POLITECNICO DI TORINO

Dipartimento di Automatica e Informatica

Master degree course in Computer Engineering (Software Engineering)

Master Degree Thesis

Formal automatic trading in the cryptocurrency era

An automatic approach to trading with cryptocurrencies

Advisor:
prof. Paolo Garza

Co-advisor:
prof. Luca Cagliero

Candidate:
Tamer Saadeh
matricola: 222201

Academic Year 2018 – 2019

Abstract

Cryptocurrency markets are highly volatile due to their lack of regulation and their always-open nature. This gives rise to the need to develop systems that can help investors limit the risks and improve the gains from the market, particularly in a downward trend. Moreover, due to the volatility of the market, it is very important for traders to act on trade signals as soon as possible, which further increases the need to an automatic system. A system was developed that analyses data from one of the biggest cryptocurrency exchanges by transactions and simulates a trader's transactions on the exchange. The system developed finds the optimal set of parameters needed to increase the return on investment for a trader.

Keywords:— cryptocurrencies, bitcoin, litecoin, etherum, finance, machine learning, data mining, trading, technical analysis, automatic trading system, financial data mining, high return investment.

Acknowledgements

I would like to acknowledge my mother, Susan Khoury, for giving me the chance to live again by donating her bone marrow to me while I was battling leukaemia in 2016. As well as the numerous times she has helped with various aspects of my life. I will be forever indebted to her kindness, caring and clear-headedness.

Second, I would like to acknowledge the doctors, nurses and staff at Città della Salute e della Scienza di Torino – Molinette for taking care and curing me during one of the worst moments. In particular I would like to extend my deepest gratitude to Dr. S. Butera and Dr. L. Brunella for the incredible care they provided me during my hospitalisation and especially during the follow up visits. Words do not do justice to how grateful I am for having them in my life and for providing me with the possibility to enjoy life again.

Last and not least, I would like to thank and acknowledge my advisors, professors P. Garza and L. Cagliero, for their time and their indispensable feedback during the development of this research.

Turin, Italy, March 31, 2019

Tamer Saadeh (tamer@saadeh.it)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Cryptocurrency markets are very volatile due to being an always-open market; as such it is very difficult for traders to follow the markets 24/7 to trade profitably. This increases the importance of an automatic trading system that can react in a timely manner to market changes and reduce any losses incurred. This became more evident after the Bitcoin (BTC) market crash in December 2017.

In this thesis, a system of simple automatic trading was developed to outperform the market; that is to have higher return on investment (ROI) with respect to a simple buy-hold-sell strategy. The system was enhanced using machine learning techniques to predict certain actions given different cryptocurrencies and market conditions.

The proposed solution consists of finding an optimal artificial intelligence classification model that is trained on a subset of the data set. Each model is trained to predict a short, a long or a hold signal class label. The training set consists of features such as the closing price as a time series or the technical indicators of the time period considered. At the end of the training, the solution predicts new signals on the test set and then simulates them. The result of the simulation is the balance of the trading account, which when compared to its initial amount, the return on investment can be computed. This result is stored for comparison with other experiments varying the model parameters and features.

The data set was classified into three main classes: long (buy), short (sell), hold (no action). Moreover, it was aggregated into larger time intervals than the base of 1-minute granularity. Then the dataset was split into two main groups: training set and test set. The training set was used for training different machine learning algorithms; such as decision tree and k-nearest neighbour. While the test set was used to perform the predictions. In addition to these parameters, the stop loss and take profit constraints were implemented to limit losses and assure a minimum gain.

Three different feature sets were considered. One set is composed of the most common technical indicators and a times series of the market volume. Another set consisted of the closing price as time series as features of the model. And finally, a set with both the closing price and volume as time series was constructed. These three feature sets were chosen to test which set would produce the highest returns for the cryptocurrency market considered. The feature set containing the technical indicators was included as it is typically used for analysing foreign exchange currencies (FOREX) and commodity futures markets. While on the other hand, the other two feature sets, consisting of the closing price and the market volume as time series to be compared to the technical analysis one in this automatic trading system. All feature sets were then compared to the simple buy-hold-sell strategy that they

exceeded.

Five main artificial learning algorithms were chosen to classify the data: k-nearest neighbour (kNN), decision tree (DT), random forest (RF), stochastic gradient descent (SGD), and passive aggressive (PA) classifiers. For each model, a subset of parameters were selected to be optimised.

A grid search was performed to evaluate the different parameters' importance. The grid explored consisted of about 70K experiments, which required that the grid be parallelised to speed up the process. The results of the grid search were analysed by choosing a model that produces the highest ROI for the different cryptocurrencies considered and for different thresholds.

1.2 Related work

Financial data mining entails two main approaches: time-series data mining and the use of technical analysis. Both approaches consider the past values of the securities analysed with the main difference being the time-series one directly using the values of the previous market price while the technical analysis one performs processing on the input data. Traders wanting to invest in cryptocurrencies should prefer investing for short periods due to cryptocurrency's high market volatility. (Abrol et al., 2016)

In Baralis et al. (2017), the authors describe a system that analyses the historical stock data to find the best stocks for intraday trading, while this is important for a trader, in this work only a single security was considered.

The technical analysis techniques that were used for this work, followed similar features described by Teixeira and de Oliveira (2010) and Chen and Hao (2017) with the parameters' inputs adjusted to the ones used to analyse futures and commodity markets, as per (Murphy, 1999, pp. 200-260). Moreover, the trading model implemented stop loss and take profit limits with the take profit an optional limit, similarly to Teixeira and de Oliveira (2010).

Many artificial intelligence models have been used to address stock financial markets such as IF-THEN rules (Dymova et al., 2016, Kim et al., 2017), artificial neural networks (Chang et al., 2009, Kara et al., 2011, Mostafa, 2010), decision trees (Chang et al., 2011, Wang and Chan, 2006), support vector machines (Żbikowski, 2015), and random forest (Laborda and Laborda, 2017, Patel et al., 2015). Moreover, some have explored the use of ensemble learning (Kwon and Moon, 2007, Pulido et al., 2014, Tsai et al., 2011), however, they mostly focused on neural networks or genetic algorithms. In this work other ensemble methods have been explored, such as the stochastic gradient descent and the passive aggressive algorithms.

While these research papers are able to obtain high return in the stock markets using technical indicators, this research considers both time-series and technical analysis to address the cryptocurrency markets, which behave differently from stock or forex markets. Also apart from Dymova et al., few others have considered the currency markets (FOREX).

Chapter 2

Data model

In order to implement the system, the input data set must be processed first to obtain the output necessary for our results; specifically, the return on investment. In this chapter the assumptions, limitations and the processing of the input data are described. Initially the raw data must be discretised into the three classes that the machine learning algorithms will use for the training and prediction. Moreover, these models are combined with different features that are better described in section 2.4.

2.1 Data collection

Datasets are collected using Coinbase's API in JSON and stored in comma-separated value (CSV) file format. The fields mentioned in section 2.1.2 are divided into columns and each minute of granularity represents a row. When data aggregation is necessary, like in section 2.1.4, they are performed on-the-fly and as such they are not stored in files. This allows for multiple different granularities to be used in parallel without the need to precompute these combinations in advance.

2.1.1 Inputs

This work analyses the cryptocurrencies Bitcoin (BTC), Litecoin (LTC), Etherum (ETH) and Bitcoin Cash (BCH) paired with the United States Dollar (USD) and the Euro fiat currencies.

2.1.2 Data sources

Data was collected from Coinbase Pro's (formally GDAX) servers using only historic data with minimum granularity of 1 minute. The attributes collected:

- time
- close price
- open price
- volume
- high price
- low price

An example input data set looks like:

	time	low	high	open	close	volume
1	2018-03-20 12:36:00	6935.02	6935.03	6935.03	6935.03	1.67

	time	low	high	open	close	volume
2	2018-03-20 12:37:00	6935.03	6935.03	6935.03	6935.03	0.27
3	2018-03-20 12:38:00	6935.02	6935.03	6935.02	6935.03	0.30
4	2018-03-20 12:39:00	6935.03	6935.04	6935.03	6935.04	0.05
5	2018-03-20 12:40:00	6935.04	6939.19	6935.04	6939.19	0.47
6	2018-03-20 12:41:00	6939.37	6939.38	6939.37	6939.38	0.68
7	2018-03-20 12:42:00	6939.37	6939.38	6939.38	6939.38	0.61
8	2018-03-20 12:43:00	6939.38	6945.00	6939.38	6941.07	1.41
9	2018-03-20 12:44:00	6941.07	6946.60	6941.07	6944.97	4.01
10	2018-03-20 12:45:00	6944.89	6948.00	6944.89	6948.00	2.03

Table 2.1: Example input data set

2.1.3 Data updating and cleaning

Coinbase's API does not provide data for time periods when no trade occurs on the exchange (Coinbase will return an empty data set). These missing timestamps from the data set are addressed by inserting them with not a number (NaNs) values. This ensures that when aggregating the data sets the computations are correctly processed.

Moreover, in order to use recent data from Coinbase, it was necessary to update the data files every so often. This ensures that the data that has already been collected is up to date, but also that this data does not contain missing data points from network issues or temporary API failures.

2.1.4 Aggregation

Data is grouped by a given granularity g by performing the following actions on the collected attribute:

- time of the granularity: first time value in group
- close price: last closing price in group
- open price: first opening price in group
- volume: sum of all market volume values in group
- high price: maximum of all high prices in group
- low price: minimum of all low prices in group

An example of the aggregated data set with granularity set to 15 minutes for the pair BTC-EUR looks like:

	time	low	high	open	close	volume
1	2018-03-20 12:00:00	6902.39	6931.17	6931.17	6920.58	22.44
2	2018-03-20 12:15:00	6920.57	6928.38	6920.57	6924.21	11.67
3	2018-03-20 12:30:00	6924.21	6946.60	6924.21	6944.97	17.41
4	2018-03-20 12:45:00	6944.89	7015.40	6944.89	6970.35	127.61
5	2018-03-20 13:00:00	6960.47	6993.10	6970.01	6985.46	48.93
6	2018-03-20 13:15:00	6931.28	6985.47	6985.47	6932.00	81.01

	time	low	high	open	close	volume
7	2018-03-20 13:30:00	6931.38	6968.22	6932.00	6964.37	28.05
8	2018-03-20 13:45:00	6910.00	6964.36	6964.36	6910.44	27.79
9	2018-03-20 14:00:00	6920.00	6945.99	6921.86	6943.50	15.66
10	2018-03-20 14:15:00	6913.99	6943.00	6943.00	6931.79	17.30

Table 2.2: Example aggregated data set

2.2 Limitations and training and test sets

To simplify the model and taking into account the limitations of trading on the Coinbase market, some considerations must be taken to account. The limits are taken into account when preprocessing the signals for the simulation in section 3.2.

Trade size limitation The trade size of each action is set to 0.1 of the cryptocurrency considered. This is a simplification as Coinbase does not provide historic data on the order book and as such we cannot consider the spread for generating an appropriate trade size.

Volume limit protections In order to minimise the impact of the system on the market, the system checks whether the transaction to be performed exceeds 10% of the current market volume. This is required to reduce the possible effects of having a high trade volume on the price of the market; in particular this is relevant when predicting future transactions that should not affect the market. Otherwise evaluating the return on investment and the computation of the ROI of the simple buy-hold-sell strategy becomes near impossible to evaluate with low margins of error.

Other assumptions and simplifications The system assumes that the market allows margin trading; that is having long and short contracts are possible. This assumption was taken as this would increase the amount of signals generated by the system. Another reason is due to some exchanges applying to allow for margin trading with regulators, however, none have been successful so far. (Loder, 2018)

Training and testing sets split The training set is chosen by considering the first 80% of the entire dataset with the remaining 20% being used for the prediction.

2.3 Discretisation

To classify data and generate the labels needed for the classification machine learning algorithms, first the percentage change of the closing price is computed to determine the immediate trend of the market. This was calculated using the following formula:

$$perc_{f,\ period}(t) = percent\_change_{f,period}(t) = \frac{f_{t+period} - close_t}{close_{t+period}} \tag{2.1}$$

With the *period* is set to $h \cdot g$, where h is the horizon in multiples of the granularity (g) and the time t at which this formula is computed. In other words the percentage change is computed over the last $h \cdot g$ rows of the input data set. In order to generate the classes for discretisation, two thresholds are necessary; specifically *long_threshold*

and *short_threshold*. Together they are used to assign values above and below them, respectively, to discretise the percentage change value generating the two classes for “long" and “short" signals. While the values of $perc(t)$ not meeting the thresholds are given the “hold" signal class. Formally this is represented by the following computation:

$$action(t) = \begin{cases} \text{perc(t)} \leq \text{short\_threshold} & short \\ \text{perc(t)} \geq \text{long\_threshold} & long \\ \text{else} & hold \end{cases} \tag{2.2}$$

$action(t)$ is the label used for training and prediction for the machine learning algorithms.

