AI in Trading

Exploring Autonomous Trading Bots with
Machine Learning and Optimisation

Nirius McDade & Ernest Zalewski

Exeter Mathematics School - 2025

September 2024 - May 2025

Contents

1 Project Introduction

Algorithmic trading, often referred to as algo trading, involves the use of computer programs and algorithms to execute financial market trades. It has fundamentally transformed the landscape of modern financial markets by improving execution efficiency, enabling high-speed trading, and reducing human error. Over the past two decades, algorithmic trading has evolved from a niche strategy used primarily by large institutional investors to a prevalent approach embraced by hedge funds, proprietary trading firms, and retail investors alike.

1.1 Historical Development

The start of algorithmic trading can be traced back to the early 1970s with the introduction of electronic communication networks (ECNs). These systems allowed for direct electronic trading without the need for traditional brokers, laying the groundwork for automation. The 1990s saw the emergence of more sophisticated algorithms designed to optimise trade execution, such as volume-weighted average price (VWAP) algorithms, which were particularly beneficial for institutional investors managing large orders [Kissell and Glantz(2003)].

By the mid-2000s, advancements in computing power and data analysis capabilities enabled high-frequency trading (HFT). These strategies rely on executing thousands of trades within fractions of a second, exploiting microsecond-level market inefficiencies. Studies such as that by Aldridge [Aldridge(2013)] have highlighted the significant role of HFT in increasing market liquidity and reducing bid-ask spreads, although concerns regarding systemic risk persist.

1.2 Key Concepts in Algorithmic Trading

At its core, algorithmic trading combines finance, mathematics, and computer science to implement strategies that leverage specific market conditions. Key concepts include:

- **Quantitative Analysis:** Algorithms often rely on statistical models to identify trading opportunities. These models range from simple moving averages to complex machine learning frameworks [Avellaneda and Lee(2008)].
- **Market Microstructure:** Understanding the mechanics of order books, price discovery, and liquidity is essential for effective algorithm design [Hasbrouck(2007)].
- **Risk Management:** Automated strategies embed predefined risk thresholds to mitigate exposure to volatile market conditions.

1.3 Current Relevance and Adoption

The adoption of algorithmic trading has grown exponentially, with estimates suggesting that it now accounts for 70-80% of US equity trading volume. This

growth is driven by increased market accessibility and the proliferation of tools that allow even retail traders to deploy algorithmic strategies. Platforms like MetaTrader, NinjaTrader, and QuantConnect have lowered the barrier to entry, enabling individuals with programming skills to compete in a space previously dominated by institutional players.

Despite its widespread adoption, algo trading remains a controversial subject. Critics argue that strategies like HFT exacerbate market volatility and create unfair advantages for those with superior technological infrastructure [Kirilenko et al.(2017)Kirilenko, Kyle, Samadi, and Tuzun].

1.4 Existing Strategies and Limitations

Algorithmic trading has been extensively studied and implemented in financial markets, with various strategies designed to exploit specific market behaviours. Despite their potential, each strategy has inherent limitations that traders must navigate. This section examines key strategies, their principles, and the challenges they face.

1.4.1 Mean Reversion Strategies

Mean reversion strategies operate on the principle that asset prices tend to revert to their historical average over time. These strategies use technical indicators such as Bollinger Bands and moving averages to identify overbought or oversold conditions, which signal potential price reversals.

Poterba and Summers (1988) provide foundational evidence for mean reversion, demonstrating that stock prices often exhibit this behaviour over extended periods. However, the practical application of mean reversion strategies is fraught with challenges. For instance, in volatile or trending markets, prices can deviate from their mean for prolonged durations, leading to substantial drawdowns. Furthermore, the assumption that past patterns will persist in the future can be undermined by structural market changes or unforeseen economic events.

1.4.2 Momentum and Trend-Following Strategies

Momentum and trend-following strategies seek to capitalise on sustained price movements in a particular direction. These strategies are predicated on the belief that assets exhibiting upward (or downward) momentum will continue their trajectory. Jegadeesh and Titman (1993) provide compelling empirical evidence for momentum effects in stock returns, showing that stocks that perform well over a 3–12 month period tend to continue outperforming [Jegadeesh and Titman(1993)].

