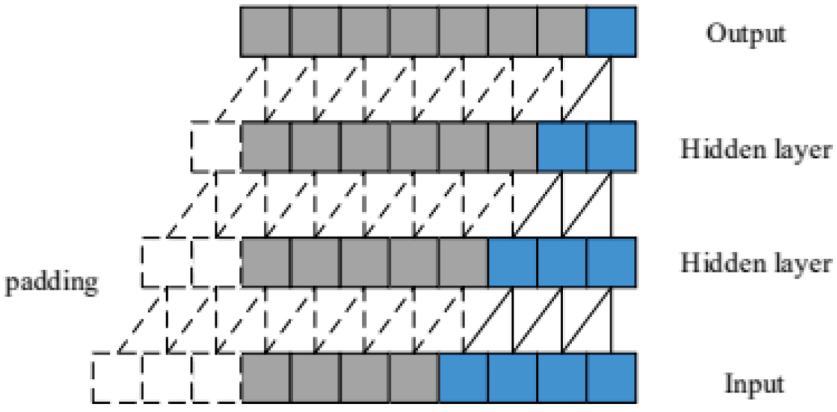


Fig. 5. Causal convolution with filter size k=2 (He & Zhao, 2019).

elements s of the sequence is defined as

$$F(s) = (x * f)(s) = \sum_{i=0}^{k-1} f(i).x_{s-d.i} \tag{25}$$

Here, d is the dilation factor, k is the filter size, and $s - d.i$ accounts for the direction of the past. Dilation can be understood as equivalent to adding a fixed step between two adjacent filter operation on elements as shown in Fig. 6. When d = 1, dilated convolution reduces to regular convolution. Larger dilation allows output at a higher level to represent a wider range of input. This approach provides two ways to increase the range of input: increase the filter size k or increase the dilation factor d. The effective history of one layer is (k-1)d. In dilated convolutions, d is increased exponentially with the depth of the network. Fig. 5 shows dilated causal convolutions with dilation factors d=1,2,4 and filter size k=3.

Residual Connections: A residual block contains a branch leading out to a series of transformations F, whose outputs are added to input x of the block. This can be represented mathematically as

$$o = Activation(x + F(x)) \tag{26}$$

The TCN's receptive field depends on the network depth n, filter size k, and dilation factor d. Stabilization of deeper and larger TCN is an important concern. A residual connection helps in the training stability of larger models. The TCN network uses a generic residual module for

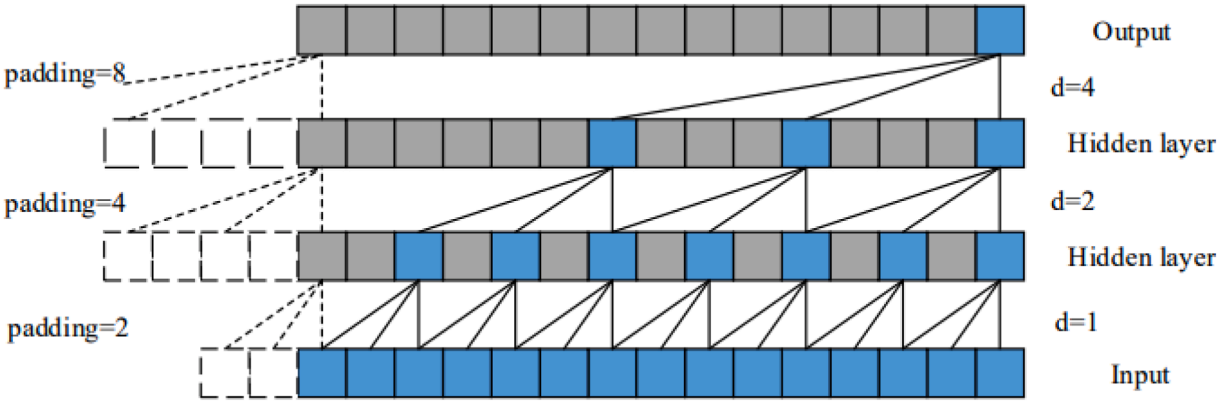


Fig. 6. Dilated causal convolution with dilation factor d=1,2,4 and kernel size k=3 (He & Zhao, 2019).

this purpose. The residual block for the TCN has two layers of dilated causal convolution and non-linearity using ReLU. For normalization, weighted normalization is used with convolutional filters. In addition, spatial dropout is added after each dilated convolution for regularization. Lastly, to account for discrepant input-output width, an additional 1×1 convolution is performed on the input to ensure element-wise an additional operation between the input and output to receive tensors of the same shape.

For this research, the TCN configuration as shown in Fig. 7 was used. The model consists of two TCN layers from the keras-tcn [3] package. The first TCN layer uses a convolutional filter size 64, kernel size 3, dilation array [1, 2, 4, 8] and ReLU activation. The second TCN layer has convolutional filter size 64, kernel size 3, dilation array [1, 2, 4, 8, 16] and ReLU activation. Each TCN layer is followed by a dropout layer with dropout probability of 0.5. The TCN network is terminated with a dense layer of unit 1 with sigmoid activation.

2.5. Trading strategy

The trading signal generated from the machine learning models are evaluated using three trading strategies: Long-only, Short-only, and Long-Short. In the Long-only strategy the asset is bought when the trading signal is 1 and sold when signal is 0. Conversely, for the Short-only strategy the asset is shorted when signal is 0 and bought to cover when the signal is 1. For the Long-Short strategy, both long and short operations are performed based on the trading signal.

Certain assumptions are made with the trading strategies to help keep complexity low and make trading signal evaluation transparent:

1. Transaction costs/commission are rounded up to 0.5 %.
2. Assume no Bid/Ask spread loss and trade slippage.
3. Previous position is closed and a new position is entered when the signal changes in the testing period.

For the purpose of benchmarking the trading strategy results, four return profiles are evaluated. First, a Buy-and-hold strategy, which involves buying and holding the asset for the entire duration of testing period, was used and will act as a lower threshold for comparison. The second, third, and fourth benchmarks are the Long-only, Short-only, and Long-Short strategies, respectively, using perfect future information of price movement directions. These benchmarks will become the upper threshold of each strategy for comparison. Evaluating a given trading strategy against the theoretically best indicator (with perfect future information) and the worst indicator will be a valuable data point in quantifying the predictive power of the new indicator.

2.6. Returns tear sheet

The value of better feature selection, modeling, and trading strategy choice collectively contribute to better return statistics. The key quantitative attributes of return statistics will be compiled in a tabular form called a tear sheet. The key quantitative attributes to evaluate performance are described in Table 3.

3. Results

Following the prediction pipeline illustrated in Fig. 1, this section provides a detailed analysis of the results in order to discuss key observations for feature selection, modeling, trading strategy, and their combined effect towards performance metrics.

