Cryptocurrency price direction prediction through ensembles of machine learning algorithms allied with percentage resampling

PEDRO FERNANDES, Instituto Superior Técnico, Portugal

This work presents a system with the aim of generating a profitable bitcoin trading strategy, based on machine learning models trained on historical data. Before defining our algorithms' target variable, a resampling was performed to group sequential data points that lead to absolute percentage price movements of around 4%. Multiple technical and time-based features were generated and the problem was then approached as a binary task. Four different machine learning models were trained - Logistic Regression, Support Vector Machine, Random Forest, and XGBoost. Different trading strategies were then generated, either directly from the individual models' predictions or from different ways of combining them, and evaluated in a 9-month trading simulation period. The results proved that every single model beat the Buy and Hold strategy used as a baseline. Furthermore, the best single-model strategy (XGBoost) had a Return on Investment of 94.78%, and the best ensemble strategy achieved 119,76% for the same metric, over a period of 9 months. These results were obtained during a turbulent period, where bitcoin's price decreased by over 30%.

Additional Key Words and Phrases: Machine Learning, Logistic Regression, Support Vector Machine, Random Forest, XGBoost, Ensemble Voting, Bitcoin, Technical Analysis

1 INTRODUCTION

Launched in 2009 [1], Bitcoin's popularity and price have skyrocketed in recent years. Due to its frequent significant price movements, bitcoin's high-risk/high-reward profile quickly attracted the attention of individual investors [2] and, more recently, institutional investors [3].

Like bitcoin, there are hundreds of other cryptocurrencies being traded every day through cryptocurrency exchanges like Binance [4]. Recently, the total cryptocurrency market capitalization (calculated by multiplying the price of the cryptocurrency with the number of coins in circulation) has spiked from 150 billion in March of 2020 to around 2.5 trillion in November of 2021 [5]. This surge in price has seen a proportional surge in interest, both from individual and institutional investors. Supported by its potential for extreme and fast profit, the cryptocurrency market has quickly become one of the largest unregulated markets in the world [6].

The cryptocurrency market is included in what is called a financial market - a system that provides buyers and sellers the means to trade financial instruments, including bonds, equities, the various international currencies, and derivatives. Due to this, many believe that it is possible to exploit the strategies utilized in other financial markets (e.g. stock or foreign exchange markets) like technical analysis and algorithmic trading. On the other hand, some experts believe that the market cannot be timed and, therefore, the best strategy would be to Buy and Hold. In this work, multiple algorithmic trading strategies were designed by machine learning models based on technical analysis, and compared to the Buy and Hold strategy.

Author's address: Pedro Fernandes, Instituto Superior Técnico, Lisbon, Portugal, pedrogsvfernandes@gmail.com.

1.1 Motivation

Due to its recency, the cryptocurrency market is still highly speculative which leads to an extremely volatile market. This means that price changes in the range of 5 to 10% in a single day are not out of the ordinary [8]. For that reason, the cryptocurrency market appears to be an interesting opportunity to apply machine learning methods using what is already known from the traditional stock market but also exploring problem-specific features.

1.2 Objectives

The main objective of this work is to build a machine learning model capable of accurately predicting the price movement of bitcoin and making a profit even when the general trading fees are accounted for.

We intend to improve on the state-of-the-art by applying, combining and improving on the findings of previous works that will be further analyzed in Section 2. We will study and compare the performance of various machine learning models that proved effective in previous works and also explore data preparation, feature engineering and hyper-parameter tuning techniques.

In our work, the task of predicting the cryptocurrency's price movement was approached as a binary classification problem with tabular data. To capitalize on the cryptomarket's high volatility and regular significant price changes, a resampling was performed by grouping 1-min candles that lead to a specific price percentage change (around 4%), and these new candles will be our target labels. To feed our model, two categories of features were used:

(1) Traditional technical analysis
(2) Time-based information

Regarding the first category, our models will operate with previously tested and well-known technical indicators such as the RSI and MACD indicators. As our data will be resampled based on percentage change, a need arises to maintain our models' notion of time, thus leading us to the second category - Time-based information.

1.3 Thesis Contribution

The main contributions from this work are:

(1) The creation of a new percentage-based resampling technique, that better aligns the target variable with the objective of a system of this nature (to be profitable);
(2) The introduction of an equation that allows for the labelling of positive and negative asymmetric candles that lead to break-even positions when they happen sequentially;
(3) Training multiple models that overwhelmingly beat the Buy and Hold strategy, during the entire 284-day test period.

1.4 Document Structure

This document is structured as follows:
In Section 2, some background concepts are discussed and related literature is reviewed. In Section 3, the proposed solution is described and each of its components is thoroughly detailed. Section 4 goes over the results and a case study, designed to find the optimal way of combining the single-model predictions. Section 5 concludes.

2 STATE OF THE ART

Although the use of technical analysis in an effort to predict price movements has been thoroughly studied in the literature, the study of machine learning techniques applied to predicting the cryptocurrency market only started in more recent years. As such, the literature regarding some domain-specific topics such as blockchain-based features is still scarce. Nevertheless, various works made important contributions that will be reviewed more thoroughly in the following subsections.

2.1 Cryptocurrency market efficiency

Fama [14] and the follow-up paper by Fama [15] introduced the Efficient Market Hypothesis (EMH), where the author states that financial markets are efficient, meaning that prices reflect all available information rendering it impossible to beat the market. This conclusion has been questioned and disputed frequently in the literature. In regards to this work's object of study, a few authors have discussed the cryptocurrency market's efficiency and its correlation to traditional asset classes such as stocks, bonds, oil, gold, and other commodities. Regarding efficiency, recent works by Andrew Urquhart [16], Bariviera et al. [17] and Vu LeTran et al. [18] share the conclusion that the cryptocurrency market is significantly inefficient, however, it has been moving towards efficiency. Correlation between the cryptocurrency market and traditional assets has been found to be weak by multiple authors over the years. Baur et al. [19], Bouri et al. [20], and Pyo et al. [21] all found weak connections between cryptocurrencies and traditional assets.

