Autonomous Bitcoin Trading

Documentation

Gymnasium Thun

26gs

Submitted by:

x

Supervised by:

Dr. Geoffrey Ostrin

August 28, 2025

Abstract

This thesis examines the development of an autonomous Bitcoin trading algorithm, based on the observation that human emotion leads traders to act irrationally in the markets. The research combines technical analysis with algorithmic trading, employing a quantitative methodology that includes data collection, strategy development using Python, and testing strategies on historical price data. The implementation involves creating a custom backtesting framework, developing rule-based trading strategies, and deploying a live trading system on a Linux based server.

Foreword

My interest in Bitcoin began in early 2022 after reading *The Bitcoin Standard by Saifedean Ammous*. The book explores the history of money and argues that Bitcoin is a revolutionary form of hard money [1] superior to traditional currencies. The book sparked an interest in me, and I was drawn into the world of crypto. As a result, I spent more and more time watching charts and losing my mind over them. The idea of making profit from these seemingly random price movements fascinated me. Trading profitably is straightforward: sell high and buy low. The practical implementation, however, is filled with uncertainty and fight against your own instincts and emotions. In addition to getting into crypto I also tried to learn to code since I wanted to make money on the side without having to leave my house. The first time I learned about automated trading algorithms I thought it was one of the hardest things to do. The combination of knowledge made me admire people who code in the finance space. It felt impossible but the more I tried the closer I got to something into the direction of a trading algorithm. With my matura thesis approaching I was pretty confident that I would be able to build an automated Bitcoin trading algorithm.

I had no idea what I was getting myself into, and if I had known before, I would probably have never started. I spent more time than I intended and completely lost myself during the summer vacation. I totally forgot about the writing part and spent a lot of time just coding. This project grew much bigger than I initially thought, but I'm proud to present both the work outcome and the learnings I gained from it.

---

[1] A form of money that maintains its value over time, offering stability and reliability

Contents

1 Introduction

Making money in the capital markets is a complex activity that requires emotional control and discipline. Most ividuals fail to make money on shorter time frames. Since computers do not suffer from psychological biases, automating the process offers the potential for more consistent decisionmaking. This thesis explores whether I can develop an autonomous trading algorithm that can generate me a consistent profit. This work does not aim to outperform professional trading firms, but rather to demonstrate the challanges of building a fully automated trading algorithm. To that end my thesis will cover the entire development cycle of an algorithmic trading strategy. The will include finding a trading strategy, backtesting the trading strategy, applying the strategy to a live market, and evaluating the results.

The work is divided in three parts: Finding a strategy, Backtesting, and Live trading.

Finding a strategy can be done in multiple ways just by watching markets over longer time periods to get ideas, reading books about how other people were able to develop strategies and steal logic from them. In my case I will try to find a strategy by analysing the orderbook delta and identifying trends on the non-stationary [2] bitcoin price.

Backtesting is the process of testing a trading strategy on historical data to evaluate its performance. It is used to see if the strategy is worth automating.

Live trading is the process of executing a trading strategy on a live market. Here I will see if my strategy works in the real market and how it will behave. The strategy will be automated in Python. A major difference is that I will use two different Python programs for the backtesting process and the live trading algorithm.

The project requires know-how in three key areas: Statistics, Programming and understanding how the cryptocurrency market operates. I need to apply statistical rules to check if my idea is worthless or has potential to make me a profit if I automate the execution process. Programming is just needed to make the process of testing or applying my ideas easy on large datasets since I have to test on time series. The algorithm will be written in Python and will run 24 hours a day seven days a week.

By the end of the project, I aim to deliver the following components:

- Backtest results

- Trading strategy

- Trading algorithm program

- Live execution results

---

[2]Non-stationary means not being still or fixed in one place; moving or changing rather than remaining constant.

2 Theoretical Background

Automating a trading strategy comes with many different challenges which are not obvius at first. In this section I will explain the basics needed to understand the trading market, algorithmic trading and challenges of creating a trading algorithm.

2.1 What is trading?

Trading means buying and selling financial assets like stocks, cryptocurrencies, or commodities, usually with the goal of making a profit. Traders try to buy at a low price and sell at a higher one, or profit from falling prices by selling high and buying back lower. Prices in financial markets change constantly, which creates opportunities and risks. There are many different types of trading, depending on the chosen timeframe, strategy, and how trades are managed. Losing is part of every strategy the goal is to lose less money than you win. As trading is often seen as an easy way to make money it is important to understand that it is not easy. It is a skill that requires a lot of time and effort to master. Type of trading strategy does not matter there is no easy or fast way to success in trading.

