Master's Thesis

A Cryptocurrency Evaluation and Trading System

Ibrahim Alshibani

Examiners: Prof. Dr. Hannah Bast
Prof. Dr. Dirk Neumann

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Computer Science
Chair of Algorithms and Data Structures
July 25th, 2019

Writing period

15. 02. 2019 – 25. 07. 2019

Examiners

Prof. Dr. Hannah Bast

Prof. Dr. Dirk Neumann

Advisers

Prof. Dr. Hannah Bast

Bernhard Lutz

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Signature

Abstract

Accurately predicting asset prices in a financial market is a very difficult task. This difficulty arises from the large number of real-world factors that drive the price. Often these factors are hard to model or follow a random behaviour. This thesis studies the predictability of a financial market of a new type of asset, cryptocurrencies. We employ two types of approaches to this problem, the first one is based on Technical Analysis. We utilized an approach based on grid search and cross validation to find the best set of parameters for the trading strategies based on technical analysis. This approach was applied to 5 different cryptocurrencies and 7 different trading strategies. All candidate strategies that resulted from the cross-validated grid search method performed well on out-of-sample data, yielding positive returns. The second approach used to tackle the problem was based on computer learning. We fitted historical data for 5 different cryptocurrencies to a Recurrent Neural Network model that uses the Long Short Term Memory (LSTM) architecture. The model was able to achieve limited accuracy improvement over a random classifier by around 5% for 3 out of 5 currencies. Complementary to the first approach, we designed a trading system that enables the user to run trading simulations on historical data, and launch a trading bot to perform automated trading based on a selected trading strategy. By comparing the performance of the trading bot to that of the offline trading simulations, we found that our offline simulation is a realistic approximation to real-world live trading.

Zusammenfassung

Die genaue Vorhersage der Vermögenspreise auf einem Finanzmarkt ist eine sehr schwierige Aufgabe. Diese Schwierigkeit ergibt sich aus der Vielzahl von realen Faktoren, die den Preis beeinflussen. Oft sind diese Faktoren schwer zu modellieren oder folgen einem zufälligen Verhalten. Diese Arbeit untersucht die Vorhersagbarkeit eines Finanzmarktes für eine neue Art von Vermögenswerten - Kryptowährungen. Wir verwenden zwei Arten von Ansätzen für dieses Problem, die erste basiert auf der Technischen Analyse. Dieser Ansatz basiert auf Grid-Suche und Cross-Validierung, um den besten Parametersatz für die Handelsstrategien auf der Grundlage der technischen Analyse zu finden. Dieser Ansatz wurde auf 5 verschiedene Kryptowährungen und 7 verschiedene Handelsstrategien angewendet. Alle Kandidatenstrategien, die sich aus der cross-validierten Rastersuchmethode ergaben, schnitten bei Out-of-Sample-Daten gut ab und brachten positive Ergebnisse. Der zweite Ansatz, der zur Lösung des Problems verwendet wurde, basierte auf Machine Learning. Wir haben historische Daten für 5 verschiedene Kryptowährungen an ein Recurrent-Neural-Network-Modell angepasst, das die Long Short Term Memory (LSTM) Architektur verwendet. Das Modell konnte eine begrenzte Verbesserung der Accuracy gegenüber einem zufälligen Klassifizierer um etwa 5% für 3 von 5 Währungen erreichen. Ergänzend zum ersten Ansatz haben wir ein Handelssystem entwickelt, das es dem Benutzer ermöglicht, Handelssimulationen auf historischen Daten durchzuführen und einen Trading-Bot zu starten, um automatisierten Handel basierend auf einer ausgewählten Handelsstrategie durchzuführen. Durch die Validierung der Leistung des Trading-Bots im Vergleich zu den Offline-Handelssimulationen, haben wir festgestellt, dass unsere Offline-Simulation eine realistische Annäherung an den realen Live-Handel darstellt.

Acknowledgements

First and foremost, I would like to thank:

- My family for always supporting me in all possible ways and for their unconditional love.

- Professor Bast for her choice of the very interesting topic and advice throughout the thesis.

- Bernhard Lutz for his very helpful feedback and input and giving me ideas on how to expand my work.

- Mrs Ursula Epe for her understanding and help that allowed me to continue my studies during the instability and war happening in my home country.

- My girlfriend Ksenia for being my rock and believing in me.

- All my friends for supporting me throughout, especially Abdullah for his thoughtful input and Manika for proof reading my thesis.

Abbreviations

- **EMH** - Efficient Market Hypothesis
- **MA** - Moving Average
- **BB** - Bollinger Bands
- **RSI** - Relative Strength Index
- **ANN** - Artificial Neural Network
- **SGD** - Stochastic Gradient Descent
- **RNN** - Recurrent Neural Network
- **LSTM** - Long Short Term Memory
- **API** - Application Program Interface
- **ROI** - Return on Investment
- **ARIMA** - Auto-Regressive Integrated Moving Average
- **TA** - Technical Analysis

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

In this thesis, we took on the difficult task of developing a cryptocurrency trading system. This system is designed to shield its user from the complexity of the cryptocurrencies and trading world. The work can be divided into two folds, the first is comprised of the practical aspect of our work. In it we developed a trading system that provides the following functionality to the user:

- Ability to run trade simulations on historical data to assess the profitability of trading strategies. The results of these simulations are also presented visually.

- Automated trading via a trading bot. This bot can be preconfigured by the user to trade according to a certain strategy.

The second fold, which is of a more theoretical nature, revolves around studying the predictability of popular cryptocurrencies prices. These are Bitcoin, Ether, Litecoin, Dash and Zcash. The first approach we used to study the predictability of cryptocurrencies is based on "Technical Analysis". Technical Analysis refers to the task of attempting to predict the future price of an asset based on historical prices of that asset. These types of trading strategies are commonly used by traders in many types of financial markets (Refer to chapter 2 for more details and references). These trading strategies are flexible in the sense that there are multiple parameters to be tweaked. For example, the window size of a Moving Average indicator can be set to different sizes. This can be problematic as there are many different variations used by traders. In order to address this, we proposed a grid search approach based on cross validation to find optimal strategies based on historical data.

The cross-validated grid search produced multiple candidate strategies which we

tested on an out-of-sample period. Simultaneously, we launched a trading bot during this period that traded based on one of the candidate strategies. After the period was over. We were able to run simulations to asses the profitability of the candidate strategies. All strategies showed a positive return on investment. Naturally, the trading bot also yielded a positive return. During this out-of-sample period, the trading bot executed multiple transactions. After the period was finished, we compared the execution time of these transactions to the execution time of the offline simulation, we found that the execution times were identical. This indicates that our offline simulation environment is a good approximation to live trading.

The second approach we used to study the predictability of cryptocurrencies was a learning-based approach. Here we fitted historical cryptocurrency prices to a Recurrent Neural Network which utilizes Long Short Term Memory cells. We formulated the problem as a classification problem, where task is to predict the market directionality. We found limited predictability for all of the cryptocurrencies, we utilized various evaluation metrics, including hypothesis testing to ensure that our results are statistically significant.

Using our pre-existing simulation environment, we ran trade simulation using the signals generated from the trained LSTM model. We compared the performance of the LSTM model to multiple baselines. The LSTM model performed well on the test set for some of the cryptocurrencies. The cryptocurrencies where the LSTM performed well, have a higher accuracy of prediction than those without.

1.1 Motivation

Why cryptocurrencies? To answer this question, it is essential to understand what exactly a cryptocurrency is:
It is a digital currency, i.e., it has no physical form (notes or coins). However, it differs from other digital currencies such as the money in a bank account or a PayPal account. This difference lies in the necessity for a central authority to exist . Cryptocurrencies are in essence a cleverly designed protocol. The first protocol named Bitcoin [3] by Satoshi Nakamoto (a pseudonym) was published in 2008. This

protocol ensures decentralization using the power of mathematics, or cryptography to be more specific. Decentralization here refers to delegating the power and authority traditionally associated with the central institution to the protocol design.

This concept of a decentralized currency is a very powerful one. By removing the need for a central authority, cryptocurrency holders are protected from possibly unfair regulations enforced by a central authority such as a bank. This is especially important if the participant is a resident of a country that's going through a recession. In many cases central banks enforce austerity measures that restrict an individual's access to their own funds. This can be seen in countries like Venezuela where cryptocurrencies are popular [4]. Another positive aspect to cryptocurrencies is transaction times. International transfers nowadays require a few days and are in some cases expensive. Bitcoin transaction take on average an hour to complete and are much cheaper [5].

As innovative and fascinating as Bitcoin technology is, in this thesis we are interested in the predictability of its price. We summarize why we chose cryptocurrencies as the target for this thesis:

1. **New Market**: Cryptocurrencies are a relatively new phenomenon and are not as extensively studied as fiat currencies or stocks.

2. **Volatility**: Bitcoin value is much more volatile than traditional fiat currencies. Yermack [6] found that in 2013 the volatility of bitcoin was 142 %. Compared to that of Euro, Yen, British Pound,and Swiss Franc which was in the range of 7 to 12 %. This volatility means that Bitcoin is a relatively risky asset. It also means that there's a great potential to make profit or lose a lot.

3. **Transaction Fees**: Transaction fees for cryptocurrencies are traditionally low. For example Kraken [7] charges fees in the range of 0% to 0.26% per transaction depending on the trading activity of the user. The larger the volume the cheaper the transaction.

4. **Abundance of High-Frequency Data**: Many cryptocurrency exchanges have free public APIs that allow users to query the entire transaction history of some assets. This free access to high frequency trading data stock market is

not available in the stock market.

1.2 Thesis Layout

In chapter 2 we explain some seminal concepts in the field of financial trading and discuss some related pieces of scientific work. In chapter 3 we describe in detail, all the trading strategies implemented by our system. We also describe some of theory behind our learning based approach. Chapter 4 examines the practical aspect of our work. How the data was crawled and processed, implementation details, programming libraries utilized, how we set up the experiments and finally the evaluation metrics we used. Finally we present our findings in chapter 5, discuss them and conclude this thesis.

2 Related Work

2.1 Efficient Market Hypothesis

The concept of Market Efficiency is mentioned in nearly all the literature we encountered while researching trading strategies. The concept was first popularized by Eugene Fama in 1970 [8]. Fama defines an efficient market as one where the price *fully reflects* all available information. This means that the price of a publicly traded asset in an efficient market cannot be consistently predicted. An implication of this is that the price of an asset follows a random walk process. This means that at each time-step there is an equal chance of the price going up or down, and that each price movement is independent from the previous price movements.

Fama describes three forms of market efficiency based on available information :

1. *Weak*-form : Information set only includes previous asset prices.

2. *Semi-weak* form: Information set includes previous asset prices and publicly available information such as news.

3. *Strong-form*: Information set includes previous asset prices, publicly available information and secret information.

Weak-form efficiency indicates that using historic price data, it is impossible to predict the future direction of an asset. This technique is commonly used by "chartists". Chartists try to find common patterns in data that are supposed to be indicative of future price movement as they believe these patterns are self-repeating. Chartists

often use technical indicators, such as the ones we discuss in section 3.1, to aid them in their decision making. The semi-weak and strong form are even stricter. The strong-form of market efficiency implies that insider trading cannot be used to beat the market.

Whether or not stock markets or foreign currency exchanges are efficient, has been fiercely debated by economists and statisticians. For a long time the EMH has been widely accepted, but with time some anomalies began to be uncovered by academics. One such example is the January Effect, Rozeff and Kinney found that returns in January for companies listed in the New York Stock Exchange were on average significantly higher than other months. This indicates seasonality[9] .

The January effect is not the only trend discussed in the literature. Another notable one is the Halloween effect, represented by the market saying "Sell in May and Go Away" . This effect implies that stock market returns are significantly lower in the months between May and October. In comparison to the remainder of the year. Bouman and Jacobsen found that this effect is in 36 out of the 37 different stock market indices that they included in their study[10].

Malkiel denies the importance of seasonal trends and effects by stating that they are too irregular [11]. Malkiel also explains that if there are profitable seasonal effects, their profitability will be diminished as soon as they are publicized as the market will adjust very quickly to this new information.

A stronger case against Market Efficiency is perhaps the wide-spread use of Technical and Fundamental analysis in both foreign exchange markets. *Technical Analysis* is the use of historic stock data. For example historic prices, trading volumes and derivatives of the two. In order to predict the direction of the price movement. *Fundamental Analysis* is the use of financial information. For example quarterly earnings (stock Trading) or Central Bank policy (foreign currency trading) in order to predict future price movements. Technical analysis violates all forms of Market Efficiency whereas Fundamental Analysis violates the semi-weak and strong form of market efficiency. If a market truly is efficient then the use of such techniques is pointless.

This wide-spread use was reported by Menkhoff and Taylor [12]. They analyze a

collection of surveys which targeted foreign exchange professionals. They summarize their findings in a set of stylized facts:

1. Almost all foreign exchange professionals use technical analysis as a tool in decision making at least to some degree.

2. Most foreign exchange professionals use some combination of technical analysis and fundamental analysis.

3. The relative weight given to technical analysis as opposed to fundamental analysis rises as the trading or forecast horizon declines.

There has also been a considerable amount of literature that examines the profitability of trading strategies based on technical analysis. This is more comparable to our work. Sullivan et al. have devised a new technique for finding profitable trading strategies from a pre-selected universe that fixes for data snooping[13]. Their technique is based on drawing multiple samples with replacement of the returns of each strategy. Although they found statistically significant returns for the best strategy compared to the benchmark, the results could not be reproduced on out-of-sample data. This was the case for both criteria used for evaluating the performance of trading strategies, Mean Return and Sharpe ratio. It was difficult for us to conduct the same experiment on cryptocurrencies for a few reasons. Firstly, our data was high-frequency and spans back only to 5 years max. Had we changed the frequency to daily as is the case with the data used by Sullivan et al., we would not have had a large enough amount of data to perform a conclusive experiment.