2.2 Data Resampling

In their work, Borges et al. [9] tested multiple machine learning models and two resampling strategies: amount resampling and percentage resampling. Various technical indicators were computed on the resampled data-sets and used as features. One of the consequences of using the resampled data was that the algorithm was significantly more active during high volume periods, something that proved to be more profitable. The conclusion was that regardless of the utilized learning algorithm, the outcome of utilizing resampled data consistently generated significantly higher returns than the traditionally used time-sampled data.

The resampling that obtained the best results was the percentage-based resampling. With this strategy, the authors grouped consecutive candles whose cumulative sum of absolute price variation would be equal or greater than a specific threshold.

This work will introduce a variation of this resampling technique, by taking the real price percentage variation between candles. This new technique will be further explained in Section 3.2.1.

2.3 Blockchain-based features

As this problem differs from traditional stock market trading, we initially intended to take advantage of domain-specific information through blockchain-based features. Although the literature on this topic is still scarce, a few authors have attempted to use blockchain-based features having achieved different levels of success.

Jaquart et al. [22] combined minutely data of four major categories - technical, asset-based, sentiment-based, and blockchain-based. The last category included the number of bitcoin transactions and the growth of the mempool (a cryptocurrency node's mechanism for storing information on unconfirmed transactions). The results indicated that the technical features were the most important for all prediction horizons (1, 5, 15, and 60 minutes). However, the relative importance of this category decreased from 80% in the 1-min horizon to less than 50% in the 60-min horizon where blockchain-based and sentiment-based features gained relevance. It is also pertinent to note that the best accuracy results were obtained for the larger prediction horizons, with a maximum accuracy of 56% for the 60-min horizon and 52% for the 1-min horizon.

Ji et al. [23] fed various deep learning models with 18 blockchain-based features and also the bitcoin price time series. The results were not promising with the models yielding a maximum accuracy of 53% and barely making a profit despite the fact that the authors did not consider trading fees. Given the high transaction fees charged in cryptocurrency markets, it is fair to question whether Ji et al.'s system could result in negative returns. Despite these results, this work brought to light an interesting finding regarding the contrast in performance between classification and regression-based algorithms, which will be later discussed.

Sebastiao et al. used ensemble learning methods and achieved an annualized return, after proportional trading costs of 0.5%, 9.62%, and 5.73%, when trading ethereum and litecoin respectively, during a bear market with daily mean returns lower than -0.20%. The ensembles included linear models, RF, SVM, and their binary counterparts. Regarding the features used, each model was optimized in the validation set by testing different sets of features. Around 1/3 of the models used network based features. On the other hand, all models used the lag returns of the tested cryptocurrencies, the day-of-the-week dummies (as the name implies, this feature provides information about the day-of-the-week) and the lagged volatility proxies proving the importance of these three features. Given the mixed results observed, blockchain-based features were not further explored.

2.4 Machine Learning Models

Various different machine learning approaches have been applied to financial markets' forecasting, some with better results than others.

As explained previously, some machine learning problems are better suited for classification tasks while others are better modeled as regression-based tasks. Therefore, it is important to start by understanding whether our problem should be approached as a classification or a regression task.

The previously mentioned authors, Ji et al. [23] tested the performance of multiple state-of-the-art deep learning approaches and

Ref.	Year	Used methodologies	Performance	Main contribution
[9]	2020	LR, RF, GTB, SVM, EV	≈55% acc., ≈1000% ROI,	Volume and percentage resampling
[22]	2021	NN, FFNM LSTM and GRU, RF, GTB, Ensemble	≈56% acc.	Blockchain-based data had more influence on large prediction horizons
[23]	2019	DNN, RNN, LSTM, CNN, DRN, Ensemble	≈53% acc.	Classification beats regression for algorithmic trading
[24]	2019	LSTM, GB	≈68% F1-score	LSTM outperformed GB
[25]	2018	LSTM and GTB	Signifcant profit even with transaction fees up to 0.2%	GTB outperformed LSTM for shorter 5-10 days windows while LSTM performed better for 50-days windows
[26]	2019	SVM, ANN	≈62% hit-rate	SVM had good results regardless of the chosen timeframe, while the ANN had poor results for some periods but abnormal returns during bull-runs.
[27]	2021	SVM, RF, Linear models	Annualized returns or 9.62% with 0.5% trading fees	Ensemble outperformed its individual parts
[28]	2019	ANN, SVM, Ensemble	62.9% acc.	Model assembling yielded positive outcomes

Table 1. Summary of the most relevant works mentioned in the state-of-the-art.

compared the results for classification and regression-based algorithms. The outcome was strongly in favor of classification as every classification-based deep learning model beat its regression-based counterpart. The authors concluded that classification models were more effective than regression models for algorithmic trading. Therefore, in our work, we will be using classification-based models.

When it comes to state-of-the-art in the machine learning field, deep learning takes the spotlight. But despite the great results obtained by deep learning techniques in fields of research like Natural Language Processing and Computer Vision, research regarding their application to Financial Markets has seen mixed results.

For instance, in the same study mentioned above, Ji et al. [23] concluded that, given the poor results obtained by all the tested models, it is still premature to solely use deep learning models for algorithmic Bitcoin trading. In another paper, Kwon et al. [24] used an LSTM and five features: open price, close price, high price, low price, and volume at each time epoch (i.e., every 10 minutes) as input to predict future price movements. The author found that, regardless of the tested cryptocurrency, LSTM always outperformed the Gradient Boosting model used as comparison. For reference, the LSTM's f1-score was approximately between 63 and 68% for the seven cryptocurrencies tested, while GB obtained an f1-score between 59 and 63% approximately. Alessandretti et al. [25] compared the performance of Gradient Boosting Decision Trees and an LSTM model tasked with predicting daily price variations of different cryptocurrencies. The authors found that the simpler models based on gradient boosting decision trees achieved the best results for short 5 to 10 days input windows. When dealing with larger 50 days windows, the LSTM model outperformed the simpler models. Souza et al. [26], with a similar task and input that included open, high, low, and close prices of Bitcoin, Gold, and Silver, compared the performance of an Artificial Neural Network and a Support Vector Machine. The results showed that, when predicting bitcoin price movement, the SVM model produced similar positive results regardless of the sample period tested with a hit-rate (percentage of profitable trades) between 57.32 and 59.75% for all three time samples. On the other hand, the ANN model proved to be more inconsistent having poor results for some periods but generating better returns in others. The ANN's hit-rate was between 51.38 and 61.73% for all 3 time samples, when predicting bitcoin's price movement. The authors conclude that the SVM strategy should be used by investors willing to achieve more conservative returns on a risk-adjusted basis but the ANN proved that it can generate greater profits during short bull runs. Nevasalmi et al. also compared an ANN model to several other simpler models in a 3-class classification task. The clear winner was the tree-based GBM with the ANN model coming at a distant second place and having only slightly better results than the other tested models.

