Using the Order Book and Machine Learning for Cryptocurrency Trading

João Guilherme Esteves de Andrade
j.guilherme.andrade@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

May 2019

Abstract

This thesis presents a new computational approach for profit optimization on cryptocurrency trading, using trade and order book data from a major digital asset trading platform of four digital currencies (Bitcoin, Ethereum, Litecoin and Bitcoin Cash) in the United States Dollar (USD) and Euro (EUR) markets. An end-to-end solution was designed, starting with the database generation, information extraction, trading algorithm, simulation and ending with a dynamic report of the results. All the parts of the system are designed in a microservice architecture to ensure that they are scalable, strongly encapsulated and tightly scoped components. Three use cases where considered for the trading system where the first two use order book volume variation to assess suitable trading periods whilst the last one uses an ARIMA model to forecast price fluctuation and consequently serve as a verification mechanism to the first two test cases during a simulation. Furthermore, a new type of trailing stop called Percentage Stop Order is introduced, allowing profit improvement. Monthly test periods were used alongside different types of trading positions to all test scenarios to find the most performing strategy according to the market's situation. The results obtained managed to surpass the gains from the Buy & Hold strategy up to 22,72% and reducing losses up to 33,24%.

Keywords: Cryptocurrency, Order Book, ARIMA, Microservices.

1. Introduction

The cryptocurrency trading realm is progressively attracting attention from communities linked to several domains of finance and computational intelligence. One of the main reasons associated with this phenomenon is the high volatility associated with the market of digital currencies. This means that finding the optimal timings to enter and leave the market is the main objective of researchers since it leads to higher profits on a short period of time. However, this task is really challenging since the cryptocurrency market is non-linear, non-stationary and heavily influenced by speculation.

In recent years, cryptocurrencies are becoming increasingly popular, getting a significant amount of media attention worldwide. Despite only being on the spotlight for a short time, the concept of cryptocurrencies and blockchain technology was introduced in 2009 with the introduction of Bitcoin [1]. The main idea was to create a decentralized shared public ledger (blockchain) that validates and saves immutably all financial transactions by using a computer network as a timestamp server, maintaining the correct order of negotiations.

There were 1474 types of digital coins at end-January 2018 and their total market capitalization was 830 billion USD [2], a consequence of an increasing public acceptance of cryptocurrencies. Nevertheless, volatility is a crucial aspect of digital markets and, as a measure of price fluctuations, has an important impact on trade planning and investment decisions. This component can be partially handled since exchange offices store "buy" and "sell" orders on an order book, providing insights into the liquidity and trading purposes [3]. Thus, the first step is to define how will this information be used and which temporal model must be chosen to adaptively learn volatility using the gathered data.

In this thesis a new approach for profit optimization on cryptocurrency trading is presented, using trade and order book data from a major digital asset trading platform (Coinbase Pro, formerly known as GDAX) of the four main digital currencies, Bitcoin (BTC), Ethereum (ETH), Litecoin (LTC) and Bitcoin Cash (BCH), in USD and EUR, to create a trading system that uses order book volume variation and machine learning to forecast price fluctuation and consequently present trading advice.

The main contributions of the proposed system are: (1) A database containing order books and trading information of the four major digital currencies built entirely from scratch (more than 70 Gigabytes of data obtained since March 2018); (2) An algorithm that pre-processes and aggregates information from order books, correlating it with confirmed trades on a monthly basis, to define the optimal start positions for long and short trading; (3) Machine Learning forecasting applied to the previous algorithm, using Autoregressive Integrated Moving Average (ARIMA); (4) An innovative type of trailing stop to optimize profits.

2. Related Work

Using Machine Learning and statistics to create algorithms for financial gains has proven to be effective in a large application scope. Despite being a relatively new area of study, we can already find several scientific articles about this subject where those that contributed the most for this assignment will be considered in this section. Before handling those papers, we will clarify the fundamental components behind the idea of Machine Learning.

2.1. Machine Learning

Data mining is described as the extraction of previously unidentified and implicit information from data, which can result in advantageous input. Machine learning specifies the technical base for data mining [4]. The collection of observations that contain one or more variables, known as attributes, is defined as a dataset. We can divide machine into two categories, supervised learning and unsupervised learning. Supervised learning includes the modelling of datasets with labelled instances that can be represented as x and y, x being a set of predictor attributes and y the dependent target attribute. A machine learning problem follows one of two methodologies: regression, when the target attribute is continuous; classification, when the target variable is discrete [5]. There are some hybrid methodologies which allow for a classification and regression at the same time, e.g. neural networks. Unsupervised learning includes modeling datasets with no established outcome or result. Since this project has a task with a known objective, predicting cryptocurrencies price fluctuation, it is a supervised machine learning task.

2.2. Supervised Learning

Supervised learning is the most common methodology in machine learning. As previously explained, supervised learning is achieved when we input variables x and an output variable Y alongside an algorithm to learn the mapping method from the input to the output. This type of learning can be algebraically exemplified by

$$Y = f(x) \tag{1}$$

The main goal of supervised learning is to obtain a mapping function so that when we have new input data, we can predict the output variables for that observation with a defined target. Supervised learning generates the mapping function based on example input-output pairs. Knowing the current answers, the algorithm makes forecasts on the training set iteratively and is rectified accordingly. When the algorithm achieves the desired level of performance, the learning process is finished. We can divide supervised learning challenges into two categories, classification and regression problems.

Classification algorithms are used when the aimed output is a discrete label, which means that we should choose them to handle problems where the objective falls under a finite set of possible results. In many use cases, such as defining whether it will rain or a gender, there are only two possible results. This is called binary classification. Situations where there are more than two classes are handled by multi-label classification, where each sample is mapped to a set of target labels, for example a book can be about art, a person and an era simultaneously.

