Signal Prediction in cryptocurrency tradeoperations: a machine learning-based approach

J. M. Toledo [a] (jefferson.morais@academico.ifpb.edu.br), D. Y. Souza [a] (damires@ifpb.edu.br)

[a] Federal Institute of Paraíba, Brazil, Avenida Primeiro de Maio, 720,Jaguaribe João Pessoa - PB, CEP: 58015-435

Corresponding Author:
J. M. Toledo
Federal Institute of Paraíba, Brazil, Avenida Primeiro de Maio, 720,Jaguaribe João Pessoa - PB, CEP: 58015-435
Email: jefferson.morasis@academico.ifpb.edu.br

Signal Prediction in cryptocurrency tradeoperations: a machine learning-based approach

J. M. Toledo[a], Damires Yluska de Souza[a]

[a]*Federal Institute of Paraíba, Brazil, Avenida Primeiro de Maio, 720,Jaguaribe João Pessoa - PB, CEP: 58015-435*

Abstract

Deciding the right time to purchase a cryptocurrency is a crucial factor in enhancing a return on a given investment. In this work, we propose the use of gradient boosting algorithms (XGBoost and LightGBM) to perform the prediction of a binary market entry signal. For that, we use as features, in addition to prices and trading volumes, some technical market indicators. We use PCA (Principal Component Analysis) to reduce the dimensionality of the training/test datasets and Bayesian optimization to tune hyperparameters of the classification models. We have verified that the resulting strategy presents better results than a simple buy-and-hold of cryptocurrencies in the portfolio.

Keywords: Cryptocurrency trading, Gradient boosting algorithms, PCA

1. Introduction

Cryptocurrencies are digital and decentralized currencies that use cryptography to record their transactions Ashford & Schmidt (2022). The concept of Bitcoin, the first and most famous cryptocurrency, was disclosed in 2008 in a work that defined cryptocurrency as a digital payment method that would allow the sending of amounts directly between parties, without the need for intermediation by a financial institution Nakamoto & Bitcoin (2008).

*Corresponding author.
Email addresses: `jefferson.morasis@academico.ifpb.edu.br` (J. M. Toledo), `damires@ifpb.edu.br` (Damires Yluska de Souza)

Cryptocurrencies have drawn the attention of investors, given the high volatility of the markets and the possibility of gains in scale. It just so happens that it is not a simple task to buy an asset at the right time to, taking advantage of its appreciation, profit from the operation. On the contrary, making the right decision to profit from a given investment is one of the most complex tasks of any investor Hitam & Ismail (2018). Choosing the wrong way (or a bad time) to enter a market can lead to substantial losses Murphy (1999). In this sense, a method that uses the study of market action is called technical analysis. It is accomplished primarily through the use of graphics, to predict future price trends Murphy (1999). Investors and specialists who trade in the stock and cryptocurrency markets mostly use such a method to guide their assets' buying and selling operations.

Recently, we have experienced the growth of Machine Learning (ML), driven not only by the availability of data but also due to the increasingly powerful processing of computers. This area of knowledge allows machines to learn from past data and make predictions Kubat (2017); Nobre & Neves (2019). Bringing this possibility to the context of this work, a question may be defined: is it possible to use ML combined with technical analysis methods to assist investment decision-making?

In the present work, we use gradient boosting algorithms to predict crypto buy signals. The objective of this work is to implement ML models to solve a binary classification problem. As a result, we observe that using our ML model allows achieving returns that are superior to a buy-and-hold investment strategy. This strategy regards the pure maintenance of the currencies in one's portfolio.

This work is organized as follows. In Section 2, we present some theoretical foundations and a literature review on the topic addressed. In Section 3, we address the development of the work, presenting the data used, how we have obtained some market variables and built some supervised ML models. The results obtained with the development of the work are presented in Section 4 and, finally, we point out, in Section 5, some final considerations and possibilities of future work.

2. Background and related works

In this section, we provide some concepts regarding ML and technical analysis. We also discuss some related works.

2.1. Gradient boosting algorithms

Boosting algorithms use an iterative sequence of weak learners, i.e., learners that are slightly better than random guessing, to obtain a strong learner Schapire (1999); Freund et al. (1999). If the base learners are decision trees, these models are called boosted trees or tree boosting.

In general, supervised learning is a mathematical method by which it is possible to, given a training dataset $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, find a function $f(\mathbf{x})$ that maps the features, $\mathbf{x}$ into a target variable y James et al. (2013); Alpaydin (2020). The predicted values of the target variable, $\hat{y}$, are determined by the ML algorithm in such way that it minimizes an objective function given by

$$obj = L(\theta) + \Omega(\theta), \tag{1}$$

where θ represents the model parameters, $L(\theta)$ is the training loss, which measures how well the model performs the predictions. $\Omega(\theta)$ is the regularization term, that is added to avoid overfitting and is related to the complexity of the model.

Gradient boosting trees are algorithms obtained by interactively adding the predictions of each tree. If we consider that the prediction of each tree is given by $f_i(\mathbf{x})$, in a boosted tree algorithm with K trees, the predicted output is given by xgboost developers

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) = \hat{y}_i^{K-1} + f_K(x_i). \tag{2}$$

If the base learners are trees, we can write each f_i as xgboost developers:

$$f_i(\mathbf{x}) = \mathbf{w}_{q(\mathbf{x})}, \quad w \in R^T, \quad q : R^d \to \{1, 2, ..., T\}, \tag{3}$$

where q is a function that maps the features into one leave and $\mathbf{w}$ is a vector of scores on leaves, T is the number of leaves and d is the dimension of each data point. In this scenario, we can also write the objective function, Eq. (1), as

$$obj = \sum_{i}^{n} l(y_i, \hat{y}_i + \sum_{k=1}^{K} \Omega(f_k) \tag{4}$$

Instead of solving directly the optimization problem of reducing the loss in Eq. (4), the parameters adjustment in Eq. (3) is done by the gradient descent algorithm Bentéjac et al. (2021).