2.4 Features

2.4.1 Base features

This consists of the open, close, low, high and volume at $t = -h \cdot g$. That is the values at the previous horizon.

2.4.2 Time-series based features

These features are the close and/or volume at every granularity set between $t = 0$ to $t = hist$ of every training datum. For example, for $hist = 5$ the following features would be considered for the close:

- $close(t)$
- $close(t + g)$
- $close(t + 2 \cdot g)$
- $close(t + 3 \cdot g)$
- $close(t + 4 \cdot g)$
- $close(t + 5 \cdot g) = close(t + hist \cdot g)$

That is to say that for a given t and $hist$ we consider the time-series leading to $t + hist$. This can be similarly performed for the volume.

2.4.3 Technical analysis features

Technical indicators are based on the hypotheses that “anything that could possibly affect the price [of a security] is actually reflected in the price of that market," that the price follows trends, and that previous trends would lead to similar future trends. (Murphy, pp. 2-4) As such by analysing and grouping the price of a security, one could study the market direction; that is when the market is trending upwards or downwards where traders should long or short a security, respectively. These indicators are split into two main categories: trend following and oscillators. Trend following indicators are tools that present a trend after it has been established in the market, while oscillators help address situations where the market prices oscillate without any significant trend.

2.4.4 Simple moving average (SMA) features

The simple moving average follows the trend of a market. This is useful as an indicator to be able to identify significant upward or downward market trends. The simple moving

average can be computed:

$$SMA_f(period) = \frac{\sum_{i=0}^{period} f(i)}{period + 1} \tag{2.3}$$

with *period* being a period over which the moving average is computed. A similar computation can be performed for the volume as well.

2.4.5 Exponentially weighted moving average (EMA) features

Exponentially weighted moving average is another trend-following indicator similar to SMA that addresses the main pitfalls of an SMA; namely it does not highlight the importance of recent price changes. Formally, this is calculated by the following formula, which is based on pandas' EWM objects:

$$\begin{gathered} EWM_f(period) = \frac{\sum_{i=0}^{period} w(i) \cdot f(period - i)}{\sum_{i=0}^{period} w_{period}(i)} \\ with \\ w_{period}(i) = \left(1 - \frac{2}{period + 1}\right)^i \end{gathered} \tag{2.4}$$

Where *period* represents the period over which the mean is computed.

2.4.6 Momentum features

The momentum indicator is an oscillator that measures the speed of price changes in the market. This is useful in identifying overbought and oversold conditions. Formally, this can be calculated as follows:

$$momentum_{f,\ period}(t) = f(t) - f(t - period) \tag{2.5}$$

Where *period* represents the period over which the momentum is computed.

2.4.7 Rate of change (ROC) features

The rate of change is similar to the momentum, however, it normalises the value between 0 and 1. This was implemented using the following:

$$ROC_{f,\ period}(t) = \frac{f(t) - f(t - period)}{f(t - period)} \tag{2.6}$$

With *period* representing the period over which the ROC is computed.

2.4.8 Relative strength index (RSI) features

The relative strength index is another oscillator similar to the momentum and ROC but addresses the issues of those indicators being sensitive to sharp changes in the market. Specifically, it counts the average number of days the market price increased and decreased and then normalises the value between 0 and 100. Formally, this is computed as follows:

$$\begin{gathered} RSI_{f,\ period}(t) = 100 - \frac{100}{1 + RS_{f,\ period}(t)} \\ with \\ RS_{f,\ period}(t) = \frac{EWM_{f_{up}}(period)}{EWM_{f_{down}}(period)} \end{gathered} \tag{2.7}$$

f_{up} represents the number of times the function f increased, with respect to its previous value while f_{down} represents the number of times the f decreased. This can be formally specified as follows:

$$f_{up} = \sum_{t=0}^{period} \begin{pmatrix} \{f(t) - f(t-1) > 0 & 1 \\ \{f(t) - f(t-1) < 0 & 0 \end{pmatrix}$$
$$and \tag{2.8}$$
$$f_{down} = \sum_{t=0}^{period} \begin{pmatrix} \{f(t) - f(t-1) < 0 & 1 \\ \{f(t) - f(t-1) > 0 & 0 \end{pmatrix}$$

2.4.9 Moving average convergence/divergence (MACD) features

The MACD indicator is an oscillator that combines the difference between two moving averages: a slow and a fast one, the moving average of that difference, and a histogram to help with identifying the divergence between the previous two. This indicator is useful in identifying market conditions that are about to change. For example when the price is increasing but the MACD line is decreasing (i.e overbought conditions), the market conditions will soon reverse (i.e. price will drop). These three sub-features are formally described below:

MACD

$$MACD(period_{fast},\ period_{slow}) = EWM_{close}(period_{fast}) - EWM_{close}(period_{slow}) \tag{2.9}$$

The $period_{fast}$ is typically 12 periods while the $period_{slow}$ is typically 26 periods.

MACD signal line

$$MACD_{signal}(period_{smoothing}) = EWM_{MACD}(period_{smoothing}) \tag{2.10}$$

Here the $period_{smoothing}$ is usually 9 periods of the MACD value computed previously.

MACD histogram

$$MACD_{histogram}(period_{fast},\ period_{slow},\ period_{smoothing}) = MACD(period_{fast},\ period_{slow}) - MACD_{signal}(period_{smoothing}) \tag{2.11}$$

2.4.10 Feature set selection

In order to evaluate which set of features are better at generating higher returns on investment for the system designed, a combination of the features described earlier were selected to be confronted with the tested models. Three distinct combinations were chosen.

Feature set 2 The parameters of the technical indicators were chosen similar to the ones commonly used by traders when analysing future markets, as detailed by Murphy, ch. 9-10. A sample of these computed features can be seen in table 2.3. Specifically, this consists of the base features (see section 2.4.1), volume as time-series, and the following technical analysis features:

- SMA (see section 2.4.4)
  - SMA_{close}(period=5)
  - SMA_{close}(period=10)
  - SMA_{close}(period=20)
- EMA (see section 2.4.5)
  - EWM_{close}(period=5)
  - EWM_{close}(period=10)
  - EWM_{close}(period=20)
- ROC (see section 2.4.7)
  - ROC_{close}(period=10)
- Momentum (see section 2.4.6)
  - $momentum_{close}$(period=10)
- RSI (see section 2.4.8)
  - RSI_{close}(period=7)
  - RSI_{close}(period=9)
  - RSI_{close}(period=14)
- MACD (see section 2.4.9)
  - $MACD(period_{short}$=12, $period_{long}$=26, $period_{smoothing}$=9)

Table 2.3: Example feature set 2

features	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
open	11 983.38	11 958.79	11 818.18	11 808.04	11 842.86
close	11 958.79	11 820.00	11 808.07	11 842.86	11 802.53
low	11 938.61	11 802.99	11 803.09	11 808.04	11 802.00
high	12 000.00	11 959.68	11 820.00	11 858.00	11 882.96
volume	9.57	14.59	11.69	11.06	10.73
close-5g-sma	12 003.61	11 952.76	11 908.58	11 882.61	11 846.45
close-10g-sma	12 025.05	12 016.23	11 990.50	11 965.78	11 937.60
close-20g-sma	11 945.41	11 941.68	11 943.98	11 950.15	11 949.35
close-5g-ema	11 992.73	11 935.15	11 892.79	11 876.15	11 851.61
close-10g-ema	11 993.78	11 962.15	11 934.12	11 917.52	11 896.60
close-20g-ema	11 962.32	11 948.35	11 934.61	11 925.65	11 913.66
close-7g-rsi	27.80	10.75	10.04	28.34	21.56
close-9g-rsi	34.96	16.49	15.60	29.42	23.78
close-14g-rsi	45.01	28.06	27.05	34.93	30.52
close-10g-roc	0.00	−0.02	−0.02	−0.02	−0.02
close-10g-momentum	15.87	−88.26	−257.30	−247.13	−281.85
close-macd-12g-26g	39.74	24.02	10.55	2.51	−6.86
close-macd-sign-12g-26g	48.90	43.33	36.15	28.93	21.36
close-macd-diff-12g-26g	−9.16	−19.31	−25.61	−26.42	−28.21
volume+1	14.59	11.69	11.06	10.73	14.41
volume+2	11.69	11.06	10.73	14.41	14.76
volume+3	11.06	10.73	14.41	14.76	16.03
volume+4	10.73	14.41	14.76	16.03	17.73
volume+5	14.41	14.76	16.03	17.73	11.52
volume+6	14.76	16.03	17.73	11.52	19.22
volume+7	16.03	17.73	11.52	19.22	56.50
volume+8	17.73	11.52	19.22	56.50	38.45
volume+9	11.52	19.22	56.50	38.45	30.21
volume+10	19.22	56.50	38.45	30.21	17.59
volume+11	56.50	38.45	30.21	17.59	16.58

Table 2.3: Example feature set 2 (continued)

features	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
volume+12	38.45	30.21	17.59	16.58	17.28
volume+13	30.21	17.59	16.58	17.28	15.98
volume+14	17.59	16.58	17.28	15.98	15.30
volume+15	16.58	17.28	15.98	15.30	27.63

Feature set 7 This consists of the base features (see section 2.4.1) and the closing price as a time-series (see section 2.4.2). A sample of these features can be seen in table 2.4.

Table 2.4: Example feature set 7

features	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
open	12 139.01	12 052.14	11 934.99	11 899.41	11 812.00
close	12 052.15	11 935.00	11 899.42	11 812.00	11 755.31
low	12 051.33	11 900.56	11 888.01	11 805.23	11 725.81
high	12 139.01	12 052.14	11 940.00	11 945.00	11 855.35
volume	17.62	30.14	12.14	26.24	20.40
close+1	11 935.00	11 899.42	11 812.00	11 755.31	11 841.48
close+2	11 899.42	11 812.00	11 755.31	11 841.48	11 653.00
close+3	11 812.00	11 755.31	11 841.48	11 653.00	11 611.22
close+4	11 755.31	11 841.48	11 653.00	11 611.22	11 734.66
close+5	11 841.48	11 653.00	11 611.22	11 734.66	11 705.08
close+6	11 653.00	11 611.22	11 734.66	11 705.08	11 705.24
close+7	11 611.22	11 734.66	11 705.08	11 705.24	11 725.12
close+8	11 734.66	11 705.08	11 705.24	11 725.12	11 879.68
close+9	11 705.08	11 705.24	11 725.12	11 879.68	11 976.22
close+10	11 705.24	11 725.12	11 879.68	11 976.22	11 894.64
close+11	11 725.12	11 879.68	11 976.22	11 894.64	11 762.14
close+12	11 879.68	11 976.22	11 894.64	11 762.14	11 719.46
close+13	11 976.22	11 894.64	11 762.14	11 719.46	11 818.49
close+14	11 894.64	11 762.14	11 719.46	11 818.49	11 886.97
close+15	11 762.14	11 719.46	11 818.49	11 886.97	11 910.13

Feature set 8 This consists of the base features (see section 2.4.1), the closing price and the volume as time-series (see section 2.4.2). A sample of these features can be seen in table 2.5.