Regression models generally present reliable results in forecasting outputs that are continuous. In other words, regression models are used when the solution to our problem can be expressed by a quantity that can be determined based on the inputs of the model instead of being restricted to a set of possible classes. Regression issues with time-ordered inputs are named time-series forecasting problems and will be thoroughly explained with the next sub-section. Some of the most popular regression algorithms are Linear Regressions and the ARIMA model [6].

2.3. Time Series

Time series are collections of data points chronologically organized. Their purpose is to understand patterns evolving over time and apply these patterns to forecast behavior in several fields (stock prices, meteorology and heart arrhythmias are some examples). Since this field can be applied in many areas science, studies on time series have become increasingly popular on the last few decades.

Most of the scientific studies about time series tackle time series modelling, which is the process of understanding and defining (mathematically) the inherent structure of the time series [7]. The usefulness of time series modelling ranges from forecasting the future [8] to describing the feature responsible for the time series but in this thesis, we will focus on the first part since our goal is to simulate the generation of data to an unforeseen future by studying the model of a time series information. When making predictions using time series, we can classify the time periods into three different categories, short-term, medium-term and long-term. While long-term refers to periods significantly distant to the last observed data, short-term denotes spans that are near the observations. An important fact to mention is the ambiguity associated with how the boundaries between each class are chosen since each application defines them in a distinguishing way using their data and objectives.

Long-term prediction is a complex challenge to work out because of the lack of information, error accumulation and the growing uncertainties [8]. These complications only intensify their negative effect as the forecasting region grows. Consequently, to elaborate a reliable time series prediction technique it is mandatory to forecast the future precisely while avoiding error accumulation as the forecasting region expands.

To correctly study time series, it is necessary to model the underlying stochastic process, a time sequence representing the evolution of a predefined system expressed by a variable whose change is subject to a random variation, that creates time series. We can use a joint probability distribution between observations to obtain a complete description of a time series, but the required calculations might be too complex, so a valid alternative would be to compute the mean and autocovariance. The mean is simply the one measure of the central tendency of observations while autocovariance calculates the statistical dependence between two observations in a time series, computing the covariance between itself at different periods of time.

A stationary time series has the property that the mean, variance and autocovariance structure do not change over time, a very desirable property on a time series. Consequently, proper estimation of the enumerated properties should be equal for all points in time, allowing forecasting for all time points in the future. Regrettably, there are two commonly found time series components that obliterate the stationary property, season and trend.

2.4. ARIMA

Autoregressive Integrated Moving Average is one of the most used time series prediction models, having a more complicated approach than the simplest model, a linear regression. ARIMA arises from the Autoregressive Moving Average (ARMA) model while the latter itself is a mixture of an Autoregressive (AR) model with a Moving Average (MA) model. The notion of both MA and AR is fundamental not only to the ARIMA model but for several time series models.

The AR model considers that the actual value of a time series is a linear combination of its prior values while the MA model defines the current value of a time series as a linear combination of several past errors. Usually the number of previous values is symbolized by p and named as the order of the autoregressive. Each one of the models uses a constant c and represents white noise as ϵ.

Combining the two explained models we obtain the ARMA(p,q) model, given by

$$y_t = c + \epsilon_t + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \tag{2}$$

where p, ϕ and y_{t-i} represent respectively the order, current parameters and previous value of the autoregressive model while q, θ and ϵ_{t-j} respectively symbolize the order, current parameters and past errors of the moving average model.

For the ARMA model to function, it assumes that the associated time series is stationary. Remembering that the objective of this project is to handle data from cryptocurrency markets which are affected by season and trend so who can we transform a non-stationary time series into a stationary time series? The mechanism to resolve this issue is called differencing and can be applied to each one of the two components. Differencing calculates the variance between consecutive observations, if a lower order differencing doesn't achieve the intended detrending or de-seasonalizing we are obliged to compute a greater order differencing.

Applying a dth order of differencing to an ARMA model results in an ARIMA model, often defined as ARIMA(p,d,q) where p represents the order of the AR, d is the order of differencing and q is the order of the MA. This model has numeral software implementations, fact that strongly contributed to the widespread utilization of ARIMA. Although the predominance of this model, its prediction power reduces as the forecasting horizon augments due to the forecast converging to the mean of the observations as the predicting region grows [9]. With this information we should be aware that this model is only adequate to short-term prediction which is the forecasting objective of this thesis.

2.5. Methodology Examples

The idea of extracting features from order book data over time and use them to create probabilistic models that simultaneously apply volatility and feature series to predict Bitcoin price fluctuations was introduced in a recent study from Tian Guo and Nino Antulov-Fantulin [10]. In each order book snapshot, the price spread, weighted spread, ask/bid volume, depth and slope and their difference were gathered. Spread is the variance between the lowest price that a seller is willing to accept (ask) and the highest price that a buyer is willing to pay (bid). Depth is defined by the number of orders on the ask or bid side while volume is the number of Bitcoins on the ask or bid side. Weighted spread is the difference between cumulative price over 10% of ask depth and the cumulative price over 10% of bid depth. Slope is estimated as the volume until α price offset from the actual traded price, where α is estimated by the bid price at the order that has at least 10% of orders with the greater price.

The authors of the referred scientific paper created a generative temporal mixture model of the volatility and trade order book data, which is able to outperform the current state-of-the-art machine learning and time series statistical models by using a gate weighting function that detects regimes when the features of buy and sell orders significantly affects the future high volatility periods [10]. The only problem with this approach was to only consider one cryptocurrency, Bitcoin, which limits the application scope. Since Bitcoin is the cryptocurrency with most volume traded daily, most of the published work in the digital currencies field applies their findings to this coin. Although being limited to one form of digital currency, these studies provide a solid knowledge base than can be applied to a larger spectrum of cryptocurrencies. Price fluctuations were studied from several perspectives such as using order book data and found that a lack of buyers generated panic amongst sellers [11], which culminated in the price crash of Bitcoin on the 10th of April 2013, or using cryptocurrency web communities data to obtain sentiment and forecast value fluctuations [12]. Price prediction studies apply some techniques used in traditional financial market like autoregression, as used by Garcia et. al. who identified word of mouth and new Bitcoin adopters as feedback loops that drive to price bubbles [13] or simply employ historical time series price information to develop an accurate trading strategy with price prediction [14].