Despite their widespread use, momentum strategies face limitations. In range-bound markets, where prices oscillate without a clear trend, such strategies are prone to frequent false signals, resulting in significant losses. Additionally, momentum trading can amplify price volatility during periods of excessive speculation, potentially exacerbating market instability.

1.4.3 Arbitrage Strategies

Arbitrage strategies exploit price inefficiencies between markets or related assets. Statistical arbitrage, for instance, involves using quantitative methods to identify correlations between assets and predict their convergence. Avellaneda and Lee (2010) highlight the mathematical models underpinning statistical arbitrage and their practical applications in equities markets [Avellaneda and Lee(2008)].

While appealing in theory, arbitrage strategies face considerable practical hurdles. The opportunities for arbitrage are typically short-lived and demand advanced infrastructure for high-speed execution. Additionally, transaction costs and diminishing inefficiencies in modern markets often erode the profitability of such strategies. Misjudging the convergence of asset prices can further result in unexpected losses, adding to the inherent risk.

1.4.4 General Limitations of Algorithmic Trading Strategies

Across all algorithmic strategies, certain universal limitations exist. Firstly, backtesting—while invaluable—cannot fully replicate live market conditions, particularly in terms of slippage and latency. Secondly, strategies relying on historical data are inherently vulnerable to overfitting, which can result in poor performance in unseen market scenarios. Finally, algorithmic trading systems require constant monitoring and refinement to adapt to evolving market dynamics.

In summary, while existing strategies offer promising avenues for algorithmic trading, their limitations highlight the need for robust optimisation and continuous evaluation. Understanding these constraints is critical for designing trading bots that can navigate the complexities of financial markets effectively.

2 The Problem

Despite the rapid advancements in algorithmic trading, there remain significant challenges in optimising trading strategies to maximise profitability and minimise risk, particularly in the context of autonomous trading bots. Current strategies, while effective in certain market conditions, often struggle to adapt to the dynamic and highly volatile nature of financial markets, especially in the cryptocurrency space.

One of the primary issues with existing strategies is their reliance on static parameters, which fail to respond to changing market dynamics. This results in suboptimal performance during periods of high volatility or when market conditions shift unexpectedly. For example, mean reversion strategies can lead to significant losses during trending markets, while momentum strategies may fail in sideways markets.

Furthermore, backtesting and historical performance evaluations of algorithmic trading systems are frequently undermined by overfitting, poor risk management, and an inability to account for unforeseen market events. The lack of

continuous optimisation and adaptability in trading bots, as well as the difficulty in accurately predicting market movements, necessitates the development of more sophisticated models that integrate machine learning, optimisation techniques, and dynamic risk management.

This project aims to address these gaps by developing an intelligent and adaptive trading bot capable of performing robust decision-making under varying market conditions. The goal is to create a strategy that balances profitability with risk mitigation, while continuously optimising itself based on historical data, market conditions, and real-time feedback.

2.1 Proposed Solution

The proposed solution leverages the 'freqtrade' library, a popular open-source cryptocurrency trading bot framework, to develop a custom trading strategy that can adapt to changing market conditions. The solution focuses on creating an intelligent bot capable of dynamically adjusting its strategy based on market signals, risk management rules, and performance feedback.

To achieve this, we will design a flexible trading strategy that maximises profitability by identifying high-potential trading opportunities while minimizing risk through well-defined stop-loss and take-profit mechanisms. Additionally, the bot will attempt to use machine learning techniques to continuously improve its decision-making process based on historical data and real-time market conditions. By integrating custom indicators and using backtesting and optimisation frameworks like 'hyperopt', the strategy will evolve to remain effective in various market environments, ensuring long-term success.

The bot will aim to:

- Maximise overall profit.
- Reduce drawdowns and exposure to risk.
- Adapt to volatile and uncertain market conditions.
- Optimise performance through continuous learning and parameter tuning.

The solution will also focus on ensuring that the bot can trade across multiple cryptocurrency pairs, balancing the portfolio based on market signals, while ensuring proper risk management and diversification.

