Data Resampling for Cryptocurrency Investment with Ensemble of Machine Learning Algorithms

Tomé Almeida Borges,
tomeborges@tecnico.ulisboa.pt,
Instituto Superior Técnico, Lisbon, Portugal

***Abstract*—This work proposes a system based on machine learning aimed at creating an investment strategy capable of trading on the cryptocurrency exchange markets. With the goal of obtaining highest returns and lowest risk, the original time series is resampled according to a closing value threshold, creating a time series more prone to obtain higher returns. These resampled series are compared to the commonly utilized fixed time interval sampling. From the resampled series, technical indicators are calculated and fed as input to four machine learning algorithms: Logistic Regression, Random Forest, Support Vector Classifier, and Gradient Tree Boosting. Each of these algorithms is responsible for generating a transaction signal. Afterwards, a simple unweighted average of the signals from these four algorithms is employed in trading as well, to improve on their results. This work demonstrates that all learning algorithms outperform the Buy and Hold (B&H) strategy in the overwhelming majority of the 100 markets tested. Nevertheless, the unweighted average obtains the best overall results, namely accuracies up to 59.26% for time resampled series. Additionally, it is concluded that the three alternative resampling methods are capable of generating far greater returns relatively to time resampled data.**

***Index Terms*—Financial Markets, Cryptocurrencies, Technical Analysis, Machine Learning, Ensemble Classification, Financial Data Resampling.**

I. Introduction

As a result of the tremendous growth and remarkably high market capitalizations reached, cryptocurrencies are now emerging as a new financial instrument and their popularity has recently skyrocketed. Despite being relatively recent, with the rise of Bitcoin, the cryptocurrency exchange market has become a global phenomenon, particularly known for its volatility and diversity, attracting the attention of many new and old investors.

Financial time series forecasting is a challenging task as these series are characterized by non-stationarity, heteroscedasticity, discontinuities, outliers and high-frequency multi-polynomial components making the prediction of market movements quite complex [1]. The complex characteristics of financial time series and the immense volumes of data that must be analysed to successfully accomplish the task of forecasting financial time series have driven the adoption of more sophisticated methods, models and simulation techniques. Lately, machine learning or data mining techniques, widely applied in forecasting financial markets, have been offering improved results relatively to simple technical or fundamental analysis strategies. Machine learning methodologies are able of uncovering patterns and predict future trends in order to identify the best entry and exit points in a financial time series with the intention of achieving the highest returns with the lowest risk.

In this work a major objective is using technical indicators as input data to forecast, better than random guessing, the dichotomous event: will a specific currency pair be bullish or otherwise (bearish or sideways) in the next instant of a time series. To achieve this goal, several supervised machine learning classification approaches are suggested. This concept has been widely used in diverse financial markets, such as stocks, bonds or Foreign Exchange markets. However, this work focuses on the cryptocurrency exchange market.

The main contributions of this paper are: development of a framework consisting of several supervised machine learning procedures to trade in a relatively new market, the Cryptocurrencies Market; compare the performance of 5 different types of forecasting trading signals amongst themselves and with a B&H strategy as baseline; compare the difference in results between a financial time series resampled according to a parameter derived from trading activity, in particular a variation percentage, to a commonly used time sampled time series as baseline.

This paper is organized as follows: in Section II the fundamental concepts and related work are discussed; Section III documents the entire proposed system architecture; In Section IV the case studies and results are presented and analysed; Section V provides the conclusions to the work developed.

II. Background & State-of-the-Art

A. *Cryptocurrencies*

In the year 2008, Bitcoin was first described and introduced the concept of decentralized cryptocurrency. Bitcoin and the remaining cryptocurrencies filled in a niche by providing a virtual currency decentralized system without any trusted parties and without pre-assumed identities among the participants that supports user-to-user transactions. Cryptocurrencies can be purchased, sold and exchanged for other currencies at specialized currency exchanges much like the Foreign Exchange market.

The cryptocurrency market is known for the large fluctuations in price, in other words, it is known for its volatility. The remarkable volatility visible in cryptocurrencies, among many reasons, is due to a lack of governance mechanisms and regulatory frameworks. To study the change of rates in multiple cryptocurrency pairs, a time series can be built from sampling the market at a fixed time rate using historical data from a specific cryptocurrency exchange platform. Using a simple application programming interface all historic data used in this thesis was retrieved from one single exchange: Binance.

There are several tools to analyse different markets, but the two major categories are *Fundamental* and *Technical* analysis.

These two approaches to analyse and forecast the market are not exclusive, they may be applied together and attempt to determine the direction prices are likely to move. Since a typical fundamental analysis cannot be executed for the cryptocurrencies market and because technical analysis is more suited for short-term trading [2], only technical analysis was employed in this work. A more elaborate description of technical analysis and each technical indicators can be found in [3].

B. Time Series Forecasting

Analysing financial time series is extremely challenging due to the dynamic, non-linear, non-stationary, noisy and chaotic nature of any financial market [4]. This analysis is carried out as an attempt to find the best entry (buy) and exit (sell) points to gain advantage over the market, increasing gains and minimizing the risk.

Machine learning procedures are capable of analysing large amounts of seemingly noisy and uncorrelated data in order to detect patterns and predict future data. Moreover, this approach provides a reaction time much faster than any human investor could deliver. In this thesis, four total multivariate[1] learning methods were used. Two of them, the *Logistic Regression* and the *Support Vector Machine* methods are linear and the other two, the *Random Forest* and the *Decision Tree Gradient Boosting* methods are non-linear. In the end a combined solution, an ensemble of these 4 algorithms, is calculated.

A brief summary of each classification model utilized in this study follows:

1) Logistic Regression: A binomial logistic regression (LR) [5] is used to model a binary dependent variable. In this type of learning algorithm, a single outcome variable Y_i follows a Bernoulli probability function that takes the outcome of interest, usually represented with the class label $Y = 1$, with probability p while the unwanted outcome, usually represented with the class label $Y = 0$, has probability $1 - p$.

The odds in favour of a particular outcome can be represented as $\frac{p}{(1-p)}$, where p stands for the probability of the wanted outcome. The logit function is capable of transforming input values in the range of 0 to 1, to values over the entire real number range, which can be used to express a linear relationship between feature values and the logarithm of the odds, also known as log-odds, as such [6]:

$$logit(P(Y = 1|x)) = \beta_0 + \sum \beta_i x_i, \qquad (1)$$

where $P(Y = 1|x)$ is the conditional probability of Y belonging to class label 1 given x_i, the feature values, β_0 is the intercept[2] and β_i corresponds to the coefficient associated with each respective feature.

Joining the log-odds and equation 1, the conditional probability can be represented as:

$$P(Y = 1|x) = \frac{1}{1 + exp(-\beta_0 - \sum \beta_j x_{ij})}. \qquad (2)$$

Equation 2 is called the logistic or sigmoid function. From this function it can be seen that $P(Y = 1|x)$ varies between 0 (as x approaches $-\infty$) and 1 (as x approaches $+\infty$). Thus, it is clear that the logistic function is able of transforming any real input into the range of 0 to 1. In fact, the class probabilities are obtained as such.