3. Architecture

This system was designed according to a multi-layered micro-services architecture built on three layers: the data layer, the application layer and the presentation layer.

Separating the project in different layers provides the ability to modify each one of them without crashing the others, reducing maintenance and enabling flexibility for application expansion. The flow of the system described can be explained through the following steps: (1) Market order books and market trades are extracted from a digital asset exchange, order books every five seconds and trades as they happen; (2) Two components, one for trades, another for order books, are in charge of Pre-Processing the information provided by the cryptocurrency exchange. They gather all the obtained information, filter it into the standard representations of the system and populate the database accordingly; (3) The Data Aggregation element will read all the data stored on the database, aggregate it into a monthly time frame so several features (for example, the maximum monthly volume on each side of the order book) can be extracted; (4) The Trading Algorithm component will receive a period of time as an argument and all the data regarding it, apply one of the three variants of the algorithm created for this thesis and generate a report with the execution results. The first alternative analyzes the best periods based on the lowest and highest volume values within the time frame to start short or long positions. On the other hand, the second relies on inversions (over a defined threshold) of total percentage in the bid or ask side of the order book to define start positions. Lastly, the third variant uses with the ARIMA model to forecast prices and determine when to invest accordingly. All methods use the percentage stop order developed for this project to improve profitability; (5) Finally, the results obtained from the Trading Algorithm running through the simulation window will be presented to the user for analysis.

3.1. Database

Since the world of cryptocurrency trading is decades younger than stock trading, we don't have the same amount of free information available on the internet. Therefore, one of the first challenges of this thesis was to establish how would the trading data be obtained. After searching for free databases containing the desired information, one of two problems always emerged. The database didn't had order book data or information was aggregated on a daily basis with maximum and minimum values. These complications alongside the scarcity of free databases lead to a different approach regarding the acquisition of trading data: scrapping information from a digital asset exchange.

There are several ways to scrape data from web trading platforms but exchanges try to simplify that process by providing free and well documented Application Programming Interface (API)s which enable their clients to easily gather knowledge. Additionally, those APIs are available in various programming languages, although their features may vary slightly between them. Two information categories were needed for this project, the buys and sells and the state of the order book within a determined time period. Evidently, we didn't gather information about all the available digital currencies because it was not in the scope of this work so we decided to pick the top four cryptocurrencies according to daily volume: BTC, ETH, LTC and BCH. We also thought that it would be interesting to compare the differences between USD and EUR digital markets so each cryptocurrency trading information was extracted on both currencies. Because of the quantity of data we defined separate documents (the Structured Query Language (SQL) table equivalent in MongoDB) for each digital currency's order books. For confirmed trades, a single document was used and the digital currency associated with it was one of the fields. To make a distinction between the USD and EUR digital markets, each document has two collections, one of each mentioned currency. Likewise, in the trades document there is a 'Buy' collection and a 'Sell' collection. The database architecture used is depicted in Figure 1.

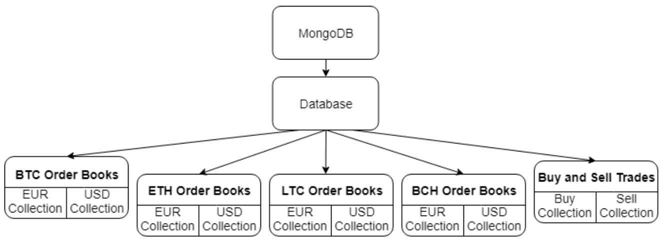


Figure 1: Database Architecture

3.2. Trading Data

After a thorough study of digital asset platforms with APIs, Coinbase Pro, formerly known as GDAX, was chosen for this master's thesis. Coinbase Pro provided a API that was easy to use and accomplished the information goals defined previously. Nonetheless, we wanted to save every confirmed traded and the best option was to use the API's web socket client but it was not available with Python so we developed a Node.js scrapper to extract trades and a Python one to gather order books. In this subsection, both programs and the information contained in every trade and order book will be clarified.

Firstly, what kind of information was retrieved from a single trade and an order book? For trades, we used the API's web socket client, starting by establishing a connection to the MongoDB database. Afterwards, the web socket is actively listening for all messages sent by Coinbase Pro and filters them for confirmed trades. These trades are documented on a message containing the 'match' type. Every time a 'match' message is caught, we will save the trade in the database according to its side (buy or sell) with the following fields: currency, trade identifier, amount, date, rate and total. On the other hand, order books were requested every five seconds since high frequency trading is not in the scope of this thesis. After configuring the connection to the MongoDB database, every five seconds the order books of the four digital currencies on both USD and EUR were saved as a pair [*date,dictionary*], where *date* refers to the time of extraction and *dictionary* contains all the ask and bid amounts and prices.

3.3. Data Flow and Data Processing

This subsection will describe the data flow and processing, after extracting trade and order book information, until sending the treated data as input to the trading algorithm. We can break down this procedure in three main steps:

1. Database Communication, that clarifies how does our solution handles the connection to a remote database and retrieves necessary information;

2. Saving Monthly Data, where we define how to gather, by parts, monthly raw data of buy or sell trades or order books;

3. Monthly Data Manipulation, composed by receiving all files generated from the previous step and using aggregation and, in some cases, resampling to create *DataFrames* that will be used as input to the trading algorithm.

