University of Hradec Králové

Faculty of informatics and management

Algorithmic Trading

BACHELOR THESIS

Author: Burliei Dmytro

Study program: Applied Informatics

Supervisor: doc. RNDr. Kamila Štekerová, Ph.D.,MSc.

Hradec Králové April 2024

Declaration:

I declare that I prepared the bachelor thesis independently and with the use of the mentioned literature.

signature

In Hradec Králové 2024-04-20

Poděkování:
Děkuji vedoucí bakalářské práce doc. RNDr. Kamile Štekerové, PhD., MSc za metodické vedení a cenné rady, které byly klíčové pro dokončení mé práce.

Abstract

Title: Algorithmic trading

This bachelor's work examines how algorithmic trading has impacted financial markets. It looks at the benefits, like increased efficiency and removing emotions from trading decisions, as well as the drawbacks, such as higher systemic risk and market volatility. The main focus is on analyzing the behavioral patterns embedded in algorithms and how they relate to market behavior. It also highlights the needs of market participants that algorithmic trading meets and suggests improvements for trading systems. The results provide practical insights into the effectiveness of algorithmic trading strategies and give recommendations for enhancing these systems. Overall, this work emphasizes the importance of technology and its capabilities in modern financial markets.

Abstrakt

Název: Algoritmické obchodování

Tato bakalářská práce zkoumá, jak algoritmické obchodování ovlivnilo finanční trhy. Zabývá se výhodami, jako je vyšší efektivita a odstranění emocí z obchodních rozhodnutí, i nevýhodami, jako je vyšší systémové riziko a volatilita trhu. Zaměřuje se především na analýzu vzorců chování obsažených v algoritmech a na to, jak souvisejí s chováním trhu. Poukazuje také na potřeby účastníků trhu, které algoritmické obchodování splňuje, a navrhuje zlepšení obchodních systémů. Výsledky poskytují praktické poznatky o účinnosti algoritmických obchodních strategií a dávají doporučení pro zdokonalení těchto systémů. Celkově tato práce zdůrazňuje význam technologií a jejich možností na moderních finančních trzích.

Key words: Algorithmic Trading, Financial Markets, Trading Efficiency, Risk Management, Market Volatility, Behavioral Analysis, Automation, Bias, Market Trend, Trading Strategies, Machine Learning, Data Analysis, Market Integrity, Technological

Table of Contents

1 Introduction

In the dynamic world of finance, the integration of trading strategies with machine learning is changing the way market operations work. Comparable to the transformative impact of computers on society in the past half of the previous century, the intersection of trading and machine learning represents a significant evolution in how financial markets function nowadays.

Starting within algorithmic trading framework, ranging from basic trading algorithms to powerful machine learning models, each component plays a crucial role. One of main ideas behind integrating machine learning in trading, is to create advanced computational techniques that improve decision-making in financial markets.

Algorithmic trading involves usage of algorithms, which are essentially sets of rules or instructions, to execute trading orders with speed and efficiency.

By incorporating machine learning, traders aim to develop algorithms that can learn from historical data, recognize patterns, and adapt to changing market conditions.

However, manual integration is tough and time-consuming. It requires deep knowledge in finance and algorithms, making it impractical for large-scale trading. The challenges vary from frequent changes in trading strategies, market conditions, new financial instruments, and short-term trading models.

This complexity makes precise manual identification with positive financial outcome difficult. Creating automated solutions for this task is also tough and expensive. The dynamic nature of algorithmic trading components such as algorithms, learning models, data processing and the constant evolution of trading strategies add to the complexity. Consequently, many components end up mislabeled or without identification, impacting the smooth operation of algorithmic trading systems.Despite many advantages, algorithmic trading is not without its drawbacks. One significant challenge is the over-reliance on historical data. Furthermore, there is a need for various evaluation metrics used to assess the performance of trading algorithms.

Metrics such as the Sharpe ratio, maximum drawdown, and return on investment will be examined closely to understand how they contribute to a comprehensive evaluation of trading performance.

The objective of this thesis is to explore the intricacies of algorithmic trading, its advantages over traditional trading methods, and the challenges it faces. The appllication will be proposed and assess how well algorithmic strategies perform in trading differenr market's assets. By dissecting the influence of algorithmic trading, this study aims to illuminate its implications for market liquidity, efficiency, and stability in the current financial landscape. The structure of chapters is as follows:

1. **Intersection of Machine Learning and Trading**: Explores the convergence of AI's predictive capabilities with financial strategies, setting the stage for the development of AT.
2. **Algorithmic Trading**: This section will explore the principles and advantages of algorithmic trading, including how it differs from traditional trading methods and the benefits it offers in terms of efficiency and profitability.
3. **Market Mechanics and Dynamics in Algorithmic Trading:** We will discuss the criteria for selecting effective trading strategies, understanding the basics of the market, analyzing the roles of buyers and sellers, identifying market trends, and explaining the importance of pips in trading.
4. **Case Studies and Examples**: Here, we will present various real-world examples and case studies of algorithmic trading to demonstrate its impact on high-frequency trading and market dynamics.
5. **Evaluation Metrics**: In this chapter, we will delve into the metrics used to evaluate algorithmic trading strategies, such as profit and loss, Sharpe ratio, maximum drawdown, and others.
6. **Challenges in Algorithmic Trading**: This part will discuss the challenges and biases inherent in algorithmic trading, such as overfitting, data snooping, and behavioral biases.
7. **Summary**: A summary will summarize the main points discussed in the previous sections, providing a brief overview before moving on to the models and results.
8. **Model development and Results**: In this section of the thesis, we will present the quantitative models that were developed and the results obtained from testing the algorithmic trading strategies.
9. **Trading strategies**: We will explore specific algorithmic trading strategies, providing details about their theoretical foundations and practical applications.
10. **Implementation**: We will discuss the practical implementation of the selected trading strategies, including the choice of programming languages and libraries,

connecting to APIs, gathering data, visualizing techniques, and explaining the step-by-step process of strategy implementation.

11. **Evaluation**: We will evaluate the effectiveness of the implemented strategies using a rigorous methodology, determining their success and identifying areas for potential improvement.

2 State of Art

2.1 Intersection of Machine Learning and Trading

Machine learning, as a subfield of artificial intelligence, found its part in algorithmic trading systems in recent years. It involves the use of statistical techniques to enable computers to learn from historical data and make predictions or decisions without being explicitly programmed for every scenario, so they can "think" on their own. In the context of trading, machine learning algorithms can be trained to identify patterns and trends in financial market data, which can then be used to make informed trading decisions.

To understand the roots of machine learning and trading, we need to examine the evolution of trading practices. In the early days, human traders relied on their intuition, technical analysis, and fundamental analysis to make trading decisions. These methods were time-consuming and subjective, creating an opportunity for automation.

With the advent of computers in the mid-20th century, trading practices began to change. The development of electronic trading platforms and high-speed communications allowed for faster and more efficient trading (6). However, it was not until the late 20th century that machine learning and algorithmic trading started to gain traction.

Machine learning refers to the development of algorithms that can learn and improve from experience without being explicitly programmed. Its application in trading began in the 1980s, with the emergence of statistical modeling techniques. These models aimed to identify patterns and predict price movements based on historical data. Although they were limited by the computing power of that time, they laid the foundation for future advancements.

The integration of algorithmic trading techniques with machine learning gained further momentum in the late 1990s and early 2000s. This period saw the rise of quantitative hedge funds, which heavily relied on mathematical models and automated trading strategies. These funds utilized machine learning algorithms to identify profitable trading opportunities and execute trades in real-time.

Advancements in data gathering and storage technologies allowed for the accumulation of vast amounts of financial data, and it became fuel for machine learning algorithms, which could extract valuable insights and patterns from these datasets.

Nowadays, there are countless areas where machine learning is used, one of them is predictive modeling, where models utilize historical data, such as price movements or market indicators, to forecast future trends and identify potential trading opportunities (4).

In addition to predictive modeling, machine learning techniques are also employed in risk management, where algorithms can be designed to continuously monitor the market and assess potential risks. By utilizing machine learning algorithms, these systems can identify unusual or unexpected behavior in real-time, helping traders mitigate potential losses and minimize risks.

To summarize, from the early days of trading based on intuition to the current era of trading driven by complex algorithms, where algorithms trade large volumes of securities in fractions of a second, it has become the norm. It is clear that machine learning and algorithmic trading have reshaped market dynamics. This historical context shows us the significance of these advancements, and points to potential risks and challenges they pose.

2.2 Algorithmic Trading

2.2.1 Principles

Algorithmic trading, also known as algo-trading, is a method of executing trades using mathematical models and automated systems. This has rapidly gained popularity in recent years due to its potential to enhance efficiency, liquidity, and profitability in trading activities.

One of the key figures in the history of algorithmic trading is David Shaw, an American computer scientist and hedge fund manager. Shaw founded D.E. Shaw & Co. in 1988, which became one of the pioneers in applying quantitative mathematical models to trading. His firm developed sophisticated algorithms that sought to identify and exploit anomalies in financial markets. Shaw's success with algorithmic trading strategies helped pave the way for the widespread adoption of this practice (10).

Proceeding to the modern era of trading, we can see how quickly algo-trading became a standard in the industry.

	% from AT	
	Volume	Message Traffic
E-mini S&P 500 Futures	51.66%	69.93%
EuroFX Futures	69.32%	83.41%
Eurodollar Futures	51.29%	64.46%
10-Yr T-Note Futures	49.88%	68.33%
Crude Oil Futures	35.34%	71.24%

Figure 1: Proportion of algorithmic trading usage in different markets for the year 2010.

Source: Algorithmic Trading and Market Dynamics, 2010

The highest trading volume was in futures such as EuroFX, followed by stock index futures such as the E-mini S&P 500. Interest rate markets including Eurodollar and 10

Year Treasury note futures are close behind with commodities such as crude oil displaying the least amount of algorithmic trading activity (6).

Throughout the years, algo-trading has increased market liquidity, however, not without its drawbacks. One of the main concerns associated with this practice is the potential concerns regarding market fairness and integrity. Some argue that big industry players, armed with advanced technology and faster access to market information, gain an unfair advantage over other market participants. This advantage, they claim (6), can lead to market manipulation and disrupt the natural functioning of financial markets.

In the United States, the Securities and **Exchange Commission (SEC)** has implemented rules and regulations to ensure fair and orderly markets. These regulations aim to promote transparency, reduce market manipulation, and reduce the risks associated with algorithmic trading (7).

In terms of future developments, algo-trading is expected to continue evolving and expanding its reach, alongside the advancements in artificial intelligence and machine learning.

2.1.2 Advantages

To explore the advantages of algo-trading, we need to have a deeper understanding of how it works and how it differs from traditional.

Essentially, it is the process of using computer programs to execute trades based on a set of pre-defined rules. These rules are often developed with the help of complex mathematical models and algorithms that consider various parameters, such as market conditions, price movements, and trading volumes.

This approach differs from traditional trading mainly in use of an algorithm to execute buy or sell orders, based on pre-defined rules. On the other hand, humans make trading decisions based on their judgment and analysis.

Considering that, here are main advantages of algo-trading (2):

- **Speed and efficiency.** Algo-trading systems can analyze and process market data in real-time, enabling traders to execute trades at lightning-fast speeds. This eliminates the need for manual intervention, reducing the risk of human errors such as emotions being involved or simple panic, and delays in trade execution.
- **Continuous 24/7 operation**, allowing for round-the-clock trading in multiple markets worldwide. This global reach gives algorithmic traders a significant advantage over manual traders who may be limited by time zones or any other physical constraints.
- **Precision**, while executing complex trading strategies, algo-trading systems can handle large volumes of data, process it efficiently, and execute trades across multiple markets simultaneously. This capability enables traders to diversify their portfolios, hedge risks, and take advantage of arbitrage opportunities.
- **Reduced Transaction Costs,** which often results in lower transaction expenses due to the absence of manual intervention and the ability to execute trades at optimal prices. This efficiency can significantly impact the overall profitability of trading strategies, especially for high-frequency trading where the volume of transactions is high.
- **Backtesting Capability** is one of the significant benefits of algo-trading - to backtest trading strategies using historical data. This process allows traders to evaluate the effectiveness of a strategy by simulating its performance under various market conditions, thereby reducing the risk of potential losses in live trading.
- **Minimized Market Impact:** Large orders, when executed manually, can significantly affect the market price, leading to less favorable execution prices. Algo-trading can break down these large orders into smaller ones, spreading them over time or across multiple markets, thereby minimizing market impact.
- **Improved Risk Management:** Algorithmic trading systems can integrate sophisticated risk management rules and algorithms to monitor and control exposure to various market risks. This includes setting stop-loss orders, trailing stops, and other

conditional triggers that automatically execute or halt trades under certain market conditions.