Table 2.5: Example feature set 8

features	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
open	12 139.01	12 052.14	11 934.99	11 899.41	11 812.00
close	12 052.15	11 935.00	11 899.42	11 812.00	11 755.31
low	12 051.33	11 900.56	11 888.01	11 805.23	11 725.81
high	12 139.01	12 052.14	11 940.00	11 945.00	11 855.35

Table 2.5: Example feature set 8 (continued)

features	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
volume	17.62	30.14	12.14	26.24	20.40
close+1	11 935.00	11 899.42	11 812.00	11 755.31	11 841.48
close+2	11 899.42	11 812.00	11 755.31	11 841.48	11 653.00
close+3	11 812.00	11 755.31	11 841.48	11 653.00	11 611.22
close+4	11 755.31	11 841.48	11 653.00	11 611.22	11 734.66
close+5	11 841.48	11 653.00	11 611.22	11 734.66	11 705.08
close+6	11 653.00	11 611.22	11 734.66	11 705.08	11 705.24
close+7	11 611.22	11 734.66	11 705.08	11 705.24	11 725.12
close+8	11 734.66	11 705.08	11 705.24	11 725.12	11 879.68
close+9	11 705.08	11 705.24	11 725.12	11 879.68	11 976.22
close+10	11 705.24	11 725.12	11 879.68	11 976.22	11 894.64
close+11	11 725.12	11 879.68	11 976.22	11 894.64	11 762.14
close+12	11 879.68	11 976.22	11 894.64	11 762.14	11 719.46
close+13	11 976.22	11 894.64	11 762.14	11 719.46	11 818.49
close+14	11 894.64	11 762.14	11 719.46	11 818.49	11 886.97
close+15	11 762.14	11 719.46	11 818.49	11 886.97	11 910.13
volume+1	30.14	12.14	26.24	20.40	32.31
volume+2	12.14	26.24	20.40	32.31	24.95
volume+3	26.24	20.40	32.31	24.95	14.96
volume+4	20.40	32.31	24.95	14.96	11.95
volume+5	32.31	24.95	14.96	11.95	12.98
volume+6	24.95	14.96	11.95	12.98	26.59
volume+7	14.96	11.95	12.98	26.59	10.42
volume+8	11.95	12.98	26.59	10.42	11.81
volume+9	12.98	26.59	10.42	11.81	13.09
volume+10	26.59	10.42	11.81	13.09	11.62
volume+11	10.42	11.81	13.09	11.62	6.53
volume+12	11.81	13.09	11.62	6.53	4.04
volume+13	13.09	11.62	6.53	4.04	1.46
volume+14	11.62	6.53	4.04	1.46	4.61
volume+15	6.53	4.04	1.46	4.61	5.70

Chapter 3

Portfolio simulation algorithm

After generating the test set using the data model described in chapter 2, the signals produced by each machine learning algorithms are simulated by acting on every signal; that is a short signal generates a short transaction, and for a long signal generates a long transaction. Before simulation can occur, however, some preprocessing of the raw signals must be performed to ensure the limits discussed previously in section 2.2.

3.1 Background

3.1.1 Implementation and technologies

Programming language choice and libraries Python was chosen due to being a powerful programming language that is enriched with the multitude of libraries to support different aspects and necessities. Such as Pandas, Numpy, and SciKit. Moreover multi-threading can be easily leveraged due to Python's convenient APIs. Each simulation was run in parallel using different threads, which have their results stored in the database.

Database For storing the results of each experiment a SQLite database was used to handle the locking, specifically for some of its ACID (Atomicity, Consistency, Isolation, Durability) properties, and its portability so that the grid search can be run remotely.

3.1.2 Machine learning algorithms

Five main machine learning algorithms where considered. These algorithms' parameters were varied and is shown in the grid parameters list, in section 4.1.2.

K-Nearest Neighbour (kNN) is a pattern recognition classification algorithm which limits the classification with a rejection factor k. It is useful when a data point is within the range of two or more different classes; including this rejection factor will limit the possible classes the label can take. (Hellman, 1970) The k value is the only parameter that was varied in the grid search performed, the values tested are presented in section 4.1.2.

Decision Tree is a classifier that represents a set of actions with their results into a tree-like structure; taking into account random events, cost and utility. They provide a set of rules that can clearly represent the data set. The main metrics parameters (defined by the criterion parameter) tested were the gini impurity and the information gain (i.e. entropy).

The gini impurity uses the Classification and Regression Trees (CART) algorithm (Breiman et al., 1984), while the information gain is used by the ID4.5 algorithm. (Quinlan, 1992)

Random Forest are an ensemble method for classification that improves upon decision trees' overfitting issue by randomly setting the predictors of each tree of the forest. (Breiman, 2001) The implementation used, diverges from the original publication by averaging the probabilistic predictions of each class [21]. The estimators parameter represents the number of trees in the forest, which was varied as shown in section 4.1.2.

Stochastic Gradient Descent (SGD) is an iterative algorithm for optimising a model with each iteration improving on the previously obtained result. Three main types of models were tested: linear models (i.e. support vector machines; "hinge" loss function), logistic regression (i.e. "log" loss function), and smoothed logistic regression (i.e. "modified_huber" loss function). (Zhang, 2004) In addition to the base model, the regularisation multiple "alpha" that is used to update the learning rate of each iteration was varied as described in section 4.1.2.

Passive Aggressive it is an iterative linear algorithm similar to the linear SGD, however it is faster to train with a regularisation parameter C and the stopping criterion (i.e. tolerance), which were varied as per section 4.1.2. Moreover, the loss function used is described by the PA-I algorithm. (Crammer et al., 2006)

3.1.3 Challenges with using Coinbase

Multiple issues were faced with the Coinbase ecosystem while developing this work. Mainly the API to retrieve the raw data sets was not reliable, for example it did not always behave as described in the documentation. Also, it changed multiple times without much advance notice. This was compounded with the rebranding of the exchange from GDAX to Coinbase Pro, where the original API endpoints stopped working soon after the announcement. Another issue faced was the unclear method used by Coinbase to evaluate commissions due. For example, initially the commissions were based on whether the transaction goes on the order book or whether it was executed directly on the exchange. This was also modified during the research to a fixed value, which helped simplify the model for simulation.

3.2 Preprocessing

Before simulating the transactions in the test set, some checks and protections must be employed. Particularly, the balance and volume limits must not be exceeded. In addition a stop loss should be applied to make sure that the system does not continue trading when a certain transaction loses more than a certain limit. For example, if the value of a "long" transaction goes below a certain threshold then this transaction must be closed as to minimise risks. Optionally, a take profit criterion can be defined as being the complement of the stop loss; that is transactions will be closed to guarantee a minimum gain on a transaction.

3.2.1 Balance and volume checks

For every step of the simulation the balance is verified to be above zero; if less or equal to zero the signal is replaced with a "hold" signal as this cannot be simulated (i.e. no

trade occurs). Moreover, when the *trade_size* does not meet the volume limits discussed in section 2.2 for “short" and “long" signals are replaced with “hold" signals.

3.2.2 Stop loss criterion

This is needed to prevent the system from keeping a transaction that has lost a significant amount of its value.

Long signals Inject a “short” signal into the list of actions if the following condition is met:

$$close(t) \leq stop\_loss \cdot (1 - close(t)) \tag{3.1}$$

That is when the $close(t)$ value goes below the stop loss (as a percentage) of the closing price of the long position we generate a short signal.

Short signals Inject a “long” signal into the list of actions if the following condition is met:

$$close(t) \geq stop\_loss \cdot (1 + close(t)) \tag{3.2}$$

That is when the $close(t)$ value goes above the stop loss (as a percentage) of the closing price of the short position we generate a long signal.

3.2.3 Take profit criterion

As a complement to the stop loss criterion discussed earlier in section 3.2.2 with the main difference being that this criterion assures a minimum gain on a transaction as a precautionary measure. For example, the system might be too slow in acting on transactions and as such it is opportune to protect every transaction when trading in a volatile market.

Long signals Inject a “short” signal into the list of actions if the following condition is met:

$$close(t) \geq take\_profit \cdot (1 + close(t)) \tag{3.3}$$

That is when the $ticker(t)$ value goes above the take profit (as a percentage) of the closing price of the long position we generate a short signal.

Short signals Inject a “long” signal into the list of actions if the following condition is met:

$$close(t) \leq take\_profit \cdot (1 - close(t)) \tag{3.4}$$

That is when the $close(t)$ value goes below the take profit (as a percentage) of the closing price of the short position we generate a long signal.

3.3 Simulation

Hold signals If a transaction has the “hold" signal class, then it is skipped. This helps speed up the simulation and adhere to the limitations of the system.

Short signals Check if a "long" position is open before the time of this "short" action. If at least one is found, close this position (if same trade size otherwise partially close or close multiple open "long" positions). If no open "long" positions are found, then open a new "short" position.

Long signals Check if a "short" position is open before the time of this "long" action. If at least one is found, close this position (if same trade size otherwise partially close or close multiple open "short" positions). If no open "short" positions are found, then open a new "long" position.

3.3.1 Commissions

In a market that allows for margin trading, two fees exist: one for the transaction and one when "borrowing" from the exchange. For example, in the case of a short margin transaction, the trader needs to borrow the security from the exchange; while in the case of long margins, the trader does not borrow. This discrepancy between the two types of transactions complicates the proper calculation of the commissions due and is highly dependent from one exchange to another. However, since Coinbase Pro does not allow the borrowing needed for margin trading, only the commissions on the transaction are needed for calculation. This is formally described by:

$$\Theta(t) = commissions(t) = transaction\_cost \cdot trade\_size \cdot close(t) \tag{3.5}$$

Where t is the time at which the transaction is simulated.

3.3.2 Stopping criterion

If the unrealised balance before closing a position (or realised when we close a position) is less than zero (i.e. initial balance exhausted), we don't trade (emit "hold"). This is implemented according to the following formula when opening or closing "long" positions:

$$balance_{stopping}(t) = balance_{stopping}(t-1) + \sigma \cdot trade\_size \cdot close(t) - \Theta(t) \tag{3.6}$$

with σ as defined by the formula 3.9 below. Instead when opening new "short" positions instead we only consider the commissions created; that is:

$$balance_{stopping}(t) = balance_{stopping}(t-1) - \Theta(t) \tag{3.7}$$

Finally when we close these "short" positions we need to take into consideration their transaction value when they where created; that is:

$$balance_{stopping}(t) = balance_{stopping}(t-1) + \sigma \cdot (\omega \cdot close(t) + \omega \cdot close(t_{open})) - \Theta(t) \tag{3.8}$$

$$\sigma = \begin{cases} -1 & long \\ +1 & short \end{cases} \tag{3.9}$$

With ω being the $trade\_size$. This is necessary as "short" positions do not affect the balance until they have been closed while "long" positions require the balance to have been realised already before performing it.

3.4 Evaluating gain/loss

Evaluating the return on investment requires the considerations of the realised gain and loss and the balance at the end of the simulation. Moreover, the number of positions that generated a gain or loss should be computed as the system should not generate too many positions that generate a loss (i.e. the number of positions with a gain should be higher than the ones with a loss).

3.4.1 Realised gain or loss

This is the gained or the lost amount for all closed positions at the end of the simulation. However, it is important to take into consideration all the commissions due to any open transaction. This can be computed as follows:

$$gain_{realised} = \sum_{i}^{\forall closed\ positions} ((value_{close}(i) - value_{open}(i)) \cdot trade\_size) - \sum_{i}^{\forall positions} \Theta(i) \tag{3.10}$$

3.4.2 Account balance

After simulating a transaction the account balance should be recomputed. This allows the implementation of the balance checks described in section 3.2.1. The account balance is computed by:

$$account\_balance_{realised} = account\_balance(t_0) + gain_{realised} \tag{3.11}$$

$account\_balance(t_0)$ is the initial investment available to the system.

3.4.3 Unrealised gain or loss

This is the possible gained or lost amount for open positions at the end of the simulation. As they represent the possible gains and/or losses, they are not included in the return on investment computation. However, this is currently not being considered when evaluating the model.

3.4.4 Evaluating return on investment (ROI)

The return on investment is computed as follows:

$$ROI = \frac{account\_balance_{realised}(t_{end}) - account\_balance(t_0)}{account\_balance(t_0)} \times 100 \tag{3.12}$$

Where t_0 is the time at the beginning of the simulations and t_{end} is the time at the end of the simulation.

3.4.5 Evaluation number of gaining and losing positions

To evaluate the number of positions that generated a gain and loss, the following was performed:

$$\begin{aligned} n_{gain} &= \sum_{i}^{\forall closed\ positions} \Big[(value_{close}(i) - value_{open}(i)) \cdot trade\_size - \Theta(i) > 0 \Big] \\ n_{loss} &= \sum_{i}^{\forall closed\ positions} \Big[(value_{close}(i) - value_{open}(i)) \cdot trade\_size - \Theta(i) < 0 \Big] \end{aligned} \tag{3.13}$$

Where $value_{open}$ is the value of the transaction when it was opened and $value_{close}$ is the value at its close. Here $[\dots]$ are the Iverson brackets, which are:

$$[P] = \begin{cases} 1 & if\ P\ is\ true; \\ 0 & otherwise \end{cases} \tag{3.14}$$

3.5 Model Evaluation

A model is profitable if its ROI is greater than zero. Moreover this is compared with the simple buy-hold-sell described below.

3.5.1 Simple buy-hold-sell strategy

This strategy "buys" cryptocurrency at the beginning of the test period holds it and then sells it at the end of the test period. This metric is useful to evaluate whether our trading system can perform better than having no trading strategy. This can be represented by the following:

$$simple\_bhs = (close(t_{end}) - close(t_0)) \cdot trade\_size \tag{3.15}$$

Where t_0 being the beginning of the prediction set time and t_{end} being the time at the end of simulation.