[1]More than two features are analysed together for any possible association or interactions.

[2]Point that intercepts the function in Y axis.

In this work the optimization problem utilized to obtain the coefficients and intercept, minimizes the following cost function [5]:

$$\min_{\boldsymbol{\beta},\beta_0} \frac{\boldsymbol{\beta}^2}{2} + C \sum_{i=1}^{n} log(exp(-Y_i(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0)) + 1), \qquad (3)$$

where $\frac{\boldsymbol{\beta}^2}{2}$ is the L2 regularization penalty, C is a parameter inverse to the regularization strength and $\sum_{i=1}^{n} log(exp(-Y_i(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0)) + 1)$ corresponds to the negative log-likelihood equation [7]. In the negative log-likelihood equation, $\boldsymbol{\beta}$ represents the coefficients associated with each respective feature value, $\boldsymbol{x}_i$, both are in a vector structure. A method of finding a good bias-variance trade-off for a model is by tuning its complexity via the regularization strength parameter, C.

2) Random Forest: A Random Forest (RF) [8] is a method of ensemble learning where multiple classifiers are generated and their results are aggregated. Random Forest is an enhancement of the method bootstrap aggregating (bagging) of classification trees.

Before all else, a classification or decision tree is a simple model that takes into account the whole dataset and all available features. These trees tend to have high variance and overfit on training data, leading to a poor generalization ability on unseen data [7].

In the bagging method however, multiple decision trees are independently constructed using random samples drawn with replacement (known as a bootstrap sample) of the dataset in order to reduce the variance. Bagging is capable of reducing overfitting while increasing the accuracy of unstable models [9].

Random Forests improves the variance reduction on bagging by reducing the correlation between trees [10]. In order to do so, an additional layer of randomness is added to the bagging procedure. Instead of using all the available features, only a random subset of features is used to grow each tree of the forest. This strategy turns out to be robust against overfitting. Reducing the amount of features will reduce the correlation between any pair of trees in the ensemble, hence, the variance of the model is reduced [8].

Let N be the number of data points in the original dataset, briefly, each tree in the random forest algorithm is created as follows [10]:

1) Draw a random sample of size N with replacement (hence) from the original data (bootstrap sample);
2) Grow a random forest tree to the bootstrapped data. Until the minimum node size is reached, recursively repeat the following steps for each terminal node of the tree:
    a) Select a fixed amount of variables at random from the whole set;
    b) Split the node using the feature that provides the best split according to the objective function;
    c) Split the node into two daughter nodes.

Each time the previous steps are repeated a new tree is added to the ensemble. Repeating this processes multiple times outputs an ensemble of trees with as many trees as this process was repeated. The predicted class probabilities of an input sample are computed as the mean predicted class probabilities of all trees in the forest.

3) Gradient Decision Tree Boosting: Gradient Decision Tree Boosting (GTB) in this work is utilized through the XGBoost framework, a scalable machine learning system for tree boosting [11].

Boosting is the process of combining many *weak classifiers*[3] with limited predictive ability into a single more robust classifier capable of producing better predictions of a target. Boosting is an ensemble method very resistant to overfitting that creates each individual members sequentially [7].

Gradient boosting replaces the potentially difficult function or optimization problem existent in boosting. It represents the learning problem as a gradient descent on some arbitrary differentiable loss function in order to measure the performance of the model on the training set [12].

In this paper the objective function for this model is applied as follows:

$$Obj = \sum_i L(y_i, \hat{y}_i) + \sum_k \Omega(f_k). \tag{4}$$

In equation 4, the first term, $L(y_i, \hat{y}_i)$, can be any convex differentiable loss function that measures the difference between the predicted label $\hat{y}_i$ and its respective true label y_i for a given instance. In this work's proposed system the log-likelihood loss will be used as loss function. Through the usage of this loss function, the calculation of probability estimates is enabled. Combining the principles of decision trees and logistic regression, the conditional probability of Y given x can be obtained [13].

The second term of equation 4, $\Omega(f_k)$, is used to measure the complexity of a tree f_k and is defined as:

$$\Omega(f_k) = \gamma T + \frac{\lambda ||w||^2}{2}, \tag{5}$$

where T is the number of leaves of tree f_k and w is the leaf weights (i.e. predicted values stored at the leaf nodes). Including equation 5 in the objective function (equation 4) forces the optimization of a less complex tree, which assists in reducing overfitting. The second term of this equation, $\frac{\lambda ||w||^2}{2}$, corresponds to the L2 regularization utilized previously in LR. λ is the L2 regularization strength and γT provides a constant penalty for each additional tree leaf.

4) Support Vector Machine: A Support Vector Machine (SVM) [14] is a classifier algorithm whose primary objective is maximizing the margin, of a separating hyperplane (decision boundary) in an n-dimensional space, where 'n' coincides with the number of features used. The margin corresponds to the distance between the separating hyperplane and the training samples that are closest to this hyperplane, the support vectors.

The hyperplane is supposed to separate the different classes, that is, in a binary classification problem, the samples of the first class should stay on one side of the surface and the samples of the second class should stay on the other side. In this work a linear hyperplane for the SVM is utilized.

Decision boundaries with large margins tend to have a lower generalization error of the classifier, whereas models with small margins are more prone to overfitting, hence, it is important to maximize the margins [6]. With the purpose of achieving a better generalization ability, a slack variable, ξ, indicating the proportional amount by which a prediction is misclassified on the wrong size of its margin is introduced. This formulation, called soft-margin SVM [14], enables controlling the width of the margin and consequently can be used to tune the bias-variance trade-off by applying penalties on samples on the incorrect side of the decision boundary according to their distance from the margin.

In order to maximize the margin, the hyperplane has to be oriented as far from the support vectors as possible. Through simple vector geometry this margin is equal to $\frac{1}{||\mathbf{w}||}$ [10], hence maximizing this margin is equivalent to finding the minimum $||\mathbf{w}||$. In turn, minimizing $||\mathbf{w}||$ is equivalent to minimizing $\frac{1}{2}||\mathbf{w}||^2$, the L2 regularization penalty term previously utilized in LR [10]. Therefore, in order to reduce the number of misclassifications, the objective function is written as follows:

$$\min_{\mathbf{w},b,\xi} \frac{||\mathbf{w}||^2}{2} + C \sum_{i=1}^{n} \xi_i \text{ s.t. } Y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i \geq 0 \ \forall_i, \tag{6}$$

where $||\mathbf{w}||^2$ is the squared norm of the normal vector $\mathbf{w}$. C is the regularization parameter, responsible for controlling the trade-off between the slack variable penalty, ξ, and the size of the margin, consequently tuning the bias-variance trade-off. In order to calculate probability estimates, Platt scaling is employed [15].

5) Ensemble Voting: The goal behind ensemble voting (EV) is to combine different classifiers into a meta-classifier with better generalization performance than each individual classifier alone. The weaknesses of one method can be balanced by the strengths of others by achieving a systematic effect [25].

In this work, a heterogeneous ensemble[4] was combined in a linear manner: the prediction's probability estimates from each different individual classifier were combined according to a simple unweighted average, giving equal probability to each individual output. This process, also known as soft majority voting [25], is the reason why all previous learning algorithms ought to yield a probability estimate for each class label, often with the cost of additional computer complexity.