Algorithmic Trading

In this thesis I will try to find success in the algorithmic trading niche, also know as Algo-trading or automated trading. This is a type of trading which uses computer programs to execute trades in finacial markets based on predefined rules or mathematical models. Inside of algorithmic trading there are again many different types. Some examples are: statistical arbitrage, trend following, and High-Frequency Trading. In terms of logic all these are not related at all and strategies are often very complex in these areas.

My strategy is best descirbed as a statistical signal-based strategy. As I try to find a metric that shows statistical evidence on which I can capitalized on. In financial markets we can profit by betting on a volatilty change [3] or directional bias. Capitalizing from directional bias is as simple as buying if your directional bias positive and selling if we have a negative directional bias. (I am reffering to positive and negative returns)

---

[3]a measure of how much and how quickly the price of a trading instrument fluctuates over time
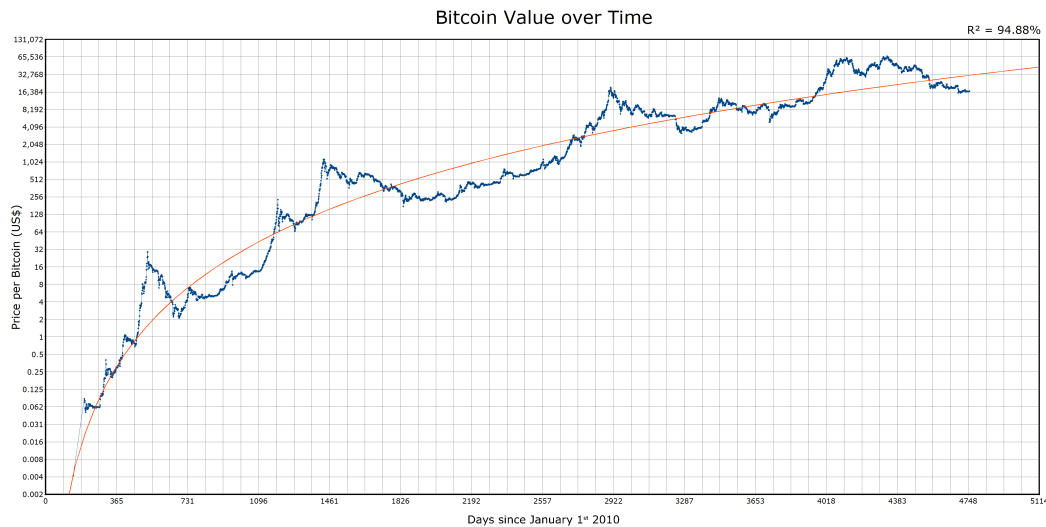
Figure 1: Bitcoin price chart

2.2   What is Bitcoin?

As my algorithm is trades the Bitcoin/USD pair, I want to explain what Bitcoin is.

Before explaining what Bitcoin is, it is important to first understand the fundamental problems of fiat money. The central issue lies in the centralization: governments and central banks hold complete control over the money supply. This control enables them to print money at will, there by increasing the supply and often causing inflation. As a result, the purchasing power of money decreases, which devalues people's savings. In addition, monetary policy [1] decisions are frequently influenced by political considerations rather than sound economic principles, leading to poor economic outcomes. Ultimately, the system depends entirely on trust in governments to manage the money supply responsibly.

In 2009 a group or individual known as Satoshi Nakamoto published the White Paper of Bitcoin. The paper outlines the concept of a decentralized digital currency that would not be controlled by any central authority. Bitcoin is decentralized, meaning that no single entity controls the network. The money supply is limited to 21 million Bitcoins. Anyone with internet access can use Bitcoin. Transactions are immutable and cannot be changed once confirmed. Bitcoin is borderless and can be sent anywhere in the world.

In the figure 1 we can see the price of Bitcoin over the years. The Y-axis is logarithmic to better visualize the strong increase in price. The X-axis shows the days since 2010.

---

[1] actions undertaken by a nation's central bank to manage the money supply and credit conditions to achieve macroeconomic goals

Why trade Bitcoin?