2.2 Efficiency of the Cryptocurrency Market

Although cryptocurrency trading exchanges are relatively new and are not as extensively studied as the traditional stock or fiat currency markets, yet some literature addressing the efficiency of Bitcoin does exist. Urquhart applied 5 different statistical tests to assess whether the Bitcoin prices behave like a random walk [14]. The data used was from 1/8/2010 to 31/7/2016. All tests show evidence of non efficiency in

that period. However, when the sample was split into two sub-samples. The results of the first sub-sample (1/8/2010 – 31/7/2013) didn't differ from the entire sample. The second sub sample (1/8/2013 – 31/7/2016) passed only 3 out of 5 tests. This could be an indication of Bitcoin prices becoming more efficient with time.

In a follow-up study to Urqhart's [14], Nadarajah and Chu have found that by simply raising the returns on Bitcoin to the power of an odd integer, the returns become market efficient [15]. They argue that scaling the entire time series to a higher odd power does not lead to any information loss. Negative returns stay negative and positive remains positive. They utilized the same set of tests as [14] and expanded them to 8, in all cases either weak or no evidence were found against the null hypothesis that Bitcoin price follows a random-walk pattern.

There has been, however, a considerable amount of work that is more directly comparable one part of our work. Namely, attempting to predict the direction of the price of cryptocurrencies using Learning-based algorithms. One such piece of work was done by Madan et al. [16], who applied a Linear Model, Random Forest and Support Vector Machine algorithms to assess the predictability of Bitcoin. Their input consisted of daily bitcoin data along with other bitcoin mining related features. They managed to reach very High accuracies of 98% and 94% percent using the Linear model and Random Forest algorithm respectively. These high accuracy values are attributed to an imbalanced dataset. However, when utilizing high granularity bitcoin price data only, the accuracy dropped to the range of 50-55%. Our work differs in that we apply our model to 5 different cryptocurrencies and we utilize a deep learning approach based on the LSTM architecture.

Hegazy and Mumford applied 6 different learning models on historical Bitcoin data [17]. Similar, to our work they utilized 2 different evaluation metrics, one based on classification hits and the other based on trade simulations. In terms of classification hits, the most successful models were Adaboost with decision trees and decision stumps (one-level decision tree) as a base classifier yielding a correct rate of 57.4% and 57.1% respectively, on the test set. Using the other evaluation metric, namely, trade simulations. The most successful method by a big margin was based on recurrent reinforcement learning. One notable difference in Hegazy and Mumford's work to ours is in the simulation method. During data preprocessing, after binning

the prices, the authors smooth the prices using weighted linear regression (For both training and test set). They had done this to reduce noise in the data. Without this smoothing, the authors claimed that the models lost their predictive power. However, this smoothed price could in certain cases not correspond to the actual price. The authors also acknowledge this fact. Therefore it's not clear if the simulation results could be applicable. We do not smooth the prices in our approach which renders it as a more realistic trading simulation.

The academic literature also included results for other cryptocurrencies. Chen et al. applied various learning approaches to Ether historical data with various success [18]. Interestingly the most successful model was ARIMA, which is commonly used for time series analysis by statisticians and economists. It achieved an accuracy of 61%. The authors also utilized an RNN model in their test. The reported a balanced accuracy score of 52.43%. The results we achieved for the same currency were better as will be further discussed in chapter 5.

3 Background

This section of the thesis discusses in detail the concepts we used in order to gain insight and to attempt to leverage the cryptocurrency trading market. In section 3.1 we describe some of the trading strategies typically used by traders in Foreign Exchange and Stock Trading market. Section 3.2 discusses the learning based approach we applied in order to learn and consequently predict the future prices of cryptocurrencies.

3.1 Trading Strategies from the Market

We describe, in this section, all the trading strategies implemented in our trading system. These trading strategies are based on Technical Analysis.

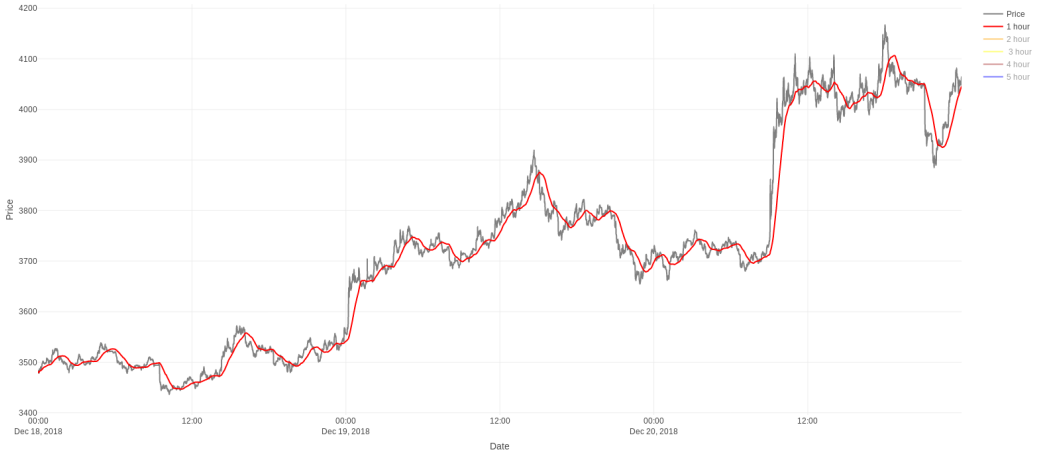
3.1.1 Cross Over Moving Average

The moving average and all of its variations are some of the most popular and widely used indicators [13]. The moving average acts as a trend indicator by calculating an average price of an asset for a fixed window size. Comparing a MA to the price line or to other MAs with different window sizes, can be useful in detecting trends. This in turn can produce some useful trading signals. We can define the MA at time step t for window size N as:

$$MA_t = \frac{1}{N} \sum_{i=0}^{N-1} P_{t-i} \tag{1}$$

where P_t is the price of an asset at time step t. Figure 1, produced by our trading system, visualizes equation 1. The figure also demonstrates the smoothing property of MAs.

Figure 1: Bitcoin price from the 18th to the 20th of December 2018 along with the 1 hour MA



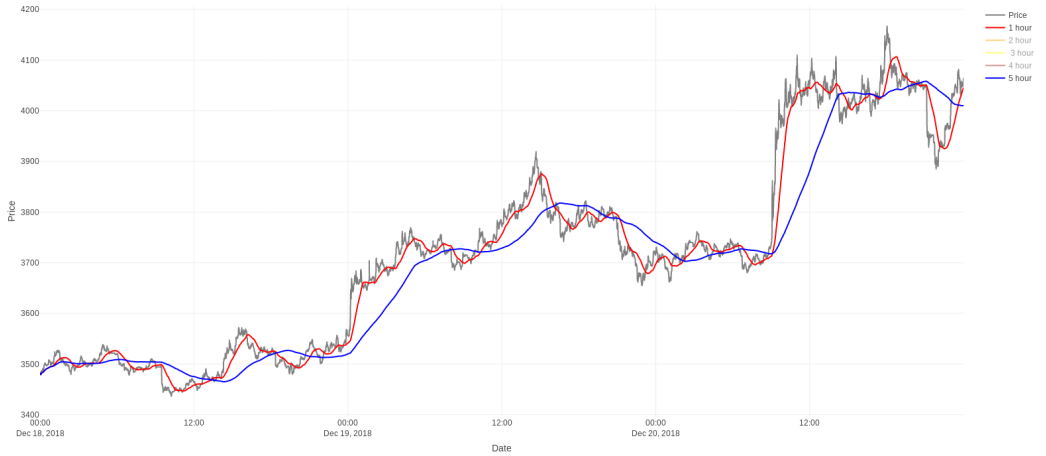
The cross over MA strategy utilizes two MAs, a short (fast) one and a long (slow) one. The short MA represents the short-term trend of the price and it should react to price changes relatively quickly in comparison to the slower moving MA. We demonstrated this idea visually in figure 2 The point of intersection of the slow and fast MA could be used as an indicator of upwards and downwards price shifts. This can be observed in figure 2. The trading strategy uses these cross-over points to generate buy and sell signals.

More formally, let's define a signal generating function S_t that takes as input short window s and long window l as at time step t:

$$S_t(s,l) = \begin{cases} 1 & \longrightarrow \quad MA(s) > MA(l) \\ -1 & \longrightarrow \quad MA(s) < MA(l) \\ 0 & else \end{cases} \tag{2}$$

The formulation above can be regarded as the simple form of the cross over strategy. We add two new parameters in order to alleviate the effects of fake signals and allow

Figure 2: Cross over strategy on Bitcoin with a 1 Hour and 6 Hour moving average



the strategy more flexibility [13]. These two new parameters are percentage filter p, and holding period h. The percentage filter specifies a threshold that should be exceeded by a certain percentage in order to generate a signal. So if p is set to 1%, a buy signal will only be generated if $MA(s)$ is greater than $MA(l) \times 1.01$ and the opposite applies for generating a sell signal, such that $MA(s)$ should be less than $MA(l) \times 0.99$. The holding period h indicates how long should a signal generating condition (for example $MA(s) > MA(l)$) be valid before a signal is actually generated.

In the literature, we found that, the cross over moving average strategy is often presented in the above form i.e. a trend following strategy. If the short MA overtakes the long one, a signal is generated. In this case the bet, is on the continuance of this trend. Hence the name *trend-following.* However, a contrarian form is also less frequently used. Pavlov et al found that contrarian MA strategies are significantly more profitable than trend-following ones on the Australian stock market [19]. Our trading system allows for the use of both trend-following and reversing forms of strategies for evaluating or trading.

There are numerous ways of calculating the average for the Moving Average indicator. One alternative method is calculating a exponentially weighted average that places greater importance on recent prices than more distant ones. This variation

is also available for use in our trading system. We calculate the exponential moving average at time step t with window size N as :

$$EMA_t(N) = \frac{\sum_{i=0}^{N-1} w_i P_{t-i}}{\sum_{i=0}^{N-1} w_i}, \tag{3}$$

Where w_i are the exponentially smoothed weights. These weights are defined as $w_i = (1-\alpha)^i$ where α is the smoothing factor. α's value is set to $\frac{2}{N+1}$ for a window of size N in practice. This is the convention we also follow in our trading system implementation.

3.1.2 Bollinger Bands

Another commonly used trading indicator are Bollinger Bands, Here an upper and lower band are used within which the price should remain. We calculate these bands based on the previous volatility of the price. It is assumed that price should remain within these bands and a "breakout" from these bands, indicates a new trend that could be leveraged. As originally presented by Bollinger in 1992 [20], when the price touches the upper band, it is overbought. This means that its only a temporary increase in price and the price should eventually revert back to normal. The same can be said about the lower band corresponding to oversold conditions, such that crossing this lower band indicates a temporary drop and the price will revert back to normal. Bollinger Bands as originally presented [20] are a trend reversing strategy. Figure 3 produced by our trading system demonstrates the strategy visually.

To compute the upper and lower bands we use the standard deviation σ_t for a time step t. It is defined as :

$$\sigma_t = \sqrt{\frac{1}{N-1} \sum_{i=0}^{N-1} (P_{t-i} - MA_t)^2} \tag{4}$$

Figure 3: Bollinger Bands in action on Bitcoin data from the 3rd to the 11th of April 2019

The upper and lower bands are then computed as :

$$\begin{aligned} UB &= MA_t + (width \times \sigma_t) \\ LB &= MA_t - (width \times \sigma_t) \end{aligned} \tag{5}$$

In the original work by Bollinger the $width$ is fixed to 2. We leave it as a variable here to allow greater flexibility. We define a signal generating function that takes as input a window size N and $width$ as :

$$S_t(N, width) = \begin{cases} 1 & \longrightarrow \quad P_t < LB_t \\ -1 & \longrightarrow \quad P_t > UB_t \\ 0 & else \end{cases} \tag{6}$$

Similar to the cross over strategy described in the previous section, we introduce two other parameters a percentage filter p and holding period h. Although originally presented as a trend reversing strategy, there's literature that provides evidence of the profitable use of Bollinger Bands as a trend following strategy [21]. Our trading system allows the use of Bollinger Bands in both variations.

3.1.3 Relative Strength Index

The Relative Strength Index, often abbreviated as RSI, was first introduced by J. Welles Wilder [22]. It is used to measure the momentum of a price, is it also called the Momentum Relative Strength Index. Momentum here refers to the rate of change of a price. To calculate the RSI, we first need to calculate the sequence C which contain all the price changes in a given window N:

$$C_t = \{P_{ti} - P_{t-i-1}\}_{t=0}^{N-1} \tag{7}$$

We calculate two subsequences out of C, C_+ and C_- to represent the sequence of gains (where there's a positive change) and losses respectively.

$$\begin{aligned} C_+ &= C_{>0} = \{x \in C \mid x > 0\} \\ C_- &= C_{<0} = \{x \in C \mid x < 0\} \end{aligned} \tag{8}$$

The relative strength RS of an asset at time step t is simply the average gain divided by the average loss for a given window N:

$$RS_t = \frac{\bar{C_+}}{\bar{C_-}} \tag{9}$$

Finally the RSI, is scaled to the range 0 to a 100 such that :

$$RSI_t = 100 - \frac{100}{1 + RS_t} \tag{10}$$

Weller also defines an oversold and an overbought line which act as an indication of when to buy and sell respectively. Thus, in this respect RSI is a trend-reversing strategy. We define a signal generating function as :

$$S_t(N, width) = \begin{cases} 1 & \longrightarrow \quad RSI_t < 50 - offset \\ -1 & \longrightarrow \quad RSI_t > 50 + offset \\ 0 & else \end{cases} \tag{11}$$

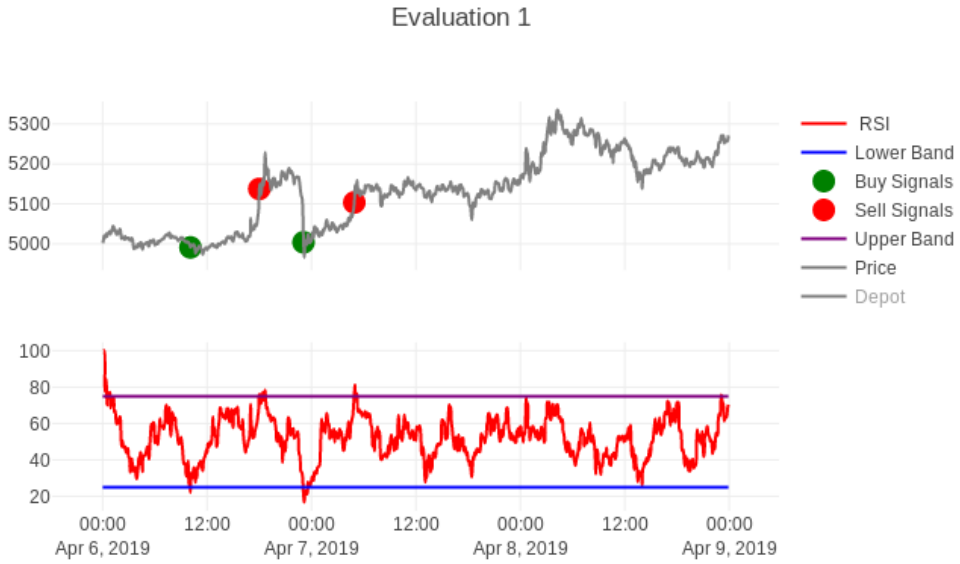


Figure 4: RSI trading strategy with a window size of 60 and offset of 25

Similar to the aforementioned strategies, here also we introduce percentage filter p and holding period h. In total, our RSI trading strategy takes as input four parameters: window size N, offset (the distance to the oversold and overbought line from 50), percentage filter p and holding period h. Figure 4 demonstrates the idea behind RSI visually.