3.5.2 Scaling ROI

To be able to compare different runs of the grid search, we must consider the ROI of the model over the same test period or have the value scaled over a year. As considering the same test period is not possible due the granularity and horizon being parameters of our grid search, the latter method was selected. The scaled ROI is computed as such:

$$ROI_{scaled} = \frac{356_{days/year}}{(t_{end} - t_{start})_{days}} \cdot ROI \tag{3.16}$$

3.5.3 Average gain per transaction

In order to be able to evaluate the strength of a model we must also consider models that generate the highest gain per transaction. This can be computed as follows:

$$\Gamma = \frac{ROI_{scaled}}{positions_{open} + 2 \cdot positions_{closed}} \tag{3.17}$$

3.6 Grid search

In order to test the multiple combinations of parameters of the system, a grid search was performed to evaluate the different data models and simulations. Due to the size of the grid, parallelisation of the different runs was necessary. This was implemented by storing the results of every run in a database with the parameters of interest being the table's primary key. In addition to those parameters, the return on investment and some basic statistics were stored. Finally, after the grid search has concluded, the database is analysed to obtain an optimal model and simulation, better described in section 3.6.2.

3.6.1 Grid database parameters

The results of each experiment that were stored in the database can be split into three groups of parameters: primary parameters, secondary parameters and statistical values. The primary parameters are the parameters that are used to setup each experiment. Likewise, the secondary parameters are used to setup each simulation, however these parameters were not varied in the grid search and were left as constants as to allow for future work to explore them instead. Finally the statistical values are the results and statistics about each run from the primary and secondary parameters. The specific primary parameters used are presented in section 4.1.

Primary parameters

- machine learning model name and parameters (see section 3.1.2)
- currency pair being traded in the simulation
- feature set id
- granularity of the data set
- horizon used for the training and prediction sets
- time range being considered
- buy and sell thresholds used
- maximum stop loss and minimum take profit

Secondary parameters

- commissions per transactions (constant set to 0.3% of the transaction's value)
- trade size of each transaction (constant set to 0.1 of the cryptocurrency)
- initial investment amount (constant set to 10000 in the fiat currency)
- initial cryptocurrency (constant set to 0)

Statistical values and results

- predicted ROI during the considered time period
- simple buy-hold-sell strategy ROI during the considered time range
- number of transactions that generate a gain and/or a loss
- total number of transactions
- number of open and closed positions
- number of long and short positions
- time used by the computer for training and simulating the experiment

3.6.2 Choosing the optimal model

To be able to understand better the results generated from the grid search and find the model parameters that are optimal, one must select a model that has the highest ROI_{scaled} among the different currency pairs and different time ranges. This is required as to ensure that the selected model is not a local minima but more importantly that it is not overfitted for that specific product or time range. Formally, this is implemented by the function FindBestModel.

This can be applied on different groups to analyse variant parameters. This assumes that the group has been grouped by *model_type* and *feature_id* as well as the other parameters of interest. For example, a group could be created by grouping the data set by *granularity*, *horizon*, *buy_threshold and sell_threshold*, and *product_id*.

Function FindBestModel(group)

Data: data_set = all data
Result: index of best model (in group) of data_set

```
1  set current_max ⟵ 0;
2  for row ∈ group do
3      set mask1 ⟵ model = row.model and roi ≥ row.roi;
4      set mask2 ⟵ mask2 or product_id ≠ row.product_id;
5      set mask2 ⟵ mask2 or since ≠ row.since or until ≠ row.until;
6      set mask2 ⟵ mask2 or horizon ≠ row.horizon;
7      set mask2 ⟵ mask2 or granularity ≠ row.granularity;
8      set mask ⟵ mask1 and mask2;
9      set filtered_set ⟵ data_set filtered by mask;
10     if filtered_set ≠ ∅ then
11         set first ⟵ filtered_set[0];
12         if first.roi > current_max then
13             set current_max ⟵ first.roi;
14             set index ⟵ first.index;
15         end
16     end
17     return index
18 end
```

Chapter 4

Results and conclusion

4.1 Grid search parameters used

The primary grid parameters consist of two different types of parameters: parameters concerning the artificial intelligence algorithms used and parameters concerning the processing of the data set before training and then for the simulation.

4.1.1 Data parameters

Currency pairs the cryptocurrencies tested were Bitcoin (BTC), Bitcoin Cash (BCH), Etherum (ETH) and Litecoin (LTC) paired with USD and Euro.

- BTC-USD
- BCH-USD
- ETH-USD
- LTC-USD
- BTC-EUR
- BCH-EUR
- ETH-EUR
- LTC-EUR

Granularity of the data set in minutes.

- 30
- 60

Horizon as multiples of the granularity.

- 24
- 48
- 72

Buy Threshold as percentage change of the closing price.

- 1%
- 3%

Sell Threshold as percentage change of the closing price.

- -1%
- -3%

Max Stop Loss as percentage of the transaction value at position opening with respect to the current closing price or disabled.

- 5%

Min Take Profit as percentage of the transaction value at position opening with respect to the current closing price or disabled.

- 10%
- disabled

4.1.2 Machine learning models

For models requiring a random state for initial conditions this was set to 2018.

k-Nearest Neighbour (kNN)

- k: 3, 5, 6, 7, 9, 10, 11, 13, 15

Decision Tree

- criterion: gini, entropy

Random Forest

- estimators: 10, 20, 50, 70, 100, 130, 150

Stochastic Gradient Descent (SGD)

- alpha: 1×10^{-5}, 1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 0.10
- loss function: hinge, log, modified huber
- max iterations: 1000

Passive Aggressive

- C factor: 0.1, 1, 10
- tolerance: 1×10^{-5}, 1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 0.10
- max iterations: 1000

4.2 Performance analysis figures

After running the grid search, the collected results were analysed to find an optimal solution. Such a solution must perform well for the different cryptocurrencies and during different time periods. These analyses consider models with high returns in different market behaviours, but also for different currency pairs which reduces the possibility of the solution being an overfitted to a single currency or time range.

The analysis focused on comparing the different results of each model for the three main feature sets considered. In order to further explore the different optimal data parameters of the model, an analysis of the horizon, granularity and buy and sell thresholds were examined overall and for each machine learning algorithm. For the different algorithms considered, the higher values of the horizon represented the highest returns on the investment: $> 200\%$ and $> 400\%$ for feature sets 2 and 7 and 8, respectively. Conversely, the granularity did not have a consistent value that performs well given the different models. However, they were consistent for the same model for the different feature sets (eg. for PA 30 minutes). Finally, the buy and sell thresholds, favoured lower thresholds for obtaining the highest returns; $> 130\%$, $> 660\%$ and $> 380\%$ for feature sets 2, 7 and 8, respectively.

In addition to these analyses, the performance of each currency pair was considered to check whether using a certain pair was more profitable than another pair; that is if

the designed system is able to better predict the classes needed for high returns. Overall, Etherum (ETH) and Litecoin (LTC) produced consistently high gain, while Bitcoin (BTC) and Bitcoin Cash (BCH) did not perform as well for some artificial models or with some feature sets. Of particular note, is the feature set 2 for the pairs BTC-EUR and BCH-EUR having negative ROI with the different algorithms used. Conversely, the other cryptocurrencies, in general, have a better performance.

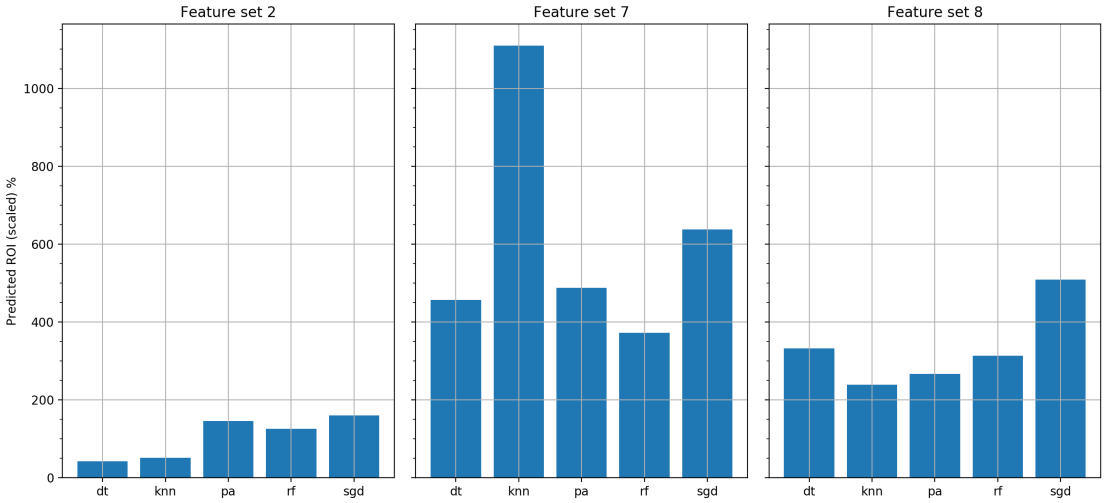

Figure 4.1: Performance by model type

In figure 4.1, the results where grouped by the model type for each feature set. From the figure, the highest return on investment is generated by the kNN model using feature set 7. This was the case when the k parameter was 3. Table 4.1, shows the overall performance results. Note that n_{gain} and n_{loss} are the average of their respective values from all the experiments having the same feature set, model and other parameters with differing time periods and currency pairs.

Table 4.1: Overall performance

feat. set	*model*	h	g	*buy*	*sell*	ROI_{scaled}	BHS	Γ	n_{gain}	n_{loss}
2	dt	24	60	1 %	−3 %	42.05	−1.77	0.07	126.15	109.50
2	knn	72	30	3 %	−3 %	50.68	−1.84	0.10	93.65	75.89
2	pa	24	60	1 %	−3 %	145.18	−2.38	0.24	60.28	49.85
2	rf	48	60	3 %	−1 %	125.64	−1.62	0.20	115.35	78.35
2	sgd	72	30	1 %	−3 %	160.08	−2.38	0.25	78.61	55.52
7	dt	48	60	1 %	−3 %	456.03	−1.77	1.06	93.57	67.32
7	knn	48	60	3 %	−3 %	1109.03	−1.50	1.83	127.52	85.84
7	pa	48	60	1 %	−3 %	487.54	−1.74	0.80	115.80	90.59
7	rf	24	30	1 %	−3 %	371.56	−1.65	0.82	94.80	70.36
7	sgd	48	30	1 %	−3 %	637.77	−1.83	1.03	125.64	94.32
8	dt	48	60	1 %	−1 %	331.48	−1.76	0.70	96.94	77.13
8	knn	72	30	3 %	−1 %	238.67	−1.77	0.44	101.30	85.17
8	pa	24	60	3 %	−3 %	265.98	−1.74	0.50	96.02	77.68
8	rf	24	30	1 %	−1 %	313.55	−1.66	0.74	79.13	59.01

Table 4.1: Overall performance (continued)

feat. set	*model*	h	g	*buy*	*sell*	ROI_{scaled}	BHS	Γ	n_{gain}	n_{loss}
8	sgd	48	60	3 %	−3 %	508.79	−1.73	0.84	124.20	96.95

Grouping by the horizon, as in figure 4.2, one can see that the higher values for the horizon perform better. Again, here the feature set 7 is the best performing one between the three options. The best model is a kNN model with $k = 3$ as can be seen in table 4.2.