- **Scalability,** which allows easily scale up to handle increasing amounts of data and more complex trading strategies. This scalability is crucial in today's rapidly evolving financial markets, allowing traders to expand their operations without a corresponding increase in operational complexity or costs.
- Unlike human traders, algorithms don't get tired, don't lose focus, or deviate from the planned strategy. This consistency ensures that the trading plan is executed **Consistency in Trading** precisely as intended, without the variations that human emotions or fatigue can introduce.

The advantages that algorithms gave to trading industry did not come without drawbacks and potential risks, which will be discussed later in this work.

2.3 Market Mechanics and Dynamics in Algorithmic Trading

3.1.1 Market Essentials

In its essence trading is the practice of buying and selling different assets to make a profit.

One of the most common methods for representing price movements in trading is through candlestick charts. A candlestick provides four key pieces of information for a given time period (16).

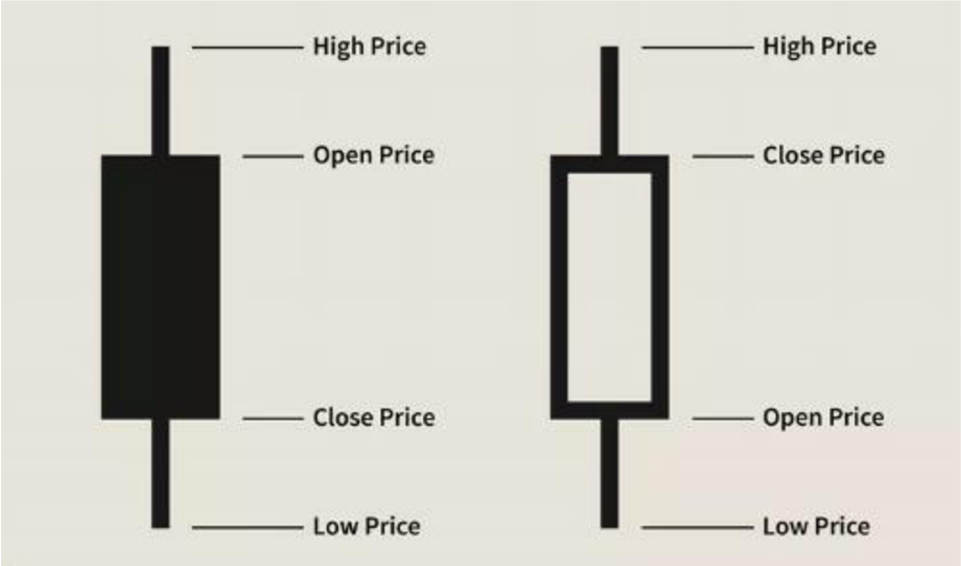


Figure 2: Candlestick body

Source: Understanding Basic Candlestick Charts

The main body of the candlestick shows the range between the opening and closing prices. A candlestick where the close is higher than the open is typically filled in or colored differently than a candlestick where the close is lower than the open.

The **"wicks"** or **"shadows"** that extend out of the body represent the high and low prices during the period. The upper shadow shows the highest price achieved, while the lower shadow shows the lowest price. Candlestick patterns can provide insights into market sentiment and potential price movements.

The colors of the candlestick body also carry meaning. A white or green body typically signifies that the closing price was higher than the opening price.

This indicates buying pressure and might be seen as a signal of upward continuation if it appears within an uptrend or potentially a reversal if it occurs after a downtrend.

3.1.2 Buyers and Sellers

The market is essentially constant competition between buyers and sellers.

Buyers, or **"bulls",** push prices up in anticipation of a currency strengthening, while sellers, or **"bears",** push prices down as they forecast a weakening. The price at any given moment reflects the latest agreement between buyers and sellers on what a currency pair is worth (17).

Volume is a crucial indicator of the strength behind price movements. A price move with relatively high volume is seen as more significant and likely to continue than a price move with low volume. Traders use various tools and indicators to identify potential buying or selling opportunities based on the actions of buyers and sellers.

3.1.3 Market Trends

Market trends in forex are broadly classified into three types (17):

- Bullish
- Bearish
- Sideway

A bullish market trend is characterized by rising prices and is typically driven by optimism and strong demand for a currency. Trading strategies in a bullish market expect prices to continue climbing and may look to enter "long" positions to profit from rising prices (17).

Conversely, a bearish market trend features falling prices and is generally driven by pessimism and strong supply exceeding demand. In a bearish market, traders expect prices to keep falling and may look to enter "short" positions to profit from the decline.

Figure 3: Gold (XAUUSD) chart against US dollar, 2024

Source: XAUUSD, tradingview.com, 2024

In finance, a bullish trend is recognized by higher highs and higher lows on a price chart, showing prices are rising over time. The Gold (XAUUSD) chart is a good example of this trend, with each peak and trough surpassing the previous ones, indicating strong investor confidence and demand for the asset.

The consistent upward movement in 2024 reflects positive market sentiment, leading to more investment as traders expect growth to continue. This momentum can attract more participants and push the trend further, as long as economic conditions and market fundamentals stay positive.

Figure 4: Gold (XAUUSD) chart against US dollar, 2024

Source: XAUUSD, tradingview.com, for year 2020 – 2021

Figure above shows (Fig. 4) a downtrend or bearish market for Gold as it was in bearish market, that can be seen in price falling continuation.

These trends will be crucial for picking a trading strategy and evolving our model, as it provides us with more reliable indicators to make informed decisions on entry and exit points. Technical analysis, including the study of chart patterns and indicators, is a key tool for identifying and confirming trends. However, it's also important to consider fundamental factors such as economic indicators and political events, especially news, that have a large impact on whether the market is going up or down (18).

3.1.4 Pips

A pip is the smallest increment by which an exchange rate can change in the forex market. It represents one-hundredth of 1% and is typically displayed in the fourth decimal place. For most currency pairs, a pip is equivalent to 1/10,000th of the price. For instance, in the USD/CAD pair, the smallest movement is $0.0001, which is equal to one pip. (15).

It's important to note that pips should not be confused with basis points (bps), which are used in interest rate markets and represent 1/100th of 1%. The value of a pip depends on various factors, including the currency pair, exchange rate, and trade value.

When your forex account is denominated in U.S. dollars and the USD is the second currency in the pair (quote currency), such as in the EUR/USD pair, the pip value remains fixed at .0001. To calculate the value of one pip in this scenario, you multiply the trade value (or lot size) by 0.0001. For example, if you have a trade value of 10,000 euros in the EUR/USD pair, the pip value would be $1. Therefore, if you bought 10,000 euros against the dollar at 1.0801 and sold at 1.0811, you would make a profit of 10 pips or $10.

On the other hand, if the USD is the first currency in the pair (base currency), such as in the USD/CAD pair, the pip value also takes into account the exchange rate. In this case, you divide the size of a pip by the exchange rate and then multiply it by the trade value (or lot size). For example, if the pip size is .0001 and the USD/CAD exchange rate is 1.2829, and you have a standard lot size of 100,000, the pip value would be $7.79. So, if you bought 100,000 USD against the Canadian dollar at 1.2829 and sold at 1.2830, you would make a profit of 1 pip or $7.79 (15).

3.1.5 Trading Strategies

There are numerous amounts of trading strategies for increasing price prediction probability, but for the sake of this work let's dive deeper into these 3: **Moving Average Crossover, Inside Bar Momentum, Candle Patterns.**
Each of them relies on different trading concepts and utilizes indicators to make predictions of price.

- **Moving Average Crossover** is one of the most straightforward and widely used technical analysis tools, that will be implemented and tested later on. It involves using two moving averages: one representing a shorter time frame and the other a longer time frame. A moving average smooths out price data to create a single flowing line, which makes it easier to identify the direction of the trend (21).

Figure 5: 20&100 days MA

Source: How to use MA to buy stock, 2024

It can bee seen (Fig. 5) how the price of 20-day MA follows price action responsively in comparison to 100-day MA for a longer time period.

In a Moving Average Crossover system, a "buy" signal is typically generated when a short-term MA crosses above a long-term MA, indicating the beginning of an uptrend. Conversely, a "sell" signal is suggested when a short-term MA crosses below a long-term MA, signaling the start of a downtrend. Traders often use the 50-period MA and the 200-period MA on their charts, looking for the strong bullish and bearish signals (21).

- **Inside Bar Momentum Strategy** is a powerful price action technique that reflects consolidation and can be a precursor to a significant breakout or breakdown. An inside bar pattern is a two-bar pattern where the inside bar is smaller and within the high to low range of the prior bar. The setup indicates a moment of 'pause' in the market, after which traders expect a breakout in the direction of the prevailing trend (22).

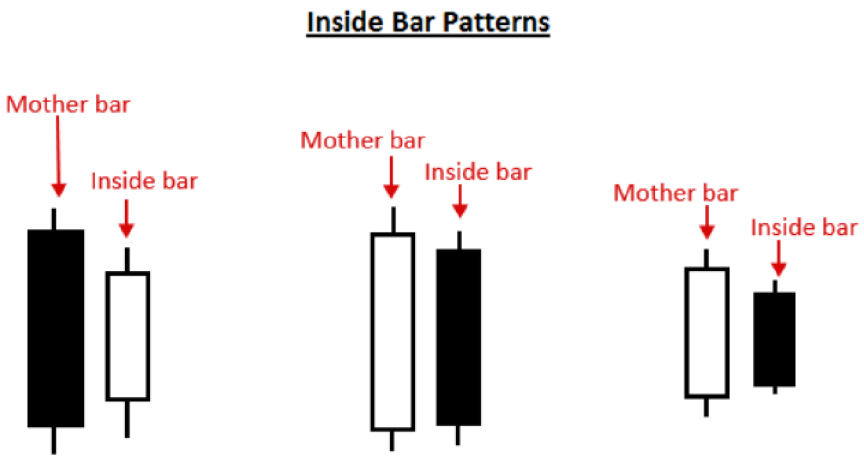


Figure 6: Inside Bar Patterns

Source: Trading inside bar strategy, 2024

Inside bars can be utilized in markets with a clear trend, aligning trades with the ongoing direction, commonly known as a 'breakout play' or an inside bar breakout pattern in price action trading. Alternatively, they can be applied against the trend from significant chart points, where they are typically labeled as inside bar reversal patterns.

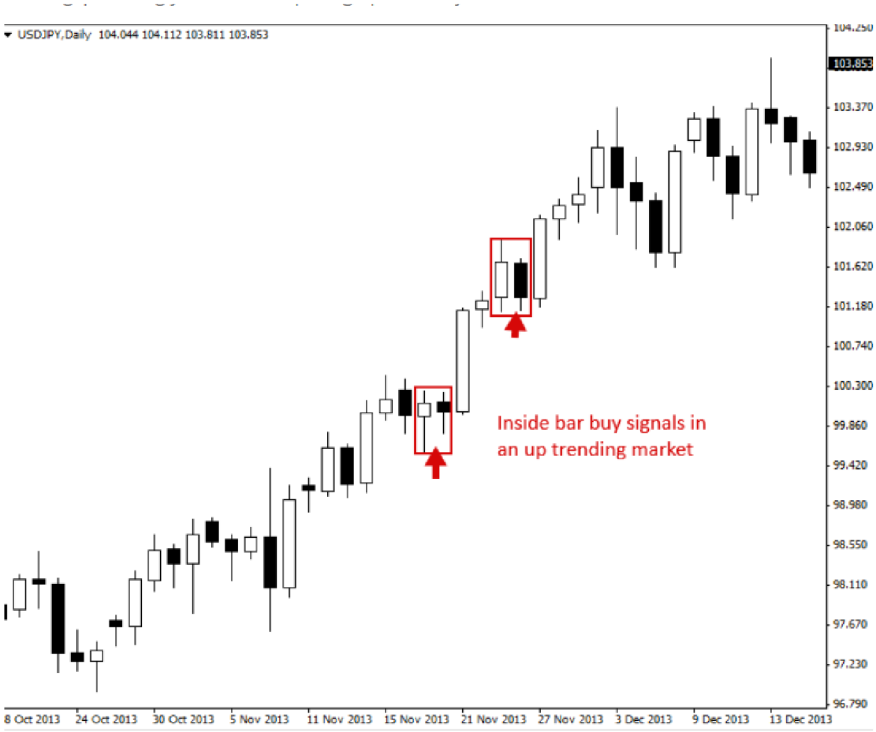


Figure 6: Using inside bar strategy in clear uptrend market

Source: Trading inside bar strategy, 2024

In this case (Fig. 6), the market was trending higher, so the inside bars would be referred to as 'inside bar buy signals. Often in strong trends like the one in the example above, we will see multiple inside bar patterns forming, providing us with multiple high-probability entries into the trend (22).