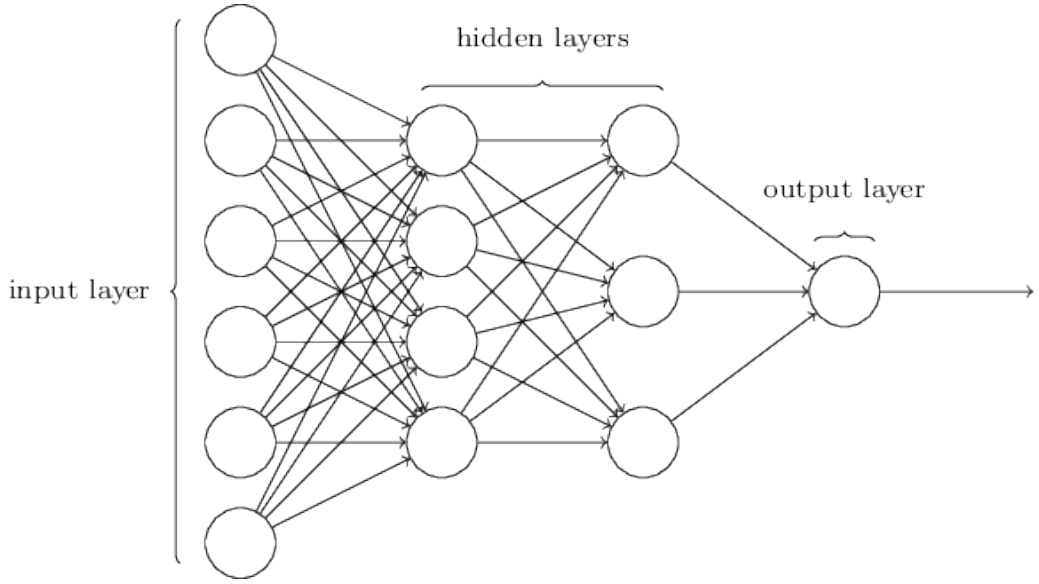
3.2 Learning Based Approach

As a part of this thesis, we utilized a Recurrent Neural Network (RNN) for predicting the future direction a price will take at each time step. In this section we describe the learning techniques used in detail. The goal of this approach is to uncover patterns in the historical price data. If such patterns exist, a learning based model should be able to learn them. Armed with this trained model it is possible to leverage this predictability to achieve a profit.

3.2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) can be defined as a computing system, the design of which is very loosely based on that of the neurons in our brains. As the name suggests, they're composed of interconnected neurons. The architecture of an ANN can be used for various types of ML tasks such as classification, regression and many others, by simply altering the structure of the network.

Figure 5: ANN Architecture. Source : "Neural Networks and Deep Learning" by Michael Nielsen [1]



We will start by defining the smallest component in an ANN, the artificial neuron. An artificial neuron takes multiple values x_j as input, multiplies them to corresponding weights for each input, offsetting them with a bias producing the output. If a neuron has j inputs, we can represent the output z of a neuron as : $z = \sum_j w_j x_j + b$. We can vectorize the notation by representing the weights and inputs as one dimensional vectors to simplify the notation : $w \cdot x + b$. An activation function, (sometimes referred to as Transfer functions), is usually used in tandem with neurons such that, the output is passed to this function. The activation functions usually chosen are non-linear. In an ANN, neurons are usually stacked vertically in layers as seen in figure 5. A neural network produces an output by performing a forward pass. A

forward pass is essentially passing the input vector x as input to the first layer then passing the output of that operation to the second layer and so on until reaching the final output layer.

By combining the vector of weights for each node in the first hidden layer such that they're concatenated together vertically together to form a matrix W. The out put of hidden layer one can be represented as : $W^\top x + b$. Building up on that logic, the dimensionality of matrix W can always be presented in the form *(Number of inputs from previous layer* $\times$ *Size of hidden layer)*. ANNs tend to perform best with a large number of hidden layers, these type of networks are called Deep Neural Networks.

3.2.2 Activation & Loss Functions

Learning Process

So far we have only discussed the flow of information in one direction in a network, the forward direction. Yet the question remaining is how do networks learn? It is useful to define a Cost or Loss function. The purpose of this function is to quantify how well the output of the network is in comparison to a training example. We measure this by comparing to the actual labels present in a training set. In a regression problem, Mean Squared Error is often used as a loss function. For n training examples, where $\hat{y}$ is the output of the neural network and y is the actual label from the training set, MSE is:

$$L_{MSE} = \frac{1}{n}\sum_{i}^{n}(y - \hat{y})^2 \tag{12}$$

Please note the equation 12 assumes that the output $\hat{y}$ is a one real number, which would work fine if the dimensionality of the output layer of the neural network is 1. However, if not, then the notation would be adjusted slightly.

Alternatively, for classification problems where a soft max layer is used (see equation 18), the output of the network is a probability distribution over the classes. In this setting, cross entropy is often used. In a binary classification problem cross entropy

can be defined as :

$$L_{CrossEntropy} = -\frac{1}{n}\sum_{i}^{n}[y \ln \hat{y} + (1-y)\ln(1-\hat{y})] \tag{13}$$

By computing a loss function, the problem of learning the weights and biases becomes that of numerical optimization. By minimizing a loss function we are able to find weights and biases that reduce the error on the training set, thus producing better predictions. One way to do this is by using Gradient Descent. Using the notation defined in [1], with a loss function L that takes as input a weight w and bias b. We can approximate the relation between L, w and b as:

$$\Delta L \approx \frac{\partial L}{\partial w}\Delta w + \frac{\partial L}{\partial b}\Delta b. \tag{14}$$

The update rules for minimizing the cost function using gradient descent are :

$$\begin{aligned} w \rightarrow w' = w - \eta\frac{\partial L}{\partial w} \\ b \rightarrow b' = b - \eta\frac{\partial L}{\partial}. \end{aligned} \tag{15}$$

Where η is the learning rate. In practice, the number of training examples can be very large (this is particularly true for ANNs). It would be very slow to compute the gradient for all n training examples, and then updating the weights after each computation. Alternatively, Stochastic Gradient Descent is used instead [23], SGD calculates the gradient for a randomly chosen batch that contains a fixed number of training samples, and updates are then carried out accordingly.

At first glance, it's not obvious how to apply gradient descent, or any optimization algorithm, on a neural network because of the shape of ANNs. In the formula above we only discussed the weights and biases of the final output layer of the network. The weights and biases in previous layers also need to be updated. In order to carry this out, the Backpropogation algorithm is used. The use of BP for neural networks was first popularized by Rumelhart et al [24]. BP makes use of the chain rule, which offers a method of calculating the derivative of composite function. The output of one

path in a network from the input layer to the output layer is in essence a composite function. BP is a systematic way of calculating the derivatives of the loss function with respect to the output nodes of all layers and in turn the weights and biases. To train a network, BP is used in tandem with an optimization algorithm like SGD, to iteratively find gradients and update the weights until convergence or a certain threshold is met.

Activation Functions

ANNs make use of activation functions for the following reasons:

1. To introduce non-linearity to the output of the neurons. If all outputs of the neurons are linear, then all hidden layers in the network could be represented as a single hidden layer. This greatly diminishes the ability of ANNs to learn more complicated functions.

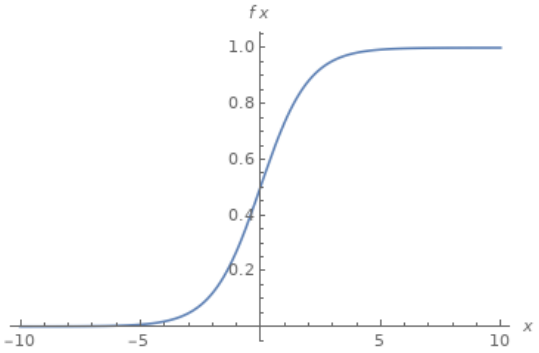
2. To scale the output of a neuron to a certain range such as -1 to 1 or 0 to 1, depending on the activation functions. This ensures a more stable learning process.

One of the most commonly used activation functions is the Sigmoid (σ) activation function, using the notation from the previous section where z is the output of a neuron, a sigmoid function applies the following to each neuron:

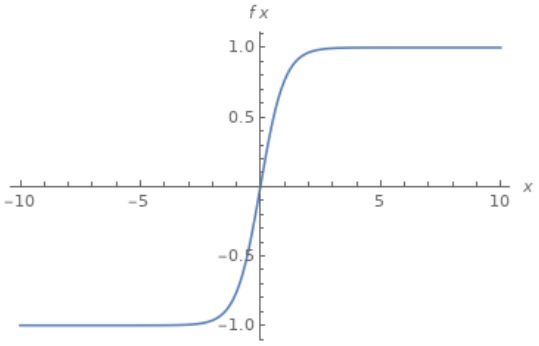
$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{16}$$

Another commonly used activation function, which is also sigmoidal in shape, is the tanh function. It is defined as:

$$\tanh(z) = \frac{(e^{z} - e^{-z})}{(e^{z} + e^{-z})} \tag{17}$$



(a) Sigmoid Function



(b) Tanh Function

Figure 6: Commonly Used Sigmoidal Activation Function. Figures created using Wolfram Alpha [25].

For classification problems, the final layer of the network is often set as a Softmax Layer. This is a layer where all the neuron output is passed through a softmax function. The softmax function differs from others discussed so far as it takes as input the output of all the neurons present in the final layer. It scales the output of all the neurons to a value between 0 and 1, such that the sum of all these outputs adds up to 1. In effect we end up with a probability distribution over all the possible classes. The softmax output for neuron j in a final layer with N nodes can be defined as

$$SoftMax(\mathbf{z_j}) = \frac{e^{z_j}}{\sum_{i=1}^{N} e^{z_i}} \tag{18}$$

Finally, we define the Rectified Linear Unit activation function or Relu for short as:

$$f(z) = \begin{cases} 0 & \text{for } z < 0 \\ z & \text{for } z \geq 0 \end{cases} \tag{19}$$

3.2.3 Neural Networks with Time Series Data

Time series data refers to sequence data with temporality. Traditional ANNs are not designed for learning from this kind of data. Although, it's possible to do so, by feeding in each point in time separately. This proves to be inefficient as the network will not be able learn any dependencies between the different points in time. A variation of ANNs called Recurrent Neural Networks, are designed specifically for this purpose. RNNs allow information from neurons to persist from one input in time x_t to the next x_{t+1}. This can be thought of as multiple copies of the same ANN arranged in sequence but of course with a new set of parameters to connect the neurons from one point in time to the next. Figure 7 shows the fundamental structure of an RNN where x_t represents the time-dependent input and A can be thought of as a node. The figure demonstrates the important property of RNNs, that the output is not only fed further into the network like a traditional ANNs. It's also fed to a copy of the ANN that takes in the next input x_{t+1}. One issue that arises with the use of RNNs is their inability to learn long-term dependencies because gradients tend to grow exponentially or vanish. Bengio et al showed through a group of experiments why this is the case [26].

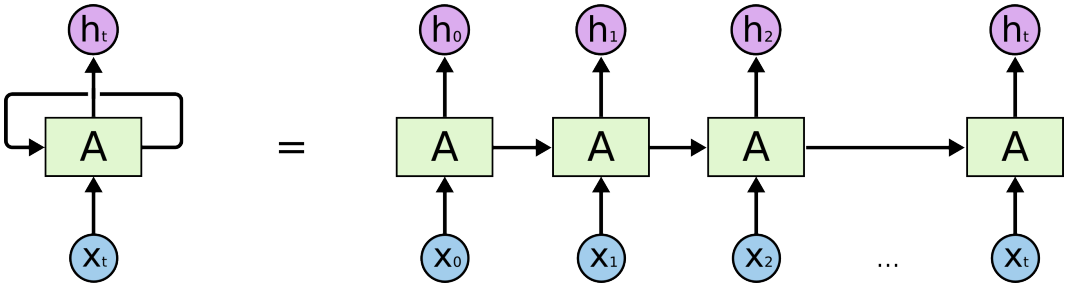


Figure 7: RNN "unrolled". Figure source : [2]

To address this issue, Hochreiter and Schmidhuber created a new RNN architecture called Long Short Term Memory [27]. LSTMs address the gradient problem by introducing gate units that control the flow of information within and between layers. They referred to these gate units together as a memory cell. Figure 8 shows the shape of this cell.