Since we were using a remote computer to store the database containing all the information obtained from the digital asset exchange, how would we access that data? There are several solutions available, since manually creating backups of the database to an external hard drive (restoring it afterwards in the machine where the trading algorithm will be executed) to establishing a permanent connection between computers that would periodically send new data from the remote database to the local one. Our decision relied on manually defining when to gather monthly data, via an Secure Shell (SSH) tunnel, by parts, to avoid memory shortage and the overhead associated with maintaining a permanent SSH connection. But what is an SSH tunnel?

Firstly, the SSH protocol is a mechanism for secure remote login from one computer to another, providing various alternative options for strong authentication while protecting the integrity and security of communications with strong encryption. SSH tunneling is a method of transporting any kind of networking data over an encrypted SSH connection. There are several Python libraries that enable us to efficiently implement an SSH tunnel which was one of the main reasons to use this approach

Prior to establishing a communication with the remote computer, we need to obtain data about digital currency trades and order books. To assist the progress of this thesis, we defined that we should obtain monthly information because it would only be gathered once and we could easily divide it into parts (by week for example) to avoid memory shortage problems. The only obvious downside of this approach is that we can only extract months prior to the current one. Since trade and order book information is provided with different specifications, despite both being in the lightweight data-interchange format called JavaScript Object Notation (JSON), we addressed each case separately.

After gathering monthly data about trades and order books, we must aggregate it since we are dealing with millions of records and the useful information can be extracted (or even created) with this method. But what does aggregation mean? In this particular case we are referencing data aggregation, that can be defined as a process in which information is obtained and expressed in a summary form. The main intent for this summary in this work is statistical analysis.

An overview of all the components responsible for data processing is illustrated in Figure 2.

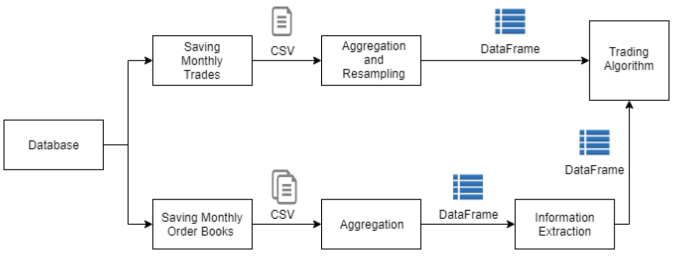


Figure 2: Data Processing

3.4. Trading Algorithm

After aggregating the monthly order book information, we had the challenge of defining what kind of data we would want to gather from the four levels of total asks and bids percentage and total volume. Firstly we defined that our trading algorithm should take advantage of the digital currency market volatility. With this aspect in mind, we established the first two use cases for our algorithm, using a percentage of the preeminent monthly volume of the order book to find periods with high demand for sales or buys and identify total volume inversions. The second use case can be defined by a shift of the total volume percentage (over a defined threshold) from the asks side to the bids side or vice versa.

We had to specify the threshold that retrieved at least fifteen periods, number delineated with my supervisor's guidance. Our objective was to find order books whose volume, from the ask or bid side, is higher or equal to the maximum monthly volume from the same order book part multiplied by the chosen threshold. The difference between the two mentioned use cases is that the first one is represented by the maximum monthly volume amount in the four levels whilst the second one only considers the top monthly volume in the front level.

To provide a better understanding of both test cases, Figures 3 and 4 illustrate simple examples of how their mechanism is used to find suitable investment periods. The first plot depicts a typical situation where the conditions for selection are met. Let us start by considering the yellow line as the total order book volume in the bids side, the blue line as the total bids orders maximum volume in the testing month and the red line as the value from where we should start a position. Point A represents the first period that would be retrieved since it follows the condition presented by the **??** equation. From this point until point C, every stage will be seen as a possible investment situation. Two additional points are worth being mentioned, the first being the fact that all volumes below the red line will not be selected and that the periods must have a distance between them of five minutes, to avoid redundancy. This will also be done for the volume on the first level of the order book, on both asks and bids side.

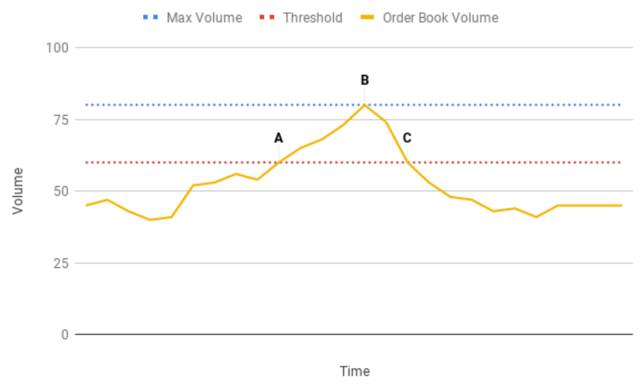


Figure 3: First Use Case Example

On the other hand, Figure 4 exemplifies the second use case. Now we are considering the total order book percentage on both sides, the red line represents the percentage of asks while the green line represents the percentage of bids. Furthermore let us consider a threshold of 20 %. Point A shows the moment where the inversion takes place since total bids percentage will be more significant than the percentage of asks orders. When the next period (B) is reached, we achieved an inversion that is equal to our threshold, so this period will be returned from the use case as a possible moment for starting a long position (since the inversion was from the asks to the bids side). The following periods in the figure will not be considered since no inversion occurred.