As an example of a gradient boosting algorithm, XGBoost (Extreme Gradient Boosting) is designed to be efficient and flexible, providing parallel tree boosting that solves ML problems in a fast and accurate way Chen & Guestrin (2016). In this algorithm, the regularization term of the objective function of Eq. (4) is given by

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2. \tag{5}$$

The regularization is an important aspect of XGBoost in order to prevent overfitting and also to reduce the time for training the model, given the reduction of the complexity of the algorithm Bentéjac et al. (2021).

XGBoost is also recognized by winning ML competitions Nielsen (2016) and building trees in a parallel way, using the CPU cores of the computers, contrasting with the traditional Gradient Boosting algorithms Nobre & Neves (2019).

Another gradient boosting tree algorithm is the LightGMB, which was developed by Microsoft, focusing on the efficiency and scalability of the ML model Ke et al. (2017). Compared to other boosting trees, LightGBM saves time and computational cost, allowing the researchers and developers to deal with big datasets Bentéjac et al. (2021). To avoid scanning all the instances of the training dataset, LighGBM introduced new methods: Gradient-based One-Side

Sampling (GOSS) and Exclusive Feature Bundling (EFB) Ke et al. (2017). The first method reduces significantly the number of training instances, while the last one bundle mutually exclusive variables, reducing the number of features Ke et al. (2017).

In this work, we use XGBoost and LightGBM to train and solve a binary classification problem, comparing their performance with the predictions of a logistic regression classifier.

2.2. Principal Component Analysis

Principal Component Analysis (PCA) is a technique used to summarize a large dataset into one with a smaller number of representative variables James et al. (2013). The reduction of features in the data is performed in such a way that the maximum amount of variance is preserved, which maintains the differences among instances Alpaydin (2020).

The PCA is, itself, an unsupervised learning algorithm, since it does not involve a target variable in the process James et al. (2013). The first principal component in PCA is a normalized linear combination of the features. The posterior components are a linear combination of the features that have "maximal variance out of all linear combinations that are uncorrelated with" the previous component James et al. (2013).

In this work, we use PCA to reduce the number of features and compare the results with that produced in a scenario without the application of such a method, similarly to what was done by Nobre & Neves (2019).

It is important to note that, before PCA is performed, the features must be centered and have mean zero James et al. (2013). Thus, we apply normalization to the features in all experiments accomplished in this work, as discussed in Section 3.

2.3. Bayesian optimization for hyperparameter tuning

While training, ML algorithms search for parameters that reduce the error (loss) for the predictions in the training dataset James et al. (2013). Before the model training phase and, while designing the ML model architecture and preparing the algorithm for the training step, the developer can set a class of variables called hyperparameters, which can impact the performance of the algorithms and also the reproducibility of the work Liashchynskyi & Liashchynskyi (2019); Feurer & Hutter (2019); Wu et al. (2019).

There are some traditional methods for searching for optimal hyperparameters, such as grid search, which trains the model for each combination of the hyperparameters in a given search space Liashchynskyi & Liashchynskyi (2019); Wu et al. (2019). These traditional methods, although, demands professional knowledge, experience and can be computationally expensive Wu et al. (2019).

In this scenario, Bayesian optimization is an effective method to solve complex problems that deal with finding maximum/minimun of a function, and saving computational costs Brochu et al. (2010); Wu et al. (2019). It is based on the Bayesian theorem. Thus it sequentially searches for the optimal point of a certain function, based on its prior evaluations Wu et al. (2019).

The Bayesian optimization of an ML model hyperparameter has been tested with many ML models and with standard datasets Wu et al. (2019), showing good results, when considering the time cost. The method also has shown good results in ML competitions, when compared with hyperparameter tuning approaches Turner et al. (2021). In this work, we use the Bayesian optimization of the Python package scikit-optimize Head et al. (2018) to search for the best hyperparameters for the gradient boosting algorithms.

2.4. Related works

Combining machine learning with technical analysis of financial markets is a challenge that has stimulated technical production on several fronts Nobre & Neves (2019); Chan Phooi M'ng & Mehralizadeh (2016); Pinto et al. (2015); Singh et al. (2019); Leippold et al. (2021). In the literature, it is possible to find studies in this field for established markets such as stock Singh et al. (2019); Leippold et al. (2021); Mehtab & Sen (2020) or futures market Pinto et al. (2015). Some investment platforms already provide customers with the ability to use supervised ML algorithms natively to carry out trading and investment operations.

The cryptocurrency market, however, is much more recent than the stock market, having only emerged a little over 10 (ten) years ago. As a result, the use of ML algorithms in this market still needs to be explored and needs further studies Zhang et al. (2021). Even so, it is possible, in the literature, to find articles that discuss, for example, the use of several algorithms to perform predictions in operations with cryptocurrencies Peng et al. (2018); Hitam & Ismail (2018); Chowdhury et al. (2020); Alonso-Monsalve et al. (2020).