2.2 Objectives

The main objectives of this project are as follows:

1. Develop a custom trading strategy using the 'freqtrade' framework tailored for cryptocurrency markets.
2. Optimise strategy parameters using the 'hyperopt' library to fine-tune the performance and minimise overfitting.

3. Backtest the strategy against historical data to evaluate its effectiveness and refine it for live trading conditions.

4. Compare the performance of the developed strategy against benchmark strategies to assess improvements in profitability and risk management.

5. Implement dynamic risk management mechanisms to minimise drawdown and optimise the risk-to-reward ratio.

6. Enable the bot to adapt to different market conditions, ensuring that it remains effective across various cryptocurrency pairs and market volatility levels.

3 Getting Started

In order to complete this project, we used the `freqtrade` Python repository. This is an extension for Python that allows us to create, analyze, and manage trading bots. A trading bot is simply a program that autonomously performs actions on an online market. By creating a custom trading strategy and applying it to this Python package, we can develop programs that automatically trade for us as long as the program is running.

3.1 Installation and Setup

To install `freqtrade`, you need to install `TA-Lib` first, a Python technical analysis library that allows us to create indicators from raw data. After installing `TA-Lib`, I proceeded to install `freqtrade` along with its dependencies.

3.1.1 Creating the User Directory

Firstly, we need to set up `freqtrade` after installation. We start by creating a folder to store our work. This is easily done by running the following command:

```
freqtrade create-userdir --userdir user_data
```

This command creates a working directory called `user_data` with the following layout:

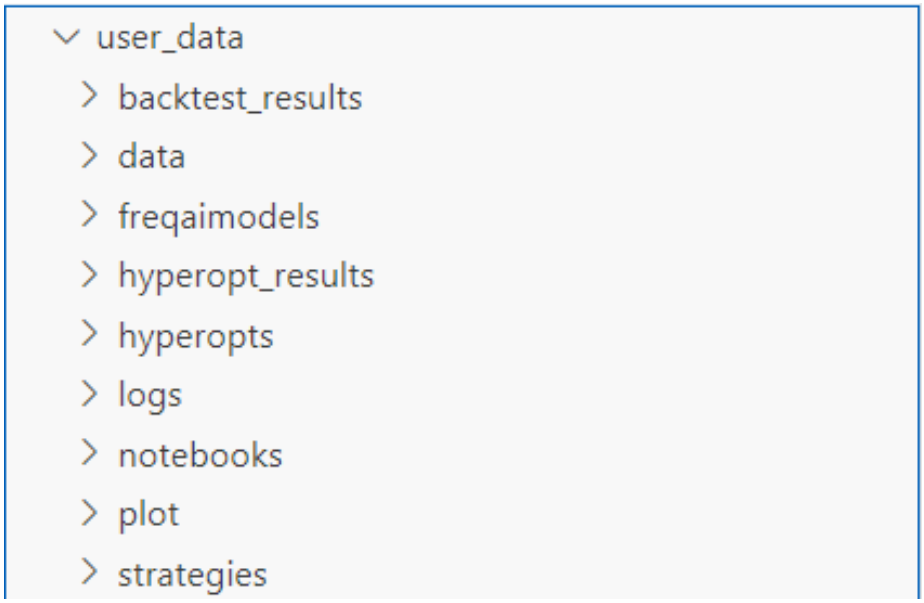

Figure 1: User Data Directory Layout

3.1.2 Creating a Configuration File

Next, we create a new configuration file using the following command:

```
freqtrade new-config --config user_data/config.json
```