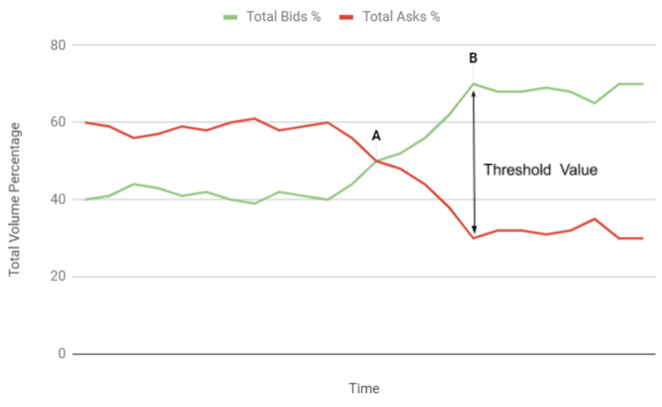


Figure 4: Second Use Case Example

Finally we needed to save this information to be able to use it in our simulation while avoiding repeating this information extraction. Nonetheless, while checking the resulting periods, we verified that the majority were consecutive which did not give us useful information so we established that we would only save periods with at least five minutes of interval between them. Afterwards, we saved the *DataFrames* containing the order books with the respective timestamps, that respected our temporal condition, in Comma-Separated Values (CSV) files.

3.5. Percentage Stop Order

After studying the standard types of limit and stop orders used, we thought that a trailing stop order was the fittest for our intentions of optimizing profits. Nevertheless, due to the volatility of the digital currency market, we needed more control over potential stop losses and be conservative to improve mean profit. The addition of dynamic limits presented itself like a solid idea, but how could we implement this concept? To solve this problem, we introduced a new variety of stop order called Percentage Stop Order (PSO).

Defining the adoption of percentages to the detriment of flat currency amounts allowed the PSO to have more flexibility and to sustain the considerable rate fluctuations of cryptocurrencies. Our stop order has conservative default percentages, using 0.5 percent as the stop gain percentage and a 2 percent stop loss. Also, the chosen threshold that provided the desired dynamic behaviour was 0.25 percent. To select this values, we tested several combinations and these produced the best results. Now we have the starting values, how does our stop order work during a trading simulation?

There are four courses of action that may be taken by the PSO algorithm. To clarify each one of them we will consider a simulation where we are placing long positions, which means that we bought currency and are expecting a growth in value. Figure 5 depicts the example where we start with 1 BTC bought at an original rate of 1000 USD. Point A illustrates a situation where the stop gain and stop loss orders were not reached nor the maximum sell rate went over the threshold to dynamically change both stop orders. When assessing the max rate sell at the B event, the PSO threshold was passed so both the upper and lower limits are updated. It is important to single out how to check if we need to update the stop order limits, given by the equation

$$rate\_sell\_max \geq (orig\_rate * (1 + thresh\_rate)) \quad (3)$$

Afterwards, the rate increases until reaching our current stop order limit in point C, at this point we sell our BTC having a profit of 0.75 percent. Finally, we decide to buy 1 BTC again in point D. The original limits are established and as we progress in time, the value of the digital currency held declines sharply so when the stop loss order is reached, at point E, we sell our position to minimize our loss.

3.6. Simulation

The simulation component is crucial in this work since it allowed us to test all the mechanisms developed and verify the results produced by our three use cases. As input, it must receive two *DataFrames*, one with the real monthly trading information containing data about buys, sells (we can use minimum, mean or maximum values) in five minute intervals, and another one regarding the

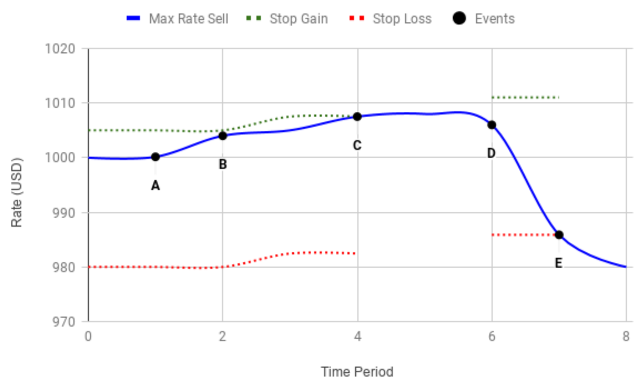


Figure 5: Percentage Stop Order Example

periods where we should invest according to the analysis performed by the selected use case. Furthermore, we define if we want to take into account fees that should be paid to the digital asset exchange and what type of positions we want to put on the market (long, short or both). A *DataFrame* containing the execution results will be output where each row referring to a time period will have a date, volume, number of trades, amount, rate, mock investor's money, mock investor's amount of digital currency, an indicator if an action was taken in that time frame and the profit percentage.

4. Evaluation and Comparison Analysis

To validate the efficiency of the considered use cases, we used a method called Backtesting. This strategy enables us, relying on previous trading data, to simulate our algorithm, assess the results and verify the mean profitability without using real money. Therefore, this method allows for better understanding of the risk associated with our approach. Additionally, if the simulation outcome is positive then our algorithm should be solid and generate profits when applied to the current market. Lastly, by adopting this procedure we can analyze several test scenarios and discover concept problems, areas for improvement and potential gains without wasting an extended period of time.

The following three use cases were performed:

1. Using a threshold over the maximum monthly order book volume to identify periods with a potential high volatility since a greater volume is correlated to a high demand for sales or buys of digital currency;

2. Discovering periods where a shift of the total volume percentage (over a defined threshold) from the asks side to the bids side or vice versa took place, which allows us to invest after an aggressive change of trend;

3. Using ARIMA to forecast prices and use the predictions as an indicator of market trend, subsequently applying this obtained information in the first two test scenarios and verify if it improves or reduces profitability.

4.1. General Setup

Despite the different elements associated with each one of the three use cases, several processes and configuration parameters are identical throughout the evaluation phase. Table 1 displays the configuration values that remained the same during the whole testing operation.