To prevent any misunderstanding, I want to clarify that I chose Bitcoin as an underlying asset for my trading strategy out of practicality and how my personal opinion about the asset is irrelevant. I read a few books explaining the reason for the creation of Bitcoin and the fundamentals behind it. I personally believe in the narrative of Bitcoin as a decentralized and inflation secure asset. However, this belief of mine plays no role in my trading algorithm. The goal of the trading algorithm is to react to signals which indicate market inefficiencies. These market inefficiencies can occur on any kind of asset in different magnitudes.

Data accessibility

In the crypto market, real time and historical data is mostly free and accessible freely through public APIs. An API is a tool that allows programs to request and receive specific data from external services in this case, used to fetch real-time Bitcoin price and orderbook data from the coinbase crypto exchange. The data I based my strategy on is totaly free I just had to create my own timeseries in the traditional market I would have paid thousands of dollars for the same data.

Modern API infrastructure

Some crypto exchanges offer modern API infastrucutre which allow to automate trading. The exchange I chose Hyperliquid offers world class python implementation of their API. This allows me to carry out trades with a simple self written python script that implements their python sdk. [4]

Entry barriers

You can start trading crypto with as little as 10 USD no matter if the exchanges offers automated trading or just manual executed trades. While in the traditional market you need atleast 10'000 USD to open an account for on a broker which offers automated strategy access.

These features make Bitcoin an interesting asset, both from a technological and financial perspective.

---

[4]A python sdk is a software development kit for python. It is a collection of tools and libraries that allow developers to interact with a specific API.

2.3 Efficient markets

Making consistent profits in financial markets is difficult. Prices move in ways that often seem unpredictable, and most traders both discretionary and systematic struggle to gain a lasting edge. [1]One explination for this is the Efficient market hypothesis (EMH) which suggest that asset prices already reflect all available information, leaving little room for profitable oppertunities.

If the EMH were fully accurate in practice, then even the most advanced trading algorithms and best discretionary traders would not be able to consistently beat the market. [2] Yet markets are not perfectly efficient, being heavily influenced by emotion, fear, and herd behaviour. Traders often fail to act on clear signals simply because it is psychologically hard to buy on a red day or sell while everybody is full of euphoria.

To make things even better depending on where you trade market can change in efficient. But still some of the most valuable assets like stocks, gold and Bitcoin prices are a reflection of human emotion pricing in information. Especially now with Trump in office a lot of volatility came in with big opportunities

Depending on the timeframe there are still a lot inefficiencies from which individuals can make a profit from. I will try to act on short term inefficiencies with my trading algorithm because I only have about one month for the live test and it would not make sense to run an algorithm which holds a trade over weeks and takes a trade every two months. In addition my initial idea when I started of with my matura thesis was already based on short term signals.

It is possible to make a profit in the markets but it is not easy, not for everyone, and needs dedication. I am pretty sure I am able to make a profit. The question rather is if it is possible to do so in the limited time I have and at what scale.

Understanding Market Edge

Before diving into strategy development, it's important to understand the concept of a market edge. An edge is a consistent advantage over the market that allows a trader to capitalize on market inefficiencies. Edge can have many different orgins some example are:

- Superior information or analysis
- Faster execution or better technology

---

[1]An edge is a consistent advantage over the market. It can be achieved through superior information, better analysis, or a unique trading approach.

[2]Beating the market means achieving a higher ROI ("Return on Investment") than a certain benchmark. A common benchmark is the S&P 500, which is an index including the 500 largest companies listed on U.S. stock exchanges. The S&P 500 has had an average yearly return of about 10.33% since 1957

- Unique trading approaches that exploit market inefficiencies

- Better risk management techniques

Without a clear edge, even the most sophisticated trading system will struggle to generate consistent profits. The goal of systematic strategy development is to identify, validate, and exploit such edges in a repeatable way.

2.4 Development of a Systematic Strategy

Idea Generation: The systematic strategy has to be based on a hypothesis that once signal X occurs the market is wrong and we can profit from the market correcting back to a fair value. This can be done by copying other strategies and applying them in a different way or in a different market. The approach I took was trying to make a discretionary strategy into a systematic one.

Backtesting: After coming up with an idea we have to make a strict set of entry and exit conditions. We test the rules on historical data. This allows the developer to evaluate how the strategy would have performed in the past. This allows us to get a hint on how the strategy would perform in the future and if it is worth automating.