Different types of trading approaches, an inside bar against the recent trend / momentum and from a key chart level. In this case, we were trading an inside bar reversal signal from a key level of resistance. The inside bar sell signal in the example below actually had two bars within the same mother bar, this is perfectly fine and sometimes occures on the charts.

- **Candlestick patterns** are fundamental to technical analysis, offering preliminary signals for potential shifts in market direction or the persistence of current trends. Key patterns like the 'Doji', 'Hammer', 'Shooting Star', and 'Engulfing' are among the various candlestick configurations closely monitored by traders. Each of these patterns, formed by one or more candlesticks, can indicate either bullish or bearish sentiments.

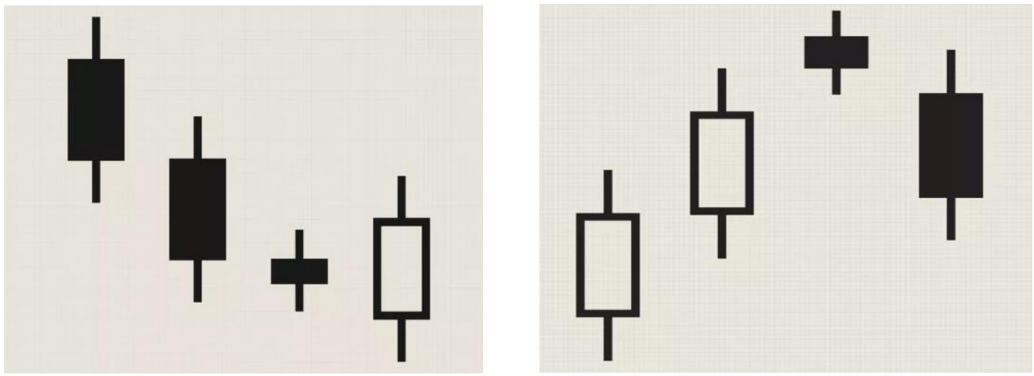


Figure 7-8: Different candle pattern examples

Source: Trading inside bar strategy, 2024

The bearish evening star on the right (Fig. 8) is a pattern that serves as a topping signal. It can be recognized when the last candle in the pattern opens below the small real body of the previous day. The small real body can be either black or white. Furthermore, the last candle closes significantly within the real body of the candle from two days prior.

This pattern indicates a slowdown in buyer activity and a subsequent shift in control towards the sellers. It suggests the possibility of further selling pressure. On the other hand, the morning star represents the bullish counterpart to the evening star (16).

A bullish engulfing pattern on the left (Fig. 8) is a two-candlestick formation that signals a potential reversal in trend. It consists of a smaller candle followed by a larger candle that 'engulfs' the body of the first. These patterns suggest a shift in momentum and are often used to identify entry and exit points in the market.

In trading we do not use candlestick patterns in isolation. Instead, they are often combined with other technical analysis tools to validate trade signals. For example, a bullish engulfing pattern found at a key support level with high volume can be considered a strong buy signal (16).

2.4 Case Studies and Examples

The intersection of machine learning and trading has garnered significant attention and interest from financial institutions and investors alike. Several case studies and examples highlight the potential of this integration and its impact on algorithmic trading.

2.3.1 High-Frequency Trading

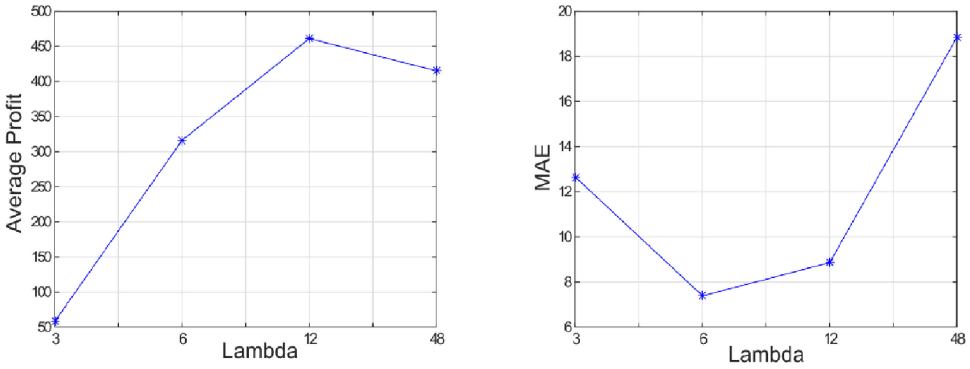
One notable case study is the use of machine learning algorithms in high-frequency trading (HFT) (4). HFT involves the execution of trades within microseconds, taking advantage of small price discrepancies and market inefficiencies.

One part of the study (4) shows how machine learning performs in high-frequency trading at different trading speeds, that are represented by λ in milliseconds.

Keep in mind that to simulate market conditions, uninformed traders are considered noise traders that trade without learning like zero-intelligence agents.

They choose their buy/sell order and order type randomly. Since firstly introduced by Gode and Sunder in 1993, zero-intelligence agents who trade randomly without learning and information have been used to examine their impact on trading mechanism and market environment.

For the benchmark HFT scenario HF there are 10 informed HF traders, 90 informed LF traders, and 900 uninformed LF traders.



(a) Average order profit (b) Price efficiency

Figure 9: Average order profit & MAE

Source: Machine learning and speed in high-frequency trading, 2022

The left graph (Fig. 9) shows how average profit correlates with trading speed. We can see that slightly longer trading speed results in more average profit, till it peaks at 12 milliseconds, where effectivity starts to decrease because of data amount that algorithm receives and tries to comprehend.

The right graph (Fig. 9) shows what trading speed has the lowest mean absolute error, which results in better predictive power of a model.

This research offers a comprehensive framework to analyze these dynamics. By integrating a Genetic Algorithm (GA) with a classifier system, it allows traders to learn and adapt based on a variety of market data, including historical prices, fundamental values, and order book dynamics.

The model demonstrates how all traders, high-frequency and low-frequency, engage with the market and each other through the limit order book, evolving their strategies via machine learning. This study sheds light on the future of market microstructure in an era where machine learning is popularized in trading.

A key finding is the balance between the information advantage and competition among informed HF traders, elucidating the observed hump-shaped relationship between trading speeds, liquidity consumption, and the profitability of HF trades, alongside price efficiency (4).

Interestingly, the study finds that market liquidity tends to decrease with the increased speed of HFT, leading to higher price volatility and wider spreads. However, the strategic trading behaviors, empowered by machine learning, offset the diminishing returns of speed-based advantages. This insight is particularly crucial for market regulation concerning HFT, suggesting that there's a profit-driven incentive for informed HF traders to avoid excessively rapid trading, thereby enhancing price efficiency (4).

In summary, the research (4) significantly contributes to the HFT literature by highlighting the role of machine learning in trading, particularly in the context of the evolving financial technology landscape. It underscores the growing prevalence of algorithmic trading among both informed and uninformed market participants.

2.3.1 Market Impact

Algorithmic trading has a significant positive impact on market conditions, primarily enhancing liquidity and, in many cases, reducing short-term volatility. Studies have consistently shown that AT is beneficial, particularly for large-cap, high-priced, and low-volatility stocks (5).

For instance, high-frequency trading algorithms (HFTs), a subset of AT, are known to provide liquidity by submitting limit orders, thereby playing a role similar to traditional market-makers. This liquidity provision is especially crucial during times of market stress or when liquidity is naturally scarce.

The impact of AT on market volatility is more nuanced. While some detractors argue that AT can exacerbate short-term volatility, numerous studies have found that AT, including HFTs, can actually reduce volatility. For example, during periods of market turbulence, HFTs have been observed to lower volatility in certain stock categories, such as small-cap stocks (4).

Moreover, AT has been shown to narrow bid-ask spreads, which is a direct indicator of improved market liquidity. This narrowing is particularly evident in stocks that are more attractive to algorithmic traders, such as those with large market capitalizations and lower volatility (6).

In summary, there is a general consensus among academics that HFTs reduce the cost of liquidity provision to the benefit of all market participants. However, the relationship between HFT activity and liquidity differs depending on certain stock-

specific characteristics. For example, the relationship is stronger for large-cap and high-priced stocks with low volatility (5).

2.3.2 Applications

There are numerous examples of algorithmic applications usage by individuals, but the biggest showcase of what incorporation of algorithms in trading can achieve is seen in big industry players, that have been in the field for a while and have a sufficient amount of resources to invest in developing their own, sophisticated, large-scale models that manage substantial portion of their portfolio.

Here are few examples (6):

- **Renaissance Technologies**, led by Jim Simons, showcases the influence of algorithmic trading in modern finance. In Greg Zuckerman's book, 'The Man Who Solved the Market,' Simons, a renowned mathematician and codebreaker, propelled his firm to the top of the financial world by following a key principle: removing emotions from investing and relying on data. This principle is deeply ingrained in the firm's DNA, as their algorithms tirelessly analyze vast datasets to uncover predictive patterns. It is this systematic and unemotional approach that has established Renaissance Technologies as one of the market's most secretive and profitable entities, consistently achieving success while others are influenced by human biases (23).
- **Two Sigma Investments** showcases the fusion of machine learning with algorithmic trading, creating adaptive trading strategies. Their technology-driven approach enables the firm to stay ahead in the rapidly evolving market environment, consistently seeking outperforming investment opportunities

Each of these companies illustrates a practical application of algorithmic trading, demonstrating its value in achieving efficiency, speed, and precision that significantly surpass human capabilities (19).

Moreover, the success of these industry leaders in algorithmic trading is not just a testament to their technological prowess, but also a reflection of a broader shift in the financial markets towards data-driven decision making. As these firms continue to refine their algorithms and computational models, they are setting new standards for efficiency and effectiveness in the financial sector.

2.5 Evaluation Metrics

To compare different algorithms and strategies, we need to highlight key points that indicate success of our model or its failure. It can be done by measuring different metrics that describe effectiveness, stability, risk, adaptability and other factors that impact decision making of a model. In this chapter I will look over the most used metrics, which often give traders and analytics a general overview of how capable their strategy and model are in real time.

2.4.1 Profit and Loss

The first and the most straightforward measure of financial performance, showing the net profit or loss over a specific period, and evaluating the immediate monetary success or failure of trading strategy. In order to have a basic understanding of how successful our model is, we need to know gross profit and losses at a specific time period and afterwards calculate overall net profit and losses (12).

Gross income can be calculated by subtracting the Cost of Goods Sold (COGS) from the Total Revenue. COGS is the cost of unsuccessful trades in our example, while total revenue is a revenue from all successful trades.

Gross profit = Total Revenue – Cost of Goods Sold (COGS)

Now, we can calculate "Net profit", which is basically gross profit after all kinds of transactional, slippage and operational expenses.

Profit and Loss is a simple measure which shows efficiency of a strategy and does not require deep understanding of statistics or graph analysis.

2.4.2 Sharpe Ratio

Sharp Ratio measures the risk-adjusted return, offering insights into the profitability of a strategy relative to its risk. It represents the additional amount of return that a trader receives per unit of increase in risk (13).

In its simplest form:

$$\textbf{\textit{Sharpe Ratio} = Rp – Rf / } \sigma p$$

Rp = return of portfolio

Rf = risk-free rate

σp = standard deviation of the portfolio’s excess return

The ratio is calculated by subtracting the risk-free rate of from the strategy's return and dividing by the standard deviation of the strategy's returns. A higher Sharpe Ratio indicates a more favorable risk-to-reward balance.

A Sharpe ratio of 1.3 for example, indicates that the return of the algorithmic trading strategy, after adjusting for its risk, is 1.3 times greater than the risk-free rate. Generally, a Sharpe ratio of 1 or higher is considered acceptable to good by investors, a ratio of 2 is very good, and a ratio of 3 or higher is considered excellent.

2.4.3 Maximum Drawdown

The drawdown by itself is the measure of the decline from a historical peak in some variable, in our case, it's the largest drop in portfolio value from peak to a trough before a new peak. Drawdown is crucial for assessing the volatility and risk inherent in a trading strategy, as it provides insights into the financial and emotional impact of high-risk investments (14).