This file determines key factors about our trading strategies. We modified the default configuration file to suit our needs. Below is our updated configuration file:

```
{
    "$schema": "https://schema.freqtrade.io/schema.json",
    "max_open_trades": 3,
    "stake_currency": "USDT",
    "stake_amount": "unlimited",
    "tradable_balance_ratio": 0.99,
    "fiat_display_currency": "USD",
    "dry_run": true,
    "dry_run_wallet": 1000,
    "cancel_open_orders_on_exit": false,
    "trading_mode": "futures",
    "margin_mode": "isolated",
    "unfilledtimeout": {
```

```
        "entry": 10,
        "exit": 10,
        "exit_timeout_count": 0,
        "unit": "minutes"
    },
    "entry_pricing": {
        "price_side": "same",
        "use_order_book": true,
        "order_book_top": 1,
        "price_last_balance": 0.0,
        "check_depth_of_market": {
            "enabled": false,
            "bids_to_ask_delta": 1
        }
    },
    "exit_pricing":{
        "price_side": "same",
        "use_order_book": true,
        "order_book_top": 1
    },
    "exchange": {
        "name": "binance",
        "key": "",
        "secret": "",
        "ccxt_config": {},
        "ccxt_async_config": {},
        "pair_whitelist": [
            "BTC/USDT:USDT",
            "ETH/USDT:USDT",
            "XRP/USDT:USDT",
            "SOL/USDT:USDT",
            "DOGE/USDT:USDT",
            "ADA/USDT:USDT"
        ],
        "pair_blacklist": [
            "BNB/.*"
        ]
    },
    "pairlists": [
        {
            "method": "StaticPairList",
            "pairs": [
                "BTC/USDT:USDT",
                "ETH/USDT:USDT",
                "XRP/USDT:USDT",
                "SOL/USDT:USDT",
```

```
                "DOGE/USDT:USDT",
                "ADA/USDT:USDT"
            ]
        }
    ],
    "telegram": {
        "enabled": false,
        "token": "",
        "chat_id": ""
    },
    "api_server": {
        "enabled": true,
        "listen_ip_address": "127.0.0.1",
        "listen_port": 8080,
        "verbosity": "error",
        "enable_openapi": false,
        "jwt_secret_key": "***",
        "ws_token": "***",
        "CORS_origins": [],
        "username": "***",
        "password": "***"
    },
    "bot_name": "freqtrade",
    "initial_state": "running",
    "force_entry_enable": false,
    "internals": {
        "process_throttle_secs": 5
    }
}
```

Most notably:

- `trading_mode` is set to `futures` as this allows the creation of strategies that can execute both long and short trades. This setting changes the pair list to include futures instead of typical spot pairs.
- The fiat currency is set to USD for ease of use, as the trading pairs use USDT, which is pegged to the dollar instead of the pound.
- `tradable_balance_ratio` is set to 0.99 instead of 1 to leave some balance for fees and commissions with the Binance brokerage.
- `api_server` enables visualization of the trading bot's actions by plotting its trades on the respective trading pair's graph. This can be accessed via the IP address listed in the config file through a web browser.

- Pairs were selected based on highest market cap, ensuring they are well-established and less risky compared to smaller market cap pairs, which are more susceptible to extreme price shifts or low liquidity.

3.2 Downloading Historical Data

After configuring `freqtrade`, we downloaded historical data to optimize our strategy using `hyperopt`, the optimization module in `freqtrade`. Data is downloaded using the following command:

```
    freqtrade download-data --config config.json --timerange 20200101-
--timeframes 1m 5m 15m 30m 4h 8h
```

This command retrieves all the pairs specified in our config file and collects price history for the indicated timeframes from January 1, 2020, to the most recent date (December 20, 2024, in this case). This allows us to test strategies across multiple timeframes and accurately replicate historical price action.