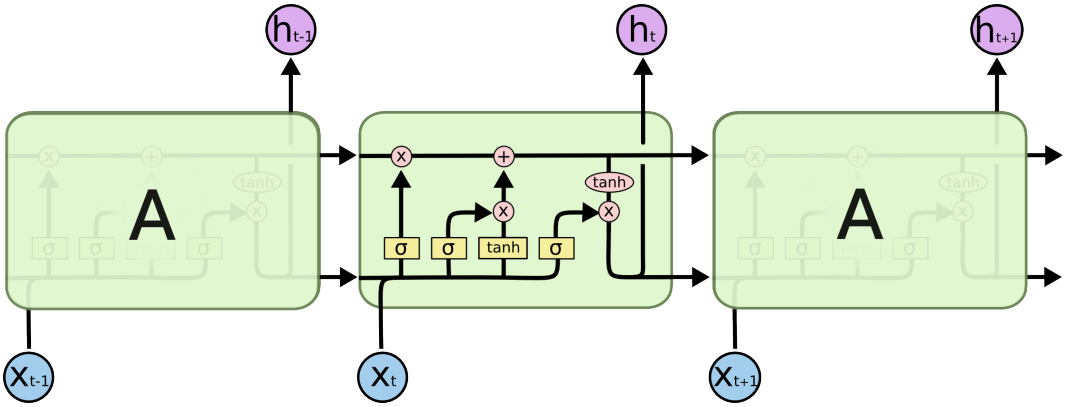


Figure 8: LSTM memory cell. Figure source : [2]

An LSTM cell is composed of 3 gate units. Each one of these gates has a unique role in the cell. The horizontal line at the top of the cell represents the flow of information between LSTM cells. This represents the selective memory in the LSTM cell. Using the notation followed in [2], this information flow between cells is stored in the Cell State C_t. The cell states can be modified at two gates. The first is the forget gate:

$$\begin{aligned} f_t &= \sigma(W_f[x_t, h_{t-1}] + b_f) \\ C_t &= f_t \circ C_{t-1} \end{aligned} \tag{20}$$

x_t is the input vector whereas h_{t-1} is the hidden state from the previous cell. If x_t is of dimensionality $(m \times 1)$ and h_t is $n \times 1$, then $[x_t, h_{t-1}]$ is a vertically concatenated vector of the two with dimensionality $((m+n) \times 1)$. W_f is the weight matrix of the forget gate it is of the dimensionality $(NumberofNeurons \times (m+n))$. Equation (20) is identical to that of a hidden layer in an ANN with a sigmoid activation function. It is also important to note that, $NumberofNeurons$ is the same for all layers in an LSTM cell, and that the cell state and hidden state share the same dimensionality.

The output of the first equation in 20 is a vector of dimensionality ($NumberofNeurons \times 1$). The sigmoid function ensures that the vector values are in range 0 to 1. When this vector is passed through the forget gate, each value determines which values to attain (closer to 1), and which to discard (closer to 0). The actual dropping of values occurs in the second equation in 20. The $\circ$ in the second equation represents a Hadamard product. A hadamard product is a special multiplication between two matrices of the same dimensionality. Each value is multiplied with the corresponding (same index) value in the other matrix. Thus, by learning the suitable weights the LSTM architecture allows an RNN to have selective memory, by dropping unuseful information.

The second gate that changes the cell state C_t is the input gate,. It is composed of two layers, a sigmoid and a tanh layer. The sigmoid layer behaves similarly as the forget gate as weights associated with this layer determine which values are affected in the cell state. However, unlike the forget gate, the sigmoid layer here determines which new values are input to the cell state C_t as opposed to which values are dropped. Finally, the tanh layer creates the new candidate values $\tilde{C}_t$. The operations of the input layer can be represented as:

$$\begin{aligned} i_t &= \sigma(W_i[x_t, h_{t-1}] + b_i) \\ \tilde{C}_t &= \tanh(W_c[x_t, h_{t-1}] + b_c) \\ \tilde{C}_t &= i_t \circ \tilde{C}_t \\ C_t &= C_t + \tilde{C}_t \end{aligned} \tag{21}$$

The final gate, called the output gate, controls the output of the cell. Similar to previous gates a sigmoid layer is used to determine which elements of the cell state C_t are going to effect the output of the cell. The operations behind the output gate are :

$$\begin{aligned} o_t &= \sigma(W_o[x_t, h_{t-1}] + b_o) \\ C_t &= \tanh(C_t) \\ h_t &= o_t \circ C_t \end{aligned} \tag{22}$$

RNNs are trained with a variation of the backpropagation algorithm called back-

propagation through time [28]. BPTT is very similar to standard BP and relies on the same principles such such as the chain rule. It is altered to consider the structure of RNNs [29].

4 Experiments

4.1 Trading System Overview

4.1.1 System Layout

Our trading system is designed to shield the user from the complex world of trading. It offers a smooth user interface that allows the user to run trade simulations on historic cryptocurrency data and launch a trading bot that trades automatically in the background for the user. The trading system currently supports five different cryptocurrencies :

1. Bitcoin [3] : The first ever cryptocurrency launched in 2011 anonymously. It has since grew to be the largest cryptocurrnecy in terms of market capitalization. It has a very active open-source community.

2. Ethereum [30]: Released in 2015. As of the time of writing, it has the second largest market capitalization after Bitcoin.

3. Litecoin [31]: Released in 2011, it is based on the Bitcoin protocol with minor alterations claiming to improve Bitcoin.

4. Dash [32]: Released in 2015. It is also based on the Bitcoin protocol.

5. Zcash [33]: Marketed at release in 2016 as an alternative to Bitcoin that provides more privacy.

The output of trade simulations is also presented visually. These trade simulations can be ran with any of the strategies detailed in section 3.1. Before running a trade simulation, the user needs to set a few global parameters that are applicable to all simulations and not specific to one. These include :

- Starting Capital : Amount to invest in USD.
- Evaluation Period : The time during which to run simulations.
- Data Origin : If the historic data is stored on disk or should be queried live.

The user is then required to set the parameters associated with each strategy. These parameters are also described in more detail in section 3.1. After settling on a promising strategy, the user then has the option to proceed to test this strategy in practice on real data. Our trading system is designed to work in tandem with the Kraken cryptocurrency exchange [7].

4.1.2 Exchanges and Live Trading

A cryptocurrency exchange is an online platform that allows users to trade cryptocurrencies in exchange for other cryptocurrencies or traditional fiat currencies(USD etc). As with any regular exchange, there are buyers (bidders) and sellers (askers). In order to buy a cryptocurrency, an exchange user should first create an order. An order is simply an instruction to buy or sell a certain cryptocurrency. There are numerous type of orders with varying levels of complexity. Two very commonly used types are Market and Limit orders.

A limit order allows the users to specify the highest price they're willing to pay to buy (hence the name limit). The order will only be executed if the price moves favourably (price goes down if buying). The opposite applies for selling. The user sets the lowest price they would sell for. When using a limit order, there's a chance that the order never (or only partially) gets executed. The execution depends on how favourable the market conditions are on the limit order. However, an advantage of

using this sort of order is that, it is likely that a better price could be secured than a Market order.

A Market order is designed to be fulfilled very quickly. However, as a trade off to this speed of execution, the order does not allow the user to specify the exact price at which to buy or sell. The price is determined as the average of the bid and ask price of the market. The order may not be executed at the best possible price but it will get executed quickly.

We motivate our choice of Kraken on the following basis :

- Founded in 2011, it is one of the oldest and largest cryptocurrency exchanges in terms of trade volume [34].

- It has a comprehensive well-documented API that covers a wide range of functionality from querying historical data to generating trade orders. This functionality is vital to our trading system.

- It supports 20 cryptocurrencies with low trading fees of 0.16%, if creating a new order that is not immediately fulfilled (limit order and its variations), and 0.26% if using market orders. The trading fees decrease with higher trade activity. It is possible to reach 0% trading fees with high trading volumes.

- It has the highest and second highest trading volume for trading pairs Ethereum/Euro and Bitcoin/Euro respectively [34]. This means that there's a large number of people trading cryptocurrencies with Euros. This ensures that trade orders generated by our trading system are likely to be executed quickly.

- It offers multiple methods of securing user accounts such as Two-factor Authentication and a Global Settings Lock.

Our trading system makes use of Market orders. The reason, as explained above, is the speed of execution. Although the system is able to handle situations where the trading order execution is delayed or never carried out, it is still in our best interest

for the order to execute as soon as a trading signal is generated. This is because the implemented trading strategies are dependent on trend. Slower executions in an up-ward market could be costly.

Kraken offers a REST API that accepts two types of HTTP requests namely POST and GET. There are two types of requests that can be sent to the API, private and public. Public requests can be initiated by anyone. They return publicly available data such as historic prices, server time and supported cryptocurrencies. We use public requests to retrieve latest market prices from the Kraken API and private requests are used to check balances and generate trade orders. Due to the sensitive nature of private requests, it is important that they utilize secure encryption techniques such that the content is not compromised. For example when a trade order is generated from a trading system, how can the exchange be sure that it was indeed the rightful user and not a malicious third party? In order to send private API requests to Kraken, a secret key needs to be generated. This key is then used to digitally sign the API requests. This scheme is widely used in security protocols to ensure the authenticity of the sender.

4.1.3 Implementation Details

The functionality of the trading system is captured with 4 classes. In this section we briefly describe these classes and how they interact with each other. The trading system was written in Python 3.6. It is designed as a web application. The front end was developed using HTML, CSS and Javascript. All the charts were generated using the charting library Plotly [35]. An object-oriented approach was used during the design of the the application. Below is a list of the Python classes created for the system:

app.py : The main file used to launch the application. The web app was developed using the Tornado and Flask web frameworks. It handles all the client side requests, queries the relevant classes and sends the results back to the client.

evaluator.py This class includes functionality for running simulations on data. It is heavily reliant on Pandas [36] and Numpy [37]. Pandas offers a wide range of

statistical and data analysis tools helpful for working with financial data. It offers some of the functionality of the R programming language in Python, most notably a DataFrame object. This object allows users to store indexed data in a 2D tabular format. Another notable asset of the Pandas library is that, it offers fast flexible methods for reading and writing to CSV files. Table 1 shows the running times of our

Average Evaluation Running Time in Seconds			
Strategy	1 Month	6 Months	1 Year
Cross Over Moving Average	0.0693	0.170	0.256196
Exponential Moving Average	0.0673	0.197	0.323787
Bollinger Bands	0.0378	0.099	0.149742
Relative Strength Index	0.0558	0.163422	0.294710

Table 1: Average Running Time of Evaluation for Each Strategy.

simulation algorithm on different lengths of time. This average was taken from 100 trials on a machine with 6 gigabytes of RAM and Intel core i5 processor. The input data simulated on, had a granularity of 1 minute. 1 Month of such data is around 44640 rows, 6 Months 262800 and a year's worth of data is around 525600.

trading_ bot.py : This is the class responsible for live trading. It is run in the background, when a trading bot instance is launched. It does not affect the performance of the rest of the application. Algorithm 1 demonstrates how the trading bot behaves.

Algorithm 1 Execution flow of Trading Bot

```
iteration = 1
number_of_transactions = 0
pending_order = False
while bot is running do
    Retrieve Latest Market Price
    Update Price History
    if Warm up time is over then
        if pending_order = True then
            if order is closed then
                number_of_transactions += 1
            else if order is expired then
                pending_order = False
            else
                pass
        else
            Calculate Technical Indicators
            if signal is generated then
                Create market order
                pending_order = True
    iteration += 1
end
```

Before the bot can actually begin to generate trade orders, it needs to have collected enough data to be able to able to generate trade signals. Therefore, we defined a warm up period. Naturally this warm up time is linked to the parameters of the trading strategy. For example, in the MA trading strategy the warm-up time is set to be the size time as the short window. For the other strategies, the warm up time is set to be the same as the window size.

util.py : Utility class, as the name suggests is utilized by all other classes. It contains all the methods used to communicate with API, perform dataset updates and data retrieval for evaluation.

4.2 Dataset

The quality of the data collected in any piece of scientific work is of extreme importance. Data collection and preprocessing is a vital step in any experiment and should not be taken lightly. The historic data for Bitcoin was collected from Bitcoincharts [38]. The website offers complete trade history for bitcoin for many exchanges. We chose data from Kraken for reasons we highlighted in section 4.1.2. This complete trade history comes in the form of a csv with 3 columns; *timestamp*, *price* and *volume*. Each row contains a completed trade (a fulfilled order). The data goes back to January 2014.

For the other cryptocurrencies, the data wasn't as readily available. Kraken, however, offers an API that allows the download of a limited amount of rows of trade history. To obtain the entire history, we wrote a Python script that repetitively queried the API to download the entire history.

Raw trades history still cannot be used for evaluation because the granularity is too high and/or irregular. To make the data uniform, it was resampled into 1 minute bins. The price for each bin was the average price for all the transaction that occurred in that minute, while the trade volumes were naturally aggregated. Table 2 illustrates information about the data.

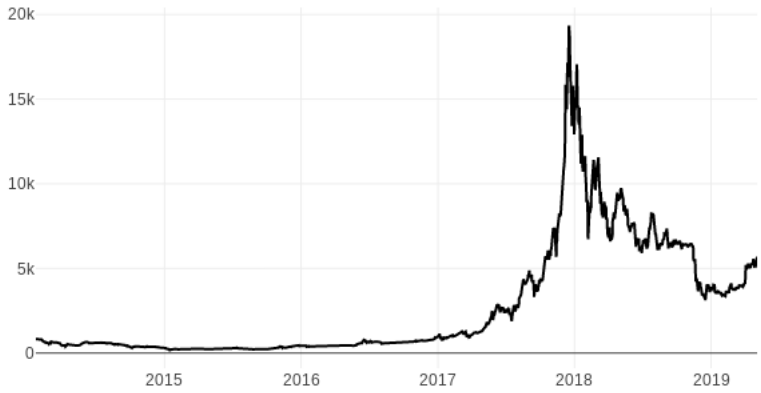
Cryptocurrency	Starting Date	Min Price (USD)	Max Price (USD)
Bitcoin	2014-01-07	175	19654.6
Ethereum	2015-08-07	0.345005	1441.39
Litecoin	2013-10-24	0.88536	367.74
Dash	2017-04-12	56.3359	1579.85
Zcash	2016-10-29	26.4332	799.4485

Table 2: Descriptive Statistics on Pre-processed Data

At some intervals, particularly in the the beginning (as the currency was first introduced to the exchange), there are extended periods of time where no transactions took place. These periods go beyond 1 minute. Naturally, after the data was processed this will lead to some missing minutes (rows). However, before the data is input into the simulation algorithm, the price is fed forward, ie. the missing rows (minutes) are

also included and their price value is taken from the latest existing minute. This is a reasonable assumption to make because if the currency was not traded for a period of time, the price will stay constant. In fact, if there are price changes, this will increase the amount of transactions as it will attract traders wanting to leverage these changes.