3.1. Feature Selection

Feature selection was performed on the in-sample dataset. A total of 196 features that were candidates for the feature selection stage. Out of these, as Table 4 shows, 120 features were selected by L1 feature selection. Boruta feature selection selected 25 features. For PCA, 20, 30, 40 and 50 dimensions were chosen for experimentation. However, since PCA features performed subpar compared to Boruta and L1, only statistics related to 20 and 30 features are reported. The chart shows the dominance of Activity features for L1 feature selection, and the dominance of Realized Value, Unrealized Value, or Stationarity features for Boruta feature selection. Table 5 list the different Boruta features for each category. Comparing the feature selection output to category wise correlation in Table 1 helps to show the prominence of feature categories in predicting price direction. Boruta based features in Table 4 and mean correlation categories in Table 1 show Realized Value and Unrealized Value categories as prominent contributors towards price direction prediction. Also, the Boruta based features in Table 4 and mean correlation categories in Table 1 show that the Mining and Activity categories offer low prediction capabilities toward the target. Owing to the lower correlation with the target, few or no features from these categories were selected by Boruta feature selection. Contrary to Boruta feature selection, however, L1 feature selection selects Activity, Realized Value, and the Stationarity based features. The selected features, as is seen in later sections, will show a large difference in prediction accuracy and help the selection of the best performing feature selection algorithm.

For subsequent stages in the prediction pipeline, the selected features in Table 6 were used. The features selected by Boruta are described here, but please refer to Appendix A for a lists of all examined features and their description.

[3] keras-tcn : GitHub - philipperemy/keras-tcn: Keras Temporal Convolutional Network.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 tcn (TCN)                   (None, 5, 64)             92992

 dropout (Dropout)           (None, 5, 64)             0

 tcn_1 (TCN)                 (None, 64)                123520

 dropout_1 (Dropout)         (None, 64)                0

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 216,577
Trainable params: 216,577
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7. TCN model summary.

Table 3
Return tear sheet to compare performance of trading strategies.

#	Performance Metrics	Description
1	Annual Return %	Geometric average of cumulative return calculated year over year
2	Annual Volatility %	Variation in asset price calculated over a year
3	Max Drawdown %	Maximum percent loss from a peak to a trough before a new peak
4	Sharpe Ratio	Risk adjusted return of a portfolio compared to a risk-free investment

Table 4
Boruta vs L1 feature selection by category.

Index	Category	Boruta	L1
1	Unrealized Value	7	13
2	Activity	1	63
3	Realized Value	12	18
4	Mining	0	10
5	Stationarity	5	16
	Total	**25**	**120**

Table 5
Features selected by Boruta based on feature category classification.

Feature Category	Feature Name
Stationarity	cdd, cdd_supply_adjusted, rcap_hodl_waves_1d_1w, rcap_hodl_waves_1w_1m, rcap_hodl_waves_24h
Realized Value	loss_sum, net_realized_profit_loss, price_usd_ohlc_o, profit_relative, profit_sum, realized_loss, realized_profits_to_value_ratio, realized_profit, realized_profit_loss_ratio, sopr_adjusted, sopr, price_ohlc_usd_c
Unrealized Value	Mvrv, mvrv_z_score, net_unrealized_profit_loss, unrealized_loss, unrealized_profit, utxo_loss_count, utxo_profit_relative
Mining	
Activity	Svl_1m_3m

3.2. Model performance

The features selected by the individual feature selection algorithms were used to train prediction models. Three models were experimented with, including the CNN-LSTM, TCN models, and the Random Forest model; the Random Forest model is used to benchmark the performance of neural network models. For the CNN-LSTM and TCN models, the training was executed for 1000 epochs, using a batch size of 50 samples and train:validation split of 9:1. For robustness of the model, the experiment was executed with multiple seed values. No significant difference in model performance was observed for different seed values. The models were trained in such a way that future information leakage is prohibited. The training and testing samples maintained their relative time series sequence. Also, the training process turned off random shuffle for both the train and test samples to avoid using future information to predict past directions. Fig. 8 shows the model accuracy for the out-of-sample period when different feature sets were used. Table 7 shows performance metrics for all combinations of feature selection and models sorted by accuracy. The CNN-LSTM model with Boruta feature selection showed the highest accuracy of 0.8203 with an F1 score of 0.8201. The next best accuracy was shown by the TCN with Boruta feature selection with accuracy of 0.7088 and F1 score of 0.6686. For the other combinations the accuracy did not cross the 0.60 threshold. The model performance for five PCA based feature sets did not show any appreciable improvement, therefore, only the PCA with 20 components is shown in Fig. 6. The L1 feature selection and PCA shows the best accuracy of 0.5606 and 0.5456, respectively.

For this experiment, the CNN-LSTM network used a lookback period 5, whereas the TCN networks used a lookback period of 3; i.e., the number of days the model should take in account for making the prediction. Fig. 9 plots the accuracy of the CNN-LSTM and TCN with Boruta feature selection against lookback period. Lookback periods from 3 to 8 were considered. The CNN-LSTM model showed the highest accuracy for lookback period of 5, whereas the TCN model showed the highest accuracy for lookback period of 3. The CNN-LSTM model consistently outperformed the TCN model over the lookback period values that were considered. Figs. 10 and 11, respectively, show precision-recall curve and ROC curve for CNN-LSTM model.

3.3. Backtesting strategies

To perform backtesting, four trading strategies were compared using the two best models previously identified: TCN and CNN-LSTM. The four trading strategies include: Long-only, Short-only, Long-Short, and Buy-and-hold (as a benchmark). The out-of-sample period from 04-19-2021 to 05-14-2023 was used for backtesting. Trading transaction costs are an important consideration. A survey shows that a transaction

Table 6
Selected features.

Feature	Description
Coin days destroyed (cdd)	Computed by multiplying number of coins in a transaction and number of days since those coins were last spent.
cdd_supply_adjusted	Coin days destroyed divided by circulating supply of total coins issued.
rcap_hodl_waves_1d_1w	Group all BTC supply in certain age bands based on their last move time. For this feature, the age band was 1 day to 1 week and then weighed by realized price.
rcap_hodl_waves_1w_1m	Group all BTC supply in certain age bands based on their last move time. For this feature, the age band was 1 week to 1 month and then weighed by realized price.
rcap_hodl_waves_24h	Group all BTC supply in certain age bands based on their last move time. For this feature, the age band was less than 24 hours and then weighed by realized price.
loss_sum	Amount of coins whose price at the time of last move was higher than current price.
net_realized_profit_loss	Net profit or loss of all moved coins.
price_usd_ohlc_o	Bitcoin open price in daily OHLC price series.
profit_relative	Percent of unique addresses for which average buy price is lower than average current price.
profit_sum	Amount of coins whose price at the time of last move was lower than the current price.
realized_loss	Total loss in USD of all coins whose price at last movement was higher than current price.
realized_profits_to_value_ratio	Ratio of realized profit and realized cap.
realized_profit	Total profit in USD of all coins whose price at last movement was lower than current price.
realized_profit_loss_ratio	Ratio of realized profit and realized loss.
sopr_adjusted	SOPR ignoring all outputs with lifespan < 1 hour.
sopr	Ratio of realized value in USD divided by value at creation in USD of spent output.
price_ohlc_usd_c	Bitcoin close price in daily OHLC price series.
Mvrv	Market Value to Realized Value is the ratio between market cap and realized cap.
mvrv_z_score	Ratio of difference in market cap and realized cap, and the standard deviation of market cap.
net_unrealized_profit_loss	Difference between relative unrealized profit and relative unrealized loss.
unrealized_loss	Total loss in USD of all coins whose price at realization time was higher than current price normalized by market cap.
unrealized_profit	Total profit in USD of all coins whose price at realization time was lower than current price normalized by market cap.
utxo_loss_count	Number of unspent transactions output with price at creation higher than current price.
utxo_profit_relative	Percent of unspent transactions output with price at creation lower than current price.
Svl_1m_3m	Spent Volume Lifespan (SVL) is total transfer volume that was last active between 1 month and 3 months ago.