On the other hand, ensemble models seem to come out on top consistently in works that provide a comparative analysis between models. Borges et al. [9] tested the performance of two linear methods - Logistic Regression and Support Vector Machine, two non-linear methods - Random Forest and Decision Tree Gradient Boosting, and an ensemble of these 4 algorithms. The results showed that the ensemble outperformed every other algorithm in both return on investment and accuracy. Sebastião et al. [27] tested Linear models, Random Forest, Support Vector Machine, and their binary versions, in a total of 6 individual models. Three ensembles were then built, based on three different voting thresholds (each model required at least 4, 5, or 6 votes to enter a long position). The conclusion was that the ensemble beat the individual models and also the Buy and Hold strategy that was used as a benchmark. Mallqui et al. [28] also found that model assembling yielded positive outcomes when predicting price movements in the cryptocurrency market.

3 SYSTEM ARCHITECTURE

In this chapter, we explain in detail the proposed solution to our bitcoin trading system. First, a brief overview of the system's main architecture is given. Then, the following subsections detail the most important modules more thoroughly.

3.1 System Overview

Our system's architecture consists of four layers, composed by five modules in total. Figure 1 is a representation of the flow of information in our proposed solution.

In the data extraction module, we simply collect the 1-minute candles from 2018-01-01 to 2022-01-27 (49 months) directly from Binance's API. This data includes, for each candle the Opening time, Opening price, Closing price, Highest price, Lowest price, Volume and Number of trades.

In the next layer, we start by calculating some technical indicators, like the RSI and MACD on the 1-minute candles. The new table is then resampled by percentage-change - the candles are grouped into larger candles that represent absolute price movements of around 4%. Afterward, the final technical and time-based features are computed. These features are computed from arithmetic transformations on the base features listed in Table 2. As this is a time-series problem, lagged features are also computed.

Next, the computed features are normalized to the [0,1] range and the training and test sets are established following a holdout set strategy. Our test-set was a period of 284 days, between 2021-04-18 and 2022-01-27, while our train set contained data from 2018-01-01 to 2021-04-18 (39,5 months). Finally, the training set is re-balanced using a random under-sampling technique.

The Hyper-parameters tuning module, receives the training set from the previous module, and, for each algorithm, it determines the best hyper-parameters for the type and amount of data we have. This is done through Bayesian optimization, using stratified K-Fold cross-validation.

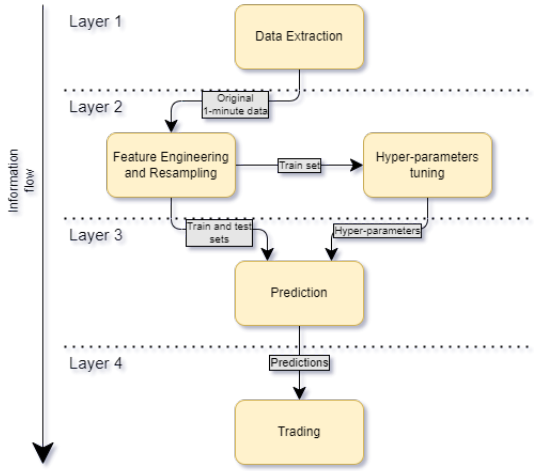

Fig. 1. System architecture overview

The recently computed features and hyper-parameters are then passed on to the prediction layer. Here, four machine learning algorithms were tested - Logistic Regression, Support Vector Machine, Random Forest, Gradient Tree Boosting - as well as multiple Ensemble Voting strategies.

The predictions are then passed through to the trading layer where a trading simulation is performed. We use the binary predictions to decide when to enter and leave the market. With every transaction, the whole value of our portfolio is used. This means, at any given time, our portfolio is composed of either 100% bitcoin or 100% money.
A more detailed description of the most important modules follows.

3.2 Feature engineering and resampling

In this step, the original data is resampled following a variation from the original approach from Borges et al [9], as explained in detail in the next Section.

Time-based and technical features are also computed in this step. The RSI and MACD features are computed before the resampling, while the others are computed after the resampling.

The target variable is also computed: each new data point gets labeled according to the price movement observed in the next candle. If the price movement, in percentage, is negative, then it will belong to class 0. Otherwise, if it is positive, it will belong to class 1.

3.2.1 Resampling. The 1-minute time-sampled raw data is passed on to this module. Here, groups of consecutive candles that represent either positive or negative price movements above a certain threshold are aggregated to form single candles. As previously mentioned, this step is loosely based on Borges et al.'s percentage-based technique. There are three two differences:

(1) The price variation is calculated as the price difference (in percentage) between the first and last candle pertaining to the same resampled candle (as opposed to the cumulative sum of every candle's absolute volatility);
(2) The technical features are computed on the original time-sampled data and then aggregated through averages (as opposed to being computed on the resampled data).

So while the previous technique's resampled candles contained similar absolute cumulative volatility, they could represent a wide range of price variations. On the other hand, the resampling technique used in this work, provides resampled candles that can represent different levels of absolute cumulative volatility but they always represent the same absolute price variation.