Table 1: General Setup Parameters

Configuration Element	Value
Test Period	Monthly
Selected Periods	July and November 2018
Digital Currencies	4
Resampling Data By	5 Minutes
Starting Money	1000 USD
Rate	Mean
Moving Average	200 Days
Considered MA Periods	3 (15 minutes)

The chosen test period is a month although we have more than a year of trading information because this time window already needs a considerable amount of hours (at least eight) to aggregate all the data collected from the digital asset exchange. Nevertheless, to improve the quality of our results, we considered two distinct months for each currency simulation on each one of the use cases. As previously mentioned, the cryptocurrencies tested were BTC, ETH, LTC and BCH.To compare our results with a standard trading method, we used the Buy & Hold strategy in the same test periods, this approach buys digital currency in the first time period and only sells at the last one. Despite using a monthly test period, we gathered more than half a million order books for each digital currency so we used a 5 minute resampling. This processed allowed us to significantly reduce the computation time needed to execute a simulation.

Regarding simulation parameters, we established that the algorithm has 1000 USD to invest in every single run that does not start with a short position (in those cases, we start with 1 unit of the specific cryptocurrency) and must always use the full amount of virtual money available in every trade. Unfortunately, due to time constraints, only the USD market was tested. Additionally, our trading method constantly uses the mean rate of the trades that took place in the five minutes of the considered period, alongside a 200 day MA that enables us to follow the market trend. Before defining a new short or long trade, our algorithm verifies the directions of the MAs of the three previous periods and allows the position if it respects the market flow. Lastly, all the investing periods determined by our use cases will be generated before we execute the simulation and we did not take into account possible transaction fees.

After running a simulation, the Return On Investment (ROI) will be calculated. In the event of finishing the test period with money invested, our algorithm uses the last saved rate of the considered cryptocurrency to sell the full amount owned. This step allows the ROI computation.

4.2. Test Case 1 - Maximum Monthly Order Book Volume Period

This test case initially considers the maximum monthly volume values at the first level of the order book and the total volume in both bid and ask sides. Afterwards, it applies a threshold to those maximum values and finds periods having a volume greater or equal to the calculated value.

Figure 6 presents the performance achieved by our trading strategy regarding the monthly ROI for each one of the four cryptocurrencies in July 2018. We considered three options for types of trades used in our algorithm: short, long and both positions. Afterwards, we use the November 2018 data to simulate the behaviour of our trading method with the note that BCH was not considered due to errors in the extracted order book information. The results of this simulation are shown in Figure 7.

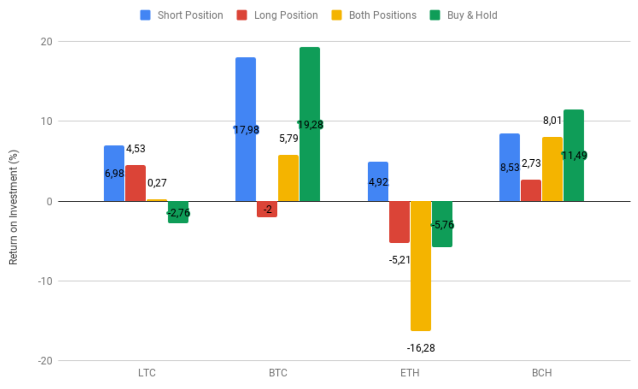


Figure 6: First Test Case Results on July 2018

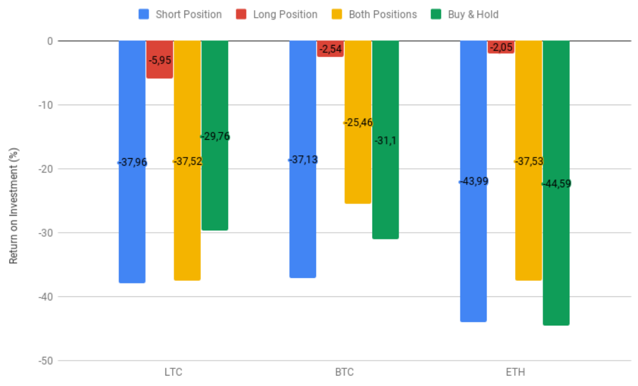


Figure 7: First Test Case Results on November 2018

The use of a percentage of the maximum order book volume to find suitable investing periods had the objective to provide more profit than the Buy & Hold trading strategy. Observing the plot illustrated in Figure 6, we can verify that our use case obtains better results, comparing to the mentioned strategy, with LTC and ETH ending the simulation with profit on July 2018. Nevertheless, the Buy & Hold ended the month with more significant gains on BTC and BCH. Another interesting note is the fact that all the test digital currencies provided solid results when running this scenario using only short positions.

Regarding the November 2018 results depicted in Figure 7, we may think that the scale is wrong since there are only negative percentages but this month brought important price decreases on several cryptocurrencies. The Buy & Hold strategy lost a major portion of the investment on all digital currencies. Only using long positions with this test case allowed us to significantly reduce our losses (e.g., lost a minimum of 35% less than any other type of strategy for ETH).

4.3. Test Case 2 - Monthly Total Order Book Percentage Inversion Of Bids And Asks

This test case considers predefined thresholds to find inversions between the total order book percentage of bids and asks. Afterwards, it saves all time periods where the inversion matched the threshold.

Figure 8 illustrates the performance achieved by using in our simulation periods where an total order book percentage inversion took place in July 2018. We considered three types of trade configurations in our algorithm: short, long and both positions. As in the previous use case, we use also tested this strategy on November 2018 data (except for BCH). The results of this simulation are shown in Figure 9. Lastly, our results were compared with the ones generated by the Buy & Hold method.