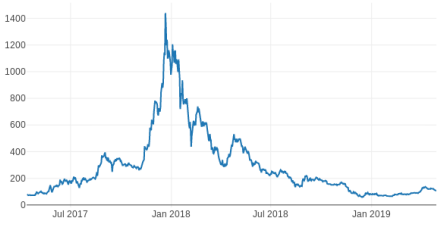
Figure 9 visually demonstrates what the processed data looks like. The data is from the starting date shown in table 2 until the end of April 2019. It is also worth mentioning that, the script for downloading the dataset was modified and offered as a feature in the trading system. This feature allows the user to update the local cryptocurrency datasets with the click of a button.



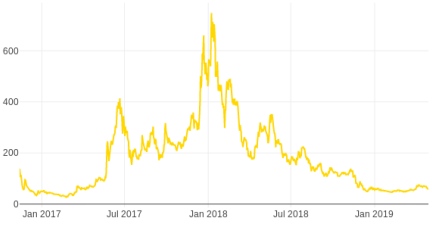
(a) Bitcoin

(b) Ethereum

(c) Litecoin



(d) Dash



(e) Zcash

Figure 9: Historical Price Data After Being Processed for All Currencies.

4.3 Evaluations

4.3.1 Grid Search Approach to Find Optimal Trading Strategy

We highlighted in section 3.1 all the trading strategies based on technical analysis that our system utilizes. These strategies are dependent on a set of parameters. These parameters define the rules that generate these signals. Variations of these strategies with different parameters are used by traders. In order to find an optimal set of parameters we opted for an exhaustive grid search approach. Grid Search is an approach typically used to find optimal hyperparameters for a Machine Learning model. All possible combinations of hyperparameters from a pre-defined space are trained with the model. The performance of the model is gauged by a pre-defined evaluation metric. The performance of the optimal hyperparameter set is typically also evaluated on a held-out set of data. This is done to avoid overfitting. Please note we were only able to run a grid search based approach due to the fast run time of our simulation algorithm. (Refer to table 1 for more details). Otherwise, a more efficient hyperparameter optimization method would have been utilized.

The choice of parameters space for our trading strategies is vital to the validity of the experiments. A problem that could arise from such a brute-force approach is data snooping. Sullivan et al [13] describes the problem as follows: "Data Snooping occurs when a given set of data is used more than once for purposes of inference or model selection. When such data reuse occurs, there is always the possibility that any satisfactory results obtained may simply be due to chance rather than to any merit inherent in the method yielding the results.". They address this issue by applying a bootstrapping-based approach called White's Reality Check [39].

The parameter space they defined is suited for low-frequency traditional stock market data. This is not suited for high frequency data. We expanded the parameter space as described in 6. As we found no similar pieces of work applied to high-frequency data. We defined this space ourselves. In order to alleviate the effects of data snooping, the parameter space was defined before the evaluation took place and has not been modified. We performed the grid search on one year of historic data with granularity of 5 mins. We divided the year into four subperiods :

Parameter	Range (In hours for time parameters)
Short Window Size	{1, 2, 3, 6, 12, 24, 36, 48, 72, 96, 120, 144, 168}
Long Window Size	{12, 24, 48, 72, 120, 168, 240, 336, 672}
Percentage Filter	{0, 0.05, 0.1, 0.5, 1, 5}
Holding Period	{0, 0.25, 0.5, 1, 1, 2, 2, 3, 4, 6, 12, 24}

Table 3: Parameter Space for Cross Over MA strategy (Both Variations)
Total Number of Combinations : 8424

Parameter	Range (In hours for time parameters)
Window Size	{1, 2, 3, 6, 12, 24, 36, 48, 72, 96, 120, 144, 168, 240, 336, 672}
Band Width	{0.5,1,1.5,2,3}
Percentage Filter	{0, 0.05, 0.1, 0.5, 1, 5}
Holding Period	{0, 0.25, 0.5, 1, 1, 2, 2, 3, 4, 6, 12, 24}

Table 4: Parameter Space for Bollinger Bands Strategy)
Total Number of Combinations : 5760

Parameter	Range (In hours for time parameters)
Window Size	{1, 2, 3, 6, 12, 24, 36, 48, 72, 96, 120, 144, 168, 240, 336, 672}
Width Offset	{10, 15, 20, 25, 30, 35, 40, 45}
Percentage Filter	{0, 0.05, 0.1, 0.5, 1, 5}
Holding Period	{0, 0.25, 0.5, 1, 1, 2, 2, 3, 4, 6, 12, 24}

Table 5: Parameter Space for Relative Strength Index Strategy)
Total Number of Combinations : 9216

Table 6: Parameter Space for Trading Strategies
Total Number of Combinations for All Trading Strategies : 23400

- **Subperiod 1** :From 2018-03-01 to 2018-05-31.
- **Subperiod 2** :From 2018-06-01 to 2018-08-31.
- **Subperiod 3** :From 2018-09-01 to 2018-11-30.
- **Subperiod 4** :From 2018-12-01 to 2018-02-28.

The evaluation metric we utilized was the return on investment(ROI)(adjusted for transaction fees) after the simulation for a subperiod was finished.

$$ROI = \frac{C_{end} - C_{start}}{C_{start}} \tag{23}$$

Where C_t is the total capital (cash + asset value) at a certain time period t. For each subperiod a grid search was performed to find the top 5 most profitable parameter combinations. These combinations were then cross validated against all the other subperiods. We defined the following criteria for then choosing promising strategies :

- A strategy (hyperparameter combination) needs to be profitable across all subperiods.
- If the first criteria is fulfilled then filter based on highest ROI.

This approach was applied to 5 different currencies for seven different types of trading strategies. The reason why it's 7 and not only 4 is because we also ran the grid search on the contrarian form of some strategies as explained in section 3.1. Figure 10 shows demonstrates this idea visually for one fold, this process is then repeated for all subperiods, except of course, the out-of-sample period.

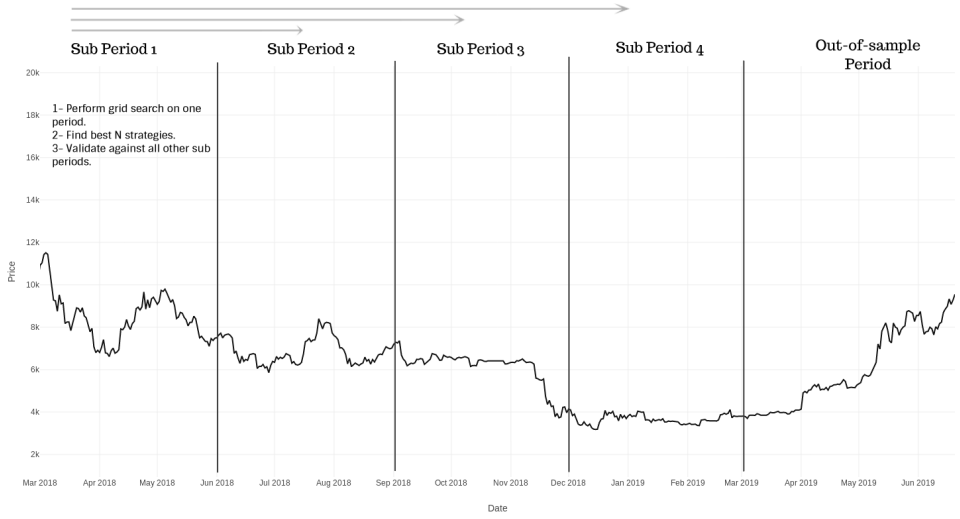


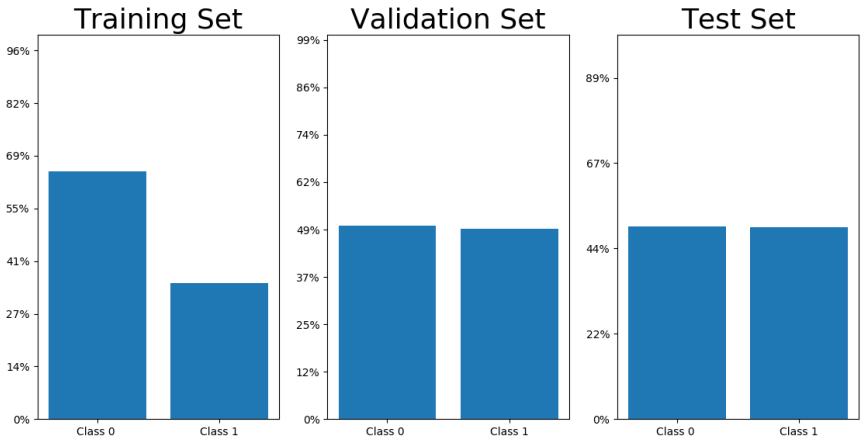
Figure 10: Cross Validated Grid Search for 1 Fold on Historic Bitcoin Price Data.

The results of this approach will be discussed in chapter 5.

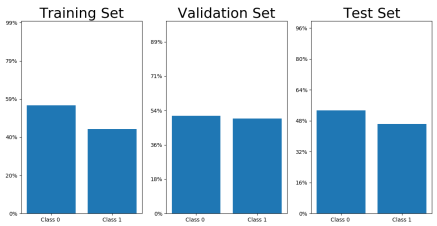
4.3.2 Learning Based Approach for Evaluating Market Predictability

Data Preprocessing

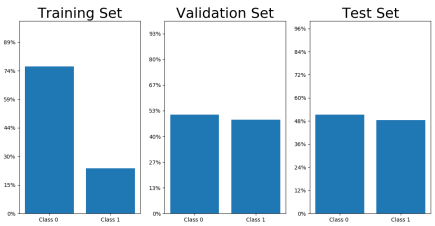
In order to assess the predictability of cryptocurrency historic prices, we defined it as a binary classification problem. If the price goes up from one time step to the next then it is of class 1, if it stays the same or goes down then it is class 0. The transformation of raw transaction data to uniform data was described in section 4.2. We considered the entire trading history of 5 cryptocurrencies from when they started trading in Kraken API until 31/05/2019. The data frequency was then decreased from 1 minute to 1 hour. This was a reasonable frequency as increasing it further would render using it in a practical scenario more difficult. Conversely, lowering the frequency would reduce the amount of data significantly to the point where it would not be sufficient for neural network training. Figure 11 shows the class distributions for the data used in our learning approach.



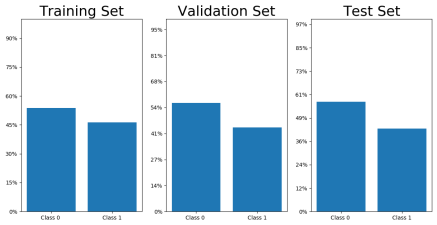
(a) Bitcoin



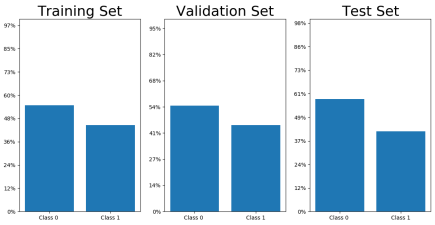
(b) Ethereum



(c) Litecoin



(d) Dash



(e) Zcash

Figure 11: Class Distributions After Data Splitting. Figure generated using Matplotlib [40]

RNN models take as input a sequence. Therefore we need to define a specific sequence length. We set this to 12 hours. As input to our network we not only considered the price but also technical indicators :

1. Short Moving Average (Window size of Sequence Length).

2. Long Moving Average (Window size of Sequence Length×2).

3. Relative Strength Index (Window size of Sequence Length) .

4. Upper and Lower Bollinger Bands (Window size of Sequence Length).

In chapter 5 we compare the performance of the network with and without these technical indicators as input.

It is standard practice to scale to the range (-1,1) the input data to ANNs and other ML models. This is called as standardization. According to LeCun et al it leads to faster convergence [41]. We standardized the data by subtracting the mean and dividing by the standard deviation as follows :

$$x' = \frac{x - \bar{x}}{\sigma} \tag{24}$$

As shown in table 2, the prices vary heavily from the earlier days of Bitcoin to its current prices. Therefore, instead of scaling down the training set using the global mean and standard deviation, we only considered mean and standard deviation within a certain window size. This window size for standardization was set to the same value as the sequence length. After preprocessing, the dataset was split into a training,validation and test set. The data was split 80:20 between train and test. Then the training set was then further split into a training and a validation set with a 80:20 ratio.