Initially, the price variation threshold was set at 4% for both positive and negative movements. But this proved not to be the optimal solution, as percentage variations are not symmetrical - e.g. a 10% price variation, followed by a -10% drop will leave us at a loss - which means a system at around 50% prediction accuracy will most likely not be profitable, even if we disregard the transaction fees. This led us to find the formula that establishes the relation between positive and negative price changes. In other words, given a positive price change, we can compute the corresponding negative price change that will lead to a break-even position, when disregarding

transaction fees. The formula follows:

$$negVariation = 1 - \frac{1}{1 + posVariation} \quad (1)$$

Substituting *posVariation* by 0.04 (4%) in the above formula leads to y ≈ 0.0384 (3.84%), which was the threshold used for the negative price variations. This small change consistently improved our results regardless of the positive variation chosen.

Figure 2 shows an example, for illustrative purposes, of a resampling procedure, both using the approach from Borges et al. [9] and also this work's approach. In this table, each separate coloured group will form a new candle, corresponding to its group number.

As explained earlier, it is easily observed that the candles provided by Borges et al.'s [9] resampling may lead to candles with no significant price change (candle 5), but always with similar total absolute cumulative volatility. On the other hand, the candles provided in this work always contain a significant price movement and, when applied to real 1-minute candles that have lower volatility, most candles will represent a similar absolute price change.

We believe our approach is more beneficial as it aligns the metrics that the models are optimized for (accuracy) with the system's objective (being profitable): with Borges et al's resampling, when training a model, an incorrect prediction on a 0.1% price variation candle will have the same weight as an incorrect prediction on a 5% price variation candle, although the latter is clearly more important than the former, when trying to create a profitable trading strategy. With the resampling proposed in this work, each data point will hold similar absolute price variations, thus, optimizing accuracy will presumably optimize returns.

As a final note, we believe the resampling introduced in this work also lines up better with the message that the technical indicators convey: an RSI value above 70 does not indicate that the price will drop in the next time-period (as implied when using traditional time-sampled data and predicting the next candle), but a high RSI over a few candles does suggest that the next significant price movement should be a negative one.

Index	Open	High	Low	Close	Absolute variation	Cumulative sum *	Assigned group (Borges et al.)	Price variation *2	Assigned group (this work)
1	23 202	23 202	20 808	20 831	10,22%	**10,22%**	1	**10,22%**	1
2	20 831	21 357	20 785	21 139	1,48%	1,48%	2	1,48%	2
3	21 139	21 692	21 077	21 517	1,79%	3,27%	2	**3,29%**	2
4	21 517	21 517	20 912	21 416	0,47%	3,74%	2	-0,47%	3
5	21 417	21 661	20 920	21 517	0,47%	**4,21%**	2	0,00%	3
6	21 516	21 832	21 171	21 365	0,70%	0,70%	3	-0,71%	3
7	21 365	21 801	21 325	21 565	0,94%	1,64%	3	0,22%	3
8	21 564	21 815	20 122	20 250	6,09%	**7,73%**	3	**-5,89%**	3
9	20 250	20 344	19 850	20 034	1,07%	1,07%	4	-1,07%	4
10	20 034	20 151	19 543	19 550	2,42%	3,48%	4	**-3,46%**	4
11	19 551	20 395	19 551	20 296	3,81%	**7,29%**	4	**3,81%**	5
12	20 296	20 558	19 560	19 793	2,48%	2,48%	5	-2,48%	6
13	19 793	20 469	19 793	20 044	1,27%	3,74%	5	-1,24%	6
14	20 050	20 203	19 584	20 126	0,38%	**4,12%**	5	-0,84%	6
15	20 132	20 428	19 765	19 953	0,89%	0,89%	6	-1,69%	6
16	19 953	20 044	19 661	19 831	0,61%	1,50%	6	-2,29%	6
17	19 832	20 018	19 595	20 000	0,85%	2,34%	6	-1,46%	6
18	20 000	20 043	19 651	19 793	1,03%	3,38%	6	-2,48%	6
19	19 793	20 169	19 687	19 894	0,51%	**3,89%**	6	-1,98%	6

Fig. 2. Resampling example.
* Group closes when 5% threshold is exceeded.
*2 From current candle to last candle in previous group, group closes when 3% threshold is exceeded.

3.2.2 Feature engineering. The final base features to be considered, described in Table 2, are then computed on the resampled data.

Two types of features were considered - technical and time-based. The former were computed before the resampling, and for each new candle, the mean values of the original candles were computed. On the other hand, the time-based features were computed after the resampling, with the intent of maintaining the notion of time within our data-set.

From the base features computed in the last step, which are listed in Table 2, multiple other features were computed through data manipulation or arithmetic operations. The full list of operations performed on the base features follows.

- Column shifts to create lagged variables;
- Subtraction (from last candle to current one);
- Percent variation (from last candle to current one);
- Rolling average;
- Exponentially weighted average;
- Division by Delta;
- Division by Close price;
- Negative and positive candle amplitude.

These operations were also performed sequentially in specific cases (e.g. the percent variation between the Closing price rolling average from the last candle to the current one).

Too many features were computed to analyse each of them individually, but the 15 most important ones listed in Figure 4 will be analysed in Section 4.2.

Feature	Description
Close	Closing price.
Open	Opening price.
High	Highest price.
Low	Lowest price.
Num_Trades	Total number of trades.
RSI_14	Average fourteen-period RSI.
RSI_60	Average sixty-period RSI.
MACD	Average MACD.
MACD_Signal	Average MACD signal.
MACD_Sub	Average difference between MACD and MACD signal.
Var_Percent	Percentage variation between the previous Closing price and the current one.
Max_price	Max. price achieved up to the given point.
Max_volume	Max. volume achieved up to the given point.
Delta	Time elapsed during the current candle.

Table 2. Base features.

3.3 Hyper-parameter tuning

The most common strategies for this hyper-parameter tuning are Grid Search and Random search. As the names imply, Grid Search works by trying every possible combination of hyper-parameters given in the search space, while Random search tries random combinations of values. Both of these have very clear drawbacks: Grid

search is very computationally expensive, while Random search can miss important combinations of values.