The drawdown is defined as:

$$D(\tau) = \max_{t \in (0,\tau)} [X(t) - X(\tau)]$$

Where **T** is a drawdown at a time, t ≥ 0 is a sequence of random variables (peaks and lows for example), with **X** (0) = 0, which means that if X is negative, it is considered to be 0. This ensures that the drawdown is never less than 0, reflecting that a drawdown represents a decrease in value and cannot be negative.

The maximum drawdown is the maximum drawdown over the history of the variable. Let's take crypto for example, maximum drawdown can be defined as the lowest market value of any coin since it was listed.

It's defined as:

$$MDD(T) = \max_{\tau \in (0,T)} D(\tau)$$

In algo-trading, it's used to align investment choices with risk tolerance levels, ensuring that strategies are suitable for their financial goals and psychological comfort, the last one plays significant role in traditional trading, we will get into this topic later in my work. A lower MDD is preferable, indicating less risk and potential loss in value (14).

Here is an example:

The highest peak of the portfolio value was $150,000**, a**fter reaching this peak, the strategy experienced a period of losses during a market downturn.

The lowest trough following the highest peak was $100,000 before the strategy began to recover.

Therefore, the drawdown from the peak to the trough was $150,000 - $100,000 = $50,000.

The MDD in this case is 33.33%, which means the strategy had a period where it lost one-third of its value before it started to recover.

2.6 Challenges Faced in Algorithmic Trading

Despite the numerous advantages of algorithmic trading, there are also several challenges that traders need to navigate in order to maximize the effectiveness of their algorithms and achieve consistent profitability. The main challenge is to deal with the complexity and dynamic nature of financial markets.

Algorithms, no matter how sophisticated, may struggle to adapt to rapid and unforeseen market changes. This section will delve into some of the key challenges encountered in algorithmic trading.

2.5.1 Overfitting Bias

In trading, overfitting occures when strategies are excessively tailored to historical data, leading to a decline in their predictive capability in real-world applications. This issue manifests in two primary ways (8):

- **Over-Optimization on Historical Data**: Strategies that are too finely adjusted to past data usually have impressive performance in backtests but often fail when applied to live markets. The root of this problem lies in the model's inability to generalize from past conditions to future, unseen market scenarios. Essentially, these models capture the noise rather than the underlying signal in historical data, mistaking random fluctuations for meaningful patterns.
- **Hindsight Bias through Parameter Tweaking**: Another form of overfitting occurs when traders incessantly modify the parameters of their models in search of improved backtest results. This process can inadvertently introduce hindsight bias, as it relies on knowledge of the full dataset to make adjustments that may not be justifiable in a real-time trading context.

Both forms of overfitting compromise the robustness and reliability of trading strategies. A strategy's ability to perform well in backtesting does not guarantee its effectiveness in live markets, primarily because these overfitted models fail to account for the unpredictable and dynamic nature of financial markets.

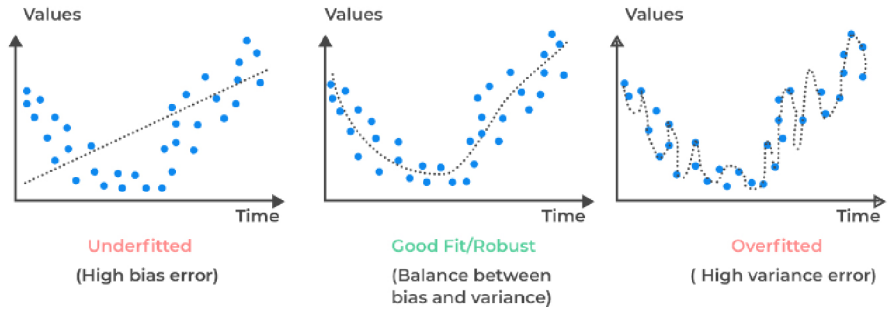


Figure 10: Types of overfittings

Source: Overfitting and Methods of Addressing it, 2021

The left-hand graph (Fig. 10) displays a model that is overly simplistic, failing to capture the complexity of the data. This results in underfitting, where the model's prediction error is high due to its inability to account for all the data points.

On the other hand, the right-hand graph (Fig. 10) illustrates **overfitting**. Here, the model is excessively complex, fitting not just the underlying pattern but also the noise within the training data. This typically leads to poor predictive performance on new data due to the model's oversensitivity to the specific details of the training set.

The central graph (Fig. 10) represents a **well-fitted** model. It strikes a balance by accurately capturing the general trend of the data without being distorted by noise. This balance suggests the model is likely to perform well on new data, having minimized both bias and variance.

To lower the risk of overfitting, we can use different strategies, here are couple of them (8):

- **Data Partitioning** is the simplest thing you can do other than keeping your model simple. Separating your data into distinct sets for training, validation, and testing, also known as in/out sample data. The training set is used to build the model, the validation set is used to tune parameters, and the testing set, which the model has never seen before, is used to evaluate performance.

- **Out-of-sample** testing involves reserving a portion of historical data, not used during the model development and parameter optimization, to test the model's predictive ability. After the model is built using the in-sample data, it's then applied to the out-of-sample data to simulate how the model would have performed in real-time. The model's performance metrics on the out-of-sample data, such as the Sharpe ratio and maximum drawdown, should not significantly differ from the in-sample performance. If the model underperforms on out-of-sample data, it might indicate overfitting, and may require refinement.
- **Walk-Forward Analysis**, which starts by optimizing the model's parameters on in-sample data, then testing the strategy on the out-of-sample data without re-optimizing the parameters. This process is then "walked forward" through the data: after the first out-of-sample test, a new in-sample test begins, immediately followed by another out-of-sample test. This cycle continues, ensuring that the strategy is consistently tested over different market conditions.
- **Cross-Validation** is similar to walk-forward analysis (Fig. 11), it involves dividing the data into parts and conducting multiple rounds of analysis, using different parts as training and validation sets in each round.

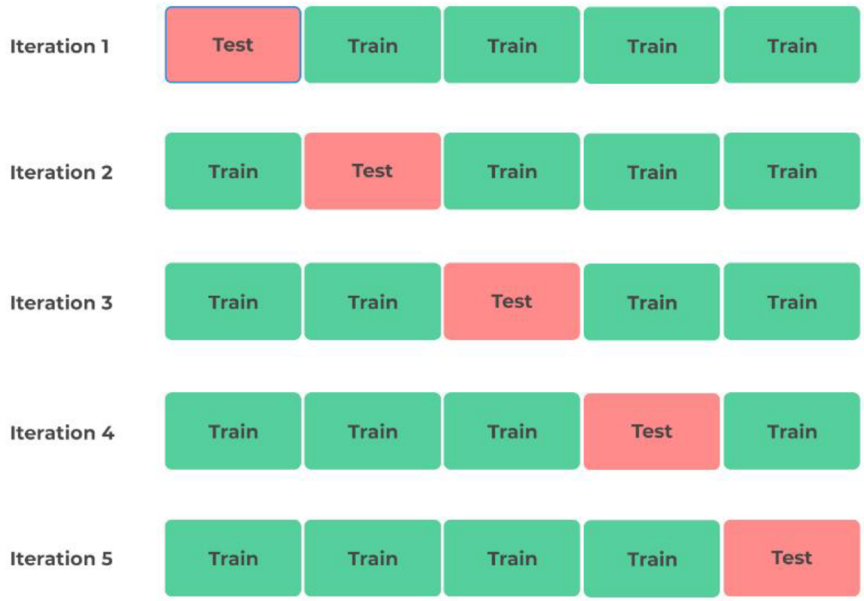


Figure 11: Cross-validation cycle

Source: Overfitting and Methods of Addressing it, 2021

By minimizing overfitting bias, we are more likely to create algorithmic trading strategy that will deliver consistent, reliable performance in live markets, ensuring that profitability and risk management are sustained over time.

2.5.2 Data Snooping Bias

"Data snooping bias is caused by having too many free parameters that are fitted ethereal patterns in the past to make historical performance look good" (3).

Another challenge in algorithmic trading is the risk of data snooping, it refers to the practice of mining historical market data to discover patterns or relationships that may not necessarily have any predictive value.

Before we dive into the problem roots, let's have a look at one example of data snooping bias in algo-trading, it is the infamous "Dow 36,000" prediction made in the late 1990s (9).

In those days, the internet was emerging as a new and exciting technology. This led to the rapid growth of internet-based companies, commonly referred to as dot-coms. Investors, who saw the potential of the internet, poured massive amounts of capital into these startups, many of which had yet to make a profit.

Around 2000, it appeared that many of these companies were not as financially successful as their stock prices suggested. This realization, combined with other economic factors, led to a loss of investor confidence. The bubble burst led to a sharp decline in the stock prices of technology companies. This crash caused many startups to go out of business and resulted in significant financial losses for investors.

Getting back to "Dow 36,000", researchers used historical stock market data to develop a model that projected the Dow Jones Industrial Average reaching 36,000 points. However, this prediction failed, as it did not account for changing market conditions and unforeseen events such as the dot-com bubble burst and subsequent market crash (9).

The most basic safeguard against data-snooping bias is a sufficient amount of backtest data relative to the number of free parameters you want to optimize. For example, let's assume that the number of data points needed for optimizing your parameters is equal to 246 times the number of free parameters your model has. If we have a daily trading model with three parameters. Then you should have at least three years' worth of backtest data with daily prices (3).

Another approach is again, testing the model on out-of-sample data. This step shows the model's true predictive power and generalizability. This approach was also mentioned in Chan's work, "The ultimate out-of-sample testing is familiar to many traders, and it is called paper trading. Running the model on actual unseen data is the most reliable way to test it" (Chan, 2008).

2.5.3 Market Dynamics and Shifts

Shifts and changing market conditions are common things for human traders, but not for algo-trading models to some degree, that's why they should be explicitly programmed for unstable and constantly changing market.

One of the main causes of market dynamics in algo-trading is technological advancements. As technology evolves rapidly, new algorithms are developed and implemented to exploit market inefficiencies. This leads to increased competition among traders and a constant need to adapt strategies. Additionally, changes in regulations or market structures can also create shifts in algo-trading dynamics.

To address these challenges, traders must update their algorithms and risk management systems. Regular monitoring of trading activities is essential to identify potential issues or anomalies promptly. Collaboration between regulators and industry players is crucial for establishing guidelines that ensure fair practices while allowing innovation.

Examples of market dynamics in algo-trading can be seen during major economic events such as interest rate announcements or geopolitical crises. These events trigger sudden shifts in market sentiment, leading to increased volatility and liquidity fluctuations.

In conclusion, understanding the causes behind market dynamics in algo-trading is vital for developing effective solutions.

2.5.4 Behavioral Biases

“Regardless of how disciplined, people often make financial decisions that are colored by behavioral biases that cause them to act on emotion or make mistakes processing information.” (18)

Behavioral biases in algo-trading refer to the irrational decision-making patterns exhibited by traders when using algorithmic trading systems. These biases can lead to significant financial losses and market instability.

One major cause of behavioral biases in algo-trading is overconfidence. Traders often believe that their algorithms are infallible, leading them to take excessive risks without considering potential downsides. This can result in catastrophic losses when market conditions change unexpectedly.

Another cause is herd mentality, where traders blindly follow the actions of others without conducting proper analysis. This behavior amplifies market volatility and can lead to price bubbles or crashes.

To address these biases, we can implement risk management measures such as setting stop-loss orders or diversifying portfolios. Additionally, regular monitoring and evaluation of algorithm performance can help identify any potential biases or flaws (18).

Several examples highlight the impact of behavioral biases in algo-trading. The flash crash of 2010 (Fig. 12) within minutes due to a combination of high-frequency trading algorithms and panic selling triggered by erroneous trades. This incident demonstrated how unchecked algorithmic trading can exacerbate market volatility.