In this scenario, we can mention the use of the algorithm called Support Vector Machines (SVM) to analyze the volatility of cryptocurrency markets Peng et al. (2018) or to predict the transactions prices Hitam & Ismail (2018). Recently, Chowdhury et al. used the algorithms K Nearest Neighbours (KNN), Artificial Neural Networks (ANN), boosted trees and the ensemble of them to predict the future price of cryptocurrencies obtaining accuracy near 92% and overcoming the classical temporal series analysis algorithms such as autoregressive integrated moving average (ARIMA) Chowdhury et al. (2020).

We can also cite the use of deep learning in the temporal series analysis of crypto prices, highlighting the role played by the Convolutional Neural Networks (CNN), which led to accuracies in the order of 75% Alonso-Monsalve et al. (2020); Zhang et al. (2021). It was possible to obtain a scheme that achieves state-of-the-art trading performance using ML algorithms and obtain investments with near 10% return Alonso-Monsalve et al. (2020); Zhang et al. (2021).

In this work, we discuss the use of the Gradient Boosting algorithm to predict cryptocurrency buy signals. To this end, we use literature works that address investments in stock markets Nobre & Neves (2019), adapting them to cryptocurrencies. We have verified that it is possible to make predictions and obtain returns superior to the usual strategies for buying and holding assets (buy and hold).

We propose an approach to predict signals of buying/selling cryptocurrencies, similar to what was done by Nobre & Neves (2019) for the stock market but considering the specificity of the crypto transactions. Differently to what has been done to the cryptocurrencies, Hitam & Ismail (2018); Chowdhury et al. (2020); Zhang et al. (2021), we do not try to predict the price of the currencies, but the market signals.

3. Proposed approach

In this section, we present the data used in this work. We also talk about obtaining derived features, setting the target variable, and training the supervised machine learning models. In Fig. 1, we synthesize all the steps we follow in our work and about which we discuss in this section. We obtain cryptocurrency trading data, calculate the market indicators and perform a split in the dataset to obtain the training and test datasets, which are used to build supervised learning models, and to evaluate them. We use the Python programming language (Van Rossum & Drake Jr (1995)) in the steps presented in Fig. 1.

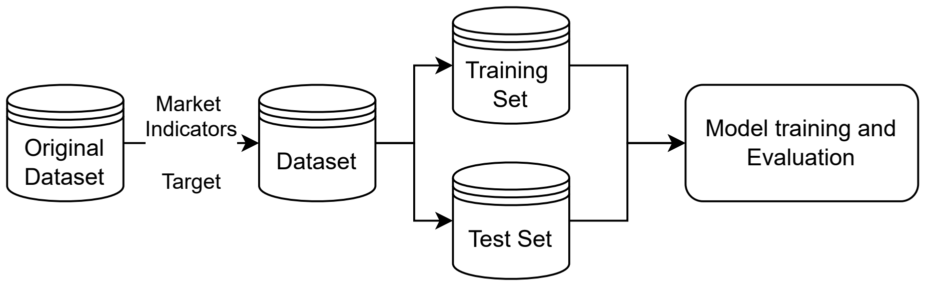

Figure 1: Blueprint of the proposed analysis.

3.1. The dataset

Similarly to what happens in the stock market, the information on cryptocurrency trades is recorded, for each time frame considered, with the data on the market opening price (Open), the closing price (Close), the trading volume (Volume), the highest price (High) and lowest price (Low) reached by the currency. It is important to point out that the dollar was used to assess the prices of all digital currencies considered in this work.

It is possible to obtain cryptocurrency trading data from several sources and, in this work, the API provided by Yahoo Finance[1] was used to this end.

In this work, we present some obtained results regarding five cryptocurrencies, namely: Bitcoin (BTC), Ethereum (ETH), Binance Coin (BNB), Cardano (ADA), and Ripple (XRP). They were chosen because they have the highest trading volumes among the oldest cryptocurrencies. With this scenario in mind, we can guarantee the greatest possible amount of data for the development of machine learning models. It is also important to point out that we use the day as the graphic operation time, and this temporal aggregation value is used as an index in the datasets.

3.1.1. The target variable

A trade operation is successful when it is possible to purchase an asset that has its value increased after the buying operation. Thus, the target variable that we use acquires a value of 1 when the closing price on the following day was higher than that of the trading day. If there is a reduction in the asset's market value on the following day, the target variable has a value of 0. The variable described here can be represented mathematically in equation Nobre & Neves (2019)

$$y_t = \begin{cases} 0, & \text{if } \frac{Close_{t+1} - Close_t}{Close_t} \geq 0 \\ 1, & \text{if } \frac{Close_{t+1} - Close_t}{Close_t} < 0 \end{cases}. \tag{6}$$

Thus, we can observe that our objective is to anticipate the raising of the close price of a cryptocurrency. If the model predicts the raising of the price in the next day ($y_t = 1$), we buy the cryptocurrency and maintain it in our wallet until the signal changes to 0, or, in other words, the model predict the decreasing of the crypto price.

[1]https://www.yahoofinanceapi.com/

As already mentioned, we use ML to solve a binary classification problem since our target variable only has two possible values (0 or 1).

3.1.2. Feature augmentation: the market indicators

Initially, as already noted, the initial cryptocurrency datasets have only the prices and volume of the coins. We know, however, that technical market analysts use, in addition to the analysis of candlestick charts resulting from these values, several other indicators to guide their decisions Murphy (1999); Pring (2002). We have therefore calculated several of these indicators thus adding them to the dataset.

We list, in Table A.7 of Appendix Appendix A, the data originally taken from the API used as source and the various technical indicators calculated from them, with a brief description of them. These features were chosen following the work of Nobre & Neves (2019).