For the reasons above, Bayesian optimization through Hyperopt was used in this work. Bayesian optimization is a probabilistic model-based technique used to find the minimum of any function. It takes into account past evaluations when choosing the next combination of hyper-parameters to try. Therefore, it chooses its hyper-parameter combinations in an informed way and can rapidly learn to focus on a smaller section of the search space, where the results are best.

The minimized function by Bayesian optimization was the K-fold validation accuracy score (with k=10), multiplied by negative one.

3.4 Prediction layer

The prediction layer receives the optimized hyper-parameters, the training set and the test set. The first two are utilized to train each model and, after the training phase is complete, the test set is received by the trained model which outputs a price movement prediction between two classes. Following, a description of the algorithms tested in this work is given:

3.4.1 Logistic Regression. A Logistic regression uses a logistic function to map input parameters to class probabilities. The output of this function will always be between 0 and 1 and corresponds to the probability of the record's true class being the positive one. In this work, the threshold considered to predict a class as positive was 0.5, not only for this algorithm but also for all others that output a probability.

Logistic regression generates a linear decision boundary, which might prove too basic for the problem we are trying to solve. Regardless, it is a good baseline for most machine learning problems and will serve as comparison for the more complex algorithms.

3.4.2 Support Vector Machine. A Support Vector Machine is a binary classification algorithm that works by finding the hyperplane in an N-dimensional plane that best separates the data points, where N is the number of features.

To find the ideal hyper-plane, the algorithm maximizes the margin between the hyper-plane itself and the support vectors (the closest points of each class).

Kernel operations determine how similar two points are, after a given linear or non-linear transformation. The original SVM algorithm is a linear classifier, but by transforming the feature space through a non-linear kernel operation, a non-linear hyper-plane can be found, thus improving results in non-linear problems.

3.4.3 Random Forest. A Random Forest is an ensemble of decision tree models. This kind of approach leverages the fact that a combination of relatively uncorrelated models will outperform its constituents [32].

To optimize an ensemble's performance, the constituents must be relatively uncorrelated so that there is a higher probability that wrong individual predictions are corrected by the majority vote. To ensure that the decision trees are uncorrelated, bagging (also called bootstrap aggregating) and feature randomness were used. Both these methods work by extracting a random sample of the available data. A brief description of each of these techniques follows:

- Bagging - A random sample of data points is selected to train each tree;
- Feature randomness - A random subset of features is considered when splitting nodes.

3.4.4 Gradient Tree Boosting. GTB is similar to RF, in the sense that it combines the efforts of multiple weak learners in order to improve the accuracy of a strong learner which is the final model. Also, as with the Random Forest algorithm, bagging and a variation of feature randomness (here, a subset of features is considered for each tree, as opposed to each split) were also applied here to keep the tree's correlation as low as possible.

However, the main difference lies in how the trees are combined. In a random forest, the trees are combined after being created, while in GTB the trees are created and added to the model sequentially, with the end goal of optimizing an objective function.

As the XGBoost package was used to implement this algorithm, the terms XGBoost and Gradient Tree Boosting were used interchangeably throughout this work.

3.4.5 Ensemble. The basic majority-vote ensemble included three of the previous algorithms - SVM, RF and XGBoost as these had the best individual results. Each of them will predict the next price movement, which will count as a vote. A simple majority vote decides the ensemble's prediction for the next candle.

Different ensemble strategies were also tested in Section 4.3 by defining two different vote thresholds: one for entering the market and one for exiting the market. Lower thresholds will account for a more active trading strategy, while higher thresholds will lead to a more passive strategy with longer periods between trades.

3.5 Trading layer

This layer receives the predictions and simulates trades accordingly. For each data-point, a prediction that the next candle will be positive is represented by a label "1", while the opposite is represented by "0". These labels are converted to trading signals by the control function f, defined as follows:

$$f = \begin{cases} "BUY" & \text{if not } currently\_in() \text{ and } label = 1 \\ "SELL" & \text{if } currently\_in() \text{ and } label = 0 \\ \text{"Do nothing"} & else \end{cases}$$

where $currently\_in()$ is a function that returns True if the system is currently holding bitcoin, and False otherwise.

As we were dealing with historical data, this simulation was done through backtest trading. Thus, two assumptions are made in this module, the same ones mentioned by Borges et al [9] in their work:

(1) Market liquidity: The markets have sufficient liquidity to complete any trade placed by the system immediately and at the current price of its placement.
(2) Capital impact: The capital invested by the algorithm has no influence on the market as it is relatively insignificant.

The trading system starts every experiment with an initial sum of 1000$. First, each model was tested individually and, afterwards, a few strategies to combine the predictions of the individual models were tested.

4 RESULTS

In this Section, a brief explanation of the evaluation methods is given, followed by the obtained results for the base models and resulting trading strategies as well as their interpretation. Lastly, a case study is presented.

4.1 Work evaluation methodology

A few data-set characteristics are summarized in Table 3.

Dataset	Start	End	Total Time	# Candles
Train set	01/01/2018	18/04/2021	~39,5 months	1255
Test set	18/04/2021	27/01/2022	~9 months	314

Table 3. Data-set characteristics.

The evaluation of our model will be split into two parts - model and financial evaluation.

The performance metrics that were used to evaluate our model are described below:

- Accuracy - The percentage of correct predictions;
- Precision - The quality of a positive prediction;
- Recall - The percentage of positive records found;
- Specificity - The quality of a negative prediction;
- F1-Score - How correct and balanced the model is.

In this work, we considered the positive class to be the one that represents positive price movements.

It is necessary to consider a few other metrics that evaluate how profitable our trading strategies are. The financial metrics considered in this work follow:

- ROI - The ratio between net income and initial investment.
- Hit-rate - The ratio of winning or profitable trades.
- Sharpe Ratio - It measures how much excess return is gained in comparison to a risk-free asset, while adjusting for the additional risk.
- Total trades - Total trades completed.
- Average time in market - Average trade duration.

.

4.2 Results Analysis

The following results are based on a trading simulation performed on a 9-month period, from April 2021 to January 2022. In this simulation, each strategy started with 1000 US dollars.

With each transaction, a fee of 0.1% was considered. This is the fee charged by Binance on a regular trading account. Table 4 summarizes the results of each strategy, with and without fees.