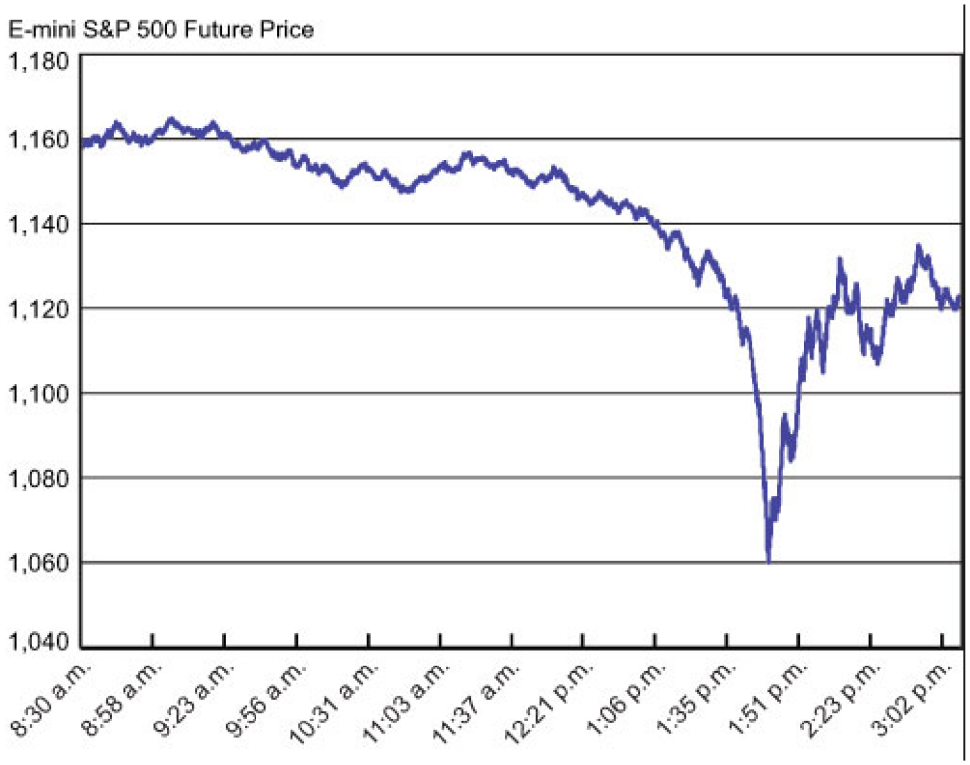


Figure 12: S&P 500 stock crash graph of 2010

Source: Liberty Street Economics, 2012

In conclusion, behavioral biases in algo-trading pose significant risks to financial markets. By understanding their causes, which are most likely connected to human nature and psychology, we can implement appropriate risk management measures, and learn from the past. These steps will help to mitigate these biases and ensure a more stable and fair-trading environment for all participants.

2.5.5 Loss Aversion Bias

"Loss aversion is rooted in a deep-seated instinctual impulse to avoid pain." (1)

Loss aversion bias significantly affects decision-making in algorithmic trading. This bias stems from the psychological phenomenon where the discomfort or pain associated with losses outweighs the satisfaction or pleasure derived from an equivalent number of gains. In the context of algo-trading, this bias can be seen in several ways, potentially undermining the effectiveness of trading strategies.

In algo-trading, loss aversion may result in algorithms that exit positions too early to avoid potential losses, missing out on substantial gains. These systems often employ tight stop-loss orders, which, while vital for risk management, may be set too close to the purchase price, leading to premature exits, especially in volatile markets.

This cautious approach can also manifest itself in a missed profitable opportunity, as the fear of trend reversals may prompt early exits from rising markets, yielding lower overall returns.

There are some ways we can counteract loss aversion bias in algo-trading. First of all, adjusting algorithm parameters helps, ensuring they strike a balance between risk and reward. This includes setting stop-loss orders and taking calculated risks based on comprehensive market analysis (1).

For manual traders intervening in algorithmic systems, psychological training to understand and manage this bias can lead to more rational decision-making.

Moreover, extensive back testing across various market scenarios can help identify if an algorithm is excessively conservative, providing valuable information into potential performance during different market trends.

2.6 Summary of Case Studies and Examples

In conclusion, the future of algorithmic trading holds great potential and promises significant advancements in the intersection of machine learning and trading. The studies and examples discussed in this essay have shown the effectiveness of using algorithms and predictive models to make informed trading decisions.

However, it is important to consider the evaluation metrics of model performance and be cautious of algorithmic trading biases, such as overfitting and data snooping. The intersection of machine learning and trading has revolutionized the financial industry by enabling market participants to analyze large amounts of data and make more accurate predictions.

Machine learning algorithms provide the ability to identify patterns and trends that human traders may miss, thus enhancing decision making and improving trading strategies. The use of neural networks, deep learning, and other advanced algorithms have proven to be highly effective in predicting market movements and identifying profitable trading opportunities.

Various studies and examples have demonstrated the success of algorithmic trading strategies. For instance, research by Ronen et al. (2019) explored the use of support vector machine (SVM) and random forest models to predict stock price movements. Their findings revealed that these models outperformed traditional statistical methods, showcasing the potential of machine learning in trading.

3. Model development and Results

In this section, we will focus on implementing the Moving Average Crossover strategy, which is a key component of algorithmic trading models. We will start by establishing a strong API connection to receive real-time market data, which is crucial for executing the strategy effectively. Then, we will gather historical price data to build the foundation of our model.

Once we have the data, we will visualize it to identify market trends and generate actionable signals using the Moving Average Crossover strategy. The main part of our investigation involves backtesting. We will simulate the strategy using historical data to predict its performance and evaluate its hypothetical success in past market conditions.

Finally, we will evaluate the results by carefully analyzing the outcomes of the backtesting. This thorough evaluation aims to provide a detailed understanding of the role and effectiveness of the Moving Average Crossover strategy in today's financial trading landscape.

3.1 Implementation

3.1.1 Programming Language and Libraries

For the development of our model, we have chosen Python. This language is well-equipped for data-centric applications, with a robust set of libraries facilitating the processing and analysis of large datasets (11).
The libraries we will leverage include:

- **Pandas** for data manipulation and analysis in Python.
- **Requests** to interact with external services and fetch data over HTTP seamlessly.
- **OANDA API** to automate trading strategy by harnessing financial market data (20).
- **Plotly** to graphically represent financial data in an interactive and intuitive manner.

3.1.2 Connecting to API

As a trading platform we will use OANDA broker, it has a great and reliable open-source API (20).

In order to access historical data, our model should have access to OANDA API to send and receive API calls. Connection is established by utilizing several personalized keys, such as:

```
API_KEY = "XXXXXXXXXXXXXXXXXXXXXX - XXXXXXXXXXXXXXXXXXXXXXX"
ACCOUNT_ID = "XXX-XXX-XXXXXXX-XXX"
OANDA_URL = 'https://api-fxpractice.oanda.com/v3'
SECURE_HEADER = {
   'Authorization': f'Bearer {API_KEY}',
   'Content-Type': 'application/json'
}
```

Code snippet 1: Connecting to OANDA api.

Source: Sample Project.

API_KEY serves as a confidential token for authentication, acting as a unique identifier. It is included in the request header to verify the requester's identity and ensure they have the necessary permissions to access the API's functionalities.

This key plays a crucial role in the security mechanism, preventing unauthorized access (20).

ACCOUNT_ID is a distinct identifier for a user's account on the OANDA platform. It is utilized to specify which account is making the request, allowing the OANDA API to apply the request to the correct account. For instance, it is used when retrieving account-specific data or executing trades. OANDA_URL represents the base URL for the OANDA API (20).

All requests to the OANDA API begin with this base URL and then append endpoint-specific paths. It points to the fxPractice environment, indicating its usage for practice or demo purposes rather than live trading.

SECURE_HEADER is a Python dictionary that contains HTTP headers to be included in the API request. The headers it contains are as follows: - 'Authorization': This header carries the API key for authentication, formatted as a bearer token.

The Python f-string syntax, f'Bearer {API_KEY}', formats the string to include the value of API_KEY in the designated position. “Content-Type" specifies the media type of the resource, which in this case is “application/json”. It indicates that the request body is formatted as a JSON object, which is a common format for exchanging data with APIs.

These parameters collectively configure the HTTP request used by your application to communicate with the OANDA API. They ensure that the requests are properly authenticated, and that the API can understand and process the request body accurately.

By using these keys, connection can be established through OANDA’s API endpoints, that are described in its API documentation (20).

```
session = requests.Session()
url = f"{defs.OANDA_URL}/accounts/{defs.ACCOUNT_ID}/instruments"
response = session.get(url, params=None, headers=defs.SECURE_HEADER)
```

Code snippet 2: Accessing OANDA API.

Source: Sample Project.

After running the code snippet above, we should get HTTP response of 200, according to API documentation, this means that connection was established successfully. (20).

3.1.3 Data Gathering

To implement our strategies, we need to obtain the next sets of data:

- **Instrument or currency pair**, represents the market data subject of interest. It is identified by a unique label, such as 'EUR_USD' for the Euro to US Dollar exchange rate. Selecting the appropriate instrument is crucial as it forms the basis for market analysis and subsequent strategy development (20).
- **Price** specifies the particular price data needed for analysis. Options typically include bid, ask, or mid prices, each indicating different aspects of market valuation.

- **The bid price** is the highest price a buyer is willing to pay, the ask price is the lowest price a seller is willing to accept, and the mid-price is the average of the bid and ask.
- **Granularity** refers to the time interval of each data point in the series, ranging from seconds to weeks. The chosen granularity reflects the temporal resolution of the data and is closely aligned with the strategic approach of the analysis—shorter intervals for high-frequency strategies, longer for trend-based analyses.
- **Count of Candles Returned** determines the number of historical data points, or 'candles', to be retrieved. The specific count corresponds to the depth of historical market behavior analysis required. A higher count allows for a more comprehensive backtest of the market's performance over time.
- **Date From** parameter specifies the start date for historical data collection. It limits the dataset to include only the market information following this date, enabling targeted analysis of market behavior within a defined timeframe.
- **Date To** parameter marks the end date for data retrieval. The resulting dataset includes market data up to this specified date. When used alongside the 'date from' parameter, it helps encapsulate the data within a precise historical segment for analysis.

For handling our trading data, we will utilize **DataFrames**, a feature of the pandas library. DataFrames arrange data in a clear, table-like format, making it easy to work with.

They will enable us to quickly sort through our collected data — such as instrument types, price points, and time intervals — and prepare it for analysis.

Creating a dictionary and obtaining tradable instruments utilizing pandas library:

```
instrument_data = []
for item in instruments:
  new_ob = dict(
    name = item['name'],
    type = item['type'],
    displayName = item['displayName'],
    pipLocation = item['pipLocation'],
    marginRate = item['marginRate']
  )
  instrument_data.append(new_ob)
```

```
instrument_df = pd.DataFrame.from_dict(instrument_data)
instrument_df.to_pickle("instruments.pkl")

instrument_df
```

Code snippet 3: Obtaining tradable instruments.
Source: Sample Project.

Running the code above will give us all currently tradable pairs, with specified parameters as a stylized data frame, and save it to pickle for future manipulations:

	name	type	displayName	pipLocation	marginRate
0	NZD_CAD	CURRENCY	NZD/CAD	-4	0.03
1	EUR_SGD	CURRENCY	EUR/SGD	-4	0.05
2	EUR_CHF	CURRENCY	EUR/CHF	-4	0.04
3	USD_CNH	CURRENCY	USD/CNH	-4	0.05
4	SGD_CHF	CURRENCY	SGD/CHF	-4	0.05
...	...	...	...	...	...
63	EUR_PLN	CURRENCY	EUR/PLN	-4	0.05
64	GBP_PLN	CURRENCY	GBP/PLN	-4	0.05
65	EUR_SEK	CURRENCY	EUR/SEK	-4	0.03
66	USD_SGD	CURRENCY	USD/SGD	-4	0.05
67	HKD_JPY	CURRENCY	HKD/JPY	-4	0.1

Figure 13: Tradable pairs with basic information
Source: Sample Project.

After that, we can select specific tradable pair and fetch candle data, also with the help of pandas. For this API request we need to specify tradable pair and candle granularity (20):

```
def fetch_candles(self, pair_name, count=None, granularity="H1", date_from=None, date_to=None):

    url = f"{defs.OANDA_URL}/instruments/{pair_name}/candles"
```

```
params = dict(
    granularity = granularity,
    price = "MBA"
)

if date_from is not None and date_to is not None:
    params['to'] = int(date_to.timestamp())
    params['from'] = int(date_from.timestamp())
elif count is not None:
    params['count'] = count
else:
    params['count'] = 300

response = self.session.get(url, params=params, headers=defs.SECURE_HEADER)

if response.status_code != 200:
    return response.status_code, None

return response.status_code, response.json()
```

Code snippet 4: Obtaining candle data for "EUR_USD" pair
Source: Sample project.