3.2. Data preprocessing

Firstly, it is important to remember that using the same dataset to train the model (learning the model parameters) and evaluating its performance usually leads to overfitting James et al. (2013). Thus, we perform the process of splitting the datasets into train dataset (which we use to train the model) and test dataset (which is used to evaluate the predictions).

In our problem, we need to make predictions to be used in future market operations. Thus, the splitting of the dataset uses temporal criteria: the first 80 % of the data is used for training and the last 20 % for testing the signal prediction.

Each variable listed in Table A.7 has its range of values. Therefore, it is necessary to normalize the values of the dataset features thus providing them with the same values range. To this end, we use the min-max normalization mentioned, the same used in Alpaydin (2020); Nobre & Neves (2019). It consists in subtracting the value of a feature (x) by its minimum value and dividing the result of this subtraction by the subtraction of the maximum and the minimum value of the feature. This operation can be represented by the following equation

$$x' = \frac{x - min(x)}{max(x) - min(x)}. \tag{7}$$

Thus, by using min-max normalization, all the features have values in the range $[0, 1]$.

Another important aspect dealt with in this work regards the calculation of the market variables, as described in Section 3.1.2, which increases the original number of features from 5 (five) to 30 (thirty). It is known that the reduction of the dimensionality of the training dataset can both allow the visualization of the model since humans can only see in three dimensions, and increase results when assessing the quality and performance of the models. In fact, in Nobre & Neves (2019), the authors used Principal Component Analysis (PCA) technique to reduce the dimensionality of a dataset and obtained the best results.

In our work, we consider two experimental scenarios: (i) with the application of PCA, by using the parameters presented in Nobre & Neves (2019); (ii) without PCA. In Section 4, we present the evaluation of the models in both scenarios.

Finally, as mentioned in Section 3.1, we use data of five different cryptocurrencies. Thus, for each one of the five datasets, we apply the described preprocessing steps, as well as the PCA dimensionality reduction when considering the first experimental scenario (with PCA).

3.3. Model training

In this section, we describe the strategies accomplished in order to train ML models along with the parameters which have been set.

We have used three supervised classification algorithms: a logistic regression classifier, a LighGBM classifier, and an XGBoost classifier. The first one, a simple statistical method, is used as a baseline, i.e., to serve as a predefined standard by which the other ML methods are measured or compared. This is because a logistic regression classifier is considered a classification method that allows a researcher to rapidly train and analyze its results Alpaydin (2020). It has been trained with the default hyperparameters of the Scikit-learn library Pedregosa et al. (2011).

LighGBM and XGBoost methods have indeed similar hyperparameters. In the present work, we tune the following ones: boosting_type, max_depth, num_leaves, subsample, colsample_bytree, and learning_rate. These hyperparameters were chosen because they have been showing increasing in the results obtained by the boosted trees and allow to prevent overfitting Brownlee (2019).

The parameter boosting_type defines the booster used in the algorithm. By default, the "Gradient Boosting Decision Tree" (gbdt) is used. We also tested the "Dropouts meet Multiple Additive Regression Trees" (dart), which uses dropout into the additive regression trees to prevent overfitting Vinayak & Gilad-Bachrach (2015).

The maximum depth of a tree can be set by the hyperparameter max_depth. The parameter num_leaves represents the maximum tree leaves for each base learner. The subsample is a hyperparameter used to prevent overfitting and represents the ratio of the training instances. Setting, for example, the subsample parameter to 0.5 means that the algorithm "would randomly sample half of the training data prior to growing trees and this may prevent overfitting" xgboost developers. Similarly, colsample_bytree is "the subsample ratio of columns when constructing each tree".

The values of the hyperparameters applied in this work are listed in Table 1.

To tune the hyperparameters of the boosted trees, we have used the Bayesian optimization algorithm Snoek et al. (2012). The Bayesian optimization uses a Gaussian process regression for global optimization of the hyperparameters and allows us to speed up the process of parameter searching. After the training process, we are able to get the best predictions of the models and their correspondent hyperparameters.

Similar to what we have performed in the preprocessing phase, the models have been trained for each one of the five cryptocurrency datasets. The hyperparameter search is performed for each dataset in our work.

Parameter	BTC
boosting_type	['gbdt', 'dart']
max_depth	Integer(5,9)
num_leaves	Integer(6,17)
subsample	[0.7, 0.8, 0.9, 1.0]
colsample_bytree	[0.8, 0.9, 1.0]
learning_rate	Real(0.05, 0.1)

Table 1: Hyperparameter search space.

In Table B.8 of Appendix Appendix B, we present the best hyperparameters for the LightGBM algorithm in each cryptocurrency dataset, for both scenarios, namely: (i) with and (ii) without the application of PCA.

In addition, we have trained the XGBoost classifier for the training data, with and without the application of PCA. The best hyperparameters obtained for the models are displayed in Table B.9.

4. Results and discussion

After training the models, we are able to evaluate the predictions using the test data.

In this paper, we have used the same metrics considered in Nobre & Neves (2019).

The accuracy measure represents the proportion of correct predictions of a model. It can be calculated by

$$\text{Accuracy} = 100 \times \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}, \tag{8}$$

where we multiply the fraction by 100 to obtain the result as a percentage.

On the other hand, the sharpe ratio can be calculated by Nobre & Neves (2019)

$$\text{Sharpe Ratio} = \frac{\text{Average returns}}{\text{Standard deviation Of returns}}, \tag{9}$$

The sharpe ratio allows the investors to compare the return of the investment with its risk.