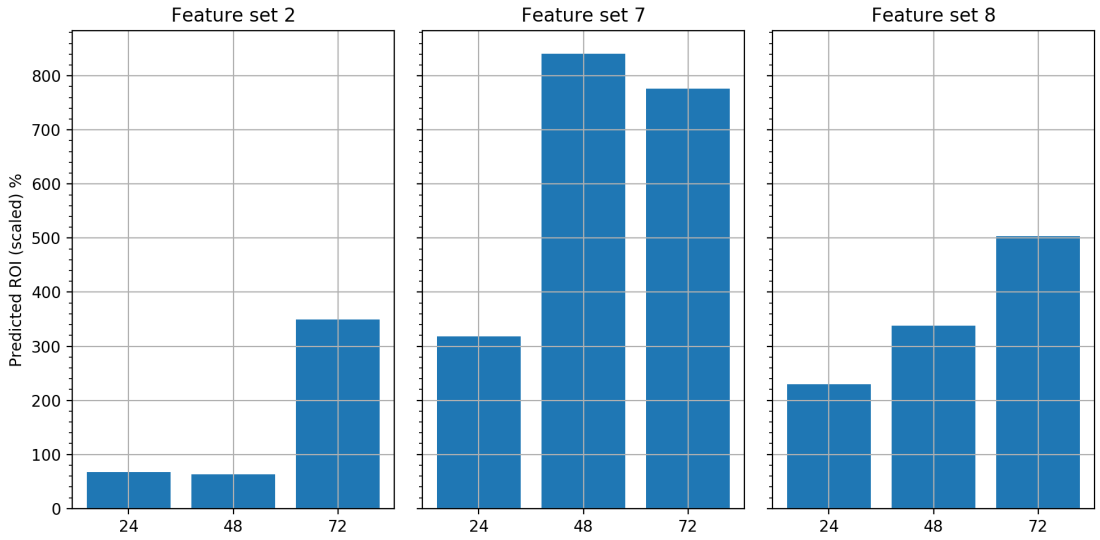

Figure 4.2: Performance by horizon

Table 4.2: Performance results by horizon

feat. set	h	*model*	g	*buy*	*sell*	ROI_{scaled}	BHS	Γ	n_{gain}	n_{loss}
2	24	knn	30	1 %	−3 %	67.47	−1.99	0.14	68.52	52.25
2	48	rf	60	3 %	−1 %	63.74	−1.97	0.11	92.98	71.25
2	72	sgd	30	1 %	−3 %	349.46	−2.70	0.41	103.11	79.62
7	24	rf	30	1 %	−3 %	318.58	−2.37	0.65	93.11	73.66
7	48	knn	60	3 %	−3 %	840.95	−1.43	1.38	128.67	90.13
7	72	knn	60	1 %	−3 %	776.38	−1.30	1.20	130.84	97.35
8	24	pa	60	3 %	−3 %	229.48	−2.54	0.49	86.83	71.88
8	48	sgd	60	3 %	−3 %	338.04	−1.34	0.61	109.11	83.83
8	72	knn	30	3 %	−1 %	503.55	−1.18	0.84	117.19	93.06

In the figure 4.3, the best currency-pairs that the system was able to predict for were Etherum (ETH) and Litecoin (LTC) when paired with US Dollars (USD). This is most likely due to the USD being more frequently traded on Coinbase, since it is based in the United States of America. The overall best model obtained for LTC using feature set 7 was an SGD with $alpha = 10^{-5}$, loss function “hinge", granularity 30 minutes, horizon 24

hours ($48 \cdot granularity$), and with buy and sell thresholds set to 1% and -3%, respectively. This model had an incredible 2050% ROI. Note that the ROI for the pairs Bitcoin Cash (BCH), Bitcoin (BTC) with Euro and BTC-USD with feature set 2 have an almost zero ROI and as such they are not visible in the figure, but they are in table 4.3.

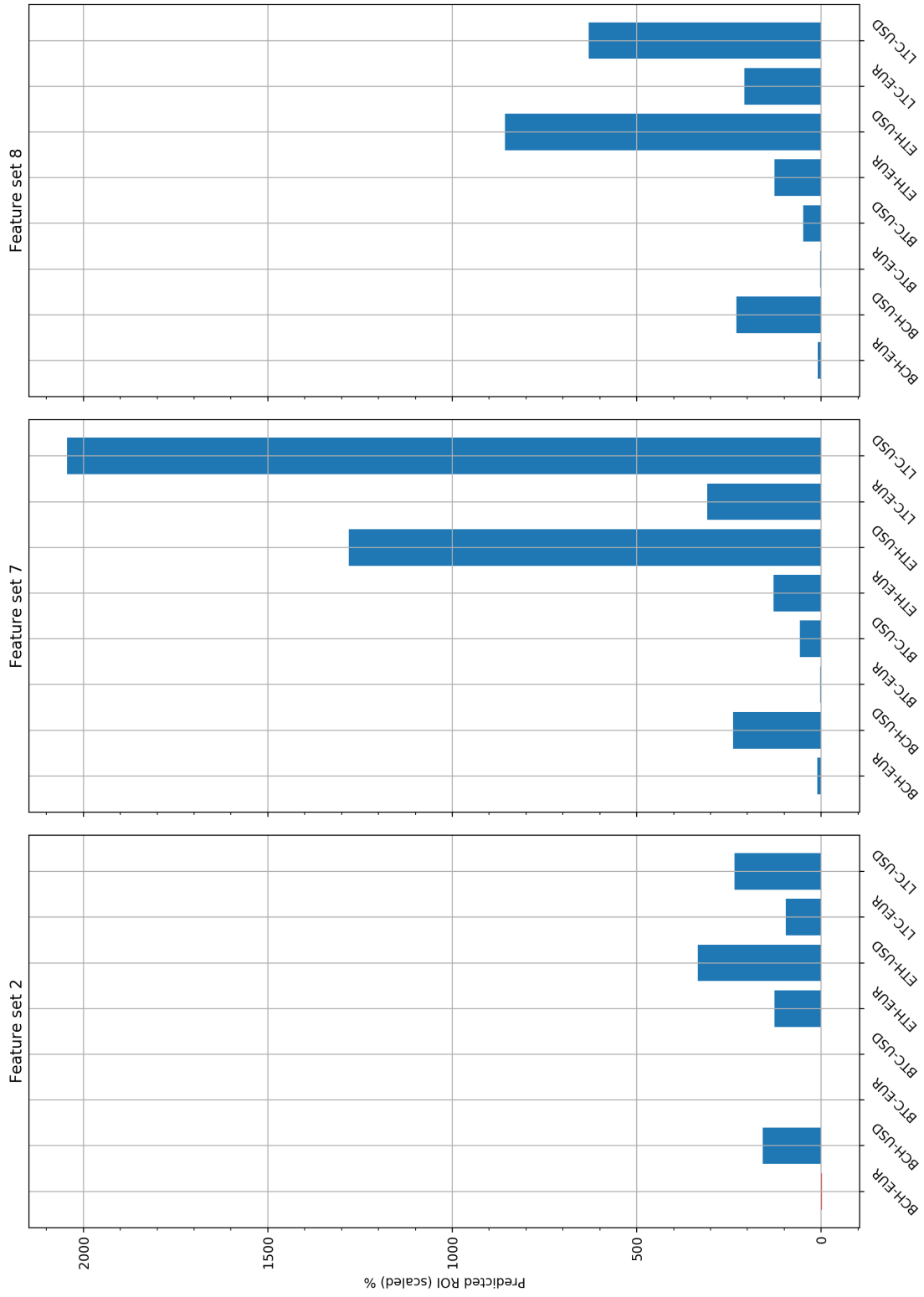

Figure 4.3: Performance by currency pair

Table 4.3: Performance results by currency pair

feat. set	*currency*	h	g	*buy*	*sell*	ROI_{scaled}	BHS	Γ	n_{gain}	n_{loss}
2	BCH-EUR	48	30	1 %	−1 %	−2.00	−8.31	−0.01	7.17	7.19
2	BCH-USD	72	30	1 %	−3 %	158.92	−4.36	0.34	87.71	73.99
2	BTC-EUR	72	60	3 %	−1 %	−0.38	−2.84	−0.00	12.18	12.43
2	BTC-USD	72	30	1 %	−3 %	1.57	−5.24	0.01	7.45	5.78
2	ETH-EUR	24	60	3 %	−3 %	126.32	0.50	0.15	139.81	87.86
2	ETH-USD	48	60	3 %	−1 %	334.02	−0.55	0.62	98.17	85.95
2	LTC-EUR	72	30	3 %	−3 %	96.10	0.01	0.11	173.48	109.42
2	LTC-USD	24	30	1 %	−3 %	234.95	−0.23	0.36	100.87	109.36
7	BCH-EUR	24	60	1 %	−3 %	9.91	−8.16	0.03	55.33	56.72
7	BCH-USD	72	60	1 %	−1 %	238.48	−4.57	0.75	69.07	45.44
7	BTC-EUR	48	60	3 %	−3 %	3.24	−3.01	0.00	264.46	271.33
7	BTC-USD	48	60	1 %	−3 %	57.77	−4.96	0.12	97.16	82.01
7	ETH-EUR	48	30	3 %	−3 %	129.42	0.87	0.15	153.44	114.56
7	ETH-USD	72	60	1 %	−3 %	1280.66	−0.79	1.86	159.61	111.99
7	LTC-EUR	24	60	3 %	−3 %	308.67	0.04	0.69	82.01	46.89
7	LTC-USD	48	30	1 %	−3 %	2044.64	−0.29	5.50	90.24	56.84
8	BCH-EUR	24	60	3 %	−3 %	9.12	−8.00	0.02	66.71	68.62
8	BCH-USD	48	30	3 %	−3 %	229.36	−4.49	0.64	81.74	58.76
8	BTC-EUR	24	30	1 %	−1 %	2.24	−2.91	0.00	216.50	223.68
8	BTC-USD	48	60	3 %	−3 %	48.84	−4.89	0.09	97.36	84.38
8	ETH-EUR	72	30	3 %	−1 %	125.95	0.88	0.15	146.39	108.94
8	ETH-USD	72	60	1 %	−3 %	857.47	−0.79	1.50	129.20	95.27
8	LTC-EUR	48	30	3 %	−3 %	208.43	0.04	0.53	70.91	46.36
8	LTC-USD	24	60	1 %	−1 %	630.02	−0.30	2.96	47.34	38.85

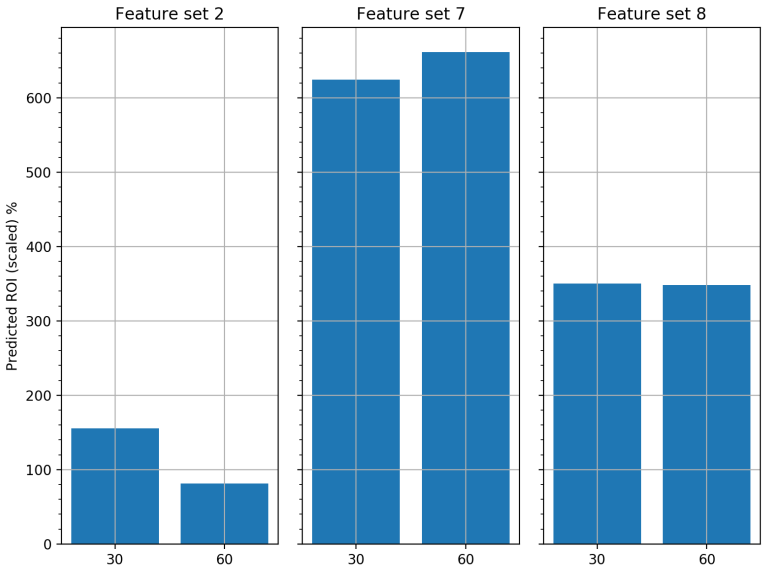

Figure 4.4: Performance by granularity (in minutes)

Table 4.4: Performance results by granularity

feat. set	g	*model*	h	*buy*	*sell*	ROI_{scaled}	BHS	Γ	n_{gain}	n_{loss}
2	30	sgd	72	1 %	−3 %	155.65	−2.21	0.22	103.78	77.13
2	60	rf	48	3 %	−1 %	81.14	−2.02	0.18	65.48	52.62
7	30	rf	24	1 %	−3 %	624.18	−1.50	0.89	142.29	103.27
7	60	knn	48	3 %	−3 %	661.22	−1.97	1.54	86.50	66.71
8	30	knn	72	3 %	−1 %	350.22	−1.56	0.52	129.72	100.97
8	60	pa	24	3 %	−3 %	348.20	−1.94	0.95	70.41	58.84

Grouping by the granularity (figure 4.4), no clear granularity value that performs well for all three feature sets. However, for feature set 2, the 30 minute granularity almost double the ROI of the 60 minute one. The results are present in table 4.4.