C. State-of-the-Art

De Prado in [26], rather than using, as is customary, fixed time interval bars (minute, hour, day, week, etc.), proposes forming bars as a subordinated process of trading activity. Fixed time interval bars often exhibit oversampling, which leads to penalizing low-activity periods and undersampling penalizing high-activity periods, as well as poor statistical properties, like serial correlation, heteroscedasticity and non-normality of returns. According to the author, alternative bars, relatively to the commonly used time bars, achieve better statistical properties, are more intuitive particularly when the analysis involves significant price fluctuations and the conclusions tend to be more robust. The author mentions, the concept of resampled

[3]Classifier whose error rate performs only slightly better than random guessing.

[4]Ensemble containing different learning techniques.

interval bars is not common yet in literature. As a matter of fact, throughout literature, no actual experimentation was found that could validate De Prado's claims.

Table I summarizes some of the most relevant studies applied to financial market analysis that were considered throughout the development of this paper's approach.

III. Implementation

A. System's Architecture

In this paper, a trading system containing 5 different methodologies for forecasting is used to detect the best entry and exit points in the cryptocurrency market. In order to do so, the direction of price, rather than price levels, is forecast in this work. This method has proven to be effective and profitable by many authors in literature. Simply put, this work attempts to solve a binary classification problem.

In this system, as shown in figure 1 the common systematic procedure to predict time series using machine learning algorithms was put into practice by following these modules in this specific order: Data module; Technical Rules module; Machine Learning module and Investment Simulator module.

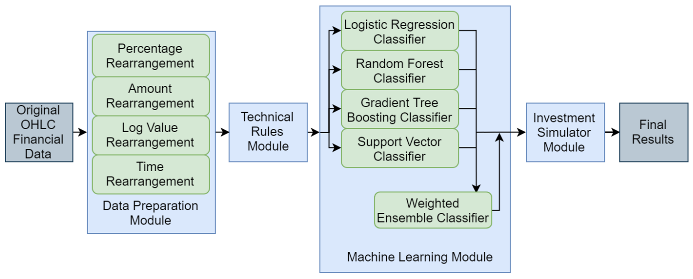


Fig. 1. Simplified overview of the proposed system's modular architecture.

B. Data Preparation Module

The starting point for this work's proposed system is a collection of homogeneously sampled time series that will be processed in this module in order to acquire informative features. In this work the several alternative bar representations attained through data resampling are to be tested and compared to the common time sampling, thus, the original time series dataset containing columns for Open time, Volume and Open, High, Low and Close prices is now to be resampled. The original datasets are as detailed as one can get from Binance: 1 sample per minute. Contrarily to the customary time representation, the resampled data, is intended to place more importance in high-frequency intervals by overrepresenting the constituting individual samples, when compared to low-frequency intervals. The penalizations for incorrect predictions can be exploited in order to favour a better performance on high-frequency intervals rather than in low-frequency intervals. From a financial perspective, it is more important to successfully forecast high-frequency periods as larger returns or losses can be obtained when compared to the more stable low-frequency periods.

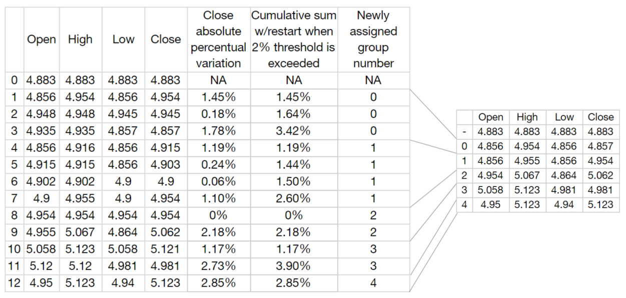
This work's resampling procedure is intended to sample data according to a fixed variation threshold rather than to a fixed time period. In this case, the fixed threshold consists on a fixed percentual variation. The execution of the resampling procedure now follows. Firstly, the percentual first order differencing of the closing values is calculated. Secondly, the sets of consecutive samples whose individual variations aggregated together reach or exceed a pre-defined threshold of absolute percentual variation must be identified. For this purpose, a customized cumulative sum of the percentual first order differencing is responsible for defining the boundaries of each group through assigning different numerical identifiers to each sample. To do so, starting on a specific sample, the total cumulative absolute

TABLE I
Summary of the most relevant works covered in the state-of-the-art.

Ref.	Year	Financial Market	Used Methodologies	Dataset Time Period (Data Frequency)	Evaluation Function	System Performance	B&H Performance
[16]	2017	Credit Default Swaps market	Genetic Algorithm	1/12/2007 - 1/12/2016 (Daily data)	ROI	87.84% (ROI)	NA
[17]	2016	12 most-volumed cryptocurrencies exchange data	Model-less convolutional Neural Network	27/08/2015 - 27/08/2016 (30-minute data)	Portfolio Value Maximization	16.3% (ROI portfolio value)	0.876% (ROI portfolio value)
[18]	2018	Bitcoin/USD exchange data	ANN (only w/long positions implemented)	31/07/2016 - 24/01/2018 (15-minute data)	ROI	6.68% (ROI))	2.28% (ROI)
[19]	2016	Bitcoin/USD exchange data	Long Short Term Memory network	19/08/2013 - 19/07/2016 (Daily data)	Accuracy, Root Mean Square Error	52.7% (Accuracy)	NA
[20]	2015	Bitcoin/USD exchange data	SVM, LR, NN	01/02/2012 - 01/04/2013 (Hourly data)	Accuracy	53.7% for SVM, 54.3% for LR and 55.1% for NN (Accuracy)	NA
[21]	2015	Bitcoin/USD exchange data	BOX-SVM, VW-SVM	09/01/2015 - 02/02/2015 (15-minute data)	Accuracy	10.58% for BOX-SVM, 33.52% for VW-SVM (ROI)	4.86% (ROI)
[22]	2019	Bitcoin/USD exchange data	SVM and Ensemble (RNN and tree classifier)	01/04/2013 - 01/04/2017 (daily data)	Accuracy and Mean Square Error	59.4% and 62.9% (Accuracy)	NA
[23]	2018	12 most liquid cryptocurrencies exchange data	RF (best performing model)	10/8/2017 - 23/6/2018 (15-minute data)	Accuracy	53% (Accuracy)	NA
[24]	2011	Electronic Industry from Taiwan	MLP, CART, LR, Voting and Bagging Methods (of the 3 single classifiers)	2005 Q4 - 2006 Q3 (Quarterly Fundamental data)	Accuracy	Voting Methods 4837%, Bagging Methods 4637% (ROI)	2970% (ROI)

variation of the specific sample as well as the consecutively posterior samples are added up until the variation sum reaches or exceeds the fixed threshold. Whenever a threshold is crossed, a group ends on the actual sample and a new one begins on the next sample, with the next numerical identifier and cumulative sum reset back to zero.