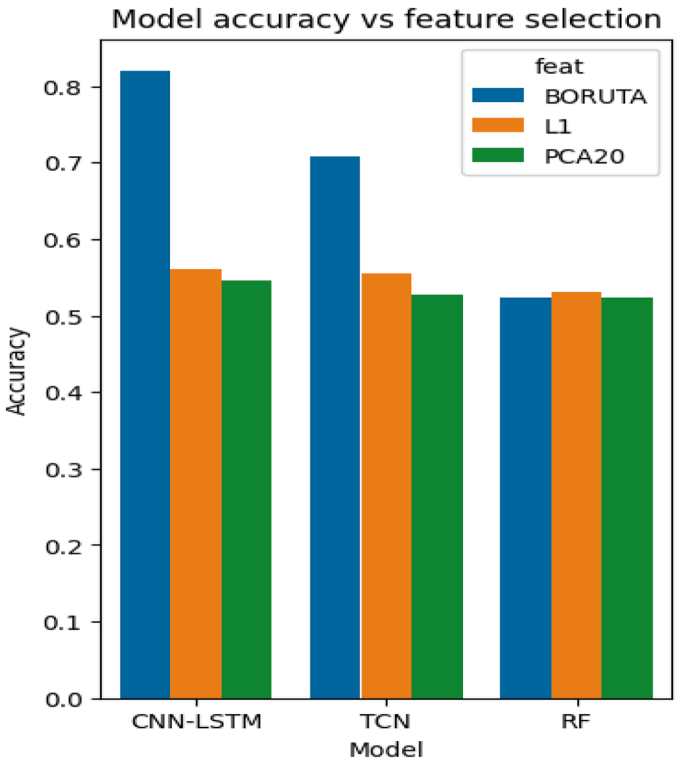


Fig. 8. Machine learning model accuracy for feature selection algorithms for the out-of-sample period.

Table 7
Machine learning model and feature selection performance for the out-of-sample period.

Index	Model	Feat Selection	Accuracy	Precision	Recall	F1-score
1	CNN-LSTM	BORUTA	0.8203	0.8010	0.8401	0.8201
2	TCN	BORUTA	0.7088	0.7483	0.6043	0.6686
3	CNN-LSTM	L1	0.5606	0.7142	0.1626	0.2649
4	TCN	L1	0.5549	0.6987	0.1571	0.2566
5	CNN-LSTM	PCA20	0.5456	0.7500	0.1056	0.1852
6	RF	L1	0.5308	0.5398	0.2378	0.3302
7	TCN	PCA20	0.5277	0.5789	0.1192	0.1977
8	RF	BORUTA	0.5243	0.7500	0.0324	0.0621
9	RF	PCA20	0.5243	0.5238	0.6044	0.1084

cost in the range 0.1 % to 0.25 % is often used in published literature (Parente et al., 2024). Nonetheless, for cryptocurrency trading this is probably too small to account for the risk factors that the transaction cost covers. Geo-political restrictions on trading platforms, volatility, trade slippage, and spread loss can all increase the trading cost. Coinbase, a popular cryptocurrency trading platform, charges anywhere between 0 % to 3.99 % per transaction[4], although other on-chain trading app fees are lower. Based on this information, it was decided to use an average trading cost of 1 %. In addition, for a fee, trading platforms such as Kraken[5] allow margin trading, which is required for holding a Bitcoin short position. For the purpose of this simulation, the cost to hold a short position is assumed to be 0.5 %. Detailed analysis of trading risks must be considered for a live trading environment as it could materially affect the trading strategy's performance. Major risk events such as liquidity risk, credit risk, and operational risk are ignored for this research.

Table 8 shows backtesting performance metrics for different strategies using no model, the perfect future information model, the TCN model, and the CNN-LSTM model. The first four rows in Table 8 that are highlighted represent the benchmark metrics. The Buy-and-hold strategy acts as the lower baseline. This strategy gives the lowest annual

[4] Coinbase uses a different fee tier for standard and professional users. Their fee structure is based on a maker-taker fee model, with liquidity takers generally paying higher fees than liquidity makers. Fees are relatively higher for the standard tier. See https://help.coinbase.com/en/exchange/trading-and-funding/exchange-fees for the fee structure.

[5] Kraken charges 0.02 % for opening a position and 0.02 % for every 4 hours in roll over fees to keep the position open. Approximating that a short position cost of 0.5 % is equivalent to holding short position for approximately four days. See https://www.kraken.com/features/margin-trading/bitcoin for the fee structure.

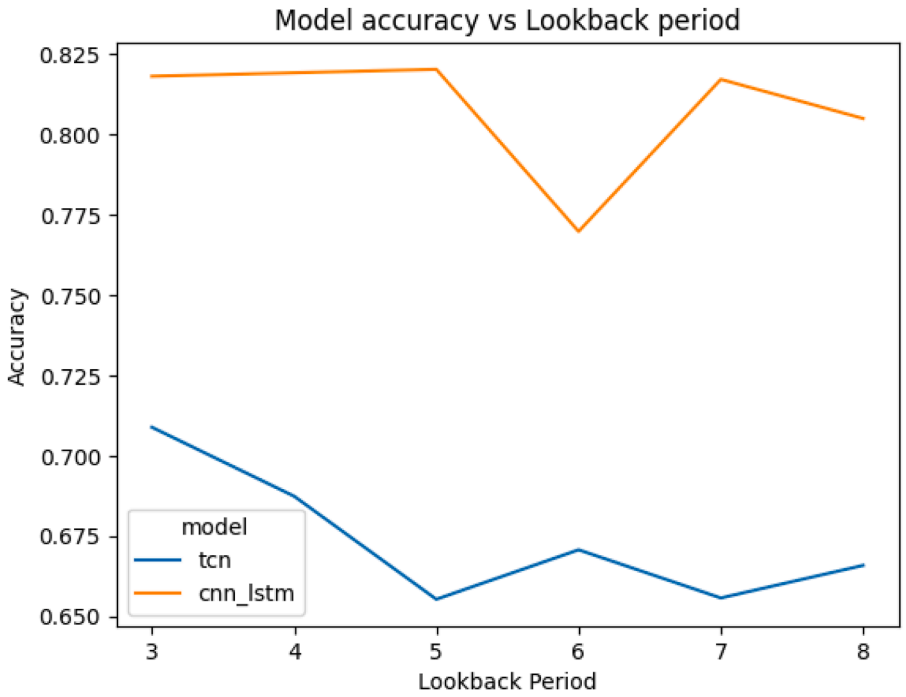


Fig. 9. Accuracy vs Lookback period for CNN-LSTM and TCN models using Boruta features.