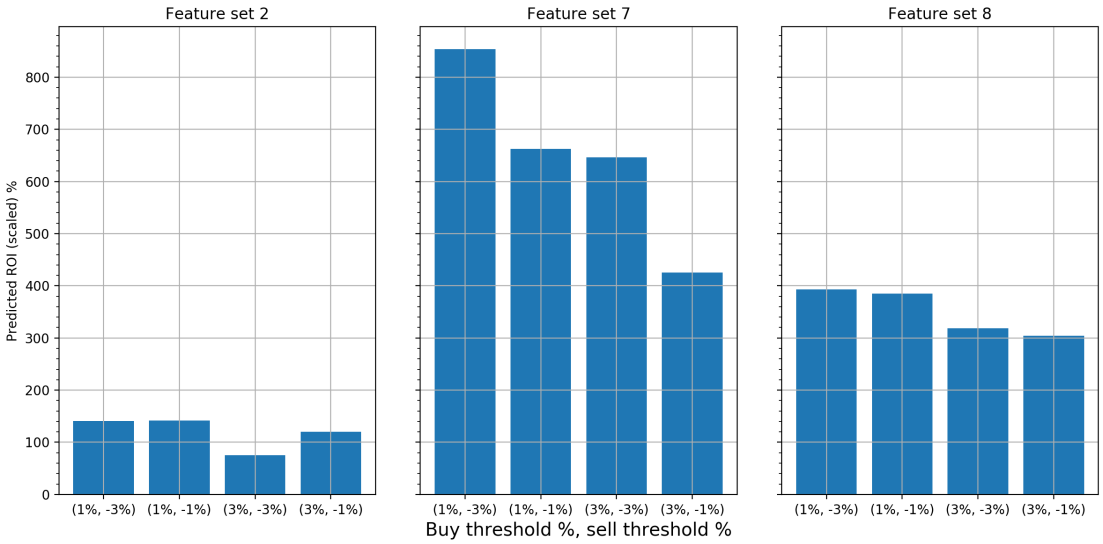

Figure 4.5: Performance by buy/sell thresholds

When considering the buy and sell thresholds used for classification, the system is able to have a positive and in most cases over 100% ROI, regardless of the thresholds, as seen in figure 4.5 and table 4.5. This is important as it shows that the system is able to perform even if the market conditions differ than the ones in the training period, which can be seen in figures 4.12 and 4.13. In particular, the overall cryptocurrency market conditions were bearish so lower and higher thresholds for sell and buy, respectively, will perform better. On the other hand, with a bullish market conditions, the opposite is true.

Table 4.5: Performance results by buy/sell thresholds

feat. set	*buy*	*sell*	*model*	g	h	ROI_{scaled}	BHS	Γ	n_{gain}	n_{loss}
2	1 %	−3 %	sgd	30	72	141.17	−2.02	0.24	90.44	62.94
2	1 %	−1 %	knn	60	48	141.39	−2.01	0.20	117.55	97.19
2	3 %	−3 %	knn	30	72	74.89	−2.23	0.17	59.33	47.76
2	3 %	−1 %	rf	60	48	120.31	−2.19	0.20	74.03	53.86

Table 4.5: Performance results by buy/sell thresholds (continued)

feat. set	*buy*	*sell*	*model*	g	h	ROI_{scaled}	BHS	Γ	n_{gain}	n_{loss}
7	1 %	−3 %	knn	60	72	853.83	−1.58	1.31	144.51	109.10
7	1 %	−1 %	knn	30	24	662.94	−1.49	1.16	121.86	87.49
7	3 %	−3 %	knn	60	48	646.75	−2.07	1.20	111.76	83.68
7	3 %	−1 %	rf	30	72	425.16	−1.69	0.76	93.78	69.40
8	1 %	−3 %	sgd	30	72	393.34	−1.57	0.67	120.50	97.98
8	1 %	−1 %	knn	30	24	385.35	−1.52	0.68	115.56	92.34
8	3 %	−3 %	pa	60	24	318.24	−2.05	0.68	92.12	74.41
8	3 %	−1 %	knn	30	72	304.34	−1.76	0.56	87.05	65.69

To further understand the results, we can group by the granularity and horizon for each model type.

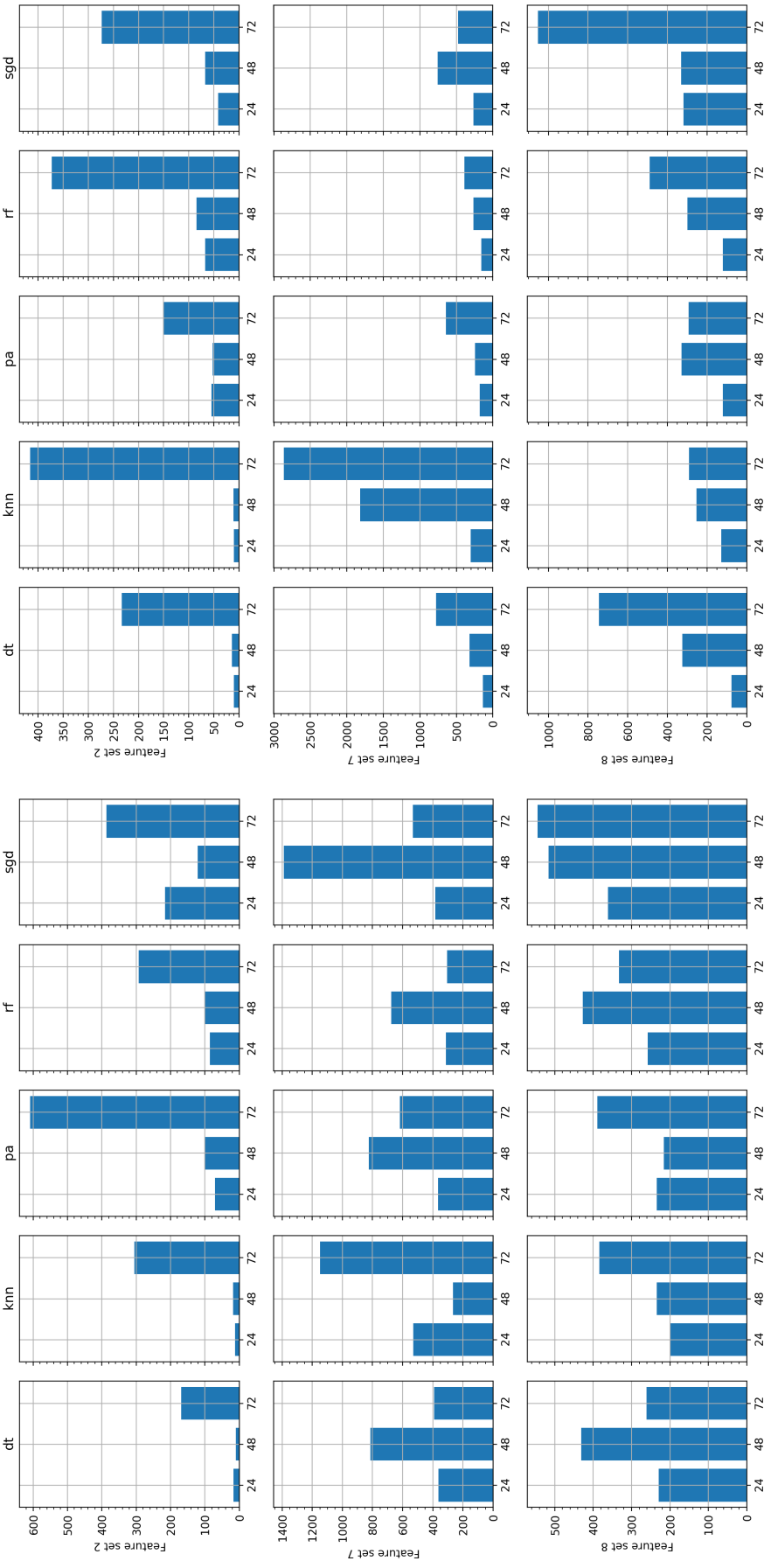

Figure 4.6: Performance by horizon given granularity is 30 or 60 minutes, respectively, for each model

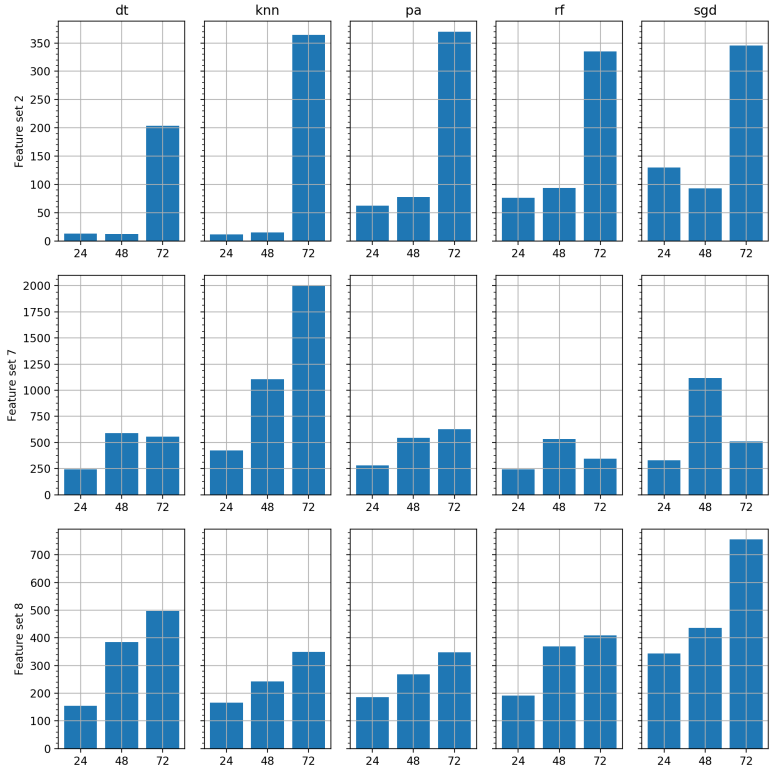

Figure 4.7: Overall performance by horizon for each model

Again here having the highest horizon (72 multiples) has the best performance among different model types and the different feature sets, as seen in figure 4.7. Furthermore, one can split the data by each granularity tested; that is for granularities of 30 and 60 minutes, the three horizon values are considered for each model, as seen in figure 4.6. The three figures confirm that the highest horizon value is the one corresponding to the highest ROI.

Similar analysis can be performed for the granularity.

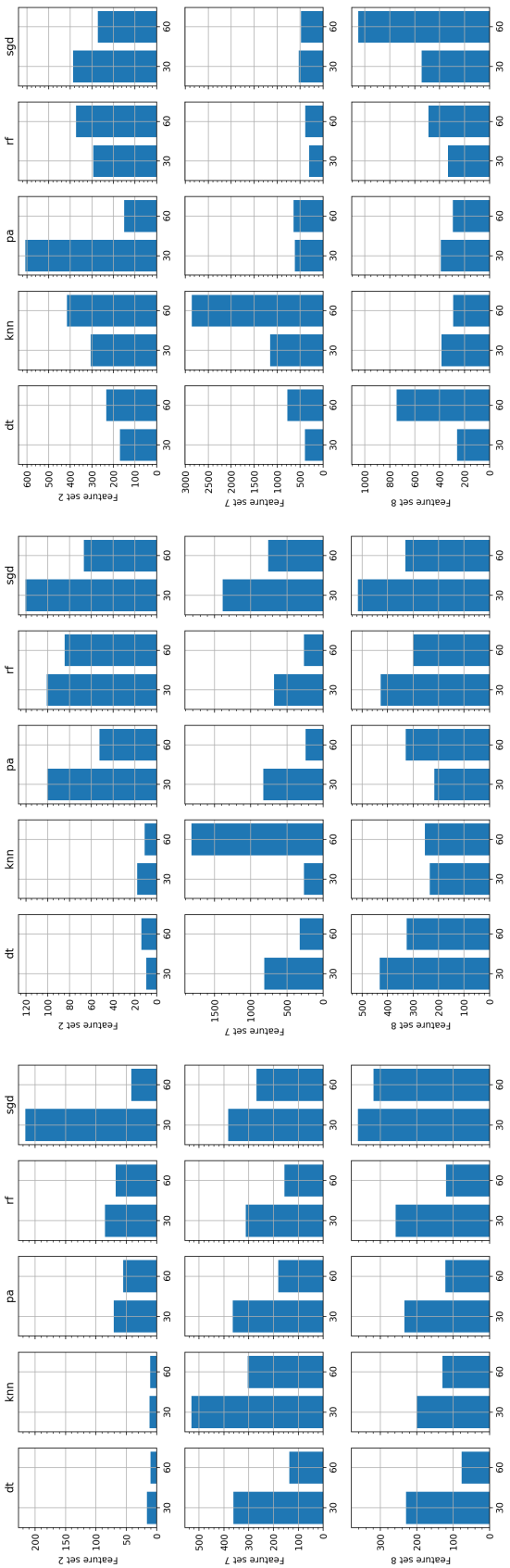

Figure 4.8: Performance by granularity given horizon is 24, 48, 72 multiples, respectively, for each model