3.3 Creating and Evaluating a Strategy

We created a simple strategy to test and evaluate. I arbitrarily selected two indicators—the RSI and MACD—and modified the sample strategy from the `freqtrade` documentation to develop this custom strategy.

```
1  import numpy as np
2  import pandas as pd
3  from pandas import DataFrame
4  from freqtrade.strategy import IStrategy, IntParameter,
   ↪  BooleanParameter
5  import talib.abstract as ta
6
7  class sample_strategy(IStrategy):
8      INTERFACE_VERSION = 3
9
10     can_short: bool = True
11
12     minimal_roi = {
13         "0": 0.1,
14     }
15
16     stoploss = -0.1
17
18     timeframe = '30m'
19
20     process_only_new_candles = True
```

```
21
22      use_exit_signal = True
23      exit_profit_only = False
24      ignore_roi_if_entry_signal = False
25
26      startup_candle_count: int = 200
27
28      buy_rsi = IntParameter(20, 80, default=30, space='buy',
        ↪  optimize=True)
29      sell_rsi = IntParameter(20, 80, default=70, space='sell',
        ↪  optimize=True)
30      short_rsi = IntParameter(20, 80, default=80, space='sell',
        ↪  optimize=True)
31      exit_short_rsi = IntParameter(20, 80, default=40,
        ↪  space='buy', optimize=True)
32
33      use_rsi = BooleanParameter(default=True, space='buy',
        ↪  optimize=True)
34      use_macd = BooleanParameter(default=True, space='buy',
        ↪  optimize=True)
35
36      max_open_trades = 3
37
38      def populate_indicators(self, dataframe: DataFrame,
        ↪  metadata: dict) -> DataFrame:
39          if self.use_rsi.value:
40              dataframe['rsi'] = ta.RSI(dataframe)
41          if self.use_macd.value:
42              macd = ta.MACD(dataframe)
43              dataframe['macd'] = macd['macd']
44              dataframe['macdsignal'] = macd['macdsignal']
45              dataframe['macdhist'] = macd['macdhist']
46          return dataframe
47
48      def populate_entry_trend(self, dataframe: DataFrame,
        ↪  metadata: dict) -> DataFrame:
49          conditions = []
50
51          if self.use_rsi.value:
52              conditions.append(dataframe['rsi'] <
                ↪  self.buy_rsi.value)
53          if self.use_macd.value and 'macd' in dataframe and
            ↪  'macdsignal' in dataframe:
54              conditions.append(dataframe['macd'] >
                ↪  dataframe['macdsignal'])
55
```

```
        dataframe.loc[
            np.all(conditions, axis=0),
            'enter_long'
        ] = 1

        conditions = []

        if self.use_rsi.value:
            conditions.append(dataframe['rsi'] >
            ↪  self.short_rsi.value)
        if self.use_macd.value and 'macd' in dataframe and
        ↪  'macdsignal' in dataframe:
            conditions.append(dataframe['macd'] <
            ↪  dataframe['macdsignal'])

        dataframe.loc[
            np.all(conditions, axis=0),
            'enter_short'
        ] = 1

        return dataframe

    def populate_exit_trend(self, dataframe: DataFrame,
    ↪  metadata: dict) -> DataFrame:
        dataframe.loc[
            dataframe['rsi'] > self.sell_rsi.value,
            'exit_long'
        ] = 1

        dataframe.loc[
            dataframe['rsi'] < self.exit_short_rsi.value,
            'exit_short'
        ] = 1

        return dataframe
```

This Python code defines a trading strategy class named `SampleStrategy`, which inherits from `IStrategy`, a base class provided by the `freqtrade` framework. The `populate_indicators` method calculates these indicators and adds them to the dataframe. The `populate_entry_trend` method defines the conditions for entering long and short positions based on the values of these indicators. For example:

- A long position is entered if the RSI is below a certain threshold and the MACD is above its signal line.

- A short position is entered if the RSI is above a certain threshold and the MACD is below its signal line.

The `populate_exit_trend` method defines the conditions for exiting long and short positions based on the RSI values. The class is designed to be optimized using `freqtrade`'s `hyperopt` optimization, with parameters like `buy_rsi`, `sell_rsi`, `short_rsi`, and `exit_short_rsi` being adjustable within specified ranges.

3.4 Optimizing the Strategy with Hyperopt

After creating the strategy, I optimized it using `hyperopt` to obtain better values for the parameters. `Hyperopt` is an optimization framework used to find the best hyperparameters for machine learning models and trading strategies. It employs a Bayesian optimization approach, which is a probabilistic model-based method for finding the minimum or maximum of an objective function.
In the context of `freqtrade`, `hyperopt` uses a machine learning regressor algorithm (ExtraTreesRegressor) to model the relationship between hyperparameters and the performance of the trading strategy. The process involves iteratively selecting hyperparameter combinations, evaluating their performance, and updating the model to better predict which combinations are likely to yield the best results. This approach efficiently narrows down the search space by focusing on promising regions, making it far more efficient than a standard linear search.