According to our parameters which are "EUR_USD" for the pair and "H1" for granularity, we get candle data in data frame:

	volume	mid_o	mid_h	mid_l	mid_c	bid_o	bid_h	bid_l	bid_c	ask_o	ask_h	ask_l	ask_c
count	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000
mean	3397.170000	1.079156	1.079736	1.078575	1.079151	1.079070	1.079657	1.078489	1.079071	1.079242	1.079820	1.078656	1.079231
std	2432.780598	0.013665	0.013641	0.013678	0.013665	0.013667	0.013642	0.013678	0.013665	0.013664	0.013641	0.013678	0.013665
min	140.000000	1.045260	1.046640	1.044850	1.045270	1.045180	1.046550	1.044770	1.045200	1.045330	1.046720	1.044930	1.045340
25%	1780.500000	1.070537	1.071075	1.069890	1.070475	1.070457	1.071003	1.069790	1.070403	1.070607	1.071160	1.069962	1.070553
50%	2864.500000	1.081590	1.082200	1.080925	1.081570	1.081500	1.082120	1.080845	1.081500	1.081675	1.082270	1.081000	1.081650
75%	4238.750000	1.088867	1.089482	1.088340	1.088843	1.088770	1.089412	1.088233	1.088770	1.088952	1.089560	1.088413	1.088923
max	21907.000000	1.112920	1.113960	1.112270	1.112940	1.112840	1.113880	1.112190	1.112860	1.112990	1.114040	1.112350	1.113020

Figure 14: Candlestick data for "EUR_USD"
Source: Sample Project.

The columns represent various market prices and volumes:

- volume: The total number of units traded within the hour.
- mid_*: The midpoint prices for the open (mid_o), high (mid_h), low (mid_l), and close (mid_c) of the currency pair, offering a consolidated view of market movement.
- bid_*: The bid prices for the open (bid_o), high (bid_h), low (bid_l), and close (bid_c), which represent the highest prices buyers are willing to pay.
- ask_*: The ask prices for the open (ask_o), high (ask_h), low (ask_l), and close (ask_c), reflecting the lowest prices sellers are willing to accept.

3.1.6 Data Visualization

Plotly, a dynamic and interactive graphing library for Python, will be utilized to represent our data visually. The extensive range of features offered by Plotly makes it an invaluable tool for generating intricate visualizations effortlessly. Its widespread adoption in various sectors underscores its adaptability, allowing for the creation of anything from basic scatter plots to intricate 3D models.

Figure 15: Stylizing obtained candle chart with plotly in python.

Source: Sample Project.

3.1.7 Strategy Implementation

Within the framework of this project, we shall execute the Moving Average Crossover strategy. As commonly understood, this strategy involves two moving averages: a shorter one and a longer one.

The essence of this approach lies in the points where these two moving averages intersect, as they indicate potential turning points in the market. To be more precise, a buy signal is generated when the short-term moving average surpasses the long-term moving average, indicating the start of an upward trend.

Conversely, a sell signal is triggered when the short-term moving average falls below the long-term moving average, indicating the beginning of a downward trend.

We will add different moving averages to our candle charts, and test each paired with each as a short and long combination throughout the period from **01.01.23** till **14.04.24**. After that, performance will be evaluated to select the pair that shows the best results.

Adding moving averages to our candle chart:

```
pair = "EUR_USD"
granularity = "H1"
ma_list = [4, 8, 16, 32, 64, 128, 256]

i_pair = instrument.Instrument.get_instrument_by_name("EUR_USD")
df = pd.read_pickle(utils.get_his_data_filename(pair, granularity))
non_cols = ['time', 'volume']
mod_cols = [x for x in df.columns if x not in non_cols]

df[mod_cols] = df[mod_cols].apply(pd.to_numeric)
df_ma = df[['time', 'mid_o', 'mid_h', 'mid_l', 'mid_c']].copy()

for ma in ma_list:
    df_ma[f'MA_{ma}'] = df_ma.mid_c.rolling(window=ma).mean()
df_ma.dropna(inplace=True)
df_ma.reset_index(drop=True, inplace=True)

df_ma.head()
```

Code snippet 5: Adding and calculating moving average points for each candle.
(Some functions are not included for better code readability)
Source: Sample project.

The original data is modified through the code by adding new columns that depict the average closing price across different time intervals.

Each interval in the list is subjected to a calculation of an average, which effectively shows the impact of short-term price fluctuations and aids in the identification of longer-term trends within the data.

These supplementary columns are consistently labeled, such as 'MA_8' for an eight-period average, thereby providing clear indication of their respective representations.

We can analyze price movements over time by using plotly and see how each MA follows uptrend or downtrend:

Figure 16: Example of different moving averages.

Source: Sample Project.

In order to make a trading decision, we must identify a crossover event and determine whether our model should place a sell or buy order.

```
def is_trade(row):
    if row.DIFF >= 0 and row.DIFF_PREV < 0:
        return 1
    if row.DIFF <= 0 and row.DIFF_PREV > 0:
        return -1
    return 0
```

Code snippet 6: Evaluating crossover event.
(Some functions are not included for better code readability)
Source: Sample Project.

The function above compares difference between short and long crossover position and returns “1” in case of shorter MA crosses above longer MA, signalizing the uptrend movement. On contrary, if shorter MA crosses under, function returns “-1”, and it could be considered as a downtrend signal. If a cross doesn't occur, we return “0” that is simply the sign of no trade.

Creating a function to evaluate pair for potential trade at a time, the inputs for this process include the currency pair (i_pair), two integers representing the short and long moving average periods (mashort and malong), and the price data (price_data):

```
def evaluate_pair(i_pair, mashort, malong, price_data):

    price_data = price_data[['time', 'mid_c', get_ma_col(mashort), get_ma_col(malong)]].copy()
    price_data['DIFF'] = price_data[get_ma_col(mashort)] - price_data[get_ma_col(malong)]
    price_data['DIFF_PREV'] = price_data.DIFF.shift(1)
    price_data['IS_TRADE'] = price_data.apply(is_trade, axis=1)

    df_trades = price_data[price_data.IS_TRADE!=0].copy()
    df_trades["DELTA"] = (df_trades.mid_c.diff() / i_pair.pipLocation).shift(-1)
    df_trades["GAIN"] = df_trades["DELTA"] * df_trades["IS_TRADE"]

    df_trades["PAIR"] = i_pair.name
    df_trades["MASHORT"] = mashort
    df_trades["MALONG"] = malong

    del df_trades[get_ma_col(mashort)]
    del df_trades[get_ma_col(malong)]

    df_trades["time"] = [parse(x) for x in df_trades.time]
    df_trades["DURATION"] = df_trades.time.diff().shift(-1)
    df_trades["DURATION"] = [x.total_seconds() / 3600 for x in df_trades.DURATION]
    df_trades.dropna(inplace=True)
```

```
return ma_result.MAResult(
    df_trades=df_trades,
    pairname=i_pair.name,
    params={'mashort' : mashort, 'malong' : malong}   )
```

Code snippet 7: Evaluating specific pair for crossover event & collecting trade data

Source: Sample Project.

Here is how the function above works:

The data is filtered by selecting only the necessary columns that are related to time, closing price, and the two specified moving averages.

The differences between the short and long moving averages **(DIFF)** are calculated, as well as the difference from the previous period **(DIFF_PREV).**

The **"is_trade"** function is applied to identify buy and sell signals based on the moving average crossover strategy.

For rows with trade signals, the profit or loss **(DELTA)** in pip movements and the actual gain or loss **(GAIN)** are calculated, taking into account the direction of the trade.

The original moving average columns are removed to declutter the dataset.

The time strings are converted to datetime objects and the duration between trades is calculated.

The final output is an instance of MAResult that contains the trade analysis results and the parameters used in the evaluation.

Multiple currencies like the British Pound (GBP), Euro (EUR), US Dollar (USD), Canadian Dollar (CAD), Japanese Yen (JPY), New Zealand Dollar (NZD), and Swiss Franc (CHF) are tested to ensure that we have enough sample and validation data to minimize data snooping bias.

To mitigate overfitting bias, we will use different Moving Averages (MAs) with periods of **4, 8, 16, 32, 64, 96, 128**, and **256**. This range allows us to capture different levels of data responsiveness, from the reactive 4-period MA to the broader 256-period MA. We will pair each currency with each MA duration to create a detailed matrix of instrument-MA combinations. This analysis aims to find the most reliable and profitable pairing for our backtesting simulations.

Our goal is to extract key insights and determine the best strategy configurations that will result in overall profitability.

Implementing our backtesting function:

```
def test():

    currencies = "GBP,EUR,USD,CAD,JPY,NZD,CHF"
    granularity = "H1"
    ma_short = [4, 8, 16, 24, 32, 64]
    ma_long = [8, 16, 32, 64, 96, 128, 256]
    test_pairs = get_test_pairs(currencies)

    results = []
    for pairname in test_pairs:
        print("running..", pairname)
        i_pair = instrument.Instrument.get_instruments_dict() [pairname]

        price_data = get_price_data(pairname, granularity)
        price_data = process_data(ma_short, ma_long, price_data)

        for _malong in ma_long:
            for _mashort in ma_short:
                if _mashort >= _malong:
                    continue
                results.append(evaluate_pair(i_pair, _mashort, _malong, price_data))

    final_df = process_results(results)
    all_trades_df = store_trades(results)

    create_excel(final_df, all_trades_df)
```

Code snippet 8: Moving Average Crossover strategy implementation.

Source: Sample Project.

Variable Initialization: a string containing the major currency codes (GBP, EUR, USD, CAD, JPY, NZD, CHF) that will be tested by the function. “granularity” - the time frame for the data, indicated as 'H1' for hourly data.

The ma_short and ma_long are lists of periods for short and long moving averages. These periods are used to calculate two sets of moving averages for each currency pair. Typically, the 'short' moving average responds faster to price changes compared to the 'long' moving average.

Pair Generation: **"get_test_pairs"** function generates all possible combinations of the specified currencies to form currency pairs for testing.

Data Acquisition and Processing: loops through each currency pair obtained from **"test_pairs"**, the function retrieves price data using **"get_price_data",** which fetches historical data based on the specified granularity.

Processes this data in **"process_data"** by calculating the moving averages specified in "ma_short" and "ma_long".

Strategy Evaluation: Nested loops for MA settings iterate over every combination of short and long moving averages. However, it only continues if the short MA period is less than the long MA period to ensure proper crossover analysis. The "evaluate_pair" function is called for each valid MA combination. It assesses the effectiveness of the trading strategy by identifying crossover points where the short MA crosses above or below the long MA, indicating potential buy or sell signals which was mentioned in previous code snippet.

Results Compilation: **"process_results"** aggregates the results of all evaluated pairs and MA settings into a final DataFrame that summarizes the performance of each strategy. The **"store_trades"** function saves all trades that occurred during the simulations for further analysis.

After running tests, we can see structured results of first 11 MA combinations:

	pair	num_trades	total_gain	mean_gain	min_gain	max_gain	mashort	malong
0	GBP_USD	2606	829.8	0.318419	-155.3	485.2	4	8
1	GBP_USD	1463	-200.7	-0.137184	-277.3	404.2	4	16
2	GBP_USD	1278	-408.6	-0.319718	-292.5	567.7	8	16
3	GBP_USD	871	2131.2	2.446843	-306.2	1390.6	4	32
4	GBP_USD	685	2668.6	3.895766	-324.4	1544.3	8	32
5	GBP_USD	595	3669.3	6.166891	-346.7	1440.0	16	32
6	GBP_USD	665	3649.8	5.488421	-290.4	1394.1	24	32
7	GBP_USD	558	2882.5	5.165771	-312.0	1155.8	4	64
8	GBP_USD	404	3406.4	8.431683	-330.6	1232.0	8	64
9	GBP_USD	332	3361.3	10.124398	-290.4	1266.0	16	64
10	GBP_USD	286	4045.3	14.144406	-273.8	1276.5	24	64

Figure 17: Backtesting results of “GBP_USD”.

Source: Sample Project.

From the figure, the **maximum drawdown** is observed in the fifth GBP/USD entry, indicating a significant loss of 346.7 pips at its lowest point.

In conclusion, the Moving Average Crossover strategy was tested extensively with the latest yearly data on different currency pairs. The next step is to evaluate the results to confirm the strategy's strength and reliability.

By examining performance and comparing expected outcomes with actual results, we can improve the strategy for possible use in real trading situations.

3.2 Evaluation

Evaluating our trading strategies is important to their effectiveness. It gives us numerical data to analyze and decide if a strategy is viable for real market use.