We have also used the return on investment (ROI) to evaluate our models. This number represents the amount of return in a particular application, i.e., in our study, it denotes the buying of a given cryptocurrency. The return of the investments guided by the algorithms discussed in this work are depicted in Figs. 2 to 6.

For BTC, for example, we can see that the buy and hold strategy would lead to a loss in the period from March to December 2021, while the signals generated by the LightGBM model, without the application of PCA, would lead to the best result. The XGBoost model with PCA would also overcome the result of buying and holding BTC. It is important to notice that, for this cryptocurrency, the logistic regression preceded by the PCA did not predict buying signals, as we can see in Figure 2.

Algorithm	Accuracy	ROI	Sharpe Ratio
Logistic regression	0.51	-29.74 %	-1.71
Logistic regression + PCA	0.48	-	-
LightGBM	0.52	- 22.73 %	-1.02
LightGBM + PCA	0.53	- 32.33 %	-1.13
XGBoost	0.49	- 38.90 %	-1.67
XGBoost + PCA	0.54	-13.68	-0.86

Table 2: Evaluation for Bitcoin.

Concerning the ETH, we can see in Fig. 3 that the model XGBoost presented the highest return on investment, while the logistic regression preceded by the PCA would lead to the worst buying signals.

In Fig. 4, it is shown that the models XGBoost + PCA, XGBoost, and

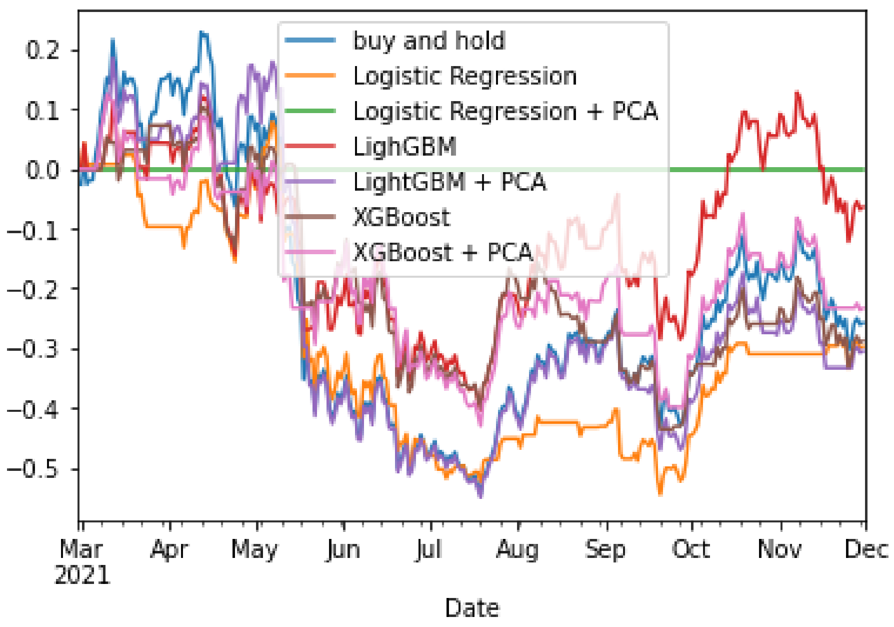

Figure 2: Investment evolution for BTC.

Algorithm	Accuracy	ROI	Sharpe Ratio
Logistic regression	0.53	6.43 %	-0.56
Logistic regression + PCA	0.44	-48.15 %	-1.85
LightGBM	0.52	70.58 %	0.93
LightGBM + PCA	0.44	-4.81 %	-0.53
XGBoost	0.53	111.21 %	0.76
XGBoost + PCA	0.43	- 20.75 %	-1.07

Table 3: Evaluation for Ethereum.

LightGBM + PCA would lead to profit in investments, while the buy and hold would let to lose.

For the cryptocurrency ADA, all machine learning signals overcome the buy-and-hold strategy. The logistic regression model had the best result, with the highest accuracy and return, but with a few buy signals.

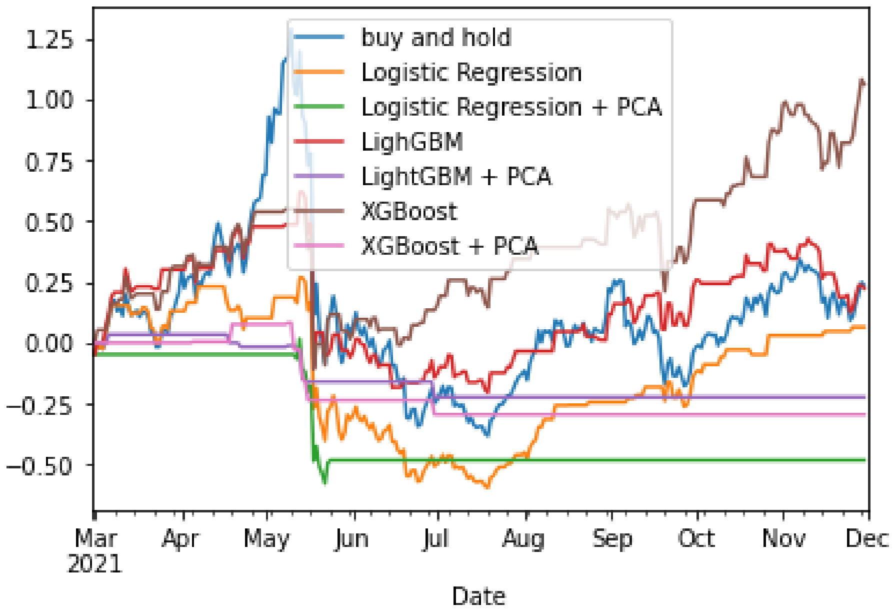

Figure 3: Investment evolution for ETH.