Figure 3 shows the portfolio value of each strategy throughout the simulation, considering the 0.1% fee. XGBoost had by far the best performance, while the Buy and Hold strategy used as a baseline was the worst performer. A more in-depth analysis will be given throughout the following Sections.

4.2.1 Model evaluation. Table 5 summarizes the results obtained by the various models, as evaluated by machine learning metrics.

XGBoost had the best accuracy out of all models with 58.28%, beating the next highest by more than 3%. The other tree-based

Strategy	Final money (0.1% fee)	Final money (no fees)
Buy and Hold	646.53	647.83
Logistic Regression	1014.70	1102.56
SVM	981.84	1041.54
Random Forests	1314.12	1516.25
XGBoost	1947.84	2297.46
Ensemble Voting	1212.19	1394.35

Table 4. Trading strategies' final portfolio value comparison.

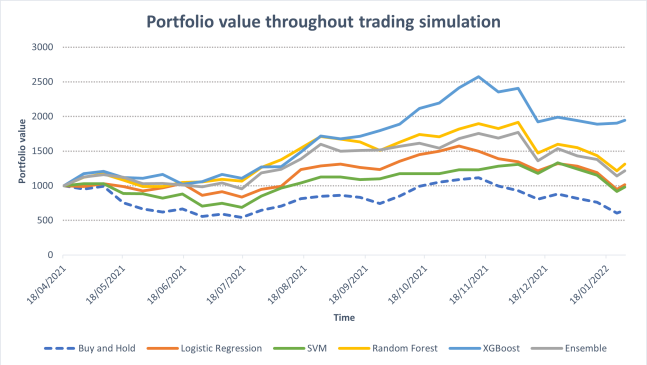

Fig. 3. Portfolio values comparison throughout the simulation period.

model - Random Forest, and the Ensemble voting model followed at 54.7% accuracy. The Logistic Regression and Support Vector Machine had significantly worse results in terms of accuracy with 52.8% and 52.2% respectively.

All models except Random forests had extremely high Recall. This tells us that the models were mostly predicting positive moves. This is especially true when looking at the two worse performing models - LR and SVM.

Figure 4 shows the feature importance plot from the best performing model (XGBoost), as measured by "weight" - the number of times a feature appears in a tree. From this graph we can see that the most used feature is simply the percentage variation from the last candle.

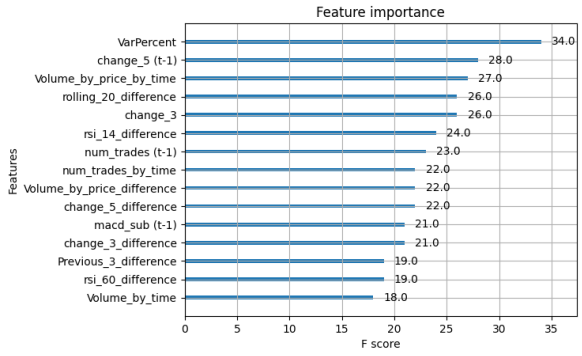

Fig. 4. Feature importance plot

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
LR	52.8	51.4	86.4	64.4
SVM	52.2	51.0	78.7	61.9
RF	54.7	57.4	32.2	41.3
XGBoost	58.28	56.9	63.8	60.2
EV	54.7	53.2	70.3	60.6

Table 5. Model predictions evaluation.

4.2.2 Financial evaluation. Table 6 shows that every model beat the Buy and Hold strategy that lost close to 35% of the initial investment during the 284 day period included in the test set. The best performer, by far, was the XGBoost model with an ROI of 94.78%. The XGBoost strategy was also the most active on the market with 83 total trades and an average TiM per trade of 2 days and 6 hours. The unbalanced

Strategy	ROI	Hit-Rate	Sharpe Ratio	# Trades	Avg. TiM
B&H	-35,35%	0	-0,150	1	284 days
LR	+1.47%	0.55	0,025	42	5.62 days
SVM	-1.82%	0.67	0,08	30	5.83 days
RF	+31.41%	0.60	0,140	72	3 days
XGBoost	+94.78%	0.57	0,337	83	2,25 days
EV	+21.82	0.57	0,116	68	3 days

Table 6. Financial evaluation of each trading strategy.

predictions made by the LR and the SVM led to a long Average Time in Market. This means these strategies had higher exposure to market volatility, but still managed to beat the Buy and Hold strategy by a significant margin.

Regarding the Sharpe Ratio, a score over 1 is usually considered good, while a score under 1 is considered bad. The best score was XGBoost with a Sharpe Ratio of 0,337. As mentioned previously, the Sharpe ratio also penalizes high positive volatility (e.g. returns larger than usual), which is a factor that worsened the tree-based trading strategies' results.

Figure 5 shows a comparison between bitcoin's price and portfolio value using XGBoost's trading strategy, throughout the 9-month test period. As intended, the percentage-based resampling led to a system that is much more active during high volatility periods. On the other hand, the average time in market per trade was much smaller during the unstable times at the beginning of the simulation, which reduces the risk of trading during these highly volatile periods.

The percentage-based resampling also created an inherent stop-loss option for our model, as after every negative 3.84% candle, a new decision point is created. This is another reason why this model responds well to high volatility periods.

4.3 Case Study - Optimal ensemble voting thresholds

This case study intends to find the optimal way to combine the single-model predictions, in order to obtain the most profitable system. Two of these strategies obtained results that beat the XGBoost trading strategy.

Various different buying and selling thresholds were tested, and also different sized ensembles. For the smaller sized ensembles, the models were picked based on their individual results.

Having different buying and selling thresholds means that, at each decision point (every time a candle closes), the trading system's control function f adopted the following behaviour:

$$f = \begin{cases} \text{"BUY"} & \text{if not } curr\_in() \text{ and } up\_vote\_sum \geq tr\_in \\ \text{"SELL"} & \text{if } curr\_in() \text{ and } down\_vote\_sum \geq tr\_out \\ \text{"Do nothing"} & else \end{cases}$$

where *curr_in* is a function that returns True, if the system currently holds bitcoin, or False if not, *up_vote_sum* (*down_vote_sum*) is equal to the sum of models that predicted the market would go up (down) in the next candle, and *tr_in* and *tr_out* are the predefined buying and selling thresholds respectively.