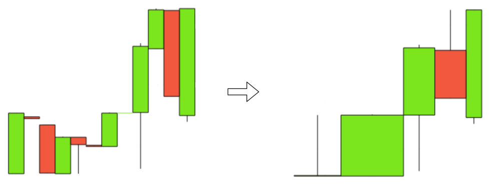
In the end, all samples of the original dataset with the same identifiers are grouped into a single sample of the final dataset. This process is done in a orderly manner that iterates through all samples (in a descendant order, where the oldest entry is at the top and the most recent is at the bottom), therefore, only consecutive samples can be grouped together. This process is illustrated in figure 2. As can be observed, whenever the value of the seventh column (Cumulative sum w/restart when threshold is reached) is equal or larger than a fixed threshold, in this case 2%, the group being numbered with the identifier $'i'$ is closed, the value for the cumulative sum is restarted and a new group with identifier $'i+1'$ is initiated in the next sample.



	Open	High	Low	Close	Close absolute percentual variation	Cumulative sum w/restart when 2% threshold is exceeded	Newly assigned group number
0	4.883	4.883	4.883	4.883	NA	NA	NA
1	4.856	4.954	4.856	4.954	1.45%	1.45%	0
2	4.948	4.948	4.945	4.945	0.18%	1.64%	0
3	4.935	4.935	4.857	4.857	1.78%	3.42%	0
4	4.856	4.916	4.856	4.915	1.19%	1.19%	1
5	4.915	4.915	4.856	4.903	0.24%	1.44%	1
6	4.902	4.902	4.9	4.9	0.06%	1.50%	1
7	4.9	4.955	4.9	4.954	1.10%	2.60%	1
8	4.954	4.954	4.954	4.954	0%	0%	2
9	4.955	5.067	4.864	5.062	2.18%	2.18%	2
10	5.058	5.123	5.058	5.121	1.17%	1.17%	3
11	5.12	5.12	4.981	4.981	2.73%	3.90%	3
12	4.95	5.123	4.94	5.123	2.85%	2.85%	4

	Open	High	Low	Close
-	4.883	4.883	4.883	4.883
0	4.856	4.954	4.856	4.857
1	4.856	4.955	4.856	4.954
2	4.954	5.067	4.864	5.062
3	5.058	5.123	4.981	4.981
4	4.95	5.123	4.94	5.123

(a) Table on the left contains the original dataset that is resampled into the dataset on the right.



(b) Left candlestick group is a representation of the original dataset and right candlestick group is arepresentation of the resampled dataset.

Fig. 2. Regrouping process illustration for a percentage of 2%.

It's worth adding that financial markets generate a discrete time series, making it impossible to achieve a perfectly even-sized resampled dataset with data from any cryptocurrency market. Nevertheless, utilizing a finer-grained periodic (smallest possible time period between data points) original data, is more likely to build a more regularly and consistently sized resampled dataset.

To resample the data according to different parameters, simply the first order difference series and a fixed threshold must be defined according to the chosen resampling parameter. In this work, besides time resampling, three alternative resampling processes were also tested. Nevertheless, relatively to utilizing a fixed amount or logarithmic amount as threshold for resampling, the percentage threshold yielded the best results. The process of resampling is analogous for any of three methods, thus, only the latter resampling procedure was described in full detail.

Lastly, because both percentage and time resampled datasets ought to have a similar amount of samples in order to generate valid comparisons, the original dataset is also resampled, according to time by simply grouping consecutive data points.

C. Technical Indicators Module

This module is responsible for generating and outputting the respective technical indicators for each data sample of the resampled time series outputted from the Data Module.

Table II lists the used technical indicators as well as the values for each parameter (if existent). In this proposed system, the values used for each parameter are the ones traditionally used or suggested. The histogram method from the MACD indicator

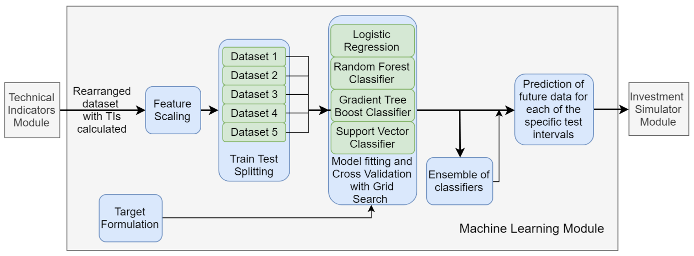
TABLE II
LIST OF TECHNICAL INDICATORS OUTPUTTED TO THE MACHINE LEARNING MODULE.

Technical Indicator
Exponential Moving Average
Relative Strength Index
Moving Average Convergence Divergence histogram
On Balance Volume
Rate Of Change
Commodity Channel Index
Average True Range
Stochastic Oscillator %K line
Stochastic Oscillator Fast %D line
Stochastic Oscillator Slow %D line
Histogram between %K line & Fast %D line
Histogram between %K line & Slow %D line

was replicated for the stochastic oscillators to indicate trend reversals (when a sample of this indicator crosses zero) as well as whether the trend is upwards or downwards.

D. Machine Learning Module

This module is the most complex and the main core of the system, as so it was divided into 3 main components, each with a different responsibility. In this system, the resampled data is firstly scaled by standardization, then the target prediction vector is defined and lastly the actual machine learning training and forecasting procedures are executed. Figure 3 contains a brief illustration of this module's main steps.



Technical Indicators Module
Rearranged dataset with TIs calculated
Feature Scaling
Dataset 1
Dataset 2
Dataset 3
Dataset 4
Dataset 5
Train Test Splitting
Logistic Regression
Random Forest Classifier
Gradient Tree Boost Classifier
Support Vector Classifier
Model fitting and Cross Validation with Grid Search
Ensemble of classifiers
Prediction of future data for each of the specific test intervals
Investment Simulator Module
Target Formulation
Machine Learning Module

Fig. 3. Machine Learning module overview.

1) Feature Scaling: Feature scaling is a data preprocessing step, applied on the technical indicator series outputted from the previous module through a technique called *standardization*. Standardization is the process of centring features by subtracting the features' mean and scaling by dividing the result by the features' standard deviation.

Feature scaling is not only important if we are comparing measurements that have different units, but it is also a general requirement for many machine learning algorithms. When the input dataset contains features with completely different units of measurement, which is the case in this system, using feature scaling avoids the problem in which some features come to dominate solely because they tend to have larger values than others [6].

2) Target Formulation: The objective of the second component of this module is defining the target discrete values to be predicted, the class label, for each data point. This process is a requirement when using methods of supervised learning.

Throughout literature, the bid-ask spread is an issue that is often overlooked. In less liquid markets, specifically in the most "exotic" cryptocurrency pairs (whose financial worth is typically in the USD cents), the percentual difference between the best buyer and the best seller may become large enough so that the common assumption of utilizing the closing value as both bid and ask values becomes quite far-fetched leading to a disparity with reality that may no longer become acceptable. However, to the author's knowledge, enquiring specific exchanges or sources for the spread of the current instant is the only method of acquiring this data free of charge. Regardless, in order to assemble a decent amount of data, years of data gathering would be required. In this work, due to the unavailability of bid-ask spread value, only transaction fees are considered. Nevertheless, it should be pointed that overlooking the importance of bid-ask spreads is not ideal.