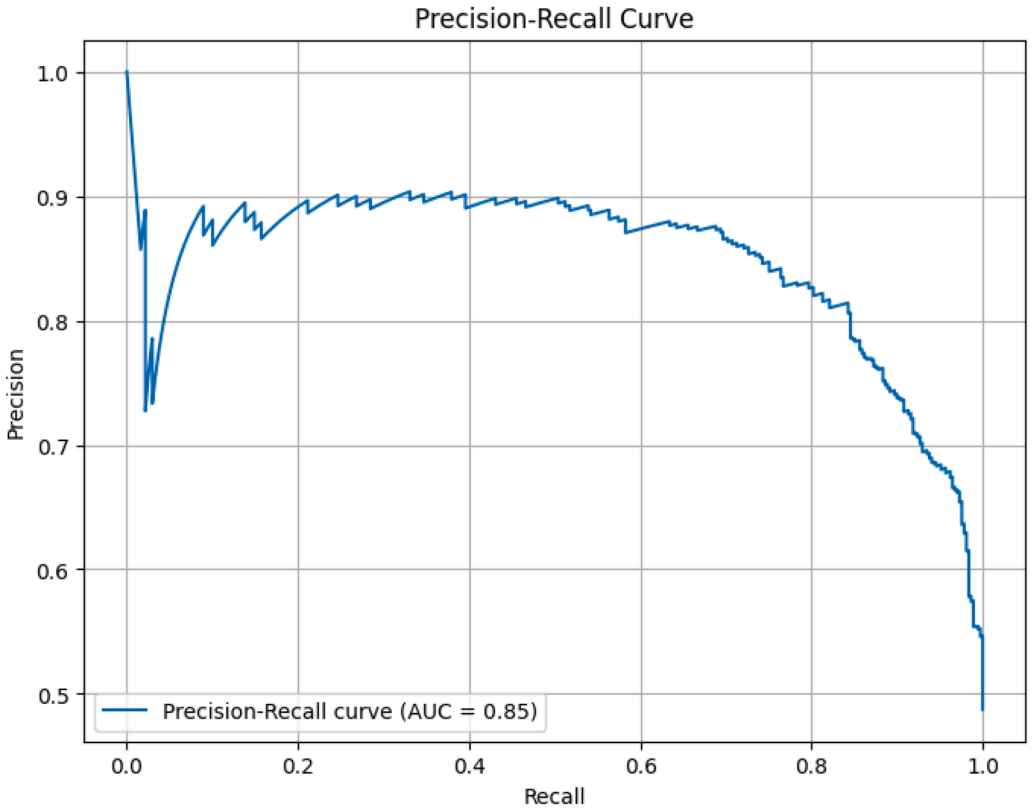


Fig. 10. Precision Recall curve for CNN-LSTM model.

return (-21.9 %), highest annual volatility (55.6 %), a negative Sharpe ratio (-0.17), and the maximum drawdown percent (-76.6 %). Ideal Long Only, Ideal Short Only, and Ideal Long and Short strategies with perfect future information represent upper thresholds for the metrics. The Ideal strategy assumes zero operation cost and uses perfect future information for price direction as a trading indicator.

The Ideal Long-only strategy shows an annual return of 1776.9 % with a Sharpe Ratio of 8.79. The Ideal Short-only strategy shows a higher annual return of 1999.4 % since the overall market trend in this period is bearish. Intuitively, the Ideal Long-Short strategy shows much higher return and Sharpe Ratio (39304.0 %, 15.23) than the Ideal Long-only or Ideal Short-only strategies since this strategy exploits price movement in

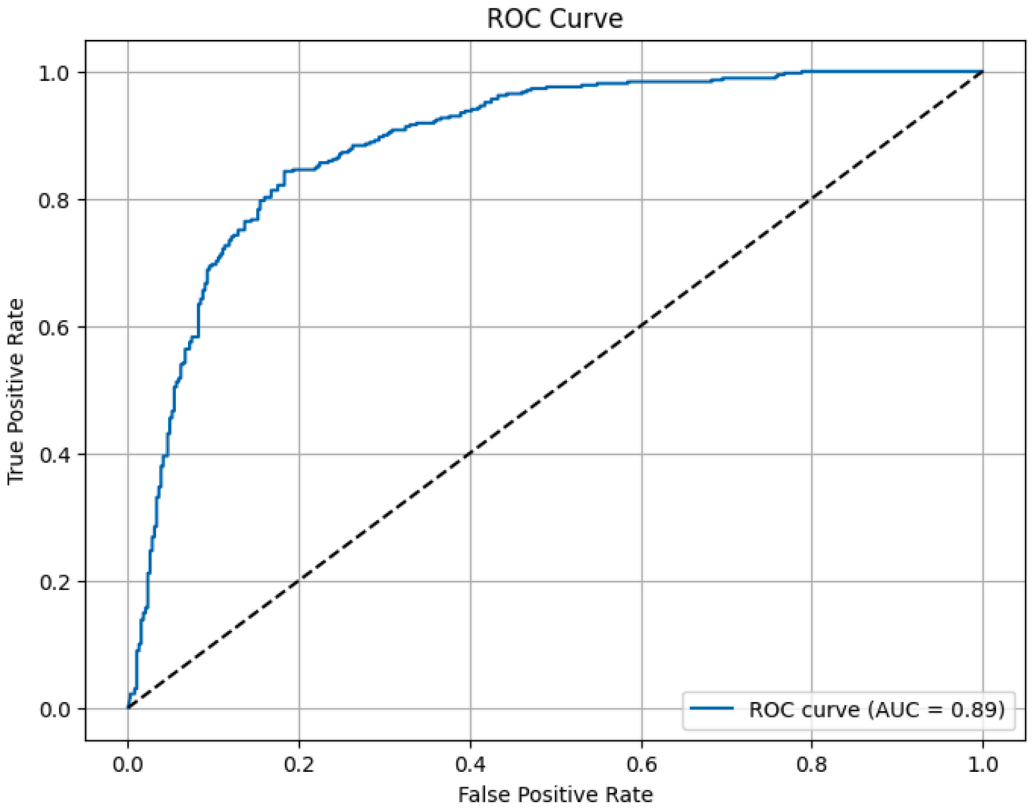


Fig. 11. ROC curve for CNN-LSTM model.

Table 8
Performance metrics of trading strategies with models.

Model	Strategy	Annual Return %	Sharpe Ratio	Maximum Drawdown %	Annual Volatility %
No model	Buy-and-hold	-21.9	-0.17	-76.6	55.6
Perfect Future Information	*Ideal Long-only*	*1776.9*	*8.79*	*0.0*	*34.2*
Perfect Future Information	*Ideal Short-only*	*1999.4*	*9.03*	*0.0*	*34.6*
Perfect Future Information	*Ideal Long-Short*	*39304.0*	*15.23*	*0.0*	*40.3*
TCN	Long-only	237.9	3.20	-27.1	40.8
TCN	Short-only	289.4	4.28	-11.4	33.2
TCN	Long-Short	547.1	3.92	-20.3	51.1
CNN-LSTM	Long-only	458.0	4.93	-18.6	36.4
CNN-LSTM	Short-only	349.0	4.87	-13.9	32.0
CNN-LSTM	Long-Short	1682.7	6.47	-16.1	46.4

both sides. Since perfect future information is assumed, the maximum drawdown is 0 for all strategies using perfect future information.