Algorithm	Accuracy	ROI	Sharpe Ratio
Logistic regression	0.45	0.65 %	16.52
Logistic regression + PCA	0.51	-54.24 %	-1.48
LightGBM	0.52	- 33.67 %	-0.32
LightGBM + PCA	0.54	72.41	0.87
XGBoost	0.50	39.76 %	0.58
XGBoost + PCA	0.54	100.02 %	1.167

Table 4: Evaluation for BNB.

Finally, we can see in Fig. 6 that all the machine learning predictions also overcome the buy and hold strategy, with LightGBM and XGBoost having the best returns.

In summary, we can observe, analyzing the tables from 2 to 6 and the figures from 2 to 6, that the investment strategies based on the boosted algorithms led

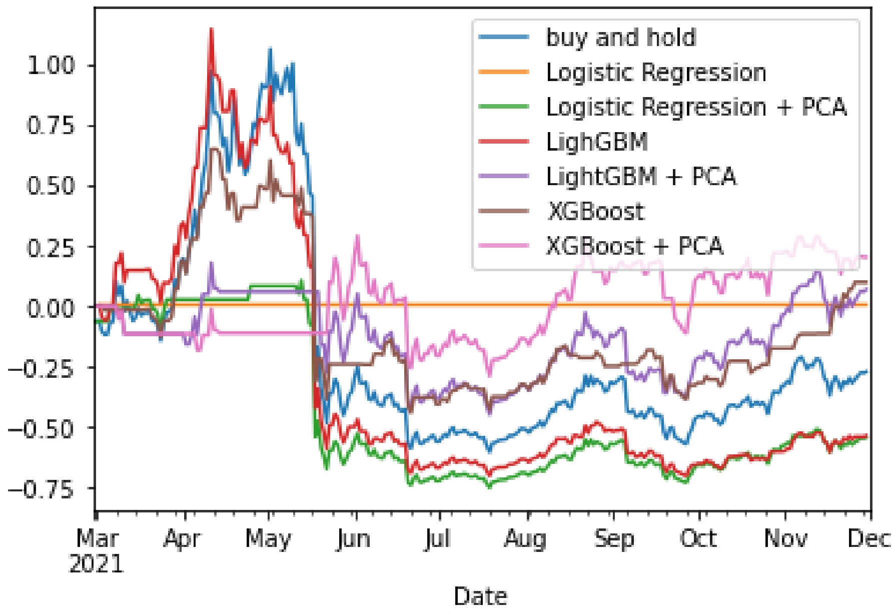

Figure 4: Investment evolution for BNB.

Algorithm	Accuracy	ROI	Sharpe Ratio
Logistic regression	0.52	- 4.41 %	-1.10
Logistic regression + PCA	0.48	- 13.57 %	-19.68
LightGBM	0.48	- 41.64 %	-2.71
LightGBM + PCA	0.47	- 4.01 %	-1.45
XGBoost	0.50	- 42.71 %	-1.74
XGBoost + PCA	0.48	- 54.08 %	-0.80

Table 5: Evaluation for ADA.

to higher returns (or lower losses) for all cryptocurrencies, excepting ADA, for which logistic regression was superior.

It is important to emphasize that the order of magnitude accuracies obtained in this work were equivalent to the ones obtained in similar experiments developed in the stock market Nobre & Neves (2019). Even though it is not

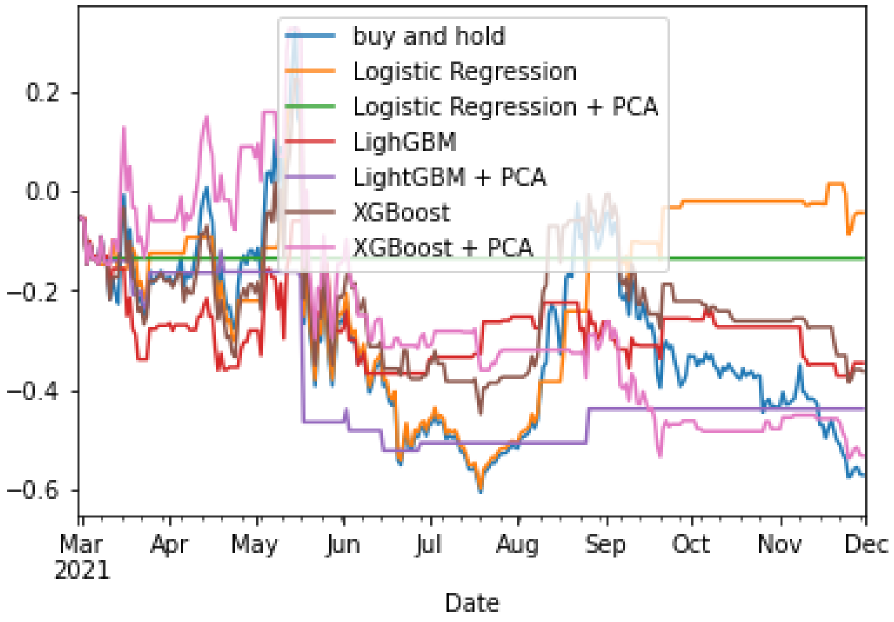

Figure 5: Investment evolution for ADA.