In this use case, we used total order book percentage inversions to extract periods to start positions with the goal of generating profit whilst comparing the results with the Buy & Hold method and the first test case. Analyzing the graph presented in Figure 8, we can conclude that while running with the two existing types of trades and the July 2018 data, this test case obtains profit on BTC and BCH unlike the Buy & Hold strategy. As the previous use case, the latter method has more profit with BCH and BTC. The type of trading strategy that provided the best outcome was using both position types since it allowed us to have profit with all four currencies. Comparing with the first use case, only in ETH while using both types of trades the results were considerably superior, a 33,24% increase in profit.

Figure 9 depicts the results obtained after running our trading simulation with the November 2018 data. As the first case, we could not get profit in all currencies with the three types of trade mechanisms. Nonetheless, using long positions obtained a lower loss compared with the previous test case with the exception of ETH. Furthermore, for LTC, all results were better than those in the first use case. For BTC using short positions brought a more important loss but the other two trade configurations reduced the deficit obtained previously.

4.4. Test Case 3 - Forecasting with ARIMA

To test this use case, we started by taking each series containing the monthly trades of all the currencies from both July and November 2018 and applying a back fill to all missing values. Next, we defined our ARIMA model. After choosing the parameters for our model, we decided to check the previsions made using a month of data, using 75% for training purposes and 25% for testing. We verified that the previsions were really distant from the

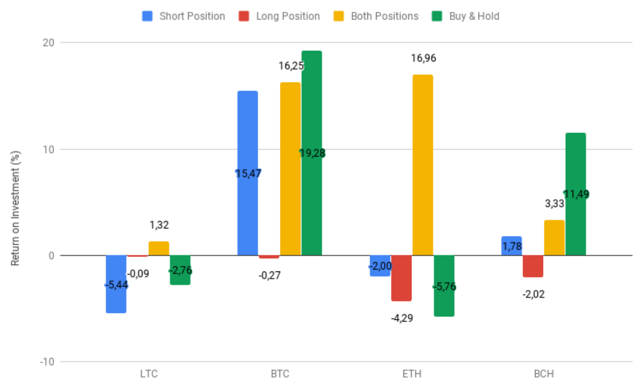


Figure 8: Second Test Case Results on July 2018

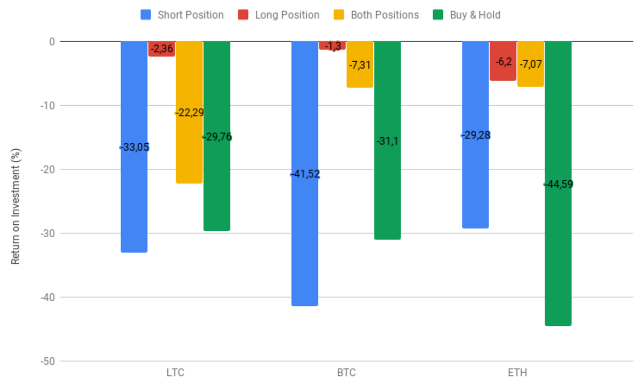


Figure 9: Second Test Case Results on November 2018

real rates but we could gather and use information from a different perspective, using the direction of the forecast instead of the value itself. To check which direction the market is going, we compare our prediction to the last predicted value, if it is a positive number, we consider the rate should increase, other wise we will consider it will decrease. We will save this information on CSV file and use this directions as an extra validation method before starting a new trading position on a simulation.

Figure 10 illustrates the performance achieved by adding ARIMA previsions to the first use case in July 2018. We still considered three types of trade configurations in our algorithm: short, long and both positions. Afterwards, we used the November 2018 data to simulate the behaviour of our trading method with the note that BCH was not considered due to the problems already mentioned. All the other test scenarios for both use cases generated the same results obtained previously.

This test case only generated different results (for BCH and BTC) when applied to the first use case on July 2018, the outcome is illustrated in Figure 10. Using only short trading positions in BTC generated the highest profit within all use cases (24,45%). This result is superior to the 19,28% of profit gained by the Buy & Hold method. Contrarily, using both types of trades in BCH results in a 5,74% profit reduction compared to the first use case without forecasting. Unfortunately, this test did not generate a different outcome for the majority of the trade type / cryptocurrency pairs since we used three quarters of the month to train our ARIMA model so it could only be tested on the last week of the month. Obviously, if our trading algorithm does not take any action in the last week, our direction mechanism is unable to act.

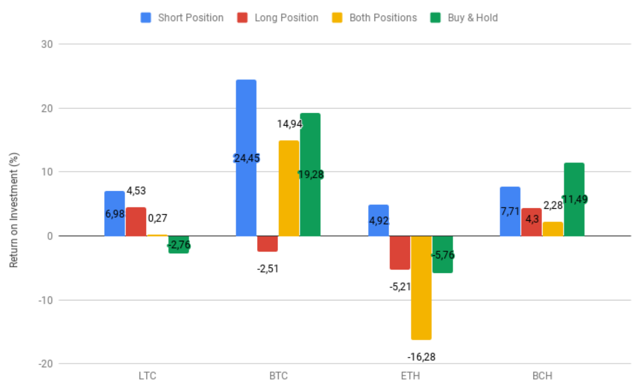


Figure 10: First Test Case Results With ARIMA on July 2018

4.5. Overall Analysis

Despite the lack of results of the last use case, the first two provided interesting outcomes that gave us a solid indication that there is value in extracting order book information and we can use it to improve profits. Before going through the positive aspects, let us address the major problems that occurred during the evaluation phase. First of all, the time needed to extract data from the database and aggregate it because it made us consider monthly test periods which is not suitable for a trading algorithm. Additionally, we should have preemptively verified the information integrity to avoid situations like the inability to execute a simulation with BCH on the November 2018 data.