Optimization and Robustness Testing: After initial tests, parameters can be adjusted. A simple example would be adjusting the holding time of a trade. Some inefficiencies can be corrected in minutes others might take a day or two it is up to us to find the right balance. Holding time is a simple example which doesn't bring a lot of risk with it if we test on different holding times. As long as we use a limited amount and have clear differences between the different holding times we can be sure that the strategy is robust. The biggest risk we face with backtesting is that we might overfit the strategy to the past data. An overfitted strategy looks great on historical data but will fail in the live market.

Live Execution: If the backtesting results are promising and the strategy appears robust, it can be deployed live. Execution is handled by a bot or script running on a server, which listens to market data and acts as soon as the conditions are met. New things have to be taken in consideration here whicha are not a problem in backtests. One such thing is the time it takes to calculate the entry and exit conditions. This is a problem, as the price might change between when the actual entry conditions occur and when the entry signal is calculated.

Throughout this process, the focus remains on consistency and measurable results. Unlike human decision-making, a systematic strategy must behave identically entry and exit conditions. This makes it possible to evaluate, improve and automate trading in a transparent way.

3 Methodology

3.1 Project Overview

The aim of this project was to design, test, and deploy a fully automated trading strategy for Bitcoin. My development process consisted of four main stages. First, historical price data was collected to backtest strategies on. In the second stage I try to develop an edge by analyzing the data I collected and define a signal from it. In the third step, the entry and exit conditions are backtested on several months of historical data using a custom Python-written framework. Finally, if the results met predefined criteria (e.g. acceptable drawdown and stable returns), the strategy was deployed live on a PaaS that connects to the exchanges where the trades are executed in real time.

3.2 Data collection

I chose to create my own dataset because it offers a key advantage: it allows me to experiment with ideas and patterns that are less likely to have been explored before as I am not limited to the data provided by a platform. This increases the chance of discovering something new that might be missed when using more commonly available datasets.

There are platforms that offer orderbook delta data for free, but only for discretionary trading (i.e. they only display the data in a chart and do not provide it as a downloadable time series). My idea was to create a Python script which hits the API in one-minute intervals and stores the data inside a Postgres database. I have been hosting the data mining script on my PaaS since March 11, 2025 Live database connection.

The creation of the time series was done through a public exchange API[1], with data fetched in one-minute intervals and stored inside a database on a PaaS[2].

In my dataset, I calculate the 1%, 2.5%, and 5% delta of the orderbook. Additionally, I include the current Bitcoin price and the corresponding timestamp. The program I wrote collects data by sending API requests to the Coinbase Exchange API at one minute intervals. It retrieves both the current Bitcoin price and a full snapshot of the orderbook at that moment. A complete snapshot is necessary to calculate orderbook deltas at different percentage depths accurately.

---

[1]API stands for Application Programming Interface. It is a tool that allows programs to request and receive specific data from external services—in this case, used to fetch real-time Bitcoin price and orderbook data from the Coinbase crypto exchange.

[2]PaaS stands for Platform as a Service. It is a cloud computing model which gives developers a platform to host their applications so that they are globally accessible on the internet and run 24/7

$$\Delta_{OB} = \sum_{i=1}^{n} V_{bid_i} - \sum_{i=1}^{n} V_{ask_i} \tag{1}$$

I got the formular for the orderbook delta calculation from: Hypeblock capital

3.3 Framework

Now before diving into the development of a systematic trading strategy I want to explain what kind of services we need and how we can use them to our advantage.

Broker

In order to buy and sell Bitcoin we need an exchange to trade on and the exchange should offer a modern API infrastructure, which allows my trading algorithm to send commands to the exchange to sell and buy Bitcoin. The most fitting exchange for this is Hyperliquid. As it offers a modern API infrastructure and a very comprehensive documentation with a lot of examples Hyperliquid Python SDK.

Hosting services

Running the algorithm on my personal computer is not an option as it would require me to have my computer running 24/7. This is not feasible and would also require me to have a stable internet connection. For this reason I need a hosting service that can run my algorithm 24/7. I chose Railway since it is very simple to use and I already know how to host my code there as I have been using it for a long time for other projects. Railway is a PaaS (Platform as a Service) which allows me to host my code on a server and run it remotely.

Backtesting Python libary

A python library is a collection of tools and libraries that allow developers to interact with a specific API. For backtesting I chose Vectorbt as it offers a comprehensive set of tools for backtesting and has a very comprehensive documentation with a lot of examples Vectorbt Documentation.