The buy and sell trading fees were taken into account when determining the ideal long position start and ending points. The trading fees applied in this system are Binance's general fee of 0.1%. In order to decide whether the signal in the next instant has one of the two possible outcomes, a deterministic binary classification is proposed: the cryptocurrency market either has a positive or a negative variation in the next instant. In the rare cases where the variation is precisely null, the outcome of the previous instant is duplicated into the current one.

A vector called *vector y* (represented in equation 7) contains the target classification for each sample in the dataset. Each entry follows a binomial probability distribution, i.e., $y \in \{0, 1\}$, where 1 symbolizes a positive or bullish signal variation and 0 symbolizes a negative or bearish signal variation. For a given currency pair, being $close_t$ the closing price for a given time period t, and $close_{t-1}$ the closing price for the previous time period, $t-1$, the target for the given time period, y_t, is defined in a probabilistic way to employ the four learning algorithms as follows:

$$y_t = \begin{cases} 0, & \text{if } Close_{t-1} \times (1 + Fee) < Close_t \times (1 - Fee) \\ 1, & \text{if } Close_{t-1} \times (1 + Fee) > Close_t \times (1 - Fee) \\ y_{t-1}, & \text{if } Close_{t-1} \times (1 + Fee) = Close_t \times (1 - Fee) \end{cases}, \quad (7)$$

Given that the utilized data was retrieved from Binance, to be truly faithful to a real case scenario, only long positions can be adopted by this system, short positions are not accepted.

3) Classifiers Fitting and Predicting: This is the last component of the Machine Learning module. The objective of this component is to create one model for each of the four classification algorithms and, in the end, creating a classifier ensemble with the output of each algorithm by averaging the sample of each classification algorithm.

Before the actual training procedure begins, both the features (matrix X) and the target vector (vector y), must be split into train dataset and test dataset. In this work, 4 splits divide the time series in 5 equal sized intervals, this is, each interval contains the same number of samples. It's worth noting that because the splits divide the number of samples, only the time rearranged series will be split in 5 time intervals with same time duration. While the test interval size is consistent for all iterations, the train set successively increases in size with every iteration, it may be regarded as a superset of the train sets from the previous iterations. Ultimately, when using this method, no future data leaking occurs and forecasting since a much earlier time point is enabled.

In order to tune the hyper-parameters of each learning algorithm, a specific type of cross-validation is used: *time series cross-validation*. It is a variation of the commonly used *k-fold* cross-validation destined specifically for time series data samples. In the k^{th} split, the first k folds are returned as train set and the $(k+1)^{th}$ fold as validation set. This employed cross-validation is identical to the previously explained process of splitting train and test data, the only differences are that rather than only 4 splits, 10 splits are done and instead of splitting train and test data, train and validation data is being split. In this work, a simple grid search was carried out in this proposed system in order to find the best hyper-parameters according to the negative log-loss metric.

4) Forecasting and Ensemble voting: At this point the second part of supervised learning, the forecasting, commences. For each specific training dataset, all learning algorithms apply the fit model on the respective testing dataset. All 4 algorithms generate a probability estimate of the respective class labels for each sample of the test dataset. The classification odds for each test sample of the 4 models, are now combined through *soft majority voting* into an ensemble. In soft majority voting, the probabilities for the prediction of each class label of the test dataset previously calculated are unweightedly averaged, designated as EV.

At last, this module outputs five different trading signals generated containing the forecasting data. Four trading signals are originated from the individual learning algorithms while the last is originated from the unweighted average of these four. All the trading signals have the same time frame, and will be simulated in a real world environment in the next module, the investment simulator.

E. Investment Simulator Module

The investment simulator model is responsible for translating the five trading signals obtained as input from predicted data in the Machine Learning module, into market orders with the purpose of simulating this system's performance in a real market environment.

The trading signal received as input, consists on a series containing the probability of each sample being classified with the label 1. This series containing probabilities is easily converted into a dichotomous class label series according to the largest likelihood. After being converted into a discrete classification series, each entry of this series is interpreted as follows: A class label of 1 represents a *buy* or *hold* signal; A class label of 0 represents a *sell* or *out* signal. Binance trading fees are taken into consideration in the trading simulator with the purpose of reproducing a real market. When a class label 1 signals *buy*, all available capital is applied in opening a long position. Similarly, a class label 0 orders the system to exit the market and retrieve all available capital. This module starts with a fixed capital for investment (by default it's one unit of quote currency) and as a method to control risk on a trade-by-trade basis, stop-loss orders were implemented.

To simulate the market orders, backtest trading is employed in this work. Because this backtest process utilizes historical data, the following two assumptions must be imposed for this proposed system:

1) Market liquidity: The markets have enough liquidity to conclude each trade placed by the system instantaneously and at the current price of their placement.
2) Capital impact: The capital invested by the algorithm has no influence on the market as it is relatively insignificant.

In the end, a series of metrics, revealed and explained in section IV-A, are calculated, plotted and stored into memory.

IV. Results

Instead of using the nearly 400 currently available pairs in Binance, only the 100 pairs with the most traded volume in USD were selected. This selection filters out many pairs that have been recently listed and don't have a large enough dataset. Moreover, it becomes more likely that the selected markets are in conformity with the two backtest trading assumptions. It should be noted that six out of the 100 markets evidently were not liquid enough to comply with the backtest requirements, thus were swapped for the six next markets with most volume.

In order to use the maximum amount of available data, no common starting date was chosen for all currency pairs. Each currency pair's data begins at the moment Binance started recording the specific currency pair. On the other hand, the ending date, is fixed at 30th October 2018 at 00h00. Each trading period in the original data, as previously mentioned, has the duration of 1 minute. The starting date varies from 14th July 2017 (Binance's official launch date), up to 12th October 2018.

The original data for the 100 selected currency pairs, prior to its usage in forecasting, must be resampled. As was explained in section III, to carry out the multiple resampling procedures, a fixed percentual variation threshold must be picked to define the approximate final size of each candle in the final dataset. In the subsequent results, a fixed percentage of 10% was chosen for the resampling procedures.

A. Evaluation metrics

The main goal of this proposed system is maximizing the negative logarithmic loss and returns while minimizing the associated risk of this work's trading strategy. With this goal in mind, the following metrics of market performance were additionally calculated for each currency pair with the intention of providing a better analysis of the obtained results:

1) Return on Investment: The return on investment (ROI) measures the amount of return gained or lost in an investment relative to the initially invested amount. The simple standard formula of this metric is represented as follows:

$$ROI = \frac{FinalCapital - InitialCapital}{InitialCapital} \times 100\%, \tag{8}$$

where $FinalCapital$ corresponds to the capital obtained from the investment bought with $InitialCapital$.

2) Maximum Drawdown (MDD): The maximum drawdown measures the maximum decline, from a peak to a through before a new peak is attained. It is used to assess the relative downside risk of a given strategy. This metric is calculated as follows:

$$MDD = \max_{t \in (StartDate, EndDate)} [\max_{t \in (StartDate, T)} (ROI_t) - ROI_T], \tag{9}$$

where ROI corresponds to the return on investment at the subscript's point in time and $\max_{t \in (StartDate,T)}(ROI_t)$ corresponds to the highest peak from the starting point until the instant T. In this work, as is customary, MDD it is quoted as a percentage of the peak value.