Using `hyperopt` requires the following command:

```
    freqtrade hyperopt --strategy SampleStrategy --timerange 20240101-
--timeframe 30m --timeframe-detail 1m --hyperopt-loss SharpeHyperOptLoss
```

Important Notes:

- `timeframe-detail` is set to 1 minute to ensure the optimization and backtesting are adequately accurate by accounting for intra-candle price movement. This prevents the model from only seeing one low and one high per candle, which could lead to misleadingly perfect entries.

The illustration below demonstrates the dangers of a high `timeframe-detail`:

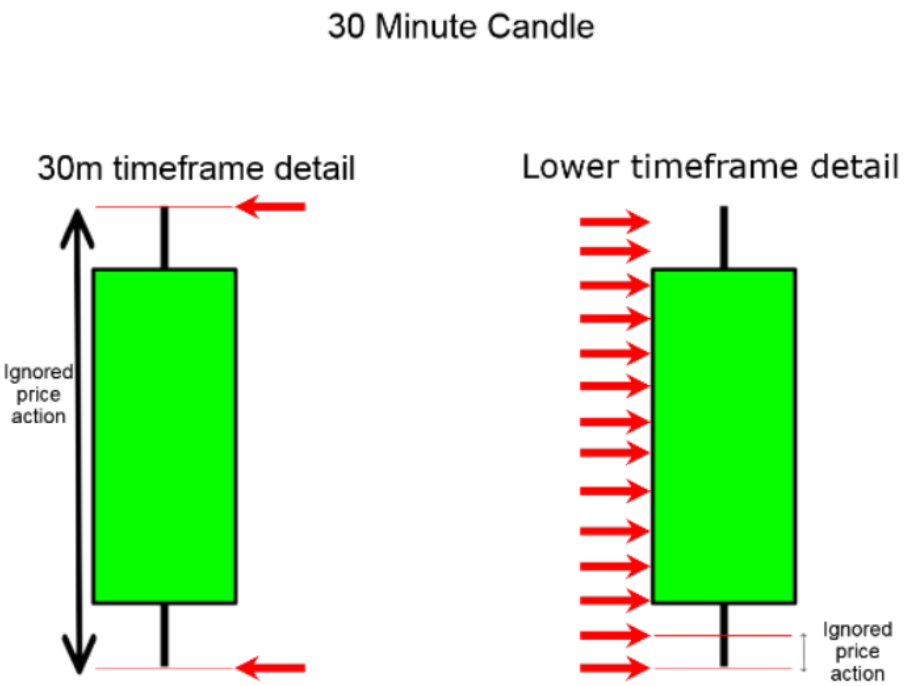

Figure 2: Comparison of Timeframe Detail Effects on Strategy Performance

After running the hyperoptimization for 100 epochs (note: most strategies should use 500+ epochs), the profit from the strategy increased from 12% to over 50% in one year.

Best	Epoch	Trades	Win	Draw	Loss	Win%	Avg profit	Profit	Avg duration	Objective	Max Drawdown (Acct)
Best	41/100	1279	710	23	546	55.5	0.09%	121.187 USDT (12.12%)	20:45:00	-0.56997	431.022 USDT (37.51%)
Best	50/100	1225	650	82	493	53.1	0.10%	426.127 USDT (42.61%)	21:15:00	-1.68128	828.303 USDT (51.74%)
Best	66/100	926	527	18	381	56.9	0.18%	526.595 USDT (52.66%)	1 day, 4:06:00	-1.79118	567.734 USDT (38.24%)

Figure 3: Hyperopt Optimization Results for Sample Strategy

3.5 Backtesting the Strategy

A further analysis of this strategy can be seen when using the backtest function for the same time range:

Enter Tag	Exit Reason	Trades	Avg Profit %	Tot Profit USDT	Tot Profit %	Avg
(”, ’roi’)		757	0.92	2654.178	265.42	
(”, ’force$_e xit'$)		3	-4.69	-72.486	-7.25	4 day
(”, ’exit$_s ignal'$)		145	-0.66	-351.637	-35.16	3 da

(”, ’stop$_{l}oss'$)		21	-19.92	-1703.459	-170.35	4 day
TOTAL		926	0.18	526.595	52.66	1 d