3.4 Defining goals

The most difficult part of developing a trading algorithm is finding a strategy that is actually worth integrating. As explained in the Efficient Markets section, it is in principle possible to find a strategy that generates profit. The problem is that finding a strategy which consistently makes money and outperforms the market is still extremely difficult.

Before I start searching, I need to define what I am searching for. To do this, I use the S.M.A.R.T. goal framework (Specific, Measurable, Attainable, Relevant, and Time-bound), as described in the book *Building Winning Algorithmic Trading Systems* by Kevin J. Davey.

Goals

- Two months time to find a strategy
- Strategy operates on the Bitcoin/USD pair using Hyperliquid
- Target Sharpe ratio between 1 and 2 [5]
- Approximately 20 trades per month
- No strict requirement to outperform the market; focus is on stability and consistency

---

[5]The Sharpe ratio is a measure of risk-adjusted return. It compares the average return of a strategy to its volatility. A higher Sharpe ratio indicates a better balance between risk and return. A Sharpe ratio between 1 and 2 is considered decent in most financial contexts.

3.5 Strategy Search Methodology

I evaluate a strategy with two different processes. First I search for a potential edge in the entry condition. Then I backtest the strategy with the entry condition that has the best performance in my randomized testing.

Initial Edge Discovery

Instead of immediately running a complete backtest on a single strategy implementation, I wrote a custom python frame work which first tests whether a potential entry condition appears to have good average returns. This is accomplished by running 1000 different variations of the same base strategy. Each variation differs in entry condition by testing different holding periods for a trade.

For example, if the base entry condition identifies 1000 potential trade entries over the historical data, each strategy variation will randomly sample different subsets of these entries (e.g., taking 20%, 80%, or 100% of all signal s). This creates 1000 unique strategies that all use the same entry condition but trade different specific instances of it.

The key insight of this approach is that if the entry condition contains no edge, the performance across the different holding periods and sample sizes should have a negative return. Now the question might come up why not revers a signal? For a reversed signal to be profitable, the absolute value of the average return per trade has to exceed the transaction cost including fees and slippage most of the time this is not the case.

However, if there is a genuine edge in the entry condition, we should see consistent performance characteristics across the variations, even though each strategy trades different specific signals.

Parameter Space Exploration

To further validate any potential edge, each of the 1000 variations is tested across different holding periods (periods between open and close of trade). This adds another dimension to the analysis, not only are we testing if the entry condition has merit [1], but also how long positions should be held to best capture any edge.

The process runs these tests in batches of 100 strategies at a time to manage computational resources efficiently. For each combination of sampling percentage and holding period, the system calculates key metrics averaged across all strategy variations. This provides robust statistical evidence of whether the entry condition has merit, as we're looking at aggregate performance across thousands of different implementations rather than a single backtest that might be influenced by luck.

---

[1]How much the test improves decision-making for a user compared to decisions made before the test.

Statistical Validation Before Detailed Backtesting

This methodology acts as a powerful filter. Only if we see good average returns, we proceed to detailed backtesting. This prevents wasting time on running every parameter combination separately and we can directly compare outcome of the different parameter combinations. It's a way of asking "Does this entry condition have any merit?" before asking "How can we best implement this entry condition?"

This approach is particularly powerful because it separates the question of whether an edge exists from the question of how to best exploit that edge. Only after establishing statistical evidence of an edge through this randomized testing do we move forward with detailed backtesting and optimization of a specific implementation.