The Long-only, Short-only, and Long-Short strategies were also studied using both the TCN and CNN-LSTM networks. The TCN model gives an annual return of 237.9 % with a 3.20 Sharpe Ratio and -27.1 % max drawdown. The Short-only strategy with the same indicator gives a higher return of 289.4 %, better Sharpe Ratio of 4.28, and a lower max drawdown of -11.4 %. The annual volatility is also reduced to 33.2 %. For the Long-Short strategy the annual return is 547.1 %, capitalizing on both up and down price movements. However, this strategy has a lower Sharpe Ratio at 3.92, higher drawdown at -20.3 % and suffers from high volatility of 51.1 %. The high Sharpe Ratio for the Short-only strategy indicates that the lower return from the Short-only strategy is a better investing choice than the Long-Short strategy with higher return.

The CNN-LSTM model performed better than the TCN across all strategies. For the Long-only strategy, the annual return was 458.0 % with a 4.93 Sharpe Ratio, which is higher than the TCN Sharpe Ratios. Similarly, the Short-only strategy gives annualized return of 349.0 %, lower than the CNN-LSTM Long-only Strategy, but considerably better than the equivalent Short-only strategy using the TCN network. The CNN-LSTM based Short-only strategy shows marginally lower volatility of 32.0 % and lower maximum drawdown of -13.9 % and Sharpe Ratio of 4.87. The CNN-LSTM provides better Sharpe Ratio performance compared to the TCN for any strategy; the worst Sharpe Ratio by the CNN-LSTM is 4.87, whereas TCN gives its best Sharpe Ratio for the Short-only strategy at 4.28. Beside Sharpe Ratio and annualized return, the CNN-LSTM performs better than TCN for annualized volatility and maximum drawdown for equivalent strategies. For the Long-only strategy, the CNN-LSTM shows annualized volatility of 36.4 %, lower than a volatility of 40.8 % produced by the TCN. Finally, the CNN-LSTM model with a Long-Short strategy had the highest annualized return of 1682.7 %. The maximum drawdown was better than the Long-only strategy, but the volatility was higher at 46.4 %. Fig. 12 shows the distribution of returns captured by the three strategies. The high F1-score of the CNN-LSTM model + Boruta feature selection algorithm helps the model to identify upside and downside price movements and also minimize false positives and negatives. This allows the Long-short strategy to capture both upside and downside price movements and therefore achieves the best annualized return. Long-only and short-only strategies on the other hand miss out on the opposite side positions and

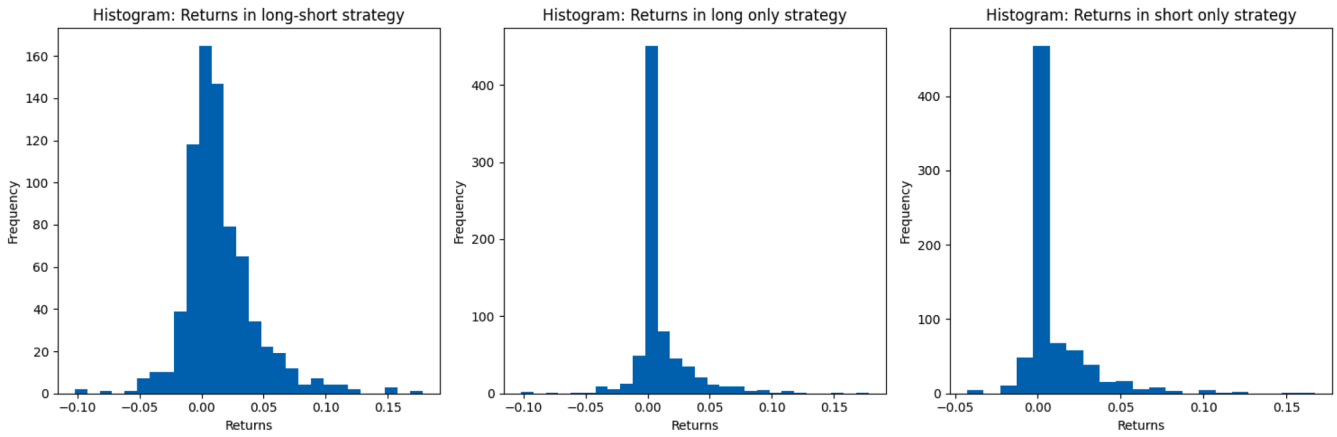


Fig. 12. Distribution of returns for three strategies using CNN-LSTM model.

therefore achieve lower annualized return; nevertheless, they perform better than other model and feature selection algorithms. Fig. 13 shows Bitcoin price against cumulative return for the testing period using the winning combination of model, feature selection, and trading strategy. A risk-taking investor choosing this strategy is being more than compensated for this risk as illustrated by an impressive Sharpe Ratio of 6.47, the highest Sharpe Ratio of all the predictive models that did not have perfect information.

4. Conclusion

On-chain metrics provide valuable information for analyzing Bitcoin trading patterns. This research demonstrated a systematic approach to evaluate 196 on-chain metrics for helping predict the price of Bitcoin. First, the on-chain features were classified in five categories to provide intuition of their predictive capabilities. Quantitative attributes of the features were also analyzed for the on-chain features. Thereafter, a price direction prediction pipeline was designed. The research compared feature selection and dimensionality reduction techniques, including L1 regularization, Boruta, and PCA to identify metrics that can predict the target. L1 feature selection selected 120 features, whereas Boruta feature selection selected 25 features. These 25 features show prominent representation of the Stationary, Realized Value, and Unrealized Value categories. With these features, three machine learning models were developed and tested: Random Forest, TCN, and CNN-LSTM. Based on the experiments, Boruta provided features that gave the best prediction performance with the TCN and CNN-LSTM model. CNN-LSTM model showed the highest accuracy of 82.03 %, while the highest accuracy achieved by the TCN model was 70.88 %. Common trading strategies were used to quantify return potential of the tested indicator models. The experiment with trading strategies revealed that both the CNN-LSTM and TCN models perform better than the lower Buy-and-hold

Fig. 13. Cumulative return of CNN-LSTM model with Boruta feature selection and Long-short strategy.

benchmark strategy. Also, the CNN-LSTM model-based indicator performs better than the TCN model-based indicator with an annualized return of 1682.7 % and Sharpe Ratio of 6.47. In general, choosing the proper feature selection approach and forecasting model can allow investors to profit from both long and short trading, even when trading traditionally volatile markets, such as the cryptocurrency markets.

This research opens up multiple avenues for future work. The model shows promising results for long and short strategies. With the recent launch of Bitcoin ETFs, holding a short position on Bitcoin is now possible with exchanged traded securities. New experiments could explore active trading strategies that better model the cost structure of holding a short position on a Bitcoin ETF. New research can also be undertaken to perform price direction prediction for other cryptocurrencies, such as Ethereum (ETH), using corresponding on-chain data. With Bitcoin, Ethereum, and other cryptocurrency based-models, research in cryptocurrency portfolio optimization can be undertaken. Alternatively, research could be conducted to define a generic pricing model based on the identified feature categories that work for multiple cryptocurrencies. A generic pricing model developed with a cryptocurrency-based portfolio optimization methodology can be combined with price forecasting models for generating alpha from a cryptocurrency portfolio. With the advent of Bitcoin ETFs, investors can also utilize the models tested in this research to incorporate Bitcoin exposure into a traditional diversified portfolio of stocks, bonds, commodities, forex, and other investable assets.