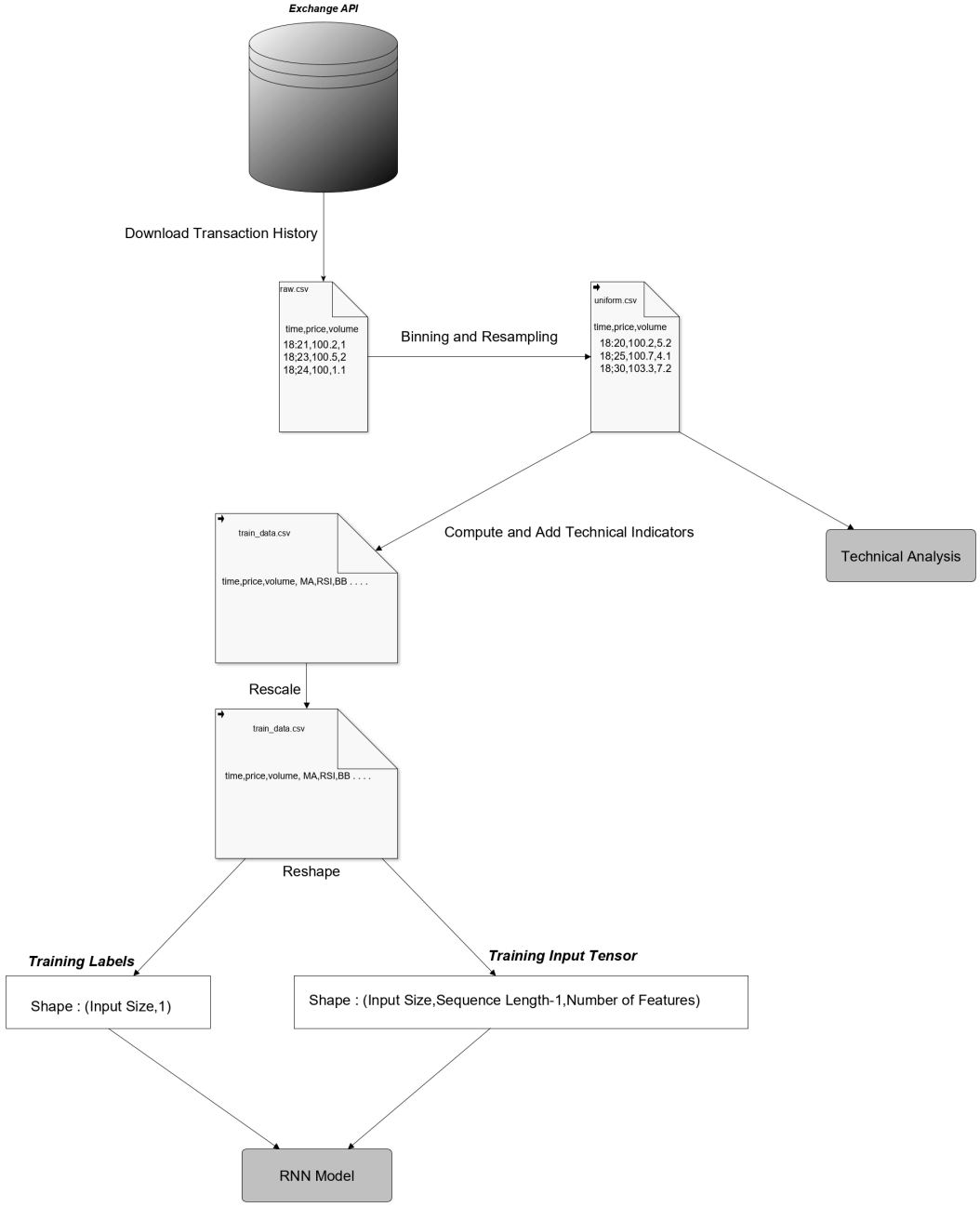


Figure 12: Data Preprocessing Steps in a Flow Diagram.

Network Architecture

We used the Keras [42] API to train and deploy the model with Tensorflow [43] as a backend. Neural Networks although extremely powerful, require a large number of hyperparameters to tune. Therefore, devising a network architecture is by no means a trivial task. Moreover, there are no clearly defined rules for computing these hyperparameters. Instead there are a group of heuristics largely dependent on trial and error. There's still a great room for optimisation and tuning to be done to our model. ANNs have shown excellent results in particular when coupled with deep learning. A network is said to be deep when there is a large number of hidden layers between the input and output network. Deeper architectures are where ANNs excel as it allows them to learn more complicated features and dependencies in the input. We opted for a stacked LSTM architecture as demonstrated in figure 13. "Stacking" LSTMs vertically is in a sense equivalent to adding more hidden layers to a traditional ANN.

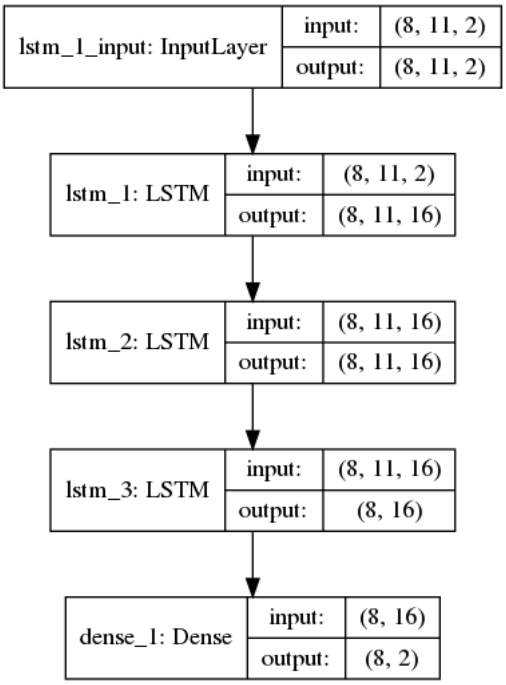


Figure 13: RNN Input and Output shapes

30 Epochs of training were used to train the network. With a tolerance of 20.

This means that the training is halted if there's no improvement in the Loss of the validation set for 20 epochs. The validation set is used here to make sure that our model doesn't overfit to the training set. Cross Entropy (Refer to equation 13) was used as a loss function, as is common in classification problems. The Adam algorithm [44] for updating weights was used instead of SGD as it performed better. A batch size of 8 was used for training. The neuron size of the LSTM cell was set to 16. These choices are based simply on trial and error. Increasing the neuron size or stacking more LSTM cells did not provide an increase in accuracy. However deeper architectures with more neurons tended to overfit to the training data after some epochs. The model learned the specific patterns special to the training set but performed poorly on the validation set. The addition of dropout [45] layers although reduced the overfitting it did not show any improvement on the validation accuracy.

4.3.3 Evaluation Metrics

This section describes the evaluation metrics we used to evaluate the predictive power of our Learning Model.

Accuracy

		Ground Truth	
		Positive	Negative
Prediction	Positive	$TruePositive$	$FalsePositive$
	Negative	$FalseNegative$	$TrueNegative$

Table 7: Confusion Matrix Layout

A confusion Matrix, as shown in table 7, will be useful for us as we describe the evaluation metrics we used. The first and simplest metric we used is Accuracy. Intuitively, accuracy is the rate of correct predictions over the total number of predictions. Using our confusion matrix we define it as :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{25}$$

One problem with Accuracy as measure of the predictive power of a model is that, it is not fair in the case of an imbalanced class distribution. If, for example, the test set has a much larger amount of one class relative to the other, and a model is biased to produce a class over the other, accuracy can be deceptively high. Thus, accuracy in this case is not indicative of the predictive power of the model on two or more classes.

Precision, Recall and F1 Score

Unlike accuracy, which does not change depending on which class is considered as positive or negative, recall and Precision differ for each class. Recall is defined as :

$$Recall = \frac{TP}{TP + FN}. \tag{26}$$

Precision is defined as :

$$Precision = \frac{TP}{TP + FP}. \tag{27}$$

F1 Score is a weighted mean of Precision and Recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{28}$$

Balanced Accuracy

Balanced Accuracy takes into account class imbalance by taking the average of the recall for each class. Balanced accuracy for our binary classification problem is formulated as :

$$BalancedAccuracy = \frac{Recall_0 + Recall_1}{2}. \tag{29}$$

Where $Recall_0$ is the recall for class 0.

Binomial Test of Significance

Hypothesis tests are statistical tests used to examine two opposite statements and determine which one is better supported by the data available. The statement which we assume to be true unless inferred otherwise, is called the Null Hypothesis. The opposing statement is called the alternative hypothesis.

In order to assess the likelihood that the null hypothesis is true, we repetitively draw samples and see how often the property that's being evaluated occurs. By drawing a large enough number of samples, it is possible to generate a probability distribution that tells us how likely a certain property occurs. Naturally the higher the number of samples drawn, the more realistic a distribution is. The number of samples drawn, is also commonly referred to as number of trials, we will refer to it as N.

Using this distribution we can obtain the likelihood of obtaining effects at least as extreme or more, given the input data. This likelihood is often referred to as p-value. If this p-value is equal to or lower than a pre-specified significance level, then the Null hypothesis can be rejected. This significance level is often set to 5% or 1% in practice. The p-value represents the likelihood of an effect occurring if the null hypothesis is true. It does not represent the probability that the null hypothesis is true.

In order to see if the LSTM classifier predictions are statistically significant, we perform a hypothesis test. We define the null hypothesis as : *the classifier is no better than a random classifier generating signals.* In our binary classification problem, a random classifier has a 50% chance of producing a correct classification. Thus we define a binomial distribution with number of trials N set to the total number of our examples in our test set and the probability of success p as 0.5 (as is the case with a random classifier). We then take the p-value associated with the number of predictions hits of the LSTM model (ie $TP + TN$) from this Binomial distribution. If the P-value is below a pre-specified confidence level, then we reject the null hypothesis and conclude that our model is better than a random classifier. In this case, the p-value tells us the likelihood that a random classifier would obtain the same number of hits on the test set.

5 Results and Conclusion

In this chapter we demonstrate and discuss the results of the experiments and evaluations applied to cryptocurrency financial data.

5.1 Grid Search Approach

The cross validation grid search approach filtered out a total of 8 candidate strategies. 2 were found for Bitcoin, 2 for Ether, 3 for Zcash and 1 for Litecoin. Table 8 defines the order of parameters used. For example MA[1h,4h,2%,0.5h] indicates a cross-over moving average with a short window of 1 hour, long window of 4 hours, 2% percentage filter and a waiting period of half an hour. Table 13 demonstrates these strategies per currency.

Strategy	Parameter 1	Parameter 2	Parameter 3	Parameter 4
MA	Short Window Size	Long Window Size	% Filter	Holding Period
EWMA	Short Window Size	Long Window Size	% Filter	Holding Period
BB	Window Size	Band Width	% Filter	Holding Period
RSI	Window Size	Width Offset	% Filter	Holding Period

Table 8: Parameters Associated with Each Trading Strategy.

The grid search resulted in no promising strategies for Dash and strategies with a very high ROI for Zcash. One explanation for this discrepancy could be that, they are some of the two least traded currencies on Kraken. This is significant because the raw transaction data that we used for our experiments was crawled from Kraken. Therefore it is highly possible that the data may not be an accurate reflection of the

Strategy	Parameters	Average ROI	Out-of-sample ROI
MA(Trend Following)	[24h, 120h, 5%, 4.5h]	7.8%	51.34%
BB(Trend Following)	[2h, 0.5, 5%, 0h]	7.1%	8.81%

Table 9: Bitcoin Candidate Strategies

Strategy	Parameters	Average ROI	Out-of-sample ROI
MA(Trend Following)	[12h, 72h, 1%, 1.5h]	16.3%	30.36%
MA(Trend Following)	[12h, 72h, 1%, 1h]	16.5%	29.69%

Table 10: Ether Candidate Strategies

Strategy	Parameters	Average ROI	Out-of-sample ROI
BB(Trend Following)	[120h, 1, 1%, 3h]	14.7%	78%

Table 11: Litecoin Candidate Strategies

Strategy	Parameters	Average ROI	Out-of-sample ROI
BB(Trend Reversing)	[672h, 1, 0.05%, 24h]	29.3%	27.2%
BB(Trend Reversing)	[336h, 2, 1%, 12h]	35.9%	6.58%
BB(Trend Reversing)	[336h, 1, 5%, 12h]	24.1%	16.68%

Table 12: ZCash Candidate Strategies

Table 13: Strategies filtered out by cross-validated gridsearch

actual trading price of an asset. The results produced for the other cryptocurrencies are more dependable. They are the most popular assets on Kraken. The statistics detailing the trading volumes of these currencies on Kraken was obtained from coinmarketcap.com [34].

Another interesting remark is that the promising strategies for Bitcoin, Ether and Litecoin, are all trend following. This indicates that it is perhaps more lucrative for traders to adapt trend following positions rather than trend reversing ones. Although, we included the RSI strategy and the exponentially weighted MA in our evaluation, no strategy passed our defined criteria. This is an indication that the simple MA and BB have a better predictive power than the other strategies.

To further evaluate the predictive power of these candidate trading strategies, they were evaluated on an out-of-sample subperiod. It was interesting for us to see how these strategies will perform on completely new data. This out-of sample-period directly followed the subperiods used for the grid search evaluation method. It started from the 1st of March and ended on the 31st of May 2019. Similar to the cross-validation folds, it is 3 months in total. The results of this test are demonstrated in table 13.

Initially, it seems that all the strategies perform exceptionally well on the out-of-sample data. However, upon a closer examination of figures 14, 15 and 16, it appears all the cryptocurrencies we considered, experience a sudden sharp increase in price. Although, this is not an uncommon occurrence in the cryptocurrency market, it does mean however, that the candidate strategies will under perform in comparison to a buy and hold strategy in this period.