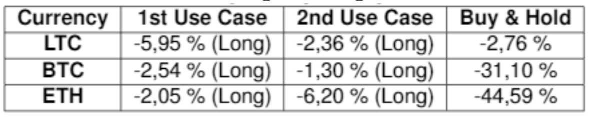
Observing the best performing trading methods using the July 2018 data in Table 2, we can infer that only establishing short positions with the first use case generates profit for all four digital currencies. Furthermore, using ARIMA for forecasting allows to improve our most profitable situation (short trades on BTC) by 6,47 %. Nonetheless, the third test scenario slightly reduced the gains with BCH while compared with the first use case. Measuring the efficiency of our strategies against the Buy & Hold method, we can verify that in the first and second use cases we achieved better results in two cryptocurrencies whilst in the last scenario we generated more profit for LTC, BTC and BCH.

Table 2: Best Trading Strategies July 2018

Currency	1st Use Case	2nd Use Case	3rd Use Case	Buy & Hold
LTC	6,98 % (Short)	1,32 % (Both)	6,98 % (Short)	-2,76 %
BTC	17,98 % (Short)	16,35 % (Both)	24,45 % (Short)	19,28 %
ETH	4,92 % (Short)	16,96 % (Both)	16,96 % (Both)	-5,76 %
BCH	8,53 % (Short)	3,33 % (Both)	7,71 % (Short)	11,49 %

Table 3 presents the outcome of the most performing trading strategies for November's 2018 data. Two notes must be addressed initially, the first one is why all the profit percentages are negative whilst the second is related to not having both third use case and BCH results. The latter remark is justified by a problem regarding the order book retrieval from the digital asset's store API since some of them were empty. We can explain the negative profits by visualizing the market trend on November 2018 because the majority of cryptocurrencies suffered significant price drops. In spite of these problems, our developed use cases managed to drastically reduce losses if compared to the Buy & Hold strategy results while using only long positions. This fact may feel strange since we are dealing with a month where the general market tendency was to decrease the rate of digital currencies but it is a proof that the PSO acts efficiently even in challenging periods.

Table 3: Best Trading Strategies November 2018



Currency	1st Use Case	2nd Use Case	Buy & Hold
LTC	-5,95 % (Long)	-2,36 % (Long)	-2,76 %
BTC	-2,54 % (Long)	-1,30 % (Long)	-31,10 %
ETH	-2,05 % (Long)	-6,20 % (Long)	-44,59 %

The results show that the first use case should be used in months with a bull market, which can be described as a period of generally rising prices, using only short positions. Contrarily, the second test scenario using only long positions should be select for periods with a bear market, where a general decline in the digital currencies rate happens over a period of time. An important aspect of the developed algorithm is the ability to overcome opposite market directions. The last test case did not produce enough data for us to assess if using the ARIMA model improves or reduces the profitability of the initial use cases. Nonetheless, the fact that we were only able to test two periods reinforces that these conclusions should be further tested to check that the outcome produced is consistent with our remarks.

Finally, this evaluation phase offered innumerable challenges and objections but we ended up creating three use cases that, in spite of needing additional examinations, substantiate our objective of extracting information from the order books of digital currencies and use machine learning to improve profits.

5. Conclusions

The presented work proposes an end to end system designed in a microservice architecture that contains a database with order books and trading information of the four major digital currencies (more than 70 Gigabytes of data), an algorithm that pre-processes and aggregates information from order books and trades to define trading positions, a trading simulator, a new type of trailing stop that optimizes profits and the usage of forecasting to check the market trend.

Three use cases were developed for our trading algorithm, the first two extract order book data to find suitable trading periods whilst the last one uses the ARIMA model with a 70% training and 30% testing periods to predict cryptocurrencies' rates and use that information on the initially mentioned test scenarios. We used monthly test periods and considered three trading configurations to find the most performing strategy according to the market's situation.

All in all, despite the reduced testing sample and innumerable challenges faced, the results were promising, surpassing the Buy & Hold strategy gains and significantly reducing losses on bear markets.

References

[1] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.

[2] O. J. Mandeng. Cryptocurrencies, monetary stability and regulation: Germany's Nineteenth Century Private Banks of Issue. Technical report, LSE Institute of Global Affairs, 2018.

[3] P. K. Jain, P. Jain and T. H. McInish. The Predictive Power of Limit Order Book for Future Volatility, Trade Price, and Speed of Trading. *SSRN Electronic Journal*, 2011.

[4] I. H. Witten and E. Frank. *Data Mining : Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 4th edition, 2005.

[5] M. D. Rechentin. *Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction.* PhD thesis, University of Iowa, 2014.

[6] G. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control.* Wiley, 1970.

[7] R. Frigola. *Bayesian Time Series Learning with Gaussian Processes.* PhD thesis, University of Cambridge, 2015.

[8] G. Bontempi, S. Ben Taieb and Y.-A. Le Borgn. Machine Learning Strategies for Time Series Forecasting. *Business Intelligence: Second European Summer School*, pages 62–67, 2013.

[9] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications: With R Examples.* Springer, 2011.

[10] T. Guo and N. Antulov-Fantulin. Predicting short-term Bitcoin price fluctuations from buy and sell orders, 2018.

[11] J. Donier and J.-P. Bouchaud. Why Do Markets Crash? Bitcoin Data Offers Unprecedented Insights. *PLOS ONE 10*, pages 1–11, 2015.

[12] Y. B. Kim, J. G. Kim, W. Kim, J. H. Im, T. H. Kim, S. J. Kang and C. H. Kim. Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies. *PLOS ONE 11*, 2016.

[13] D. Garcia, C. Tessone, P. Mavrodiev and N. Perony. The digital traces of bubbles: feedback cycles between socio-economic signals in the Bitcoin Economy. *Journal of The Royal Society Interface 11*, 2014.

[14] M. Amjad and D. Shah. Trading Bitcoin and Online Time Series Prediction. *IPS 2016 Time Series Workshop*, pages 1–15, 2017.