Authors' information

Ritwik Dubey (rdqhk@mst.edu) received his Bachelor in Technology, Electronics and Communications Engineering, from MNNIT Allahabad, India, and a M.S in Electrical and Computer Engineering from the University of Florida. His research interests include artificial intelligence, engineering management, and financial markets. In addition to working in the area of financial risk management, his research has focused on utilizing computational intelligence and machine learning to understand and predict financial markets, particularly the cryptocurrency markets.

Dr. David Enke (enke@mst.edu) received his B.S. in Electrical Engineering and M.S. and Ph.D. in Engineering Management, all from the Missouri University of Science and Technology. He is a Curators' Distinguished Teaching Professor of Engineering Management and Systems Engineering at the Missouri University of Science and Technology, as well as the director of the Laboratory for Investment and Financial Engineering. His research interests are in the areas of investments, derivatives, financial engineering, financial risk management, portfolio management, algorithmic trading, hedge funds, financial forecasting, volatility forecasting, neural network modeling and computational intelligence.

Availability of data and materials

The datasets analyzed in this study are available from Glassnode (https://glassnode.com/). On-chain metrics were also collected from Glassnode and are available at https://studio.glassnode.com/metrics. The code for the developed models is available upon request.

CRediT authorship contribution statement

Ritwik Dubey: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **David Enke:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no competing interests and nothing to declare.

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.

Appendix A

List of on-chain features with brief descriptions. These features are obtained from Glassnode. Please refer to the website (https://studio.glassnode.com/catalog) for more information about each feature.

Category	Column	Description
Activity	active_1d_1w	Circulating supply moved between 1 day and 1 week
	active_1m_3m	Circulating supply moved between 1 month and 3 months
	active_1w_1m	Circulating supply moved between 1 week and 1 month
	active_24h	Circulating supply moved in last 24 hours
	active_3m_6m	Circulating supply moved between 3 months and 6 months
	active_6m_12m	Circulating supply moved between 6 months and 12 months
	active_count	Active addresses
	count	Total unique addresses
	exchange_net_position_change	30d change in supply held by exchange wallets
	gini	Gini coefficient for distribution of coins over addresses
	herfindahl	A metric to quantify decentralization
	min_100_count	Number of unique addresses with at least 100 coins
	min_10k_count	Number of unique addresses with at least 10k coins
	min_10_count	Number of unique addresses with at least 10 coins
	min_1k_count	Number of unique addresses with at least 1k coins
	min_1_count	Number of unique addresses with at least 1 coins
	min_point_1_count	Number of unique addresses with at least 0.1 coins
	min_point_zero_1_count	Number of unique addresses with at least 0.01 coins
	msol	Median Spent Output Lifespan

(continued on next page)

(*continued*)

Category	Column	Description
	new_zero_count	**Number of unique addresses** with positive amount of coins
	rate	Amount of transaction per second
	receiving_count	Number of unique addresses receiving funds
	sending_count	Number of unique addresses sending funds
	size_mean	Mean size of transaction in timeperiod
	size_sum	Total size of transaction in timeperiod
	soab_1d_1w	Spent output age band between 1 day and 1 week
	soab_1h	Spent output age band less than 1 hour
	soab_1h_24h	Spent output age band between 1 hour and 24 hours
	soab_1m_3m	Spent output age band between 1 month and 3 months
	soab_1w_1m	Spent output age band between 1 week and 1 month
	soab_3m_6m	Spent output age band between 3 months and 6 months
	soab_6m_12m	Spent output age band between 6 months and 12 months
	sol_1d_1w	Total spent output created between 1 day and 1 week
	sol_1h_24h	Total spent output created between 1 hour and 24 hours
	sol_1h	Total spent output created less than 1 hour ago
	sol_1m_3m	Total spent output created between 1 month and 3 months
	sol_1w_1m	Total spent output created between 1 week and 1 month
	sol_3m_6m	Total spent output created between 3 months and 6 months
	sol_6m_12m	Total spent output created between 6 months and 12 months
	spent_output_types_share_multisig_share	Spent output on multiple public keys
	spent_output_types_share_nonstd_share	Spent output on undefined method
	spent_output_types_share_p2pk_share	Spent output using public key
	spent_output_types_share_p2pkh_share	Spent output using hash of public key
	spent_output_types_share_p2sh_nonsegwit_share	Spent output using script ash multisig
	spent_output_types_share_p2sh_p2wpkh_share	Spent output using script hash using nested witness public key hash
	spent_output_types_share_p2sh_p2wsh_share	Spent output using script shash using segwit
	spent_output_types_share_p2tr_share	Spent output using taproot
	spent_output_types_share_p2wpkh_share	Spent output using Witness Public Hash Key
	spent_output_types_share_p2wsh_share	Spent output using Witness Script Hash
	stock_to_flow_deflection	Ratio of current Bitcoin price and stock to flow model
	stock_to_flow_ratio_ratio	Ratio of circulating Bitcoin supply and flow of newly mined coins
	supply_by_txout_type_total_other	Transaction output supply - other
	supply_by_txout_type_total_p2pk	Transaction output supply - p2pk
	supply_by_txout_type_total_p2pkh	Transaction output supply - p2pkh
	supply_by_txout_type_total_p2sh	Transaction output supply - p2sh
	supply_by_txout_type_total_p2tr	Transaction output supply - p2tr
	supply_by_txout_type_total_p2wpkh	Transaction output supply - p2wpkh
	supply_by_txout_type_total_p2wsh	Transaction output supply - p2wsh
	svab_1d_1w	Spent Volume Age Band moved within timeperiod
	svab_1h	Spent Volume Age Band moved within timeperiod
	svab_1h_24h	Spent Volume Age Band moved within timeperiod
	svab_1m_3m	Spent Volume Age Band moved within timeperiod
	svab_1w_1m	Spent Volume Age Band moved within timeperiod
	svab_3m_6m	Spent Volume Age Band moved within timeperiod
	svab_6m_12m	Spent Volume Age Band moved within timeperiod
	svl_1d_1w	Total transfer volume of coins within timeperiod
	svl_1h_24h	Total transfer volume of coins within timeperiod
	svl_1h	Total transfer volume of coins within timeperiod
	svl_1m_3m	Total transfer volume of coins within timeperiod
	svl_1w_1m	Total transfer volume of coins within timeperiod
	svl_3m_6m	Total transfer volume of coins within timeperiod
	svl_6m_12m	Total transfer volume of coins within timeperiod
	transfers_between_exchanges_count	Total count of transfers between exchanges
	transfers_from_exchanges_count	Number of on-chain withdrawals from exchanges
	transfers_to_exchanges_count	Number of on-chain deposits to exchanges
	transfers_volume_adjusted_mean	Mean value of transfer adjusted by change volume
	transfers_volume_adjusted_median	Median value of transfer adjusted by change volume
	transfers_volume_adjusted_sum	Total volume of transfer adjusted by change volume
	transfers_volume_between_exchanges_sum	Total amount of coins transferred between exchanges
	transfers_volume_exchanges_net	Difference in volume flowing in and out of exchanges
	transfers_volume_from_exchanges_mean	Mean value of a transfer from exchange addresses
	transfers_volume_from_exchanges_sum	Total amount of coins transferred from exchange addresses
	transfers_volume_mean	Mean value of transfers
	transfers_volume_median	Median value of transfers
	transfers_volume_sum	Total amount of coins transferred on-chain
	transfers_volume_to_exchanges_mean	Mean value of transfer to exchange address
	transfers_volume_to_exchanges_sum	Total amount of coins transferred to exchange addresses
	transfers_volume_within_exchanges_sum	Total amount of coins transferred within wallet of same exchange
	utxo_count	Total number of UTXO in network
	utxo_created_count	Number of UTXO created
	utxo_created_value_mean	Mean amount of coins in new UTXOs
	utxo_created_value_median	Median amount of coins in new UTXOs
	utxo_created_value_sum	Sum of amount of coins in new UTXOs
	utxo_spent_count	Number of spent UTXO
	utxo_spent_value_mean	Mean amount of coins in spent UTXO