Walk-forward testing

Even if we search for a strategy by using this multi parameter testing framework. In order to prevent to being lucky with a good looking parameter combination I run the same strategy on new data. This way I can see if the strategy is still profitable and if it is I can be sure that it is not just a lucky hit. First test is done on 40'000 minutes of data if the strategy is profitable I run it on 60'000 minutes of data. If the strategy is still profitable I run it on 200'000 minutes of data. 95% of the time the strategy goes from a good looking strategy to a bad one.
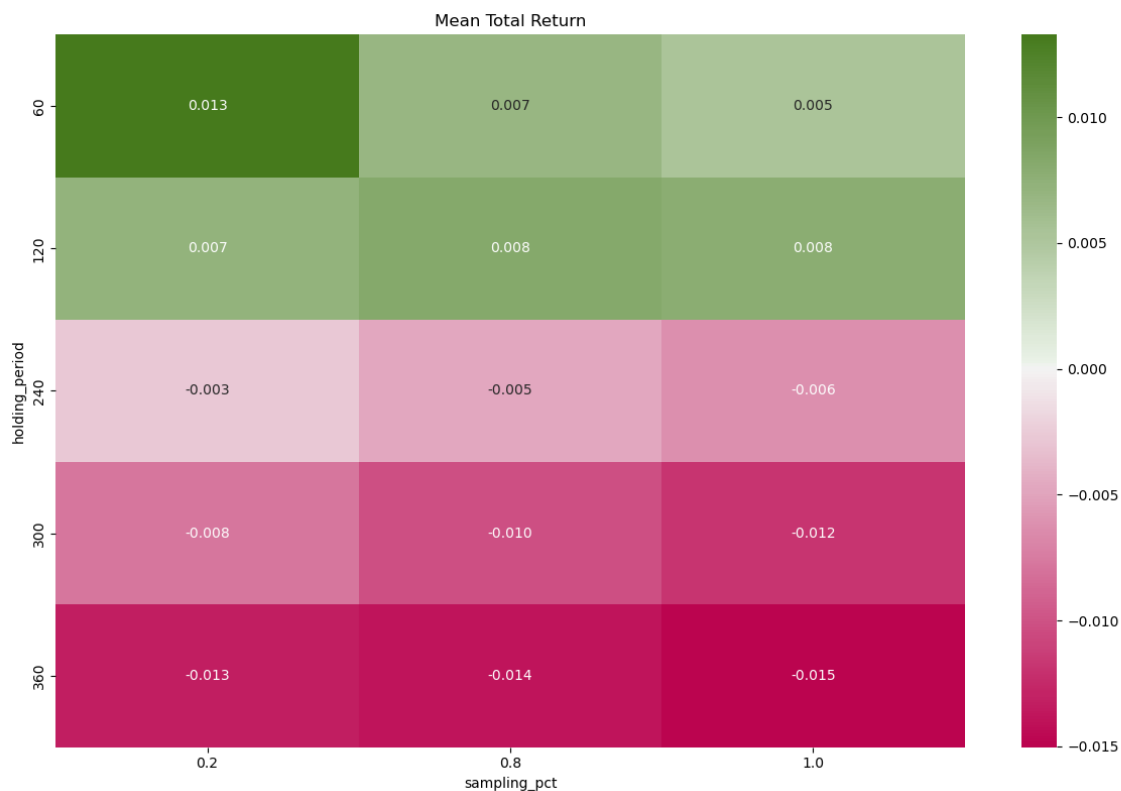
Figure 2: Backtest average return comparison

This is a visualization of the backtest results. The x-axis shows the sample size and the y-axis shows the average return over the different test runs. A value is colored green if the average return is positive and red if it is negative. The stronger the color the higher or lower the average return is.

Right here I just wanted to see what the average return of a P_t is which is inside of an uptrend.

```
# Entry conditions
entries = df['trend'] == 'Uptrend'
```

3.6 Strategy example

For showcasing purposes I will run a backtest on a strategy which is based on two different moving averages. We buy once the short moving average crosses the long moving average from below and sell once the short moving average crosses the long moving average from above.