(a) MA(Trend Following)[24h, 120h, 5%, 4.5h]

(b) BB(Trend Following)[2h, 0.5, 5%, 0h]

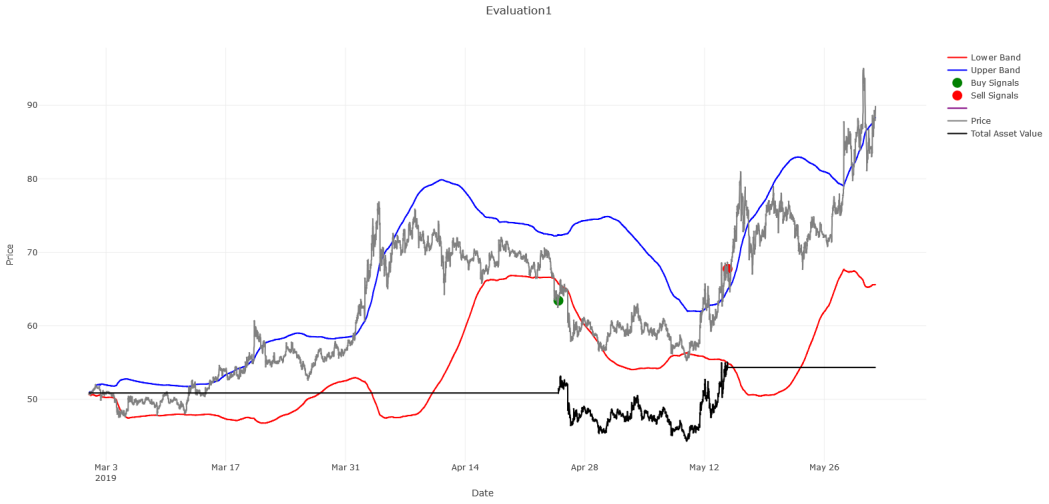
Figure 14: Performance of optimized strategies on out-of-sample Bitcoin data

(a) Ether: MA(Trend Following)[12h, 72h, 1%, 1.5h]

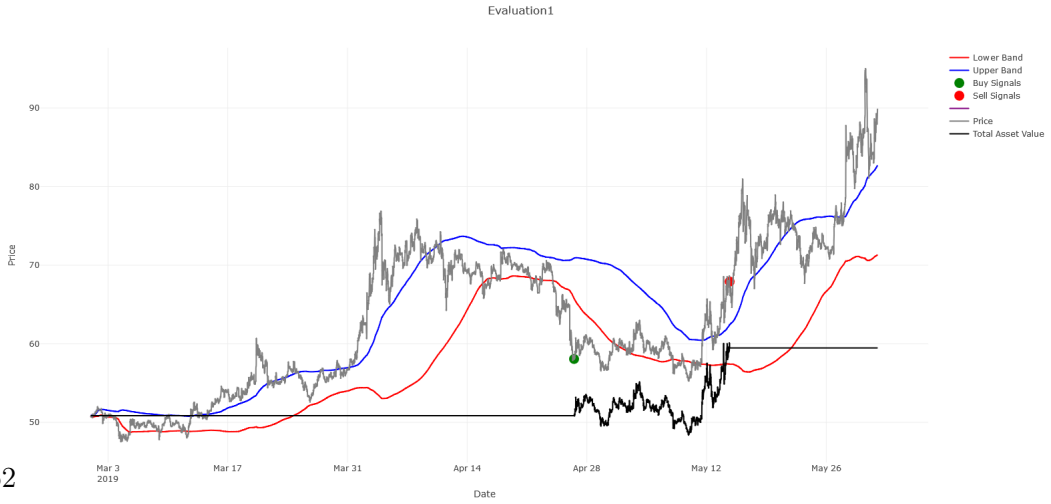
(b) Litecoin: BB(Trend Following)[120h, 1, 1%, 3h]

Figure 15: Performance of optimized strategies on out-of-sample Ether and Litecoin data

(a) BB(Trend Reversing)[672h, 1, 0.05%, 24h]



(b) BB(Trend Reversing)[336h, 2, 1%, 12h]



(c) BB(Trend Reversing)[336h, 1, 5%, 12h]

Figure 16: Performance of optimized strategies on out-of-sample Zcash data

5.1.1 Live Trading Bot

One of the goals we set initially was to develop a trading bot that's able to trade cryptocurrencies based on a pre-defined strategy. In order to ensure that our bot traded as expected, we launched it at the beginning of the out-of-sample subperiod. The trading bot was configured to trade with the strategy MA(Trend Following)[24h, 120h, 5%, 4.5h]. The reason we chose this particular strategy over others, although others were more profitable as seen in table 13, was because it was the most profitable Bitcoin strategy. Bitcoin has the advantage of being the most actively traded cryptocurrency on Kraken. Thus picking the strategy that performed well on Bitcoin, was a safer choice as it meant a higher chance of the bot orders getting executed quickly.

(a) Bitcoin

Figure 17: Performance of Trading Bot during Live Simulation
MA(Trend Following)[24h, 120h, 5%, 4.5h]

We launched the bot for around 3 months (the same size of the cross validation folds used to find the optimal strategy). Figure 17 visualizes this trading run. The data used to plot the graph was obtained from the logs produced by the bot during the simulation. This log file was then passed as input to our simulation environment. Three trades were executed by the bot during this run. By comparing the execution

Type	Trading Bot (Live)	Offline Simulation
Buy	Apr $2^{nd}18:20 CEST$	Apr $2^{nd}18:20 CEST$
Sell	May $18^{th}04:30 CEST$	May $18^{th}04:30 CEST$
Buy	May $27^{th}15:55 CEST$	May $27^{th}15:55 CEST$

Table 14: Execution Time of Trade Orders

time of all the trade orders executed by the bot to that of the trade orders produced by our simulation environment (which was ran after the bot finished trading), we found that all orders were initiated at the same time. This indicates that our simulation environment is a very good approximation of real-life automated trading. Another note-worthy observation is that, the market orders initiated by the bot were executed almost immediately. This enabled the bot to leverage the fast changing market conditions to induce a profit. The ROI for the live trading run was 47.7% (adjusted for transaction fees), the amount initially invested was 50 euros.

Please note that the slight difference between the offline simulation result and the live trading run, demonstrated in figures 14 and 17 and the different ROI, is due to the fact that our offline simulation was based on local data crawled from Kraken API in USD. Whereas our trading bot traded in Euros. Another difference is that our local datasets were processed differently, this is described in more detail in section 4.2. The bot updates its internal trade price history by taking the average of the Ask and Bid price. As this is how the price is set by the Kraken trading engine for a *market* order.

5.2 Learning Based Approach

5.2.1 Model Performance

In order to evaluate the predictive power of the LSTM model, we used multiple evaluation metrics described in section 4.3.3. We trained 2 models for each currency. The first model was trained with only the price information as input, whereas in the second model we fed in TA indicators such as MAs, RSI and Bollinger Bands as well. The reasoning behind this, is to see if including these indicators would improve the

Test Set	Accuracy	Balanced Accuracy	Avg. Precision	Avg. F1
Bitcoin	54.9%	54.9%	54.9%	54.8%
Ether	55.4%	55.2%	55.3%	55.1%
Litecoin	53.6%	53.6%	53.6%	53.6%
Dash	53.7%	50.8%	51.9%	43.7%
Zcash	54.2%	50.4%	50.8%	46.0%

Table 15: Performance of LSTM model trained with only historical prices

Test Set	Accuracy	Balanced Accuracy	Avg. Precision	Avg. F1
Bitcoin	54.3%	54.3%	54.3%	54.3%
Ether	53.6%	53.4%	53.5%	53.2%
Litecoin	53.4%	53.0%	53.4%	52.1%
Dash	51.6%	50.8%	50.9%	50.6%
Zcash	53.8%	50.7%	51.0%	48.0%

Table 16: Performance of LSTM model trained with historical prices + technical indicators

prediction accuracy of the model. Table 15 demonstrates the performance of the model trained only with price information. The precision and F1 score shown in table 15 is the average of the precision and F1 for each class (each class being considered the positive class). Balanced Accuracy is just the average recall. It is clear that the LSTM model does have limited predictive power, particularly for Bitcoin, Ether and Litecoin. The values for Accuracy and Balanced accuracy are very close. This is because the test set for these currencies are evenly distributed as shown in figure 11. The predictive power drops significantly for Dash and Zcash. This could be related to the low trading volumes of these two currencies on Kraken.

Comparing Tables 15 and 16, it is clear there's no significant improvement when adding TA indicators. In fact the balanced accuracy is lower or the same when including technical indicators as input to the network. This is by no means conclusive evidence that Technical Analysis tools have no predictive power. There could be multiple reasons for this lack of improvement. Firstly, it could be that network architecture needs to be changed to be able to learn better using multiple sequences (price + TA indicators). As previously mentioned, tuning ANNs is very hard as there's a large number of hyperparameters to be optimized. The second reason involves the hyperparameter selection for our Technical Analysis trading strategies. These include

window size, band width, holding period etc. The window size for these strategies was set 12. The holding period and percentage filter was set to 0. This setting limited the predictive power and flexibility of TA trading strategies.

Although it would be interesting to include some of the optimized trading strategies from section 5.1 as input to the network and assess the performance afterwards, there are some obstacles. Firstly the grid search to find these optimal strategy was performed on high-frequency 5 minute time interval data whereas the LSTM model was trained on 60 minute interval data.

Another aspect to consider is the signal generating frequency of the two approaches. In the grid search approach, trading fees were included in the search therefore strategies which produce many signals were penalized and had a lower adjusted return on investment. Thus, the optimized strategies tended to have stricter conditions which did not produce a large amount of signals.

The training setting for the LSTM model is, however, different. The task was to predict the market directionality, whether the prices are going up or down. Thus it is possible (although not likely), that the model would generate a signal for each time step ie., if the model prediction constantly alternates between 1 (up) and 0 (down). Therefore, it is not clear if these optimized strategies would improve the predictive power of the LSTM model.

We performed a binomial test of significance to ensure that the limited predictive power of the LSTM model is statistically significant according to a certain confidence level. The number of trials was set to the total size of the test set. Table 17 demonstrates these values. All the results are significant at the 5% significance level. At the 1%, all but one model performs significantly better than random.

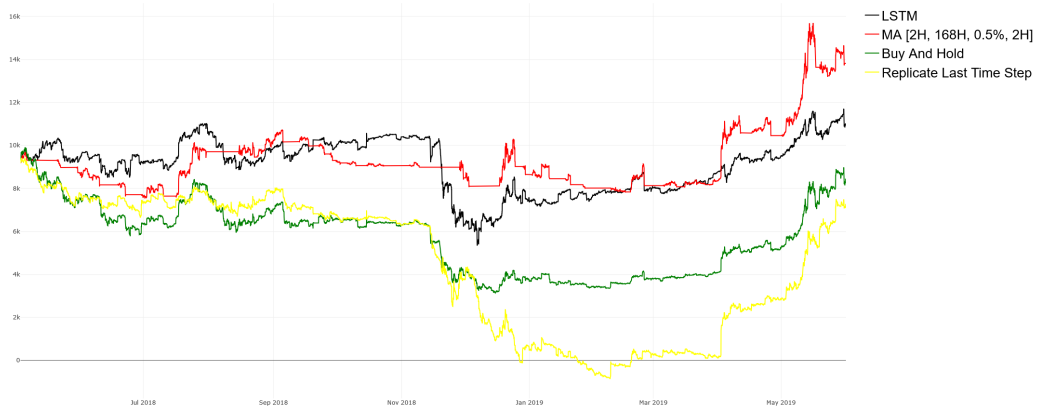
Models trained with only historical price data	**P-value**
Bitcoin	2.9e-19%
Ether	1.4e-16%
Litecoin	2.0e-9%
Dash	6.4e-04%
Zcash	1.4e-06%
Models trained with prices + technical indicators	**P-value**
Bitcoin	6.7e-15%
Ether	2.0e-07%
Litecoin	9.8e-09%
Dash	4%
Zcash	2.0e-05%

Table 17: P-values resulting from binomial test applied to LSTM predictions

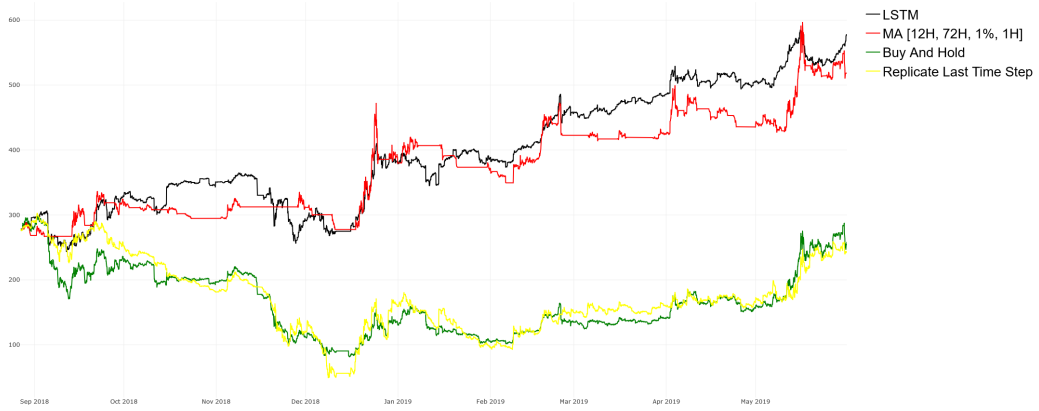
5.2.2 Trading Simulations

To further evaluate the predictive power of the LSTM model, we conducted trading simulations based on the signals generated by the model. The simulations were performed on the test set. The simulation rules are as follows, the starting capital is the price of the currency at the first time step of the evaluation. If the model predicts that the market price will go up, then a buy signal is generated and the starting capital will be used to buy the currency. If a sell signal is generated all currency is sold for the current price. We compared the LSTM model against multiple baseline strategies, the first is simply a Buy and Hold strategy, at the beginning of the simulation one unit of currency is purchased. The second is the 'Replicate Last' strategy, in this strategy, if the market price went up in the last time step then a buy signal is generated and vice versa, it is a simple trend following strategy. The third baseline is a MA strategy, where the parameters were optimised using the grid search cross validation approach described in the section 4.3.1. The cross validation period was the year previous to the beginning of the test set. 4 cross validation folds were used, where each fold is around 3 months of data. Multiple successful strategies were found for Bitcoin, Ether and Litecoin. However, none were found for Dash and Zcash.

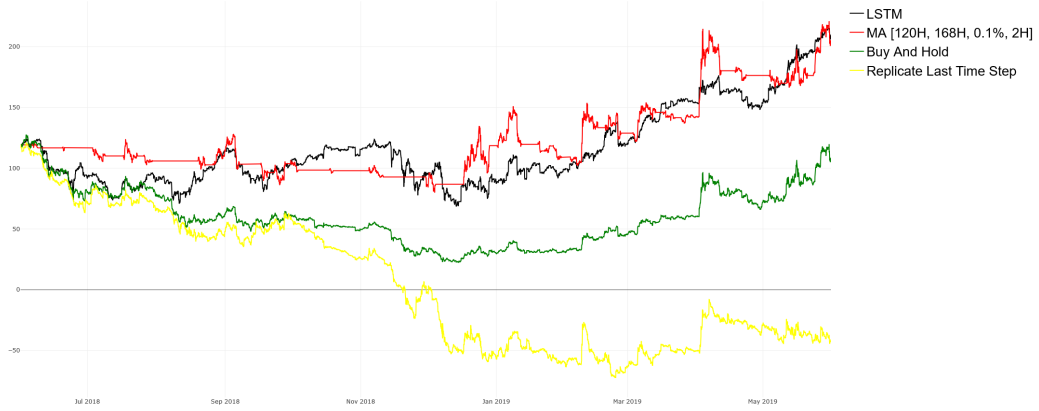
The results of these simulations are shown in figures 18 and 19. These figures demonstrate the development of the equity line for the duration of the test set. The equity line represents the total value of assets (cash and cryptocurrency value). The trades performed during this period affect the equity line. It is important to mention that the equity line does not reflect the trading fees imposed by such trades. The LSTM model was trained with the purpose of generating a prediction for each time step, this leads to a larger number of signals compared to the grid search approach which takes into account transaction fees. Naturally, this makes the grid search approach biased towards trading strategies with stricter conditions. Some cryptocurrency exchanges allow for trading without transaction fees when trading with large sums of currencies. Table 18 shows the number of trades for each simulation.