(*continued on next page*)

(*continued*)

Category	Column	Description
	utxo_spent_value_median	Median amount of coins in spent UTXO
	utxo_spent_value_sum	Sum of amount of coins in spent UTXO
	velocity	Ratio of on-chain transaction volume and market cap
Mining	block_count	Number of blocks created in main blockchain
	block_height	Total number of blocks created
	block_interval_mean	Mean time between mined blocks
	block_interval_median	Median time between mined blocks
	block_size_mean	Mean size of all blocks in timeperiod
	block_size_sum	Total size of all blocks in timeperiod
	difficulty_ribbon_compression	Standard deviation of a set of moving average mining difficulty
	fee_ratio_multiple	Ratio of total miner revenue and transaction fees
	puell_multiple	Ratio of daily issuance value and 356 day MA of daily issuance
	revenue_from_fees	Percent of miner revenue from fees
	revenue_sum	Total miner revenue
	stock_to_flow_ratio_daysTillHalving	Days till halving
	thermocap	Aggregate amount of coins paid to miner
	volume_mean	Mean fee per transaction
	volume_median	Median fee per transaction
	volume_mined_sum	Total amount of newly minted coins
	volume_sum	Total fee per transaction
Realized Value	asol	Average Spent Output Lifespan or average age of spent transaction output
	balanced_price_usd	Difference between Realized price and Transfer price
	investor_capitalization	Difference of Realized cap and Thermo cap
	loss_sum	Amount of coins who moved to a lower price
	marketcap_realized_usd	Value of supply at prices when they last moved
	net_realized_profit_loss	Difference of Realized Profit and Realized Loss
	pi_cycle_top_ma111	111 day MA of Bitcoin price
	pi_cycle_top_ma350×2	2×350 day MA of Bitcoin price
	price_realized_usd	Realized cap divided by current supply
	price_usd_ohlc_c	Bitcoin closing price in 24 hour period
	price_usd_ohlc_h	Bitcoin opening price in 24 hour period
	price_usd_ohlc_l	Bitcoin low price in 24 hour period
	price_usd_ohlc_o	Bitcoin high price in 24 hour period
	profit_relative	Percent of unique address with average buy price lower than current price
	profit_sum	Circulating supply in profit
	realized_loss	Total loss of all coins that moved to a lower price
	realized_profits_to_value_ratio	Ratio of Realized profits and Realized cap
	realized_profit	Total profit of all coins that moved to higher price
	realized_profit_loss_ratio	Ratio of all coins moved to profit and loss
	realized_volatility_1_month	1 month Rolling standard deviation of returns from mean return of market
	realized_volatility_1_week	1 week Rolling standard deviation of returns from mean return of market
	realized_volatility_1_year	1 year Rolling standard deviation of returns from mean return of market
	realized_volatility_2_weeks	2 weeks Rolling standard deviation of returns from mean return of market
	realized_volatility_3_months	3 months Rolling standard deviation of returns from mean return of market
	realized_volatility_6_months	6 months Rolling standard deviation of returns from mean return of market
	rhodl_ratio	Ratio of 1 week and 1-2years Realized cap HODL wave
	seller_exhaustion_constant	Product of percent supply in profit and 30 day price volatility
	sopr_adjusted	Ratio of realized value and value at creation of spent output with lifespan $>$ 1 hr
	sopr	Ratio of realized value and value at creation of spent output
Stationarity	average_dormancy	Ratio of coin days destroyed and total transfer volume
	average_dormancy_supply_adjusted	Ratio of coin days destroyed and total transfer volume adjusted for supply
	cdd90_age_adjusted	90 day rolling sum of CDD normalized by time
	cdd90	90 day rolling sum of CDD
	cdd	Product of number of coins in transaction and number of days since last spent
	cdd_supply_adjusted_binary	1 if more adj CDD destroyed than mean CDD, else 0
	cdd_supply_adjusted	CDD normalized by circulating supply
	coin_blocks_created	Product of value of coin and number of blocks it was unspent
	coin_blocks_destroyed	Product of number of coins and number of blocks since last spent
	cvdd	Cumulative Value Days Destroyed; ratio of CDD and market age
	cyd	365 day rolling sum of CDD
	cyd_supply_adjusted	Supply adjusted 365 day rolling sum of CDD
	dormancy_flow	Ratio of CDD and total transfer volume
	hodl_waves_1d_1w	Percent of Bitcoin that moved in this time period
	hodl_waves_1m_3m	Percent of Bitcoin that moved in this time period
	hodl_waves_1w_1m	Percent of Bitcoin that moved in this time period
	hodl_waves_24h	Percent of Bitcoin that moved in this time period
	hodl_waves_3m_6m	Percent of Bitcoin that moved in this time period
	hodl_waves_6m_12m	Percent of Bitcoin that moved in this time period
	liveliness	Ratio of CDD and sum of all coin days
	rcap_hodl_waves_1d_1w	HODL wave weighted by realized price
	rcap_hodl_waves_1m_3m	HODL wave weighted by realized price
	rcap_hodl_waves_1w_1m	HODL wave weighted by realized price
	rcap_hodl_waves_24h	HODL wave weighted by realized price
	rcap_hodl_waves_3m_6m	HODL wave weighted by realized price
	rcap_hodl_waves_6m_12m	HODL wave weighted by realized price
	reserve_risk	An indicator to gauge confidence of long term holders relative to price

(*continued on next page*)

(*continued*)

Category	Column	Description
Unrealized Value	accumulation_trend_score_score	score closer to 1 when accumulation increases, closer to 0 when accumulation decreases
	balance_1pct_holders	Percent of supply held by top 1 % addresses
	balance_exchanges	Total amount of coins held on exchange addresses
	balance_exchanges_relative	Percent of supply held on exchange addresses
	current_adjusted	Circulating supply adjusted by lost coins
	current	Total amount of coins created
	deltacap_usd	Difference between Realized cap and average of Market cap
	inflation_rate	Percent of new coins divided by current supply
	issued	Amount of new coins added to current supply
	marketcap_thermocap_ratio	Ratio of market cap and thermo cap
	marketcap_usd	Product of current supply and USD price
	mvrv	Ratio of Market cap and Realized cap
	mvrv_z_score	z score of MVRV
	net_unrealized_profit_loss	Difference of Relative Unrealized Profit and Relative Unrealized Loss
	nvts	Ratio of market cap and 90 D MA of daily transaction volume
	nvt	Ratio of market cap and transferred on-chain volume
	price_drawdown_relative	Percent drawdown from last high
	probably_lost	Inactive coins since launch of BTC exchange
	provably_lost_cumsum	Cumulative sum of lost bitcoins
	unrealized_loss	Total loss of all coins whose price at realization time was higher than current price
	unrealized_profit	Total profit of all coins whose price at realization was lower than current price
	utxo_loss_count	Number of UTXO with creation time price higher than current price
	utxo_profit_count	Number of UTXO with creation time price lower than current price
	utxo_profit_relative	Percent of UTXO with creation time price lower than current price

References

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *ArXiv Preprint ArXiv: 1803.01271*.