Python implementation

```
import vectorBT as vbt
fast_ma = vbt.MA.run(price, 10)
slow_ma = vbt.MA.run(price, 50)
entries = fast_ma.ma_crossed_above(slow_ma)
exits = fast_ma.ma_crossed_below(slow_ma)
```
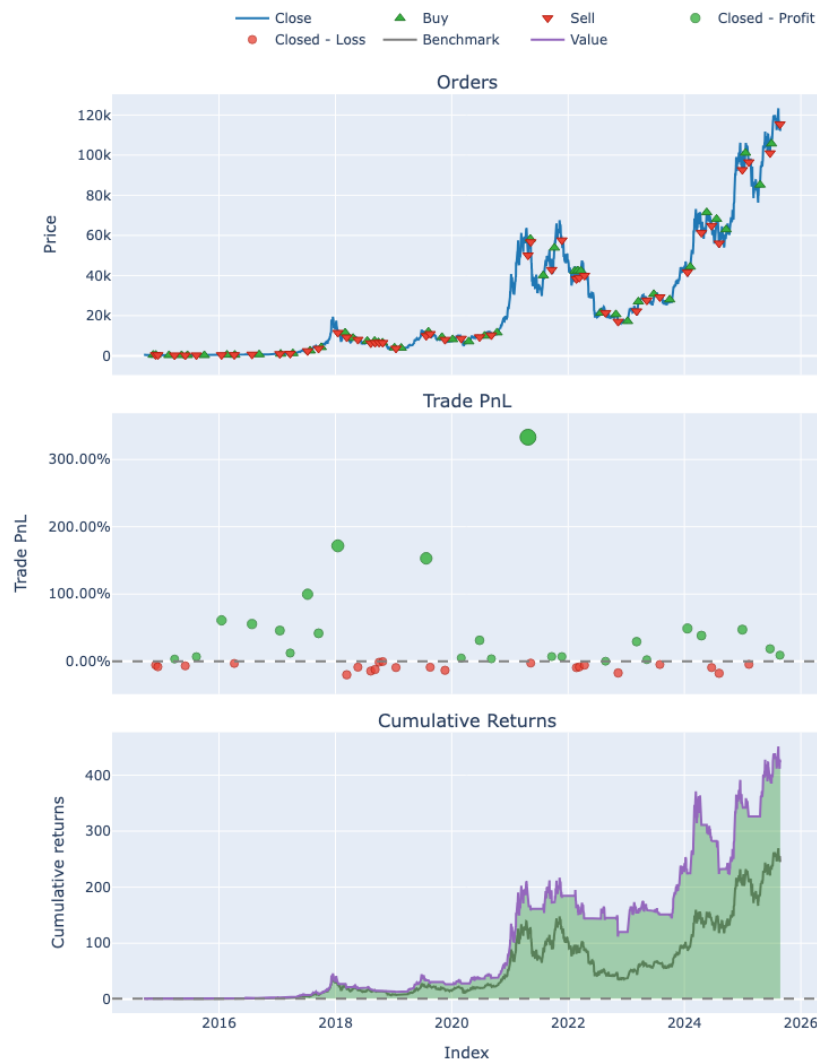

Figure 3: MA cross strategy

3.7 Final strategy

The area where I saw the most potential was the orderbook delta (see Equation 1). My hypothesis was that, since it directly reflects the relationship between buyers and sellers at specific price levels, it could serve as the basis for a statistical signal. By running volatility and directional bias tests under various orderbook delta conditions, I aimed to identify patterns that might be predictive of short-term price movements.

Figure 4: orderbook delta visualization showing price action (top) and corresponding delta values (bottom)

As shown in Figure 4, the orderbook delta provides valuable insights at extreme values. The chart displays two key components:

- The price action (top chart) shows the actual Bitcoin price movements
- The orderbook delta (bottom chart) represents the imbalance between buy and sell orders

A high orderbook delta value indicates substantial buy orders below the current price. This creates a strong support level, as sellers would need significant pressure to break through these orders. Even if sellers manage to push through, these accumulated buy orders often create buying pressure that can lead to price support or potential rebounds.

Strategy backtest implementation

Sharp ratio

To compare different strategies we should not ourselvs to only compare the total return a strategy achieves. Especially when we are not trying to outperform the market. Investing is associated with risk a good comparison between different strategies is to compare how much return we get for the risk we take. The sharpe ratio does exactly this. It is a measure of the return per unit of risk. I used this as the main metric to compare strategies. Following formular is used to calulate the sharpe ratio:

$$\text{Sharpe ratio} = \frac{\text{Average return } t}{\text{Standard deviation } t^6} \cdot \sqrt{98280}^7 \tag{2}$$

There are different variations of the sharpe ratio. The orginal version as defined by WIlliam Sharpe, uses the return of a strategy compared to a risk free rate, divided by the standard deviation of the return. How ever in the crypto industry the is no universally accepted risk free rate.

$$\text{Sharpe ratio} = \frac{\text{Average return } t \text{ - Risk free rate}}{\text{Standard deviation } t} \tag{3}$$

Backtesting results

- average sharpe ratio 1.55 [8]

- Over a time period of 137 days a return of 2.7% was achieved

Backtest parameters

- A stop lost of 2%

- Exit is 120 minutes after entry or if a regime change occurs

- fees of 0.0003% and slippage of 0.0001%

- accumulate $=$ True [9]

---

[7]98280 is used to annualize the Sharpe ratio we do this in order to compare the Sharpe ratio with the Sharpe ratio of other strategies or funds