(a) Bitcoin



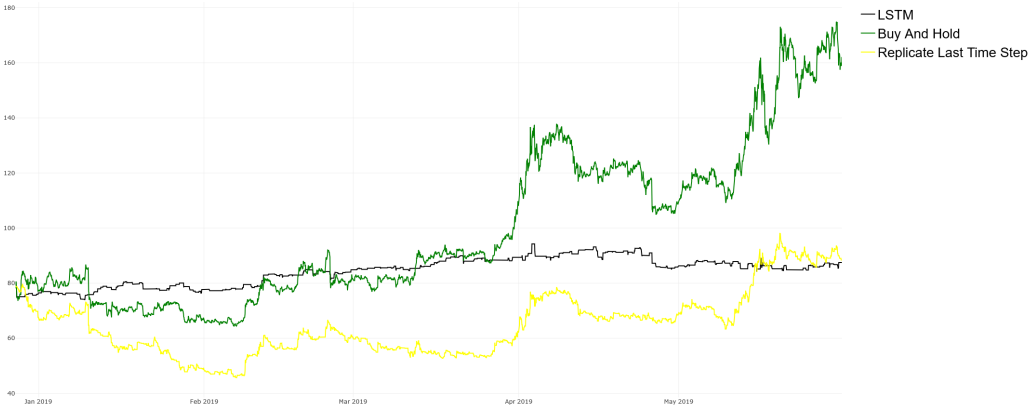
(b) Ether



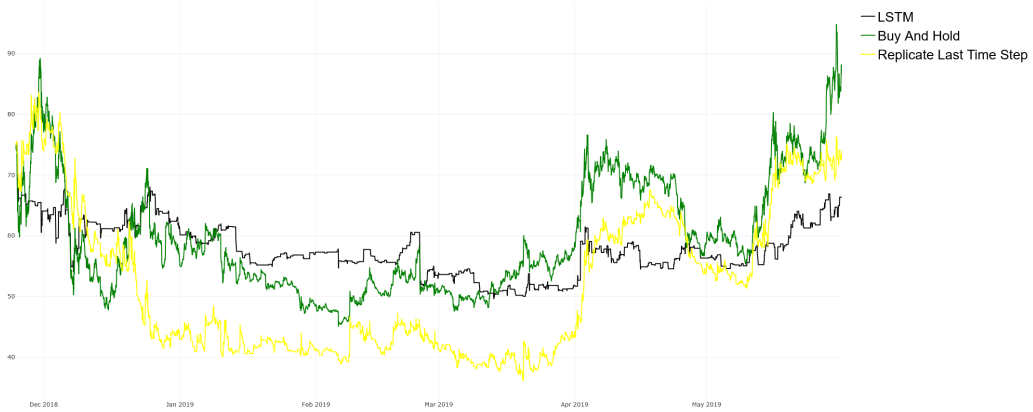
(c) Litecoin

Figure 18: Equity line of a trading simulation on Bitcoin, Ether and Litecoin test sets.

Examining figure 18 more closely, a theme present in all currencies, is the LSTM model and MA strategy outperforming the two remaining baselines. The Replicate Last strategy performs very poorly for Bitcoin and Litecoin. The equity for the Replicate Last strategy is negative at some points, the simulation environment allows for borrowing money that is not available, hence the negative equity.



(a) Dash



(b) Zcash

Figure 19: Equity line of a trading simulation on Dash and ZCash test sets.

Figure 19 shows the equity lines for the remaining two currencies we performed evaluations on, Dash and Zcash. The equity line for the LSTM model remains fairly

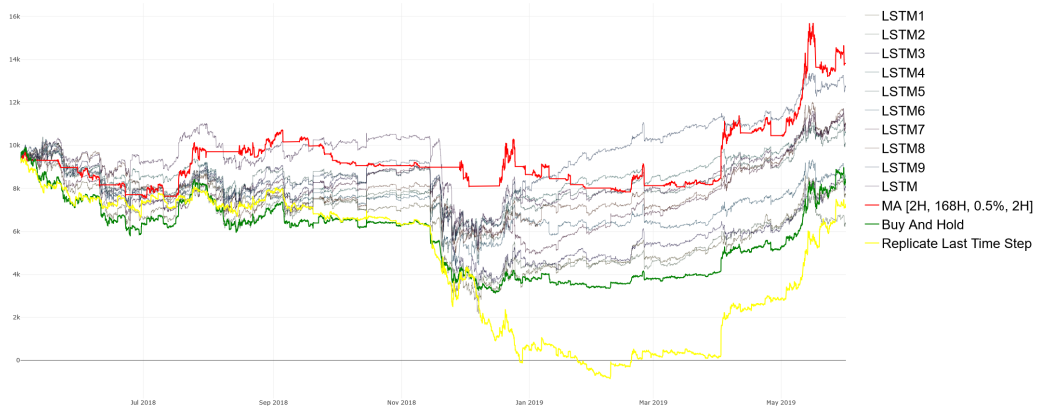
constant for Dash throughout the length of the simulation. It appears that the signals generated from the LSTM model aren't able to capitalize on the market price increases. The same can be seen for Zcash. Although the line is not as constant, it under performs compared to the baselines.

Table 15 shows that Zcash and Dash have significantly lower balanced accuracies than Bitcoin, Ether and Litecoin. This could be the reason that the equity line generated by LSTM model signals performed much better on the first Bitcoin, Dash and Litecoin in comparison to Dash and Zcash.

Currency	**LSTM**	**Moving Average**	**Buy and Hold**	**Replicate Last**
Bitcoin	3778	70	1	5157
Ether	2766	66	1	3559
Litecoin	2502	55	1	4740
Dash	528	-	1	1905
Zcash	860	-	1	2313

Table 18: Number of signals generated for each simulation

To further confirm that the results of the simulation are significant and not born by chance, 9 other LSTM models were trained on bitcoin data and the signals they produced were passed to the simulation environment. The results of this simulation are presented in figure 20.



(a) Bitcoin

Figure 20: Equity line of a trading simulation on Bitcoin with multiple LSTM models.

The balanced accuracies the for 9 trained models on the test set had a standard deviation of 0.4%.

5.3 Concluding Remarks and Future Work

Financial markets price predictions is not a trivial task. There have been numerous attempts to do so but with very limited success. This difficulty arises from the difficulty of encapsulating all market dynamics and capturing the interactions between them within one model. We employed two different techniques to perform this task on a relatively new market, the cryptocurrency market. The first, based on Technical Analysis tools, yielded good results on an out-of-sample test set. Yet it under performed compared to a Buy and Hold strategy as the market was doing well during this period. However, we were able to gauge the performance of our trading bot which was launched at the beginning of this period. We were able to conclude from the bot performance that our offline simulation environment is a good approximation to real world trading.

The second approach which we employed, based on Machine Learning, has shown limited predictive power of around 5% balanced accuracy improvement compared to a random classifier for 3 out of 5 currencies, namely Bitcoin, Ether and Litecoin. By running trade simulations on the test set, which was about a year in length. We observed that the LSTM model performed very well on the 3 currencies with limited predictive power. As for the other currencies, namely Dash and Zcash, the LSTM model under performs compared to the baselines.

We compared the LSTM model to multiple baselines, one of which was the grid search cross validation approach based on Technical Analysis. The grid search approach performed well (relative to the other baselines) on the test set. The test set was much longer than the previous out-of-sample period. Thus the good performance of the approach cannot be attributed to the a period where the market was performing particularly well.

We summarize the main contributions of this thesis as follows:

1. An application with a GUI that enables the user to perform realistic trade simulations offline and update local historical price data.

2. A trading bot that can be configured via the GUI, to trade automatically for the user in the background. Using this we were able to validate our simulation environment by comparing its performance to that of the trading bot.

3. A cross-validated grid search approach which has performed well on out-of-sample data for multiple cryptocurrencies.

4. An extensive analysis of the predictability of multiple cryptocurrencies using a learning-based approach.

At the end of this thesis, we find that there are some remaining routes which can be explored. These are:

1. Investigating the effect of financial news on cryptocurrency prices. If some

degree of correlation is present, then it would be interesting to see if including this as input to our RNN model would offer any improvement in the directional accuracy. There has been already some work in this direction for traditional stock markets. Shumaker and Chen investigate the effect of using 3 different textual analysis techniques in combination with historical market prices to predict the price 20 minutes after an article is released [46]. They achieved 57.1% directional accuracy using this approach.

2. Experiment further with different RNN architectures and to further tune the parameters of the model. There are many variations of LSTM cells, an example would be Gated Recurrent Units as proposed by Cho et al. [47]. It would be interesting to see if there is a significant improvement in accuracy with these modifications.

3. Enabling the trading bot to be able to connect to multiple cryptocurrency exchanges as opposed to just Kraken. This would allow for more freedom when it comes to the choice of exchange.

4. Enhance the bot by implementing live trading based on the signals generated by the LSTM model. So far the trading bot supports only trading strategies based on Technical Analysis. It would be interesting to see how the bot will perform in this setting on a live environment, particularly because the LSTM model tends to produce a higher number of signals than the TA approach.

Bibliography

[1] M. A. Nielsen, *Neural networks and deep learning*, vol. 25. Determination press San Francisco, CA, USA:, 2015.

[2] C. Olah, "Understanding lstm networks – colah's blog https://colah.github.io/posts/2015-08-understanding-lstms," 2015.

[3] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[4] "Latin america and turkey have the most cryptocurrency users https://news.bitcoin.com/latin-america-and-turkey-have-the-most-cryptocurrency-users-poll-shows/," Jun 2019.

[5] "How long do bitcoin transactions take? https://coincentral.com/how-long-do-bitcoin-transfers-take/," Aug 2018.

[6] D. Yermack, "Is bitcoin a real currency? an economic appraisal," in *Handbook of digital currency*, pp. 31–43, Elsevier, 2015.

[7] "Kraken cryptocurrency exchange https://www.kraken.com/."

[8] B. G. Malkiel and E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *The journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[9] M. S. Rozeff and W. R. Kinney Jr, "Capital market seasonality: The case of stock returns," *Journal of financial economics*, vol. 3, no. 4, pp. 379–402, 1976.

[10] S. Bouman and B. Jacobsen, "The halloween indicator," sell in may and go away": Another puzzle," *American Economic Review*, vol. 92, no. 5, pp. 1618–1635, 2002.

[11] B. G. Malkiel, "The efficient market hypothesis and its critics," *Journal of economic perspectives*, vol. 17, no. 1, pp. 59–82, 2003.

[12] L. Menkhoff and M. P. Taylor, "The obstinate passion of foreign exchange professionals: technical analysis," *Journal of Economic Literature*, vol. 45, no. 4, pp. 936–972, 2007.

[13] R. Sullivan, A. Timmermann, and H. White, "Data-snooping, technical trading rule performance, and the bootstrap," *The journal of Finance*, vol. 54, no. 5, pp. 1647–1691, 1999.

[14] A. Urquhart, "The inefficiency of bitcoin," *Economics Letters*, vol. 148, pp. 80–82, 2016.

[15] S. Nadarajah and J. Chu, "On the inefficiency of bitcoin," *Economics Letters*, vol. 150, pp. 6–9, 2017.

[16] I. Madan, S. Saluja, and A. Zhao, "Automated bitcoin trading via machine learning algorithms," *URL: http://cs229. stanford. edu/proj2014/Isaac% 20Madan*, vol. 20, 2015.

[17] K. Hegazy and S. Mumford, "Comparitive automated bitcoin trading strategies," *CS229 Project*, 2016.

[18] M. Chen, N. Narwal, and M. Schultz, "Predicting price changes in ethereum," *International Journal on Computer Science and Engineering (IJCSE) ISSN: 0975-3397 Vol. 7 No. 4 Apr 2015*, 2018.

[19] V. Pavlov and S. Hurn, "Testing the profitability of moving-average rules as a portfolio selection strategy," *Pacific-Basin Finance Journal*, vol. 20, no. 5,

pp. 825–842, 2012.

[20] J. Bollinger, "Using bollinger bands," *Stocks & Commodities*, vol. 10, no. 2, pp. 47–51, 1992.

[21] C. Lento, N. Gradojevic, and C. Wright, "Investment information content in bollinger bands?," *Applied Financial Economics Letters*, vol. 3, no. 4, pp. 263–267, 2007.

[22] J. W. Wilder, *New concepts in technical trading systems*. Trend Research, 1978.

[23] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[24] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[25] E. W. Weisstein, "Tree. From MathWorld—A Wolfram Web Resource." Last visited on 13/4/2012.

[26] Y. Bengio, P. Simard, P. Frasconi, *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] P. J. Werbos *et al.*, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[29] D. Britz, "Recurrent neural networks tutorial, part 3 – backpropagation through time and vanishing gradients," Apr 2016.

[30] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger,"

Ethereum project yellow paper, vol. 151, pp. 1–32, 2014.

[31] C. Lee, "Litecoin," 2011.

[32] E. Duffield and D. Diaz, "Dash: A privacycentric cryptocurrency," 2015.

[33] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," *Tech. rep. 2016–1.10. Zerocoin Electric Coin Company, Tech. Rep.*, 2016.

[34] "Cryptocurrency exchange rankings https://coinmarketcap.com/rankings/exchanges/."

[35] P. T. Inc., "Collaborative data science," 2015.

[36] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference* (S. van der Walt and J. Millman, eds.), pp. 51 – 56, 2010.

[37] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, Austin, TX, 2010.

[38] "Bitcoin charts http://api.bitcoincharts.com/v1/csv/."

[39] H. White, "A reality check for data snooping," *Econometrica*, vol. 68, no. 5, pp. 1097–1126, 2000.

[40] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[41] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48, Springer, 2012.

[42] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[43] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[46] R. P. Schumaker and H. Chen, "Textual analysis of stock market prediction using breaking financial news: The azfin text system," *ACM Transactions on Information Systems (TOIS)*, vol. 27, no. 2, p. 12, 2009.

[47] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.