Tables 7 and 8 summarize the obtained results for ensembles containing three and two models respectively. Analysing the results,

(Sell / Buy)	# Trades	Avg. TiM	ROI	Hit-Rate
(1 / 2)	103	1 days 13:39:49	63,01%	0,6
(1 / 3)	75	1 days 10:41:52	119,76%	0,64
(2 / 2)	68	2 days 23:53:05	21,82%	0,57
(2 / 3)	53	2 days 15:47:54	87,9%	0,58
(3 / 2)	24	10 days 19:22:32	13,96%	0,58
(3 / 3)	19	11 days 01:11:09	11,91%	0,63

Table 7. Three-model ensemble simulation results, given different Buy and Sell thresholds.

(Sell / Buy)	# Trades	Average TiM	ROI	Hit-Rate
(1 / 1)	108	1 days 18:31:13	69,64%	0,59
(1 / 2)	85	1 days 18:05:48	107,04%	0,61
(2 / 1)	58	4 days 09:24:41	26,64%	0,53
(2 / 2)	52	4 days 07:52:32	56,16%	0,5

Table 8. Two-model ensemble simulation results, given different Buy and Sell thresholds.

one can observe that independently of the number of models utilized in the ensemble, the best result always pertained to the experiences that had simultaneously the highest possible "Buy" threshold and the lowest possible "Sell" threshold.

Generally speaking, the final portfolio value increased with higher "Buy" thresholds and with lower "Sell" thresholds. Both of these fluctuations lead to a smaller average time in market, which may have led to a more risk-averse trading system. Also, in machine learning terms, having more models agreeing before entering the market would also increase our "Buy moment" precision, while allowing for less models to sell our position increased our "Sell moment" recall.

Comparing these ensemble models to the previously mentioned majority vote ensemble (the (2 / 2) three-model ensemble in this experience), we can conclude that a few other ensemble strategies greatly outperformed it.

Despite XGBoost having abnormal returns when compared to the other members of the ensemble, it was still outperformed by two of

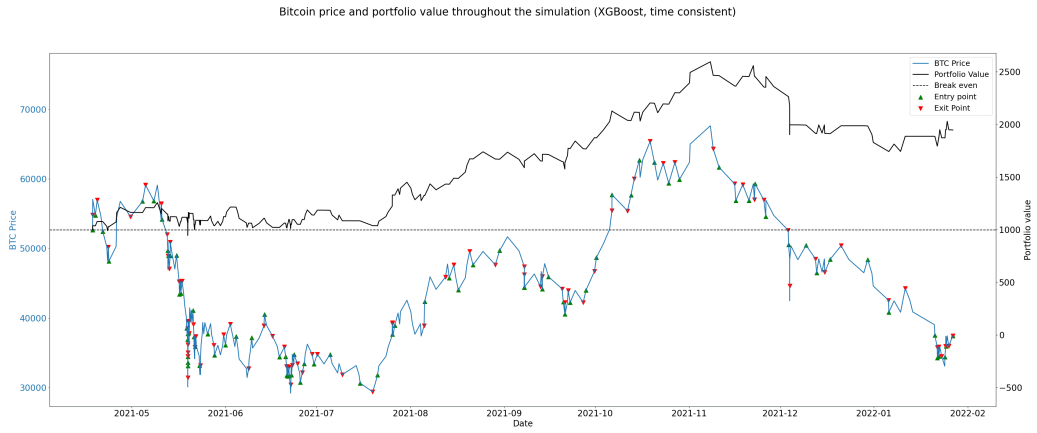

Fig. 5. XGBoost trading strategy entry graph.

the ensemble strategies. This goes in accordance with the findings of multiple papers ([9], [27], [28]) where ensembles outperformed their constituents.

The best overall result was obtained with the three-model ensemble, with a buying threshold of 3 and a selling threshold of 1. It achieved an ROI of 119,76%, and made 75 trades with 0,6 hit-rate (45 winning trades), during the simulation period of 284 days. It had the lowest average time in market per trade, with a value inferior to 1 day and 11 hours.

4.4 Overall Analysis

The best overall result was obtained by an Ensemble Voting strategy, composed of three models and with a Buy threshold of 3 and a Sell threshold of 1 with an ROI of 119.76% over 284 days. The best single model strategy was the one based on XGBoost's predictions, achieving an ROI of 94.78% over the same period. These are really promising results when compared to the baseline Buy and Hold strategy that yielded a negative ROI of -35.35% over the same period.

As for the presented case study, we found the optimal way of combining our single model predictions. Furthermore, it proved that ensembles with larger Buy thresholds and smaller Sell thresholds were more successful than other ensemble strategies.

5 CONCLUSION

5.1 Summary and Achievements

In this work, a solution that massively outperformed the market was proposed. Its design combined the findings of multiple state-of-the-art systems and introduced a new resampling approach that better lines up the target variable with the goal of being profitable.

Four machine learning models were trained and a fifth model - Ensemble Voting - was derived from their predictions. The XGBoost model was the clear winner with 58.28% accuracy.

Multiple trading strategies were derived from the model predictions and tested in the 9-month simulation period, from April 2021 to January 2022. In this period, while the market dropped by more than 35%, our single model strategies' ROI ranged from -1.82% to +94.78%. Again, the tree-based models generated the winning trading strategies, being both more active and more profitable.

In our case study, the best way to combine the single-model predictions was investigated, and two strategies actually outperformed the best single-model trading strategy with ROIs of 107.04% and 119.76%. We concluded that strategies with high "Buy" thresholds and low "Sell" thresholds were consistently more profitable.

These results are very promising, especially given the very turbulent times in which the system was tested that included both a bull and a bear market. Despite this, a few opportunities for improvement still exist, and these will be detailed in the following subsection.

5.2 Future Work

In this subsection, some of our system's limitations and their possible solutions are addressed, along with suggestions that may lead to performance improvements in future work.