By evaluating carefully, we can find ways to improve, avoid overfitting, and gain insight into how a strategy performs in different market situations. This evaluation is crucial for creating a strong trading model that can be sustainable.

Let's start by examining the overall profits and the total trades made during the backtesting phase for the initial 10 moving average pairs starting from well-performed:

CROSS	mashort	malong	pair	num_trades	total_gain
MA_16_32	16	32	GBP_USDGBP_CADGBP_JPYGBP_NZDGBP_CHFEUR_GBPEUR_...	4314	6706.6
MA_8_16	8	16	GBP_USDGBP_CADGBP_JPYGBP_NZDGBP_CHFEUR_GBPEUR_...	8989	4698.0
MA_8_32	8	32	GBP_USDGBP_CADGBP_JPYGBP_NZDGBP_CHFEUR_GBPEUR_...	4978	3588.4
MA_4_32	4	32	GBP_USDGBP_CADGBP_JPYGBP_NZDGBP_CHFEUR_GBPEUR_...	6381	3487.2
MA_4_16	4	16	GBP_USDGBP_CADGBP_JPYGBP_NZDGBP_CHFEUR_GBPEUR_...	10266	3096.2
MA_24_32	24	32	GBP_USDGBP_CADGBP_JPYGBP_NZDGBP_CHFEUR_GBPEUR_...	4906	-2411.3
MA_4_8	4	8	GBP_USDGBP_CADGBP_JPYGBP_NZDGBP_CHFEUR_GBPEUR_...	18327	-2904.7
MA_64_256	64	256	GBP_USDGBP_CADGBP_JPYGBP_NZDGBP_CHFEUR_GBPEUR_...	621	-4621.0
MA_8_64	8	64	GBP_USDGBP_CADGBP_JPYGBP_NZDGBP_CHFEUR_GBPEUR_...	3140	-5406.1
MA_4_64	4	64	GBP_USDGBP_CADGBP_JPYGBP_NZDGBP_CHFEUR_GBPEUR_...	4201	-7985.9

Figure 18: Backtesting results

Source: Sample Project.

The best performing moving average for all tradable pairs is "**MA_16_32**" with 67% of profitable trades. While the worst MA combination not shown in the figure is "**MA_32_96**" with total of **1612 trades**, loss of **-27954 pips** and only 14%.

We can notice that number of trades and ma_short/ma_long ration are correlated with total gain in our scenario. When studying different combinations of moving averages, it seems that extreme values, whether they are very small or very large, are not as effective. This could be because these values don't often cross over each other. For example, larger moving averages may not give many trading signals, which could lead to less-than-ideal entry points and fewer trades. On the other hand, very small moving averages may give too many signals, increasing the chances of false positives. Therefore, it is beneficial to find a balance in selecting moving average periods to improve the timing and frequency of trades.

Let’s pick a tradable pair and analyze models' performance step by step. As an example, CAD_CHF with **16** and **32** moving averages:

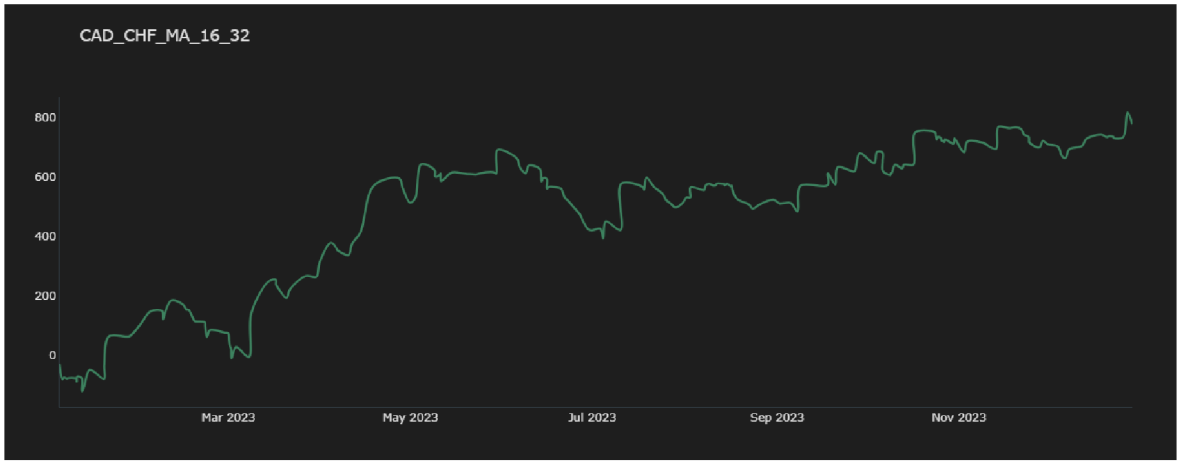


Figure 19: "CAD_CHF_MA_16_32" gains through time

Source: Sample Project.

The chart (Fig. 19) shows a steady increase in performance when using the **16_32** moving average settings for the CAD_CHF currency pair. Since March 2023, there has been a clear upward trend in profits, showing that the model was successful in taking advantage of favorable market movements.

Although the growth is not constant, there are periods of market fluctuations; overall, the trend is positive, indicating that the chosen moving average settings have been effective in capturing profitable trades and ended up with **778 pip profit** and total of **213 trades**. This highlights the model's ability to generate consistent growth during the testing period.

In contrast, the CAD_CHF currency pair shows a downward trend throughout 2023 when using the **64_128** moving average settings. The graph reveals a continuous loss in value starting from March, with some stabilization between May and September before further declines occur.

Although there are occasional upward ticks indicating brief periods of gains, they are not enough to offset the overall negative trend. Based on the visual data, this trading strategy has moments of profitability but is not consistently effective for this currency pair and timeframe. Resulting in **–794 pip loss** with only **54 trades**.

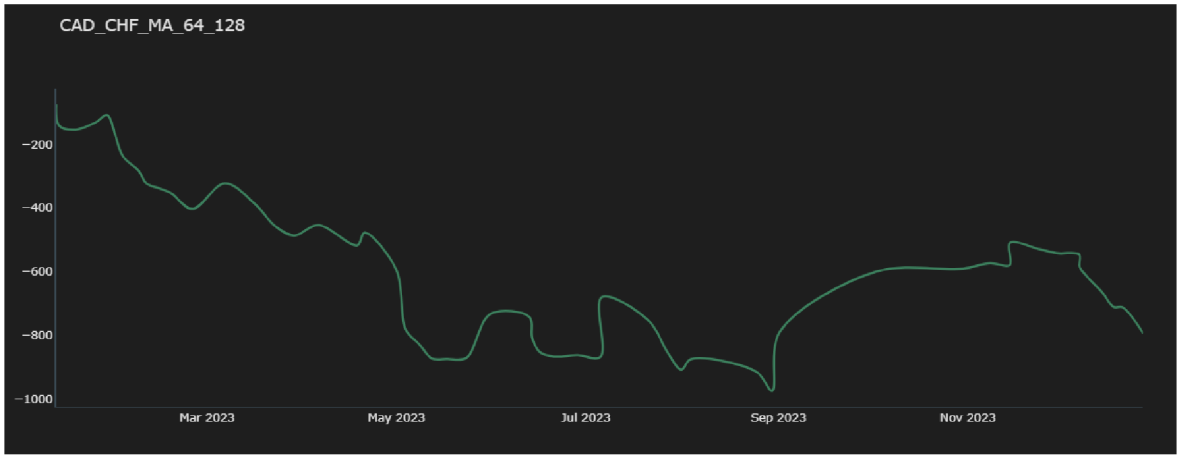


Figure 20: "CAD_CHF_MA_64_128" gains through time

Source: Sample Project.

This graph (Fig. 20) shows how a trading strategy using the **64_128** moving average settings on the CAD_CHF currency pair performed in 2023. The graph indicates a downward trend starting in March, showing a continuous decrease in value.

From May to September, the trend stabilizes somewhat before declining further. Occasionally, there are small increases, indicating brief periods of gains. However, these increases are not enough to offset the overall negative trend. These MA values were not consistently effective for this currency pair and timeframe. Therefore, further evaluation and improvement are necessary.

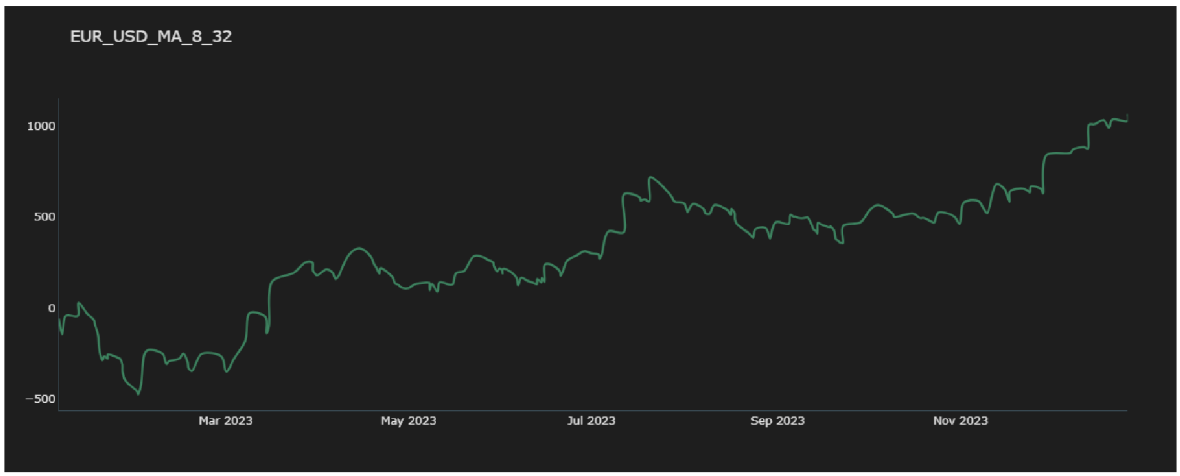


Figure 21: "EUR_USD_MA_8_32" gains through time

Source: Sample Project.

The displayed graph (Fig. 21) showcases the gains accrued from employing an **8_32** moving average strategy on the EUR_USD currency pair. Initially, the model encountered losses, however, from March onwards, it demonstrates a steady recovery, characterized by an uptrend. By May, gains begin to outpace losses, with a consistent rise through to November.

The pattern of higher highs and higher lows indicates a strong bullish trend. We can suggest that the **8_32** moving average parameters are effective for this pair, capturing profitable moments in market movements while providing resilience against volatility throughout the year, that can be seen in absence of major drawdowns.

Backtesting showed **215 trades**, with a total gain of **1065 pips.**

Finally, we will examine the 4_8 moving average, with both parameters set to the shortest timeframe combination:

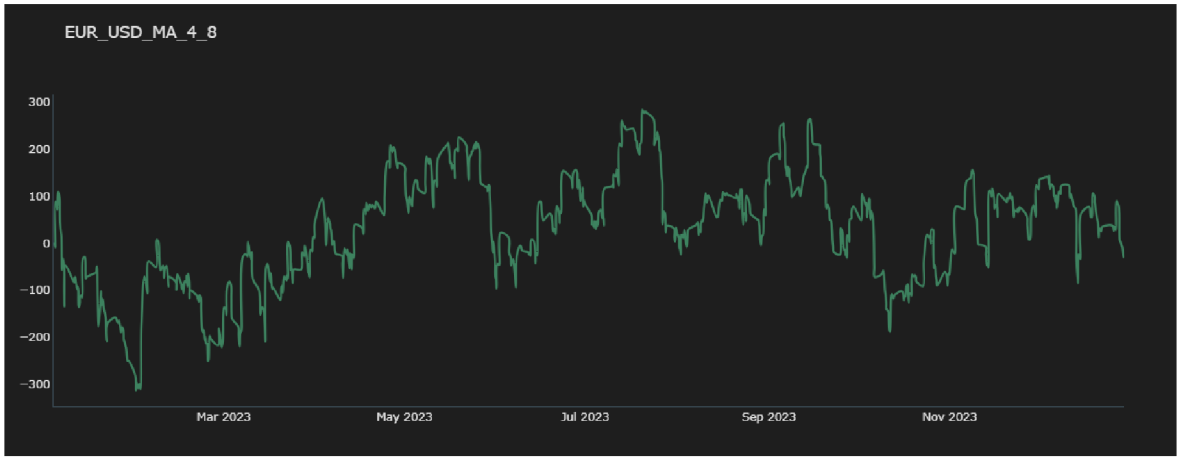


Figure 22: “EUR_USD_MA_4_8” gains through time

Source: Sample Project.