[8]The $\overline{SR}$ is the average Sharpe ratio over different walkforward periods

[9]This allows us to open multiple trades at the same timei

Code entry conditions

```python
# Entry conditions
entries = (df['context'] == 'b') & (df['trend'] == 'Uptrend')



# Exit after 120 minutes
time_exit = exitSlots.shift(120).fillna(False)

# Exit on trend change
trendChange = (df['trend'] != df['trend'].shift(1))

#Or statment to exit
exits = trendChange | time_exit
```

Entry conditions calculations

I have to give more context about the entry conditions. **df_temp** is just a variable to which I assign a timeseries dataframe [10] with different kind of metrics inside of it. The entry conditions are defined by two boolean conditions:

Outlier context

```
df_temp['outlier_context'] == 'b'
df_temp['trend'] == 'Uptrend'
```

As shown in Figure 5, the dataframe contains various metrics that we use for our trading decisions:

```
df15 len:  212458
                       delta1     delta2_5      delta5       price          ema  directional_pressure  is_outlier  outlier_signal_strength  outlier_context  trend
timestamp
2025-03-11 23:37:01  -1.786011  -29.817527  -51.350122   82681.515          NaN                   NaN           0                      0.0                     NaN
2025-03-11 23:38:01  -9.159898  -28.886296  -59.164552   82658.855  82681.515000                  NaN           0                      0.0                     NaN
2025-03-11 23:39:01   0.963591  -17.596764  -48.747833   82758.125  82681.483550                  NaN           0                      0.0                     NaN
2025-03-11 23:40:02 -31.762113  -37.069179  -48.975283   82838.605  82681.589922                  NaN           0                      0.0                     NaN
2025-03-11 23:41:01   1.521426  -15.890205  -46.926685   82742.585  82681.807847                  NaN           0                      0.0                     NaN
```

Figure 5

The column $is\_outlier$ is a binary signal column if the current current orderbook delta value Δ_t is further than two standard deviations from the mean $\overline{\Delta}_{1440}$ of the orderbook delta. $outlier\_context$ is a string column that has a value of **b** if Δ_t has a postive Z score and **s** if it has a negative Z score.

Market Structure and Trend Identification

To identify trends in Bitcoin's price, we use a combination of price movement analysis and moving averages. The main challenge is to distinguish between real trends and temporary price movements. We start by converting our 1-minute price data into OHLC [11] 240 minute data to better see the overall market direction.

Price Movement Analysis

The core of our trend analysis looks at how price moves over time. We use two main indicators:

1. The derivative (rate of change) of a moving average:

$$\frac{d}{dt}\text{MA} = \text{Current MA value} - \text{Previous MA value}$$

---

[10]A dataframe is a two dimensional array with rows and columns

[11]OHLC stands for Open, High, Low, and Close. It is a type of data that is used to represent the price action of an asset over a period of time and is also known as a candlestick chart.

17

When this derivative is positive, it indicates an upward trend, and when negative, a downward trend.

2. Price levels comparison: We compare current prices with previous highs and lows using a window of time (w). This helps us confirm if we're really in a trend:

$$\text{Trend}_t = \begin{cases} \text{Uptrend} & \text{if price is making higher highs} \\ \text{Downtrend} & \text{if price is making lower lows} \\ \text{Ranging} & \text{otherwise} \end{cases}$$

Support and Resistance

We also look at support and resistance levels to strengthen our trend analysis:

- Resistance: Price levels where buying pressure tends to stop price from falling further

- Support: Price levels where selling pressure tends to stop price from rising further

The slopes of these levels help us determine trend strength. For example, rising support and resistance levels indicate a strong uptrend, while falling levels suggest a downtrend.

Final Trend Determination

To make the final decision about the trend, we combine all these factors:

$$\text{Final Trend} = \begin{cases} \text{Uptrend} & \text{if } \frac{d}{dt}\text{MA} > 0 \text{ and making higher highs} \\ \text{Downtrend} & \text{if } \frac{d}{dt}\text{MA} < 0 \text{ and making lower lows} \\ \text{Previous Trend} & \text{if moving average confirms} \\ \text{Ranging} & \text{if no clear direction} \end{cases}$$

This approach helps us filter out market noise and identify real trading opportunities. We only take trades when all these factors align, which increases our chances of success.

4   Results

For some context I will now compare the backtest results with the results of the live test.

5   Discussion

6 Appendix