Algorithm	Accuracy	ROI	Sharpe Ratio
Logistic regression	0.49	2.76 %	0.29
Logistic regression + PCA	0.48	-	-
LightGBM	0.50	36.12 %	1.49
LightGBM + PCA	0.49	- 13.15 %	-0.85
XGBoost	0.50	29.09 %	1.58
XGBoost + PCA	0.50	5.09 %	0.48

Table 6: Evaluation for XRP.

uncommon to build supervised ML models that obtain more than 90% of accuracy, the values obtained in this work are comparable with the literature and we need to accentuate the good results obtained in the problem domain area. The behavior of the market is influenced by human emotions and cannot be easily predicted Murphy (1999)

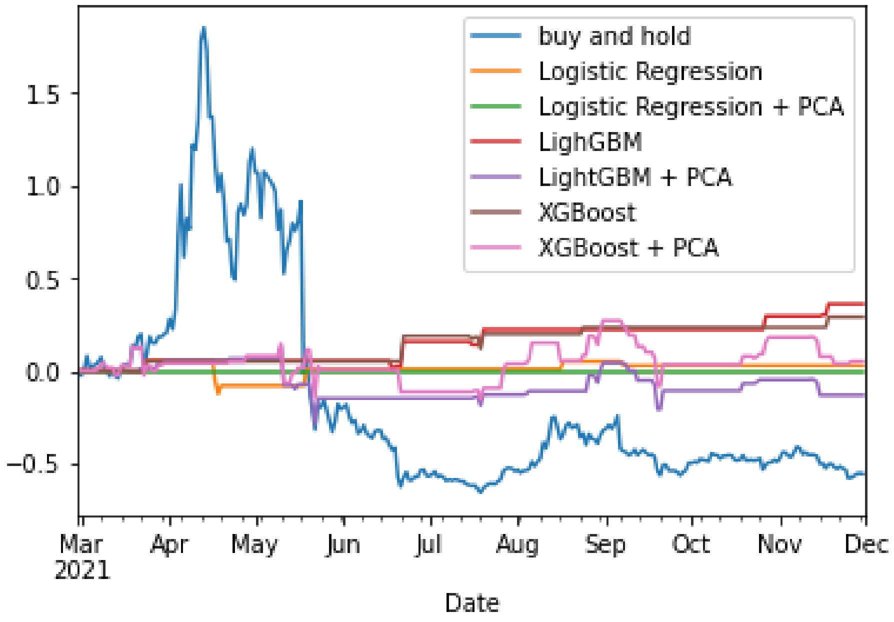

Figure 6: Investment evolution for XRP

5. Concluding remarks and perspectives

Machine learning algorithms have been used in several branches of human knowledge. In this work, we present the use of gradient boosting algorithms to predict cryptocurrency purchase signals, which generate higher returns than the purchase and maintenance of the currency in the portfolio (a strategy known as holding).

It is important to highlight that the present work is limited to the extraction and processing of data from cryptocurrencies, as well as the development and evaluation of machine learning models. For the techniques presented here to be successfully used in cryptocurrency trading operations, some aspects still need to be developed. It is essential, for example, to deal with the implementation (deployment) of the solution, to define how to carry out the integration of the prediction system with the portfolio of possible users so that trading operations can be carried out. It is also important to address the availability of a deployed system, as it needs to constantly monitor opportunities to purchase digital assets.

It should also be noted that the prediction of cryptocurrency buy signals can be associated with other machine learning algorithms. It is possible, for example, to forecast the prices of cryptocurrencies by combining time series analysis techniques with machine learning. It is also possible to try to make the model presented here more robust with the combination (ensemble) of some algorithms as well.

Data Availability Statements

All data analyzed in this work is publicly available on the internet through the website https://www.yahoofinanceapi.com.

Appendix A. Market indicators

In Table A.7, we briefly list and describe the market indicators which were used as features for training the ML models. The complete description of these variables can be founded in the technical analysis bibliography (Murphy, 1999; Pring, 2002).

Appendix B. Best hyperparameters

References

Alonso-Monsalve, S., Suárez-Cetrulo, A. L., Cervantes, A., & Quintana, D. (2020). Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. *Expert Systems with Applications*, *149*, 113250.

Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

Ashford, K., & Schmidt, J. (2022). What is cryptocurrency? URL: `https://www.forbes.com/advisor/investing/what-is-cryptocurrency/`.

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, *54*, 1937–1967.

Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, .

Brownlee, J. (2019). Xgboost with python. *Machine Learning Mastery*, .

Chan Phooi M'ng, J., & Mehralizadeh, M. (2016). Forecasting east asian indices futures via a novel hybrid of wavelet-pca denoising and artificial neural network models. *PloS one*, *11*, e0156338.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

Chowdhury, R., Rahman, M. A., Rahman, M. S., & Mahdy, M. (2020). An approach to predict and forecast the price of constituents and index of cryptocurrency using machine learning. *Physica A: Statistical Mechanics and its Applications*, *551*, 124569.

xgboost developers (). Xgboost documentation. `https://xgboost.readthedocs.io/en/stable/`. Accessed: 2022-02-07.

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In *Automated machine learning* (pp. 3–33). Springer, Cham.

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, *14*, 1612.

Head, T., MechCoder, L. G., Shcherbatyi, I. et al. (2018). scikit-optimize/scikit-optimize: v0. 5.2. *Version v0*, *5*.

Hitam, N. A., & Ismail, A. R. (2018). Comparative performance of machine learning algorithms for cryptocurrency forecasting. *Ind. J. Electr. Eng. Comput. Sci*, *11*, 1121–1128.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* volume 112. Springer.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*.

Kubat, M. (2017). *An introduction to machine learning*. Springer.

Leippold, M., Wang, Q., & Zhou, W. (2021). Machine learning in the chinese stock market. *Journal of Financial Economics*, .