3.5.1 Summary Metrics

Metric	**Value**
Backtesting from	2024-01-01 00:00:00
Backtesting to	2025-01-05 19:00:00
Trading Mode	Isolated Futures
Max open trades	3
Total/Daily Avg Trades	926 / 2.5
Starting balance	1000 USDT
Final balance	1526.595 USDT
Absolute profit	526.595 USDT
Total profit %	52.66%
CAGR %	51.79%
Sortino	1.34
Sharpe	1.79
Calmar	7.11
Profit factor	1.18
Expectancy (Ratio)	0.57 (0.06)
Avg. daily profit %	0.14%
Avg. stake amount	387.094 USDT
Total trade volume	358448.908 USDT
Long / Short	925 / 1
Total profit Long %	52.25%
Total profit Short %	0.41%
Absolute profit Long	522.476 USDT
Absolute profit Short	4.119 USDT
Best Pair	ADA/USDT:USDT 33.35%
Worst Pair	SOL/USDT:USDT -5.67%
Best trade	XRP/USDT:USDT 16.99%
Worst trade	SOL/USDT:USDT -20.75%
Best day	129.324 USDT
st day	-234.136 USDT
s win/draw/lose	144 / 129 / 98
Avg. Duration Winners	6:28:00
Avg. Duration Loser	2 days, 9:23:00
Max Consecutive Wins / Loss	15 / 8
Rejected Entry signals	23,693
Entry/Exit Timeouts	0 / 0

Min balance	916.774 USDT
Max balance	1,875.071 USDT
Max % of account underwater	38.24%
Absolute Drawdown (Account)	38.24%
Absolute Drawdown	567.734 USDT
Drawdown high	484.508 USDT
Drawdown low	-83.226 USDT
Drawdown Start	2024-03-14 06:51:00
Drawdown End	2024-08-05 01:10:00
Market change	129.36%

Figure 4: A Plot of Some of the Trading Activity

3.6 Analysis and Conclusion

While our strategy achieved results significantly above the typical stock market benchmark, the market of the selected crypto pairs increased by over double our profit. This indicates that a simple buy-and-hold strategy would have been more competitive than our trading bot. Therefore, we need to further refine our bot to outperform the crypto market.

4 Code Appendix

References

[Aldridge(2013)] Irene Aldridge. *High-frequency Trading: A Practical Guide to Algorithmic Strategies and Trading Systems*. Wiley, 2013. URL `https://pdfroom.com/books/`

high-frequency-trading-a-practical-guide-to-algorithmic-strategies-and-trading-systems/Vo75Xpwe5aG.

[Avellaneda and Lee(2008)] Marco Avellaneda and Jeong-Hyun Lee. Statistical arbitrage in the u.s. equities market. *Quantitative Finance*, 10(7):761–782, 2008. URL https://ssrn.com/abstract=1153505.

[Hasbrouck(2007)] Joel Hasbrouck. *Empirical Market Microstructure: The Institutions, Economics, and Econometrics of Securities Trading*. Oxford University Press, 2007. URL https://www.forexfactory.com/attachment/file/794018?d=1317095892.

[Jegadeesh and Titman(1993)] Narasimhan Jegadeesh and Sheridan Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48(1):65–91, 1993. URL https://doi.org/10.1111/j.1540-6261.1993.tb04702.x.

[Kirilenko et al.(2017)Kirilenko, Kyle, Samadi, and Tuzun] Andrei Kirilenko, Albert S. Kyle, Mehrdad Samadi, and Tugkan Tuzun. The flash crash: High-frequency trading in an electronic market. *The Review of Financial Studies*, 30(11):2226–2256, 2017. URL https://dx.doi.org/10.2139/ssrn.1686004.

[Kissell and Glantz(2003)] Robert Kissell and Morton Glantz. *Optimal Trading Strategies: Quantitative Approaches for Managing Market Impact and Trading Risk*. Amacom, 2003. URL https://www.scribd.com/document/748567351/Robert-Kissell-Morton-Glantz-Optimal-Trading-Strategies-Quantitative-Approaches-for-Mana