- In this work, when our system received either a "Buy" or "Sell" signal, all of its portfolio value was changed from one asset to the other. Instead, one could invest only a percentage of its portfolio at a time. This percentage could be fixed or defined by a given indicator or even by the prediction probability, outputted by the model.
- With the purpose of achieving better "Buy" signal precision, one could set the positive prediction probability threshold to a number larger than the default value of 0.5. Different thresholds could be set for entering or leaving the market, similar to what was done in this work with the different ensemble voting thresholds.
- Regarding the features used, a plethora of different features could be used. One could use different technical indicators, blockchain-based features or even social networks sentiment analysis.
- Only long positions were considered in this work. One could potentially improve the results by also considering short positions.

REFERENCES

[1] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." Decentralized Business Review (2008): 21260.

[2] Hileman, Garrick, and Michel Rauchs. "Global cryptocurrency benchmarking study." Cambridge Centre for Alternative Finance 33 (2017): 33-113.

[3] Forbes article, https://www.forbes.com/sites/lawrencewintermeyer/2021/08/12/institutional-money-is-pouring-into-the-crypto-market-and-its-only-going-to-grow/?sh=6fb127ad1459. Last accessed 6 Jan 2022

[4] Binance homepage, www.binance.com. Last accessed 23 Nov 2021

[5] Coinmarketcap - Market capitalization evolution, https://coinmarketcap.com/charts/. Last accessed 23 Nov 2021

[6] Foley, Sean, Jonathan R. Karlsen, and Tālis J. Putniņš. "Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies?." The Review of Financial Studies 32.5 (2019): 1798-1853.

[7] Coinmarketcap - Listed cryptocurrencies, https://coinmarketcap.com/. Last accessed Jan 6 2022

[8] Adhikari, Ratnadip, and Ramesh K. Agrawal. "An introductory study on time series modeling and forecasting." arXiv preprint arXiv:1302.6613 (2013).

[9] Borges, Tome Almeida, and Rui Ferreira Neves. "Ensemble of machine learning algorithms for cryptocurrency investment with different data resampling methods." Applied Soft Computing 90 (2020): 106187.

[10] SVM example, https://en.wikipedia.org/wiki/Support-vector_machine Last accessed Jan 5 2022

[11] KNN example, https://scikit-learn.org/0.22/auto_examples/neighbors/plot_classification.html. Last accessed Jan 5 2022

[12] Kirkpatrick II, Charles D., and Julie A. Dahlquist. Technical analysis: the complete resource for financial market technicians. FT press, 2010.

[13] Iansiti M and Lakhani K R 2017 The Truth About Blockchain Harvard Business Review, Harvard University, hbr.org/2017/01/the-truth-about-blockchain. Last accessed Nov 22 2021.

[14] Fama, Eugene F. "Efficient capital markets a review of theory and empirical work." The Fama Portfolio (2021): 76-121.

[15] Eugene F. Fama. “Efficient Capital Markets: II.” The Journal of Finance, vol. 46, no. 5, [American Finance Association, Wiley], 1991, pp. 1575–617, https://doi.org/10.2307/2328565.

[16] Urquhart, Andrew. "The inefficiency of Bitcoin." Economics Letters 148 (2016): 80-82.

[17] Bariviera, Aurelio F. "The inefficiency of Bitcoin revisited: A dynamic approach." Economics Letters 161 (2017): 1-4.

[18] Le Tran, Vu, and Thomas Leirvik. "Efficiency in the markets of crypto-currencies." Finance Research Letters 35 (2020): 101382.

[19] Baur, Dirk G., Kihoon Hong, and Adrian D. Lee. "Bitcoin: Medium of exchange or speculative assets?." Journal of International Financial Markets, Institutions and Money 54 (2018): 177-189.

[20] Bouri, Elie, et al. "On the hedge and safe haven properties of Bitcoin: Is it really more than a diversifier?." Finance Research Letters 20 (2017): 192-198.

[21] Pyo, Sujin, et al. "Predictability of machine learning techniques to forecast the trends of market index prices: Hypothesis testing for the Korean stock markets." PloS one 12.11 (2017): e0188107.

[22] Jaquart, Patrick, David Dann, and Christof Weinhardt. "Short-term bitcoin market prediction via machine learning." The Journal of Finance and Data Science 7 (2021): 45-66.

[23] Ji, Suhwan, Jongmin Kim, and Hyeonseung Im. "A comparative study of bitcoin price prediction using deep learning." Mathematics 7.10 (2019): 898.

[24] Kwon, Do-Hyung, et al. "Time series classification of cryptocurrency price trend based on a recurrent LSTM neural network." Journal of Information Processing Systems 15.3 (2019): 694-706.

[25] Alessandretti, Laura, et al. "Anticipating cryptocurrency prices using machine learning." Complexity 2018 (2018).

[26] de Souza, Matheus José Silva, et al. "Can artificial intelligence enhance the Bitcoin bonanza." The Journal of Finance and Data Science 5.2 (2019): 83-98.

[27] Sebastião, Helder, and Pedro Godinho. "Forecasting and trading cryptocurrencies with machine learning under changing market conditions." Financial Innovation 7.1 (2021): 1-30.

[28] Mallqui, Dennys CA, and Ricardo AS Fernandes. "Predicting the direction, maximum, minimum and closing prices of daily Bitcoin exchange rate using machine learning techniques." Applied Soft Computing 75 (2019): 596-606.

[29] Nevasalmi, Lauri. "Forecasting multinomial stock returns using machine learning methods." The Journal of Finance and Data Science 6 (2020): 86-106.

[30] Martin, James H. “Logistic Regression.” Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Pearson Prentice Hall, Upper Saddle River, NJ, 2009.

[31] Bishop, Christopher M. Pattern Recognition and Machine Learning. New York :Springer, 2006.

[32] Random forest explanation, https://towardsdatascience.com/understanding-random-forest-58381e0602d2. Last accessed Jan 13 2022

[33] XGBoost documentation, https://xgboost.readthedocs.io/en/stable/tutorials/model.html. Last accessed Jan 5 2022