3) Sharpe Ratio: The Sharpe Ratio is a method for calculating the risk-adjusted return. This ratio describes the excess return received for holding a given investment with a specific risk. The sharpe ratio is calculated as follows:

$$SharpeRatio = \frac{ROI - R_f}{\sigma}, \tag{10}$$

where ROI corresponds to the return on investment, R_f is the current risk-free rate, 3.5%, and σ is the standard deviation of the investment's excess return.

4) Sortino Ratio: The Sortino Ratio is a modification of the Sharpe ratio metric. In contrast to Sharpe ratio, the Sortino ratio includes only negative variations. The formula for calculating this metric is identical to the formula represented in equation 10, however, the denominator corresponds only to the standard deviation values observed during periods of negative performance.

5) Classification Parameters: Besides the four presented metrics used to evaluate the performance of an investor, the following parameters are also used in the classification of this proposed system:

- Percentage of periods in market: Percentage of time periods where a long position was in effect out of all the available time periods of the testing set;
- Percentage of profitable positions: This parameter is calculated by dividing the number of trades that generated a profit (with fees included), by the total number of trades; This probability is complementary to the percentage of non-profitable positions;
- Average Profit per position: Average percentual profit or loss per position;
- Largest Percentual Gain: Most profitable position;
- Largest Percentual Loss: Greatest loss.

B. Case Studies

In this section, the case studies and the main results obtained through the application of the described strategies are presented.

1) General Overview: First of all, a general idea of the overall performance obtained with each different methodology independently of the resampling method utilized is provided. To achieve this, whilst utilizing the B&H strategy as benchmark, the five trading signals generated by the four individual learning algorithms and the ensemble voting method are individually averaged and subsequently compared against each other. The results are represented in table III.

TABLE III
COMPARISON BETWEEN THE BUY & HOLD STRATEGY AND EACH OF THE FIVE METHODOLOGIES EMPLOYED.

Parameter	B&H	LR	RF	GTB	SVC	EV
Average Obtained Results (for all markets and resampling methods)						
Final ROI	-10.5%	519%	295%	335%	538%	**615%**
Accuracy	40.20%	53.50%	53.62%	53.51%	53.38%	**56.28%**
Negative Log-Loss	-20.6	-0.7031	-0.6992	-0.6918	-0.6975	**-0.6829**
Periods in Market	100%	56.0%	52.9%	55.0%	50.4%	**39.9%**
Profitable Positions	19.5%	**60.2%**	56.2%	58.3%	58.8%	57.6%
Profit per Position	-10.6%	0.57%	0.31%	0.33%	0.61%	**0.69%**
Largest Gain	**35.6%**	17.6%	17.7%	17.3%	18.4%	15.0%
Largest Loss	-46.0%	-14.4%	-15.0%	-15.1%	-14.5%	**-13.2%**
Max Drawdown	79.9%	57.6%	60.6%	62.0%	54.7%	**49.3%**
Annual Sharpe Ratio	-0.164	0.769	0.413	0.312	0.848	**0.945**
Annual Sortino Ratio	0.169	2.374	1.665	1.407	2.568	**2.821**

Through analysis of table III, it is clear that the B&H strategy yields the worst results. Out of all individual learning algorithms, the SVC generates the highest ROI and is the least risky, as so, it can be considered as the best individual learner. LR follows in terms of the metric ROI but is quite risky as a forecasting method. Both RF and GTB performed averagely and both yielded modest ROIs. Nonetheless, EV is by far the most robust alternative. As can be seen, the trading signal obtained from EV on average outperforms the remaining ones according to the majority of metrics.

The obtained accuracies are on par or, for the case of EV, exceed the accuracy of most papers regarding cryptocurrency exchange market forecasting throughout the state-of-the art (table I). The risk measures and returns on investment are also superior on average.

In conclusion, there is no clear performance hierarchy for each of the 4 individual learning algorithms. Nonetheless, the trading signals generated by the EV methodology obtain the top performances out of all tested methods. In any case, all five methodologies clearly outperform the plain B&H strategy.

2) Time Resampling: Time resampling is the most widely used method of resampling, therefore, it is the first sampling method analysed to be considered as the comparison baseline.

Firstly, figure 4, contains for each time instant, the respective average ROI of all 100 analysed markets for each trading signal. In this figure it may be observed that overall, most methodologies are quite conservative when the market suddenly rises, hence a B&H strategy earns more in these periods. On the other hand, this conservative behaviour is also responsible for minimizing losses in sudden price drops, contrarily to the B&H strategy.

Secondly, table IV contains the general statistics obtained for the time resampling method. In this table, a better method cannot be clearly defined. Relatively to EV, SVC achieved a

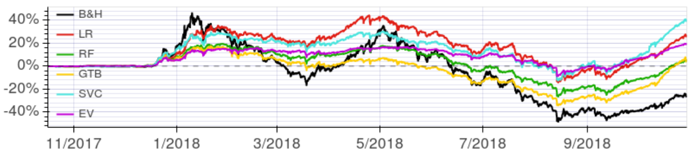


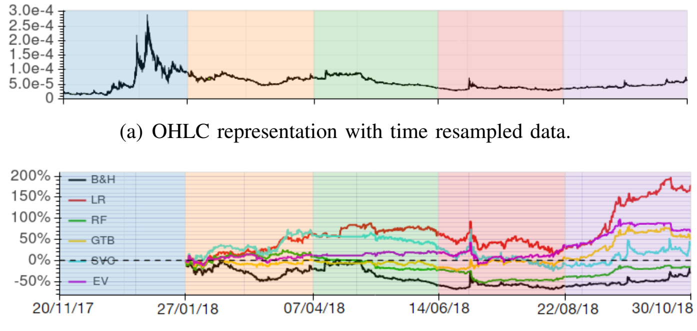
Fig. 4. Average accumulated ROI [%] for each instant from the starting until the last test point with time resampled data.

TABLE IV
AVERAGE OBTAINED RESULTS FOR THE B&H AND EACH OF THE FIVE METHODOLOGIES EMPLOYED FOR TIME RESAMPLING.

Parameter	B&H	LR	RF	GTB	SVC	EV
Average Obtained Results (all markets are considered)						
Final ROI	-27.9%	25.6%	3.92%	3.30%	**39.5%**	18.7%
Accuracy	37.40%	54.77%	55.58%	54.84%	54.70%	**59.26%**
Negative Log-Loss	-21.6	-0.6931	-0.6896	-0.6890	-0.6757	**-0.6746**
Periods in Market	100%	51.4%	44.4%	48.9%	44.9%	**27.7%**
Profitable Positions	15.0%	**57.6%**	50.8%	53.4%	56.0%	55.0%
Profit per Position	-27.9%	0.05%	0.01%	-0.02%	**0.09%**	0.06%
Largest Gain	20.8%	18.1%	**20.8%**	19.6%	18.4%	15.3%
Largest Loss	-48.7%	-15.2%	-14.5%	-14.9%	-16.2%	**-12.3%**
Max Drawdown	77.0%	55.6%	56.8%	59.6%	52.5%	**42.5%**
Annual Sharpe Ratio	-0.352	0.075	-0.135	-0.327	0.228	**0.288**
Annual Sortino Ratio	-0.137	0.608	0.330	-0.074	0.985	**1.102**

worse predictive power and is slightly more risky, yet obtained over twice the profits. Nonetheless, EV stands out due to the better accuracies and NLL as well as due to the remarkably small percentage of periods in market and top values in the risk metrics, thus suggesting that this is the least risky alternative.

