Creating an AI-based trading bot

for cryptocurrencies markets on Poloniex

Autonomous project

Sébastien Cararo

Gap Year – May 2020

Table of contents

I – Abstract

I created a cryptocurrencies trading bot in python using the Poloniex API. The system was able to generate profit on backtested simulations and is currently being tested for validation in real-life trading.

II – Introduction and problem overview

Bitcoin is a cryptocurrency created in xx by xx. It is currently the largest held cryptocurrency asset across the world and, as of the 13th of June, 2020, a market cap of xx billion dollars. Some may think it can revolutionize the monetary system, as it is a decentralized asset. …

It can also be traded in order to bet on the price avolutions over time. Trading bot are a useful way to make profit without being emotionally involved in the decision process. Furthermore, it is a good way for non-specialists – like me – to generate profit.

I decided to conduct multiple studies and trading strategies throughout this work, in order to explore theses possiblities and to choose the best one to make profit. A trading bot is a great tool for creating passive incomes, as from the moment it is entirely built and fully operating, it is supposed to be making trades autonomously and hence not requiring time investment from the user. Furthermore, it also removes the emotional side from trading, which is generally a great sign as emotions, especially during stressful situations, might lead traders to make biased decisions under pressure. On the other hand, the trading bot is lacking a few information, such as sentiment analysis, special events and news announces.

III – Pipeline presentation

IV – Data source

In order to obtain models as precise as possible, especially for short-term variations such as 1%. The poloniex API in python ([xx]) is a great tools to take buy and sell orders, but it also features multiples functions such as historical data. The data are available throught different timesteps (in seconds : 300, 900, 1800, 7200, 14400, 86400) and from the year 2015. I then asked for the 5-min Bitcoin data from January 1st of 2015 until March 23rd of 2020, containing the following information for each 5-min time step :

Field	Description
date	The UTC date for this candle in miliseconds since the Unix epoch.
high	The highest price for this asset within this candle.
low	The lowest price for this asset within this candle.
open	The price for this asset at the start of the candle.
close	The price for this asset at the end of the candle.
volume	The total amount of this asset transacted within this candle.
quoteVolume	The total amount of base currency transacted for this asset within this candle.
weightedAverage	The average price paid for this asset within this candle.

<u>Table xx :</u> Output for historical data request on Poloniex ([xx] – Poloniex API documentation)

close	date	high	low	open	quoteVolume	volume	weightedAverage
7405.0	16/11/2017 08:35	7415.56705894	7360.00000007	7365.0	21.55844607	159209.20511175	7385.00375188
7399.00000001	16/11/2017 08:40	7415.56705884	7385.43573889	7415.0	27.49793162	203593.35868841	7403.95174087
7395.0	16/11/2017 08:45	7405.0	7395.0	7402.24401974	16.21509078	120005.59893429	7400.85890128
7405.0000001	16/11/2017 08:50	7410.00000004	7386.94453482	7395.0	7.80580921	57731.03404846	7395.90636861
7420.55037798	16/11/2017 08:55	7424.335108	7402.0	7408.9999999	34.31409057	254376.25602325	7413.1720176
7425.0	16/11/2017 09:00	7429.11735298	7410.00000009	7423.99999999	66.83573313	496078.0566095999	7422.34779776
7442.0	16/11/2017 09:05	7442.21324774	7421.86142151	7421.86142151	23.04951484	171420.83566639	7437.06914684
7446.99999997	16/11/2017 09:10	7449.0	7421.099420899999	7442.0	38.56054511	286883.06715657	7439.80839322
7438.0	16/11/2017 09:15	7450.0	7436.16022967	7446.99999997	57.40625173	427561.91204007	7448.00259823
7421.10681754	16/11/2017 09:20	7440.0	7411.19999903	7439.99999999	6.21939699999999	194754.94911932	7427.89581008
7430.99999988	16/11/2017 09:25	7439.99999985	7421.10681754	7421.10681754	23.47427649	174448.73187999	7431.48492582
7431.00000044	16/11/2017 09:30	7439.9999999	7430.0	7430.99999988	8.52377513	63370.67840717	7434.57886214
7437.47847159	16/11/2017 09:35	7443.2	7430.0	7430.2569002	2.75189469999999	169268.13912404	7439.73815613
7443.2	16/11/2017 09:40	7450.00000001	7435.79745242	7437.47847159	28.76639623	214173.24830107	7445.25823077
7450.0	16/11/2017 09:45	7450.00000011	7440.00000006	7443.2	10.27007034	76505.66749248	7449.38106163
7450.49650337	16/11/2017 09:50	7450.49650337	7441.0	7449.99999999	13.2621354	98789.23704975	7448.96911923
7457.28	16/11/2017 09:55	7457.28	7450.0	7450.49650337	15.25788605	113705.90816863	7452.27142187

<u>Table xx :</u> Overview of the data received after calling the API

V – Defining Y, computing predictive variables (X)

(a) Defining Y

(b) Computing predictive variables

In order to accurately predict the price trends, I had to create some finance-related variables that would describe at best the signal evolution over time. To do this, the variables needed to reflect both short-term and long-term tendencies (*bullish* or *bearish*) along with the volume and volatility evolutions. Therefore I documented myself towards such finance metrics, and I found out a few key metrics that are commonly used for financial time-series analysis :