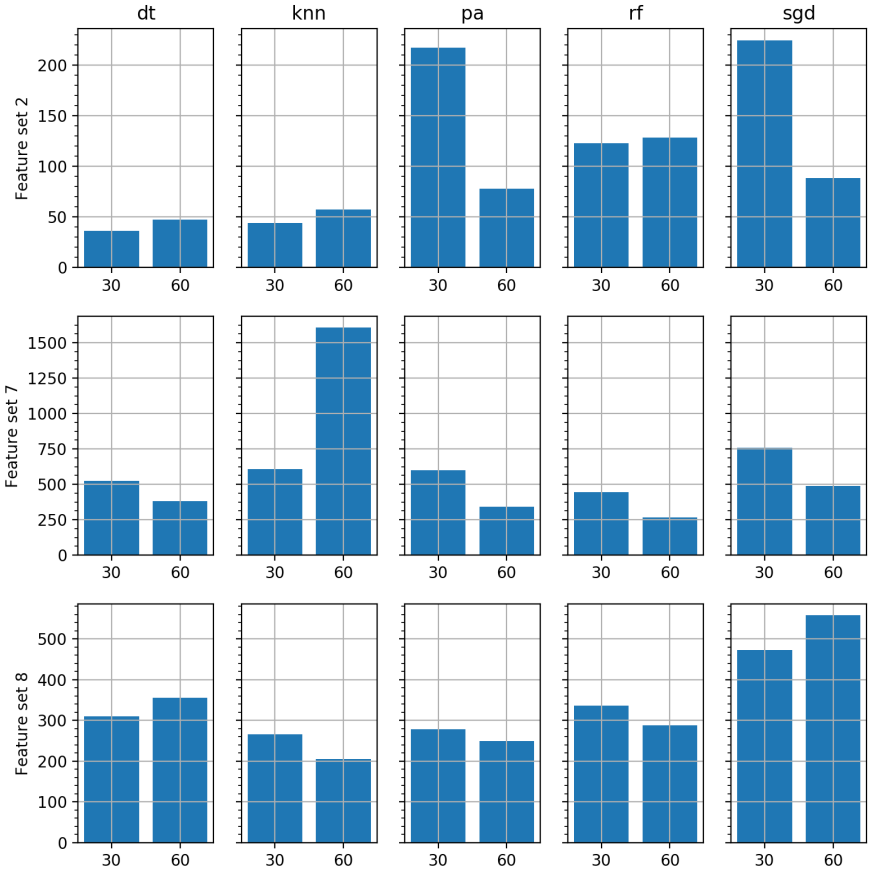

Figure 4.9: Overall performance by granularity for each model

For granularity, on the other hand, there isn't a clear optimal parameter when considering all the models, as in figure 4.9. In other words, the system based on different parameters (eg. feature sets, thresholds, ...) will have a different optimal granularity. This is also confirmed by the figure 4.8.

Moreover, one can analyse the performance of every single currency pair for each model.

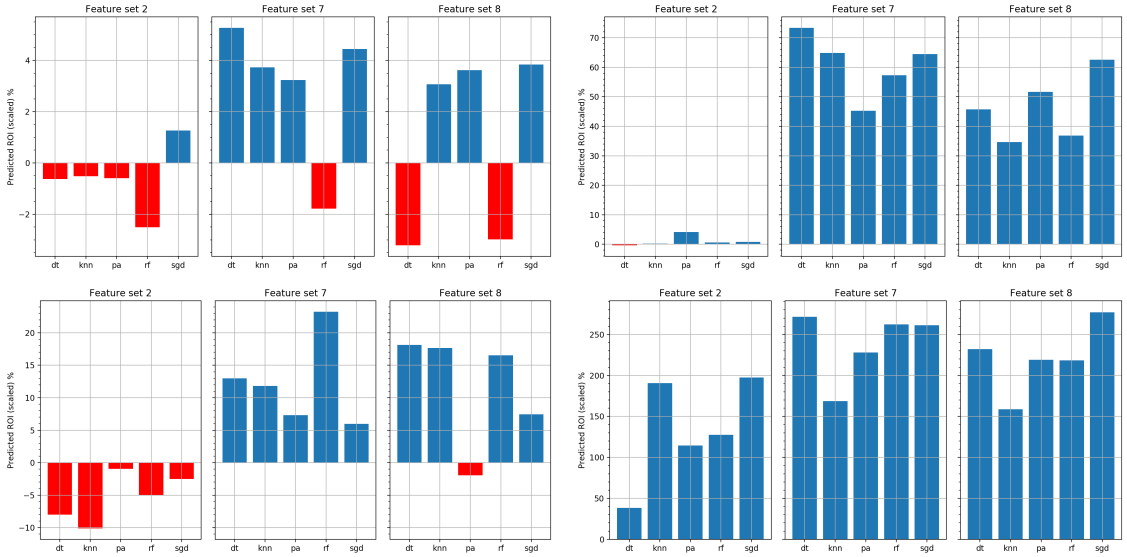

Figure 4.10: Performance by model type for BTC-EUR (top left), BTC-USD (top right), BCH-EUR (bottom left) and BCH-USD (bottom right)

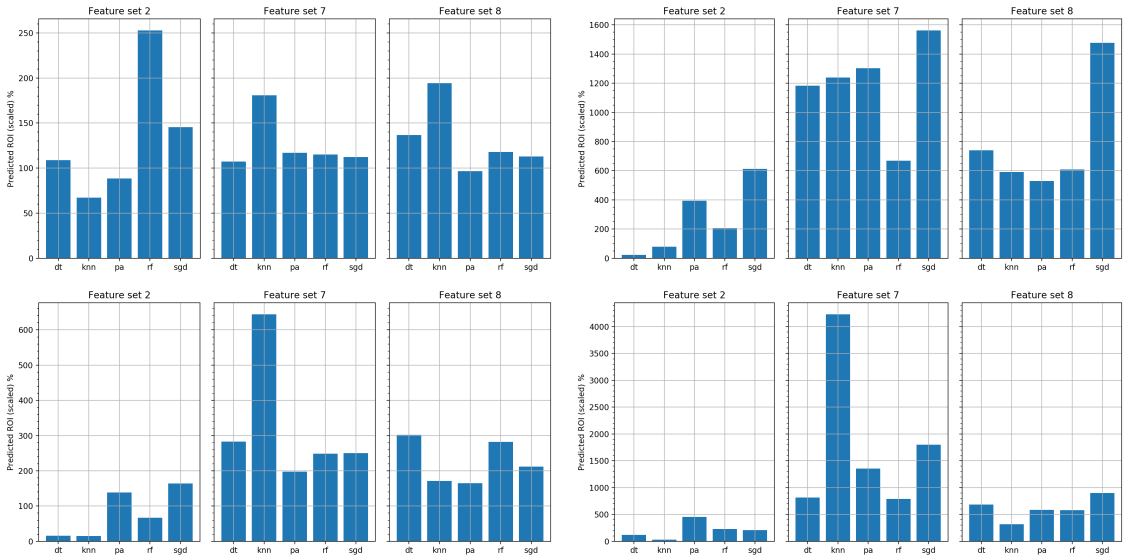

Figure 4.11: Performance by model type for ETH-EUR (top left), ETH-USD (top right), LTC-EUR (bottom left) and LTC-USD (bottom right)

Overall and in general, from the figures 4.10 and 4.11 no significant distinctions can be seen between the different feature sets; the performance is more or less the same. However, with certain currency pairs, specifically BTC-EUR/BTC-USD and LTC-EUR/LTC-USD, feature set 2 performs much worse than the time-series based features. Differences between the same cryptocurrency with a fiat currency can be attributed to fluctuations in the fiat markets. Moreover, in the case of the BCH-EUR pair due to having a lower number of transactions in comparison to the BCH-USD pair the results are more susceptible to noise factors. The noise can be reduced with the use of technical analysis, however, since the

BCH market was bullish the feature sets based on the time-series alone had higher returns.

4.3 Closing prices

In order to be able to better understand the performance of the models presented, the closing prices of the Bitcoin-Euro pair and Litecoin-Euro pair are shown in this section. The time periods tested, 01/01/2017 - 01/06/2018 and 01/01/2018 - 01/06/2018, present both bullish and bearish markets, which is important when evaluating the models as training the models on only one of the market conditions cause the models to be overfitted to such market conditions and would not perform well if they differ.