Thirdly, the ROI evolution for a well performing market is represented in figure 5 and its statistics are shown in table V. In these figures, the top subfigure contains a candlestick representation of the utilized resampled data for each cryptocurrency pair. As can be seen, the background is divided into five different coloured periods. Each coloured period represents a different sub-dataset of the train-test splitting procedure. This splitting procedure divides the data in five intervals with the same amount of samples. Additionally, it is also visible that the ROI line only begins at the start of the second dataset, which is in accordance with the fact that the first sub-dataset is only used for the training procedure.



(a) OHLC representation with time resampled data.

(b) Chronological evolution of the ROI [%] for the 5 calculated trading signals and B&H strategy.

Fig. 5. ROI variations for currency pair POEETH (Po.et/Ethereum) with time resampling applied.

From figure 5 and table V it is clear that the trading signal from the LR outperformed the remaining signals. EV obtained

TABLE V
RESULTS OBTAINED WITH TIME RESAMPLING FOR THE POEETH MARKET.

Parameter	B&H	LR	RF	GTB	SVC	EV
Final ROI	-28.0%	**180%**	-15.6%	51.8%	39.7%	71.3%
Accuracy	39.02%	55.51%	58.75%	58.36%	50.99%	**60.06%**
Negative Log-Loss	-21.06	-0.6851	-0.6732	-0.6840	**-0.6660**	-0.6730
Periods in Market	100%	60.9%	35.2%	41.4%	73.3%	**31.0%**
Profitable Positions	0%	**63.5%**	52.2%	54.4%	55.3%	55.7%
Profit per Position	-28.0%	**0.205%**	-0.02%	0.060%	0.048%	0.102%
Largest Gain	NA	3.55%	21.3%	16.7%	**25.9%**	16.7%
Largest Loss	-28.0%	-20.6%	-20.3%	**-7.7%**	-21.0%	-20.5%
Max Drawdown	70.5%	27.14%	63.51%	**19.16%**	57.27%	26.93%
Annual Sharpe Ratio	0.222	**2.190**	-0.057	1.174	0.900	1.471
Annual Sortino Ratio	0.368	**3.405**	-0.152	1.869	1.383	2.130

a clear second place with mostly above average results. In any case, the B&H strategy was outperformed by the remaining trading signals.

In conclusion, taking all figures and tables of this procedure into account, the developed strategy is not flawlessly suited for this type of resampling. Neither B&H nor any of the five remaining models achieved absolutely brilliant results. In any case, it may be concluded the EV methodology is the superior one. It's worth praising EV's high predictive performances as can be perceived by the high accuracies and NLLs, implying that this system has predictive potential for this resampling method. This suggests that if this same machine learning formulae is combined with a more fine-tuned strategy, there is potential to achieve more impressive results, risk and return-wise.

3) Percentage Resampling: The results from this type of resampling, were only narrowly outperformed by amount resampling in terms of predictive power and final returns. Nevertheless, on average these results are noticeably less risky. Due to its low risk and only slightly lower profits and predictive power, this resampling method originates a more desirable outcome, hence, it is worth carrying out a more in depth analysis of the obtained results.

Firstly, a temporal graph showing the instantaneous average ROI of the 100 analysed markets for each of the methodologies is presented in figure 6. In this type of resampling, although

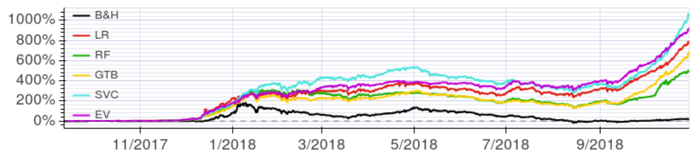


Fig. 6. Average accumulated ROI [%] for each instant from the starting until the last test point with percentage resampled data.

not as significant as time resampling, the drop in the markets from May until September 2018 is also visible. Once again, the signal from B&H was clearly surpassed by all remaining trading signals in terms of profits.

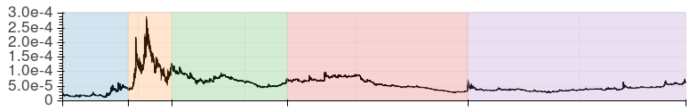
Secondly, a table containing the general statistics obtained for the percentage resampling method follows in table VI. In this table, even though, on average EV's returns are slightly inferior to SVC, the former distinctly obtains the highest predictive power and the lowest risk out of all other trading signals, thus it may be concluded that the EV method clearly generates

TABLE VI
AVERAGE OBTAINED RESULTS FOR THE B&H AND EACH OF THE FIVE METHODOLOGIES EMPLOYED FOR PERCENTAGE RESAMPLING.

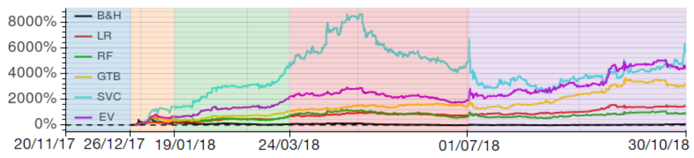
Parameter	B&H	LR	RF	GTB	SVC	EV
Average Obtained Results (all markets are considered)						
Final ROI	16.4%	792%	494%	692%	**1063%**	923%
Accuracy	41.38%	53.03%	53.00%	53.06%	52.64%	**55.23%**
Negative Log-Loss	-20.2	-0.7112	-0.7028	-0.6928	-0.7187	**-0.6866**
Periods in Market	100%	56.7%	55.8%	58.0%	52.3%	**43.3%**
Profitable Positions	25.0%	**61.9%**	55.2%	58.0%	59.7%	59.0%
Profit per Position	**16.4%**	0.82%	0.51%	0.67%	1.17%	0.97%
Largest Gain	**56.3%**	20.9%	15.0%	16.3%	19.1%	13.6%
Largest Loss	-39.9%	-13.2%	-14.2%	-14.2%	-13.9%	**-12.7%**
Max Drawdown	80.5%	56.6%	59.9%	60.5%	55.4%	**50.1%**
Annual Sharpe Ratio	0.066	1.297	0.868	0.824	1.360	**1.496**
Annual Sortino Ratio	0.605	3.907	2.911	2.689	3.765	**4.325**

the trading signals with the top results. Once again, the B&H strategy clearly obtained the worst average performance.

Thirdly, the ROI evolution for the same specific market, relatively to the previous section, is represented in figure 7 and its statistics are shown in table VII.



(a) OHLC representation with percentage resampled data.



(b) Chronological evolution of the ROI [%] for the 5 calculated trading signals and B&H strategy.