- Bodysize
- Shadow size
- Percent change
- Moving avergaes [xx] : slow-MA (xx periods) and fast-MA (xx periods)
- Bollinger bands[xx] : top, bottom, range, percentage
- MACD [xx]
- RSI [xx]
- Support and resistance trendlines : thanks to the *trendy* [xx] package in Python, I was able to compute support and resistance levels for the signal in the form of linear lines (one above the signal and one below th signal). I used the slope from each line and used it as input for my model, plus I also used the expected value according to each line. I also used the difference between these two values. [*insert graph*]

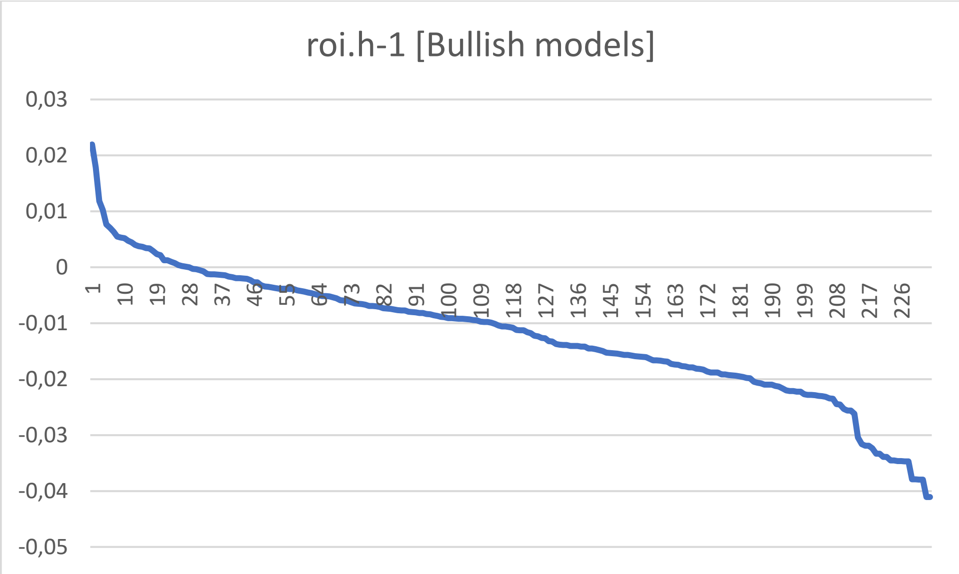
VI – Dimension reduction using Principal Component Analysis (EigenVectors Decomposition)

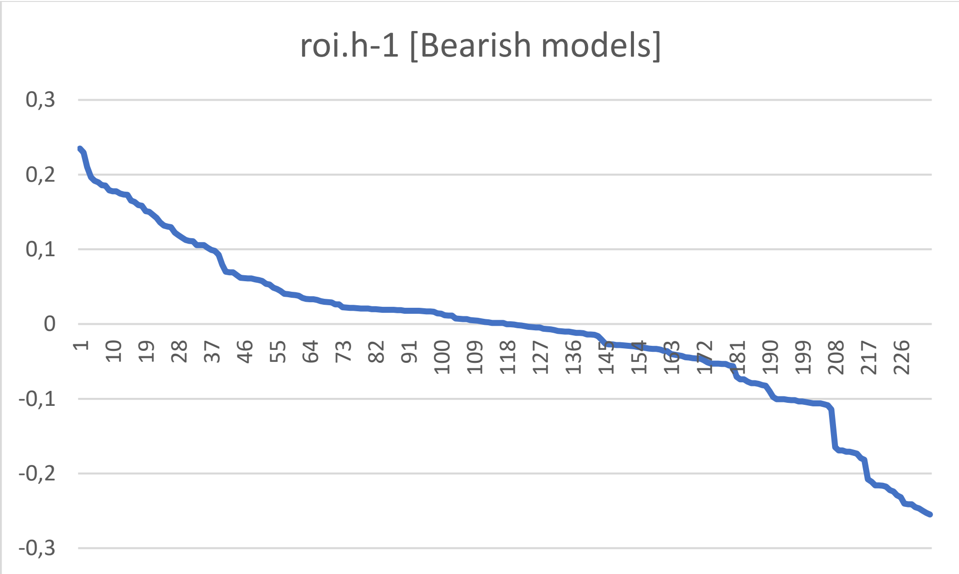
VII – Machine learning

(a) Various models and hyperparameters

I chose to use a feedforward neural networks, composed of xx hidden layers and I used xx as activation function.

(b) Results comparison [Check results]





It is easier to predict when will the bitcoin drop rather than predicting when it will rise.

VIII – Creating a trading bot on Poloniex

(a) Principle

(b) Backtesting our strategy

(c) Deployment and real trading results

IX – Future improvements

X – Conclusion

XI – Sources

[xx] – Poloniex API in Python

https://pypi.org/project/poloniex/

[xx] – Poloniex API documentation

https://docs.poloniex.com/#returnchartdata

XII – Appendices