Critien, J. V., Gatt, A., & Ellul, J. (2022). Bitcoin price change and trend prediction through twitter sentiment and data volume. *Financial Innovation, 8*(1), 45. https://doi.org/10.1186/s40854-022-00352-7

Dutta, A., Kumar, S., & Basu, M. (2020). A gated recurrent unit approach to bitcoin price prediction. *Journal of Risk and Financial Management, 13*(2), 23. https://doi.org/10.3390/jrfm13020023

Fang, F., Ventre, C., Basios, M., Kanthan, L., Martinez-Rego, D., Wu, F., & Li, L. (2022). Cryptocurrency trading: a comprehensive survey. *Financial Innovation, 8*(1), 13. https://doi.org/10.1186/s40854-021-00321-6

Gourieroux, Christian &, Hencic, Andrew &, & Jasiak, Joann. (2020). Forecast performance and bubble analysis in noncausal MAR(1,1) processes. *Journal of Forecasting, 40*. https://doi.org/10.1002/for.2716

Gourieroux, C., Hencic, A., & Jasiak, J. (2018). *Forecast performance in noncausal MAR (1, 1) processes*, researchGate, https://doi.org/10.13140/RG.2.2.25969.53601.

Gourieroux, C., & Jasiak, J. (2015). *Semi-parametric estimation of noncausal vector autoregression. Working papers 2015-02*. Center for Research in Economics and Statistics.

Gourieroux, C., & Jasiak, J. (2016). Filtering, prediction and simulation methods for noncausal processes. *Journal of Time Series Analysis, 37*(3), 405–430.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction, 2*. Springer.

Hencic, A., & Gourieroux, C. (2015). Noncausal autoregressive model in application to bitcoin/USD exchange rates. In V.-N. Huynh, V. Kreinovich, S. Sriboonchitta, & K. Suriya (Eds.), *Econometrics of Risk* (pp. 17–40). Springer International Publishing. https://doi.org/10.1007/978-3-319-13449-9_2.

He, Y., & Zhao, J. (2019). Temporal convolutional networks for anomaly detection in time series. *Journal of Physics: Conference Series, 1213*(4), Article 042050. https://doi.org/10.1088/1742-6596/1213/4/042050

Ho, T. K. (1995). Random decision forests. In *, 1. Proceedings of 3rd International Conference on Document Analysis and Recognition, 1* (pp. 278–282). https://doi.org/10.1109/ICDAR.1995.598994

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Jaquart, P., Dann, D., & Martin, C. (2020). Machine learning for bitcoin pricing — A structured literature review. *WI2020 Zentrale Tracks* (pp. 174–188). GITO Verlag. https://doi.org/10.30844/wi_2020_b4-jaquart

Jaquart, P., Dann, D., & Weinhardt, C. (2021). Short-term bitcoin market prediction via machine learning. *The Journal of Finance and Data Science, 7*, 45–66. https://doi.org/10.1016/j.jfds.2021.03.001

Kaiser, L. (2019). Seasonality in cryptocurrencies. *Finance Research Letters, 31*. https://doi.org/10.1016/j.frl.2018.11.007

Kim, Y., bin, Kim, J. G., Kim, W., Im, J. H., Kim, T. H., Kang, S. J., & Kim, C. H. (2016). Predicting fluctuations in cryptocurrency transactions based on user comments and replies. *PloS One, 11*(8), Article e0161197.

Kim, G., Shin, D.-H., Choi, J. G., & Lim, S. (2022). A deep learning-based cryptocurrency price prediction model that uses on-chain data. *IEEE Access, 10*, 56232–56248. https://doi.org/10.1109/ACCESS.2022.3177888

Koo, E., & Kim, G. (2021). Prediction of Bitcoin price based on manipulating distribution strategy. *Applied Soft Computing*, 110. https://doi.org/10.1016/j.asoc.2021.107738

Kursa, M. B., & Rudnicki, W. R. (2010). Feature selection with the boruta package. *Journal of Statistical Software, 36*(11), 1–13. https://doi.org/10.18637/jss.v036.i11

Lea, C., Flynn, M. D., Vidal, R., Reiter, A., & Hager, G. D. (2017). Temporal convolutional networks for action segmentation and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 156–165).

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *, 86. Proceedings of the IEEE* (pp. 2278–2324). https://doi.org/10.1109/5.726791

Li, Y., Jiang, S., Li, X., & Wang, S. (2022). Hybrid data decomposition-based deep learning for Bitcoin prediction and algorithm trading. *Financial Innovation, 8*(1), 31. https://doi.org/10.1186/s40854-022-00336-7

Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020). A CNN-LSTM-based model to forecast stock prices. *Complexity, 2020*, 1–10.

Marthinsen, J. E., & Gordon, S. R. (2022). The price and cost of bitcoin. *The Quarterly Review of Economics and Finance*. https://doi.org/10.1016/j.qref.2022.04.003

McNally, S., Roche, J., & Caton, S. (2018). Predicting the price of bitcoin using machine learning. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)* (pp. 339–343).

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*. https://doi.org/10.1007/s00521-020-05129-6

Nakamoto, S. (2008). Bitcoin whitepaper. *URL: Https://Bitcoin.Org/Bitcoin.Pdf-(: 17.07. 2019)*.

Ng, A. Y. (2004). Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning* (p. 78).

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 Systems with Applications, 198*, Article 116804. https://doi.org/10.1016/j.eswa.2022.116804

Parente, M., Rizzuti, L., & Trerotola, M. (2024). A profitable trading algorithm for cryptocurrencies using a neural network model. *Expert Systems with Applications, 238*, Article 121806. https://doi.org/10.1016/j.eswa.2023.121806

Philippas, D., Rjiba, H., Guesmi, K., & Goutte, S. (2019). Media attention and Bitcoin prices. *Finance Research Letters, 30*, 37–43. https://doi.org/10.1016/j.frl.2019.03.031

Ranjan, S., Kayal, P., & Saraf, M. (2023). Bitcoin price prediction: A machine learning sample dimension approach. *Computational Economics, 61*(4), 1617–1636. https://doi.org/10.1007/s10614-022-10262-6

Shen, D., Urquhart, A., & Wang, P. (2019). Does twitter predict Bitcoin? *Economics Letters, 174*, 118–122.

Shlens, J. (2014). *A tutorial on principal component analysis*. https://doi.org/10.48550/arxiv.1404.1100.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological), 58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x