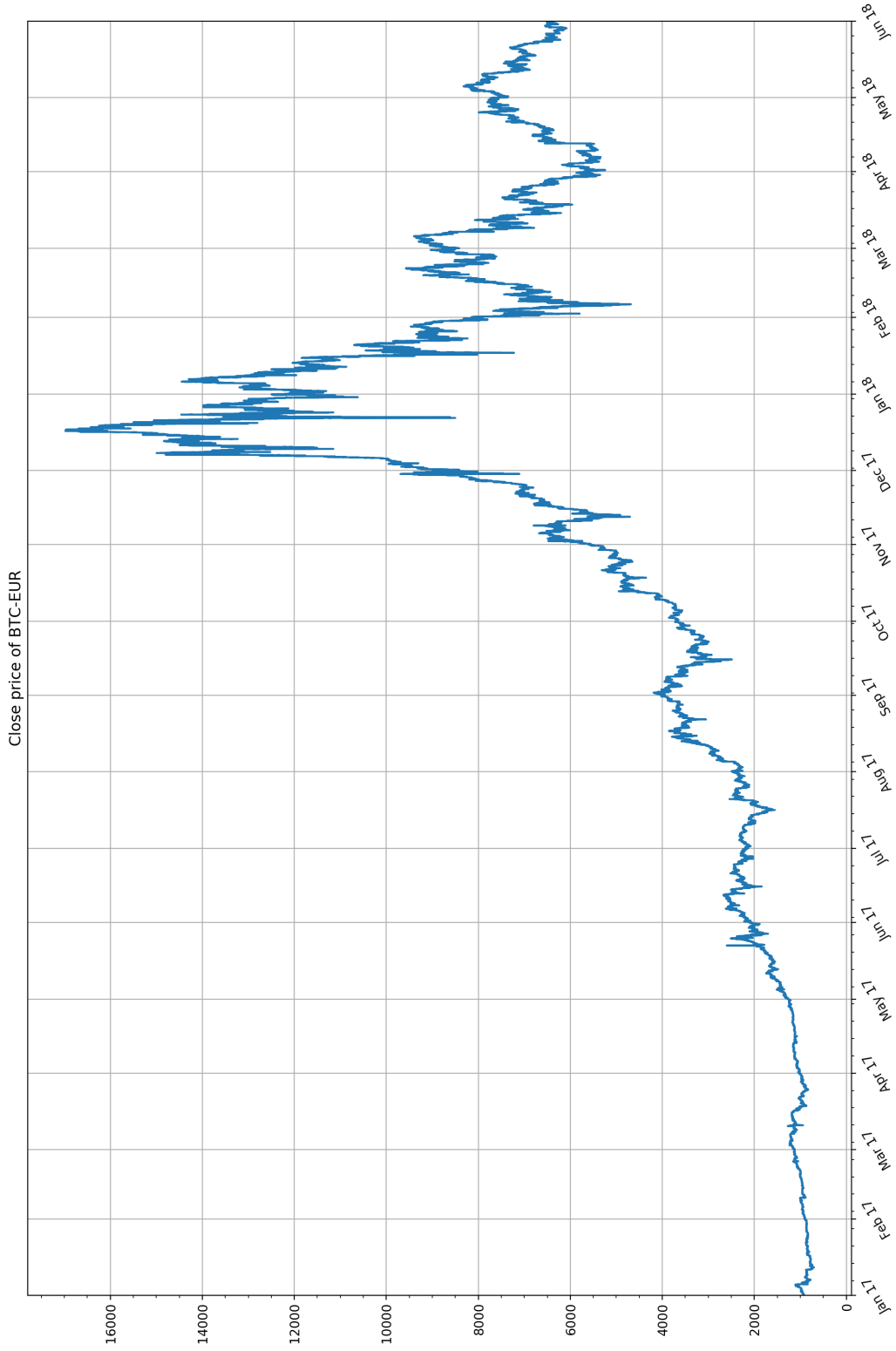

Figure 4.12: Closing price of BTC-EUR

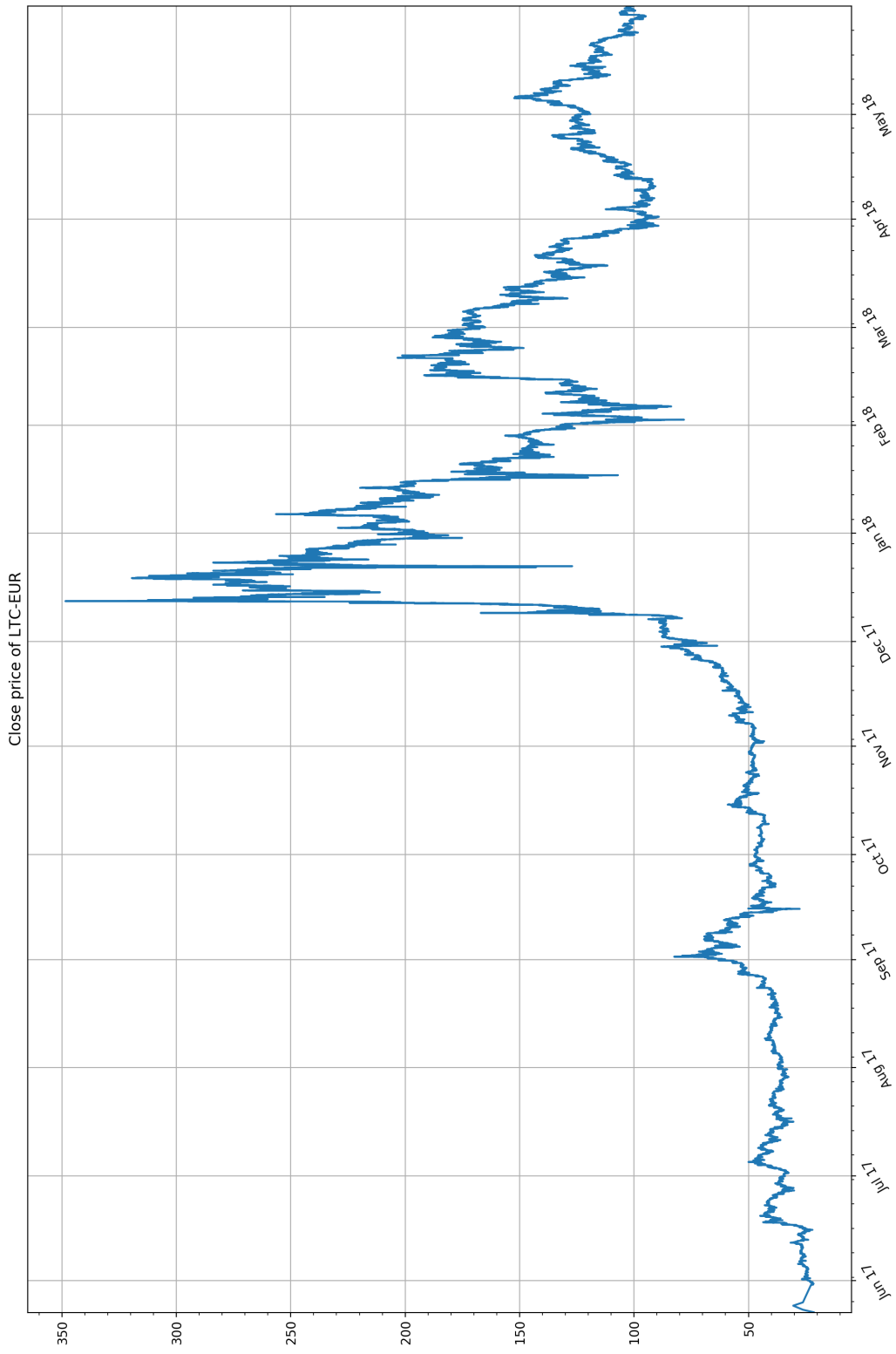

Figure 4.13: Closing price of LTC-EUR

4.4 Conclusion and future work

A system was developed to trade automatically to increase investors' profits in the cryptocurrency markets. The model was designed to compare different cryptocurrencies with diverse set of machine learning techniques. The data analysed was aggregated from Coinbase Pro's market. Different feature sets were considered for classifying and analysing the data; like closing price times-series and technical analysis. Moreover, different parameters were used to implement data processing, such as the buy and sell thresholds for classification and the granularity of the data set. The system then simulates the transactions from the predicted data set to get the results of each experiment. To properly analyse the results of the simulations performed, a method was developed to naturalise the results and compare different models with each other, given different currency pairs and time ranges.

The combination of algorithms and parameters considered for finding the ideal parameters was limited to five algorithms (decision tree, kNN, passive aggressive, random forest, and SGD). This was necessary as to limit the time needed to perform the grid search and as such further work is necessary in the field to explore other models, like multilayer perceptron (MLP), in addition to testing more parameters; for example testing out a larger range of thresholds for classification. Moreover, since the system expects that market supports future-like contracts, future systems should explore markets that do not support contract trading. Another improvement could be having the take profit and stop loss being a dynamic parameter; that is the percentage of the transaction is not fixed during the entire experiment.

In conclusion, the results of the various experiments show that using the model described earlier, the system was able to generate the highest return on investment margins when using a simple closing price time-series over the more common techniques used by many traders (i.e. technical analysis). In particular the model that has the best performance, considering different currency pairs and different time ranges, was the kNN model with k=3. The data parameters used to prepare the data sets to be analysed overall favoured higher values for the horizon, specifically 48 and 72 multiples. On the other hand, the results of the granularity parameter were inconclusive as to which of the two values tested performed better. Given the bearish market conditions, the -3% sell threshold performs better than the -1% threshold. However, even with symmetric buy and sell thresholds (i.e. $\pm 1\%$ and/or $\pm 3\%$), the system still performs as well. The developed automatic trading system was able to outperform the simple buy-hold-sell strategy by high margins.

Bibliography

Samir Abrol, Benjamin Chesir, and Nikhil Mehta. High frequency trading and us stock market microstructure: A study of interactions between complexities, risks and strategies residing in u.s. equity market microstructure. *Financial Markets, Institutions & Instruments*, 25(2):107–165, 2016. doi: 10.1111/fmii.12068. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/fmii.12068. 1.2

Elena Baralis, Luca Cagliero, Tania Cerquitelli, Paolo Garza, and Fabio Pulvirenti. Discovering profitable stocks for intraday trading. *Information Sciences*, 405:91 – 106, 2017. ISSN 0020-0255. doi: 10.1016/j.ins.2017.04.013. URL http://www.sciencedirect.com/science/article/pii/S0020025516307290. 1.2

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://link.springer.com/article/10.1023%2FA%3A1010933404324. 3.1.2

Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and regression trees*. Chapman and Hall, January 1984. ISBN 978-0412048418. 3.1.2

Pei-Chann Chang, Chen-Hao Liu, Chin-Yuan Fan, Jun-Lin Lin, and Chih-Ming Lai. An ensemble of neural networks for stock trading decision making. In De-Shuang Huang, Kang-Hyun Jo, Hong-Hee Lee, Hee-Jun Kang, and Vitoantonio Bevilacqua, editors, *Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence*, volume 5755, pages 1–10, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04020-7. doi: 10.1007/978-3-642-04020-7_1. URL https://link.springer.com/chapter/10.1007%2F978-3-642-04020-7_1. 1.2

Pei-Chann Chang, Chin-Yuan Fan, and Jun-Lin Lin. Trend discovery in financial time series data using a case based fuzzy decision tree. *Expert Systems with Applications*, 38 (5):6070 – 6080, 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.11.006. URL http://www.sciencedirect.com/science/article/pii/S0957417410012431. 1.2

Yingjun Chen and Yongtao Hao. A feature weighted support vector machine and k-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications*, 80:340 – 355, 2017. ISSN 0957-4174. doi: 10.1016/j.eswa.2017.02.044. URL http://www.sciencedirect.com/science/article/pii/S0957417417301367. 1.2

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal Of Machine Learning Research*, 7:551–585, March 2006. ISSN 1532-4435. URL http://www.jmlr.org/papers/volume7/crammer06a/crammer06a.pdf. 3.1.2

Ludmila Dymova, Pavel Sevastjanov, and Krzysztof Kaczmarek. A forex trading expert system based on a new approach to the rule-base evidential reasoning. *Expert Systems with*

Applications, 51:1 – 13, 2016. ISSN 0957-4174. doi: 10.1016/j.eswa.2015.12.028. URL http://www.sciencedirect.com/science/article/pii/S0957417415008301. 1.2

M. E. Hellman. The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):179–185, July 1970. ISSN 0536-1567. doi: 10.1109/TSSC.1970.300339. URL https://ieeexplore.ieee.org/document/4082319. 3.1.2

Yakup Kara, Melek Acar Boyacioglu, and Ömer Kaan Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert Systems with Applications*, 38(5): 5311 – 5319, 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.10.027. URL http://www.sciencedirect.com/science/article/pii/S0957417410011711. 1.2

Youngmin Kim, Wonbin Ahn, Kyong Joo Oh, and David Enke. An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms. *Applied Soft Computing*, 55:127 – 140, 2017. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2017.02.006. URL http://www.sciencedirect.com/science/article/pii/S1568494617300741. 1.2

Y. Kwon and B. Moon. A hybrid neurogenetic approach for stock forecasting. *IEEE Transactions on Neural Networks*, 18(3):851–864, May 2007. ISSN 1045-9227. doi: 10.1109/TNN.2007.891629. URL https://ieeexplore-ieee-org.ezproxy.biblio.polito.it/document/4182391. 1.2

Ricardo Laborda and Juan Laborda. Can tree-structured classifiers add value to the investor? *Finance Research Letters*, 22:211 – 226, 2017. ISSN 1544-6123. doi: 10.1016/j.frl.2017.06.002. URL http://www.sciencedirect.com/science/article/pii/S1544612316303968. 1.2

Asjylyn Loder. Bitcoin etfs keep trying, despite regulators' rejections. *Wall Street Journal*, September 2018. URL https://www.wsj.com/articles/bitcoin-etfs-keep-trying-despite-regulators-rejections-1537754701. 2.2

Mohamed M. Mostafa. Forecasting stock exchange movements using neural networks: Empirical evidence from kuwait. *Expert Systems with Applications*, 37(9):6302 – 6309, 2010. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.02.091. URL http://www.sciencedirect.com/science/article/pii/S0957417410001302. 1.2

John J. Murphy. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. Penguin Group, January 1999. ISBN 978-0735200661. 1.2, 2.4.3, 2.4.10

Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4): 2162 – 2172, 2015. ISSN 0957-4174. doi: 10.1016/j.eswa.2014.10.031. URL http://www.sciencedirect.com/science/article/pii/S0957417414006551. 1.2

Martha Pulido, Patricia Melin, and Oscar Castillo. Particle swarm optimization of ensemble neural networks with fuzzy aggregation for time series prediction of the mexican stock exchange. *Information Sciences*, 280:188 – 204, 2014. ISSN 0020-0255. doi: 10.1016/j.ins.2014.05.006. URL http://www.sciencedirect.com/science/article/pii/S0020025514005246. 1.2

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, October 1992. ISBN 978-1558602380. 3.1.2

scikit-learn developers. 1.11. ensemble methods — scikit-learn 0.20.3 documentation: 1.11.2.1 random forests, 2019. URL https://web.archive.org/web/20190331074612/https://scikit-learn.org/stable/modules/ensemble.html#random-forests. [Accessed: 31/03/2019]. 3.1.2

Lamartine Almeida Teixeira and Adriano Lorena Inácio de Oliveira. A method for automatic stock trading combining technical analysis and nearest neighbor classification. *Expert Systems with Applications*, 37(10):6885 – 6890, 2010. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.03.033. URL http://www.sciencedirect.com/science/article/pii/S0957417410002149. 1.2

Chih-Fong Tsai, Yuah-Chiao Lin, David C. Yen, and Yan-Min Chen. Predicting stock returns by classifier ensembles. *Applied Soft Computing*, 11(2):2452 – 2459, 2011. ISSN 1568-4946. doi: 10.1016/j.asoc.2010.10.001. URL http://www.sciencedirect.com/science/article/pii/S1568494610002516. The Impact of Soft Computing for the Progress of Artificial Intelligence. 1.2

Jar-Long Wang and Shu-Hui Chan. Stock market trading rule discovery using two-layer bias decision tree. *Expert Systems with Applications*, 30(4):605 – 611, 2006. ISSN 0957-4174. doi: 10.1016/j.eswa.2005.07.006. URL http://www.sciencedirect.com/science/article/pii/S095741740500148X. 1.2

Kamil Żbikowski. Using volume weighted support vector machines with walk forward testing and feature selection for the purpose of creating stock trading strategy. *Expert Systems with Applications*, 42(4):1797 – 1805, 2015. ISSN 0957-4174. doi: 10.1016/j.eswa.2014.10.001. URL http://www.sciencedirect.com/science/article/pii/S0957417414006228. 1.2

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 116–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015332. URL http://doi.acm.org/10.1145/1015330.1015332. 3.1.2