Fig. 7. ROI variations for currency pair POEETH (Po.et/Ethereum) with percentage resampling applied.

TABLE VII
RESULTS OBTAINED WITH PERCENTAGE RESAMPLING FOR THE POEETH MARKET.

Parameter	B&H	LR	RF	GTB	SVC	EV
Final ROI	64.2%	1436%	850%	3145%	**5747%**	4416%
Accuracy	42.59%	56.60%	52.33%	**56.71%**	53.92%	56.54%
Negative Log-Loss	-19.83	-0.6860	-0.7058	-0.6894	**-0.6824**	-0.6859
Periods in Market	100%	46.5%	64.4%	**43.8%**	59.5%	44.9%
Profitable Positions	100%	60.9%	54.1%	61.0%	**62.3%**	61.3%
Profit per Position	**64.2%**	1.6%	1.0%	3.7%	2.1%	4.9%
Largest Gain	**64.2%**	2.6%	2.5%	4.8%	6.4%	7.3%
Largest Loss	NA	**-2.2%**	-8.9%	-3.7%	-8.2%	-4.3%
Max Drawdown	89.98%	**16.60%**	54.60%	20.03%	70.94%	23.92%
Annual Sharpe Ratio	1.041	3.969	2.661	3.868	3.404	**3.970**
Annual Sortino Ratio	2.510	9.326	6.125	10.192	10.829	**11.927**

From figure 7 and table VII it can be concluded that the signal from EV only obtained top results for the risk ratios. Results regarding ROI and especially predictive power are slightly inferior to the top values. In this example the top values are dispersed throughout the different trading signals, which goes to show that when utilizing varied learning algorithms, one's weakness can be overcome by another learning algorithm and, ultimately only the best traits of these algorithms are hopefully noticeable in the EV, which in this case, did not happen. The opposite, where the EV outperforms most metrics of the remaining trading signals, is also frequently verified.

To conclude, resampling the original dataset according to a percentage is preferable than resampling according to a logarithmic amount or a simple amount. Relatively to a logarithmic amount, percentage obtained better returns and accuracies and less risk. Relatively, to an amount, despite obtaining slightly inferior returns, resampling according to a percentage is notably less risky. Lastly, relatively to the time procedure, even though the returns were plainly superior, the predictive power is clearly inferior. As a matter of fact, time rearrangement on average, holds the lowest ROI, but also obtains the best accuracies and negative log-losses out of all 4 resampling methods experimented, a possible explanation is because most investors utilize similar investment strategies and prediction algorithms based on similarly time sampled data. Furthermore, market manipulation is entirely permitted in cryptocurrency exchange markets. Because the algorithms and strategies utilized by investors are related and due to their collective influence, the current ongoing time series is constantly being heavily impacted. Hence, when an algorithm similar to the ones who collectively crafted and whose influence is deeply embedded in a given time series intends to backtest trade with this same time series, it seems plausible that its overall predictive power is enhanced.

Nevertheless, to assign this phenomenon as the unambiguous source of this issue would require further investigation.

V. Conclusion

In this work a system combining several machine learning algorithms with the goal of maximizing predictive performance was described, analysed and compared to a B&H strategy. All aspects of this system, namely the target formulation, were designed with the objectives of maximizing returns and reducing risks always in mind. To validate the robustness of this system's performance, this system was tested on 100 different cryptocurrency exchange markets through backtest trading.

Based upon this work, it may be concluded that all four distinct learning algorithms consistently bared positive results, better than random choice or the B&H strategy. Nonetheless, the trading signal generated by the ensemble voting method produced by far the best results.

The outcome of utilizing data resampled according to alternative metrics, namely data resampled according to a fixed percentage, proved to consistently generate significantly higher returns than the commonly used time sampled data. Data resampled according to a logarithmic amount and to an amount, did not obtain results as good as a percentage did, hence, these were not analysed in detail in this paper. Nevertheless, the returns from both also exceeded the returns obtained with time resampling. The overall results were in accordance with what was anticipated: rearranging financial time series according to an alternative metric does in fact construct a series prone to generating larger returns, which is in accordance with one of the main intentions of this work.

It is worth noting that the return on investment obtained in this system with the alternative procedures is in some markets excessive. This fact is attributed to the absence of bid-ask spread data, as was mentioned in section III. Putting it differently, this work shows that the three alternative methods offer more profitable ROIs relatively to time resampling, however, it is unlikely that this system would do as well, return-wise, in an actual real scenario in face of bid-ask spreads.

References

[1] Esmaeil Hadavandi, Hassan Shavandi, and Arash Ghanbari. Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting. *Knowledge-Based Systems*, 23(8):800–808, 2010.

[2] Rodolfo C Cavalcante, Rodrigo C Brasileiro, Victor LF Souza, Jarley P Nobrega, and Adriano LI Oliveira. Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, 55:194–211, 2016.

[3] John J Murphy. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.

[4] Yaser S Abu-Mostafa and Amir F Atiya. Introduction to financial forecasting. *Applied Intelligence*, 6(3):205–213, 1996.

[5] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[6] Sebastian Raschka. *Python machine learning*. Packt Publishing Ltd, 2015.

[7] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[8] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: data mining, inference, and prediction, springer series in statistics, 2009.

[11] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[12] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[13] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C Weng. A note on Platt's probabilistic outputs for support vector machines. *Machine learning*, 68(3):267–276, 2007.

[16] João Manuel Ribeiro Cardoso and Rui Neves. Investing in Credit Default Swaps using Technical Analysis Optimized by Genetic Algorithms. 2017.

[17] Zhengyao Jiang and Jinjun Liang. Cryptocurrency portfolio management with deep reinforcement learning. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 905–913. IEEE, 2017.

[18] Masafumi Nakano, Akihiko Takahashi, and Soichiro Takahashi. Bitcoin technical trading with artificial neural network. *Physica A: Statistical Mechanics and its Applications*, 510:587–609, 2018.

[19] Sean McNally, Jason Roche, and Simon Caton. Predicting the price of bitcoin using machine learning. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 339–343. IEEE, 2018.

[20] Alex Greaves and Benjamin Au. Using the bitcoin transaction graph to predict the price of bitcoin. *No Data*, 2015.

[21] Kamil Żbikowski. Using volume weighted support vector machines with walk forward testing and feature selection for the purpose of creating stock trading strategy. *Expert Systems with Applications*, 42(4):1797–1805, 2015.

[22] Dennys CA Mallqui and Ricardo AS Fernandes. Predicting the direction, maximum, minimum and closing prices of daily bitcoin exchange rate using machine learning techniques. *Applied Soft Computing*, 75:596–606, 2019.

[23] Erdinc Akyildirim, Ahmet Goncu, and Ahmet Sensoy. Prediction of cryptocurrency returns using machine learning. 2018.

[24] Chih-Fong Tsai, Yuah-Chiao Lin, David C Yen, and Yan-Min Chen. Predicting stock returns by classifier ensembles. *Applied Soft Computing*, 11(2):2452–2459, 2011.

[25] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[26] Marcos Lopez De Prado. *Advances in financial machine learning*. John Wiley & Sons, 2018.