Liashchynskyi, P., & Liashchynskyi, P. (2019). Grid search, random search, genetic algorithm: A big comparison for nas. *arXiv preprint arXiv:1912.06059*, .

Mehtab, S., & Sen, J. (2020). A time series analysis-based stock price prediction using machine learning and deep learning models. *International Journal of Business Forecasting and Marketing Intelligence*, *6*, 272–335.

Murphy, J. J. (1999). *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.

Nakamoto, S., & Bitcoin, A. (2008). A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, *4*.

Nielsen, D. (2016). *Tree boosting with xgboost-why does xgboost win" every" machine learning competition?*. Master's thesis NTNU.

Nobre, J., & Neves, R. F. (2019). Combining principal component analysis, discrete wavelet transform and xgboost to trade in the financial markets. *Expert Systems with Applications*, *125*, 181–194.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, *12*, 2825–2830.

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

Pinto, J. M., Neves, R. F., & Horta, N. (2015). Boosting trading strategies performance using vix indicator together with a dual-objective evolutionary computation optimizer. *Expert Systems with Applications*, *42*, 6699–6716.

Pring, M. J. (2002). *Technical analysis explained: The successful investor's guide to spotting investment trends and turning points*. McGraw-Hill Professional.

Schapire, R. E. (1999). A brief introduction to boosting. In *Ijcai* (pp. 1401–1406). Citeseer volume 99.

Singh, S., Madan, T. K., Kumar, J., & Singh, A. K. (2019). Stock market forecasting using machine learning: Today and tomorrow. In *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)* (pp. 738–745). IEEE volume 1.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*.

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., & Guyon, I. (2021). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 Competition and Demonstration Track* (pp. 3–26). PMLR.

Van Rossum, G., & Drake Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Vinayak, R. K., & Gilad-Bachrach, R. (2015). Dart: Dropouts meet multiple additive regression trees. In *Artificial Intelligence and Statistics* (pp. 489–497). PMLR.

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., & Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, *17*, 26–40.

Zhang, Z., Dai, H.-N., Zhou, J., Mondal, S. K., García, M. M., & Wang, H. (2021). Forecasting cryptocurrency price using convolutional neural networks with weighted and attentive memory channels. *Expert Systems with Applications*, *183*, 115378.

Variable	Description
Original dataset	
Open	Market opening price
Close	Market opening price
Volume	Trading volume
High	highest price (High)
Low	Lowest price
Added market Indicators	
RSI (relative strength index)	Measures the relative internal strength of a stock or cryptocurrency against itself
MACD (Moving Average Convergence / Divergence)	Evaluates the trading deviation of a market, using two exponential moving averages
PPO (Percentage Price Oscillator)	Percentual difference between two moving averages
Momentum	Difference between the closing price in a given time and the close price some periods ago
CCI (Commodity Channel Index)	Informs if the price is above or below the historic average price
ROC (Rate of Change)	Rate at which a price changes over a given period of time
Stochastic (Stochastic Oscillator)	Shows the relation between the close price and the high-low range in some period
Williams %R	Compares the close price with the past high/low prices in some period
Simple Moving Average (20, 50, 100)	Ratio between the sum of prices and the number of observations in a given period
Exponential Moving Average (20, 50, 100)	Exponentially weighted moving average. Places greater significance in recent periods
Bollinger Bands (Upper, Middle, and Lower)	Envelopes at a standard deviation level above and below a moving average
OBV (On-Balance Volume)	Indicates the tendency of a price through the cumulative volume of the cryptocurrency
Chaikin Oscillator	Measures the accumulation-distribution line of MACD
MFI (Market Facilitation Index)	Shows the price movement for each volume unity
ATR (Average True Range)	Measures the market volatility by considering gaps in prices

Table A.7: Original data and market indicators.

Table B.8: Best parameters for LightGBM models.

	LightGBM					LightGBM + PCA				
Parameter	BTC	ETH	BNB	ADA	XRP	BTC	ETH	BNB	ADA	XRP
boosting_type	gbdt	gbdt	dart	dart	dart	gbdt	gbdt	dart	gbdt	dart
max_depth	7	5	5	6	6	6	7	6	7	6
num_leaves	15	9	15	12	8	9	16	12	13	10
subsample	1.0	0.8	0.9	0.9	0.7	1.0	0.7	0.7	0.8	0.7
colsample_bytree	0.9	1.0	1.0	0.9	0.8	1.0	0.9	0.9	0.9	0.9
learning_rate	0.06466	0.06437	0.07246	0.09094	0.07387	0.09433	0.07118	0.07681	0.08503	0.06851

Note: We present the parameter for both scenarios, namely with the application of PCA and without it.

Table B.9: Best parameters for XGBoost models.

	XGBoost					XGBoost + PCA				
Parameter	BTC	ETH	BNB	ADA	XRP	BTC	ETH	BNB	ADA	XRP
boosting_type	gbtree	gbtree	gbtree	gbtree	dart	dart	dart	dart	dart	dart
max_depth	6	7	9	6	6	9	5	6	6	7
num_leaves	15	13	13	9	7	7	13	10	14	17
subsample	1.0	0.9	0.9	1.0	0.8	0.9	1.0	0.7	0.7	0.9
colsample_bytree	0.8	0.8	0.9	1.0	1.0	0.9	0.8	0.9	0.9	0.8
learning_rate	0.05563	0.06144	0.08164	0.07447	0.07854	0.05223	0.09217	0.05063	0.07111	0.07817

Note: We present the parameter for both scenarios, namely with the application of PCA and without it.