The chart (Fig. 22) shows how a strategy using **4_8** period moving averages performed on the EUR_USD currency pair. This strategy is more volatile and less consistent compared to longer period moving averages. There are many instances where the profit and loss threshold are crossed, indicating frequent changes in trade signals.

However, the strategy doesn't have a clear trend, as the gains are not consistently higher than the losses or otherwise. It stays around the breakeven point,

going up and down within a narrow range. This suggests that further optimization or the use of additional indicators is needed to improve the decision-making process for entering and exiting trades on a shorter timeframe. As a result, we got **885 trades** and a –**22.9 pip profit-loss**.

4 Results

Algorithmic trading backtesting is very quick, often completing in seconds, even with a lot of data. For instance, analyzing 5000 candles or more, which is a big part of historical price data, is done rapidly.

After testing different currency pairs and adjusting strategy parameters, analyzing Moving Average (MA) combinations shows a detailed connection between the specific periods used and trade profitability. The data clearly demonstrates how various MA settings impact trade success, providing valuable insights into fine-tuning algorithmic strategies.

MA_16_32	14	67%
MA_8_16	9	43%
MA_8_32	9	43%
MA_4_32	9	43%
MA_4_16	9	43%
MA_24_32	11	52%
MA_4_8	7	33%
MA_64_256	6	29%
MA_8_64	7	33%
MA_4_64	4	19%
MA_8_256	5	24%
MA_4_256	3	14%
MA_32_256	5	24%
MA_16_256	7	33%
MA_4_128	3	14%
MA_4_96	4	19%
MA_8_96	3	14%
MA_16_64	5	24%
MA_16_96	3	14%
MA_24_256	5	24%
MA_8_128	3	14%
MA_24_64	3	14%
MA_64_128	2	10%
MA_16_128	3	14%
MA_64_96	2	10%

Figure 23: Profitable trades % with different MA pairs.
Source: Sample Project.

For example, the **MA_16_32** combination (Fig. 23) stands out as highly effective, with a 67% success rate with 14 profitable pairs. This indicates that this particular MA setup is skilled at identifying real market trends versus random fluctuations, offering strong signals for traders to enter and exit positions.

On the other hand, broader MA combinations like **MA_64_128** and **MA_64_96**, with only a 10% success rate only with 2 profitable pairs each, suggest a strategy that might be too slow to adapt to rapid market changes. This delay could lead to taking bad trades or missed opportunities and an inability to capitalize on short-term market movements.

A moderate gap between short and long MAs, such as the **MA_16_32** mentioned earlier, seems to be ideal for aligning trade timing with market conditions, increasing the chances of profitable trades. Well-balanced MA pairings, especially those with success rates above 40%, offer a practical and potentially more profitable approach for algo-trading strategies utilizing MA crossover.

However, even the least effective combinations of Moving Averages (MA) can make money with certain currency pairs. This shows that each trading instrument is unique; what may not work well in one situation could work in another. Traders need to understand that every instrument reacts differently to market forces and trading strategies, so a one-size-fits-all approach won't work. It's important to customize strategies based on the specific characteristics and behaviors of each asset to maximize trading performance.

The success of Moving Average (MA) strategies is not uniform across different currency pairs. In a fast-moving market like GBP/JPY, the "**MA_64_128**" combination may not respond quickly enough, leading to suboptimal performance. However, in more stable markets like EUR/USD, where trends change slowly, a delayed signal from the same MA combination may not be a missed opportunity but can prevent false entries and result in unexpected gains.

The "**MA_32_96**" combination, with a wide gap between its short and long periods, may miss quick market reversals. However, for currency pairs with longer

periods of trend stability or less market noise, this combination can filter out irrelevant fluctuations and generate profitable trades.

These examples highlight the importance of considering the unique characteristics of each traded instrument in algorithmic trading strategies.

5 Conclusion

Algorithmic trading has embedded itself into the core of modern finance (2). Its impact can be seen in the smooth trade execution, fast transaction speeds, and thorough market analysis it provides (5). By combining math, data analytics, and computational power, it has changed how trades are done and set a new standard in the financial industry focused on efficiency, speed, and accuracy. This shift from traditional trading floors to computational finance environments highlights the importance of algorithmic trading. It greatly influences market behavior and global financial trends (6). The evolution of trading algorithms illustrates the critical balance between complexity and utility. While sophisticated models are capable of intricate analysis, there's growing recognition that simplicity often prevails. Overly complex algorithms can suffer from overfitting, where models are too finely tuned to historical data and thus fail in real-world trading, hindering their decision-making efficacy.

Simplicity, therefore, emerges not just as a design preference but as a strategic imperative for creating resilient and adaptable trading strategies (8).

The results additionally strengthen the concept that even basic models, like those relying on moving average crossovers, can generate profits when combined with careful risk management. Achieving success in trading is not solely dependent on the intricacy of models, but rather on their deliberate implementation and the strategic reduction of risks. The effectiveness of uncomplicated models serves as evidence to the principle that in the domain of algorithmic trading, simplicity can often be more advantageous (3).

When it comes to strategies, they have diverse implications on market dynamics. It possesses the capability to augment market liquidity and narrow bid-ask spreads, thereby promoting seamless and more effective price discovery mechanisms.

The impacts of algorithmic trading are not uniform and can vary depending on market circumstances, algorithmic design, and the specific strategies implemented.

Our results additionally strengthen the concept that even basic models, like those relying on moving average crossovers, can generate profits when combined with careful risk management.

Achieving success in trading is not solely dependent on the complexity of models, but rather on their deliberate implementation and the strategic reduction of risks.

The effectiveness of uncomplicated models serves as evidence to the principle that in the domain of algorithmic trading, simplicity can often be more advantageous.

The results of backtesting also showed us the importance of tradable instruments.

Different types of assets have varying levels of volatility, which can make trading more complex. Highly liquid markets need strategies that can handle complex dynamics and correlations, while volatile markets require strong risk management and quick-reacting algorithms.

Markets affected by diverse economic and sector-specific factors need algorithms that can process a wide range of data to take advantage of opportunities. T

rading strategies must align with the specific characteristics and volatility of the assets being traded to be effective (6).

As we move forward into the future, the importance of computers and algorithms in is becoming more apparent in the finance and global markets.

Algorithmic trading is a prime example of this shift, as it improves market efficiency and responsiveness through fast and accurate operations.

Algorithmic strategies will likely dominate the future of financial markets, as they can handle in amounts and speed that humans cannot match.

While this may not sound like something out of science fiction, it is clear that machine learning will play a crucial role in shaping investments and economic interactions on a global level.

6 Resources

1. Liberto, D. (2024, April 26). Loss aversion: Definition, risks in trading, and how to minimize. Behavioral Economics. Reviewed by R. C. Kelly, Fact checked by Y. Perez. Available at: https://www.investopedia.com/terms/l/loss-psychology.asp
2. Chan, E. P. (2013). Algorithmic Trading: Winning Strategies and Their Rationale. John Wiley & Sons. Available at: https://pdfroom.com/books/ernest-chan/Jr2ELAEAgyv
3. Chan, E. (2008). Quantitative Trading: How to Build Your Own Algorithmic Trading Business. John Wiley & Sons... Available at: https://www.academia.edu/28488586/Quantitative_Trading_Ernest_P_Chan
4. Doe, J., & Smith, A. (2022). Machine learning and speed in high-frequency trading. Journal of Financial Markets, 68. Available at: https://www.sciencedirect.com/science/article/pii/S0165188922001439
5. Doe, J., & Smith, A. (2022). The impact of algorithmic trading on market quality: Evidence from the Johannesburg Stock Exchange. Journal of Economic Perspectives, 35(4). Available at: https://doi.org/10.1080/10293523.2022.2090056
6. CME Group. (2010, July 15). Algorithmic Trading and Market Dynamics. Available at: https://www.cmegroup.com/education/files/Algo_and_HFT_Trading_0610.pdf
7. Liberty Street Economics. (2012, May). The Flash Crash, Two Years On. Liberty Street Economics. Retrieved January 27, 2024. Available at: https://libertystreeteconomics.newyorkfed.org/2012/05/the-flash-crash-two-years-on/
8. AnalystPrep. (2021, March 6). Overfitting and Methods of Addressing it. AnalystPrep. Retrieved January 27, 2024. Available at: https://analystprep.com/study-notes/cfa-level-2/quantitative-method/overfitting-methods-addressing/
9. Smith, B. M. (2003). A History of the Global Stock Market: From Ancient Rome to Silicon Valley. Retrieved 2024. Available at: https://www.oreilly.com/library/view/python-machine-learning/9781789955750/
10. Mallaby, S. (2010). More Money Than God: Hedge Funds and the Making of a New Elite. Retrieved 2024. Available at: https://bailiping.github.io/assets/docs/Books/MoreMoneyThanGod.pdf
11. Raschka, S., & Mirjalili, V. (2019). Python Machine Learning - Third Edition. Packt Publishing. Available at: https://www.oreilly.com/library/view/python-machine-learning/9781789955750/

12. Fernando, J. (2024, March 05). Profit and Loss Statement Meaning, Importance, Types, and Examples. Investopedia. Reviewed by Julius Mansa. Fact checked by Yarilet Perez. Available at: https://www.investopedia.com/terms/p/plstatement.asp
13. Fernando, J. (2024, January 30). Sharpe Ratio: Definition, Formula, and Examples. Investopedia. Reviewed by Margaret James. Fact checked by Katrina Munchiello. Available at: https://www.investopedia.com/terms/s/sharperatio.asp
14. Hayes, A. (2024, April 07). Maximum Drawdown (MDD) Defined, With Formula for Calculation. Investopedia. Reviewed by Gordon Scott. Fact checked by Yarilet Perez. Available at: https://www.investopedia.com/terms/m/maximum-drawdown-mdd.asp
15. Hayes, A. (2024, February 29). What Are Pips in Forex Trading and What Is Their Value? Investopedia. Reviewed by Gordon Scott. Fact checked by Kirsten Rohrs Schmitt. Available at: https://www.investopedia.com/terms/p/pip.asp
16. Mitchell, C. (2024, February 28). Understanding Basic Candlestick Charts. Investopedia. Reviewed by Gordon Scott. Fact checked by Kirsten Rohrs Schmitt. Available at: https://www.investopedia.com/trading/candlestick-charting-what-is-it/
17. Kramer, L. (2022, May 20). An Overview of Bull and Bear Markets. Investopedia. Reviewed by Julius Mansa. Fact checked by Kirsten Rohrs Schmitt. Available at: https://www.investopedia.com/insights/digging-deeper-bull-and-bear-markets/
18. Parker, T. (2023, June 30). 4 Behavioral Biases and How to Avoid Them. Investopedia. Reviewed by Robert C. Kelly. Fact checked by Pete Rathburn. Available at: https://www.investopedia.com/articles/investing/050813/4-behavioral-biases-and-how-avoid-them.asp
19. Two Sigma. (2022, July 5). Webinar: Machine Learning Models of Financial Data. Available at: https://www.twosigma.com/insights/webinar/machine-learning-models-of-financial-data
20. OANDA Corporation. (2024, April). Getting Started. In OANDA v20 REST API Development Guide. Available at: https://developer.oanda.com/rest-live-v20/development-guide/
21. Mitchell, C. (2024, April 5). How to Use a Moving Average to Buy Stocks. Investopedia. Reviewed by Charles Potters. Fact checked by Vikki Velasquez. Available at: https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp

22. Fuller, N. (2024). Inside Bar Trading Strategy. Price Action. Available at: https://priceaction.com/price-action-university/strategies/inside-bar/
23. Stevens, P. (2019, November 5). The secret behind the greatest modern day moneymaker on Wall Street: Remove all emotion. CNBC. Available at: https://www.cnbc.com/2019/11/05/how-jim-simons-founder-of-renaissance-technologies-beats-the-market.html
24. Dmytro Burliei. (2024). Forexbot. Available at: https://gitlab.com/ellisholfee26054/forexbot

7 Attachements

Sample project - https://gitlab.com/ellisholfee26054/forexbot

8 Assignment of work

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2023/2024

Studijní program: Aplikovaná informatika
Forma studia: Prezenční
Obor/kombinace: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

Jméno a příjmení:	**Dmytro Burliei**
Osobní číslo:	**I2000340**
Adresa:	Palachova 1129/17, Hradec Králové – Nový Hradec Králové, 50012 Hradec Králové 12, Česká republika
Téma práce:	Algoritmické obchodování
Téma práce anglicky:	Algorithmic Trading
Jazyk práce:	Angličtina
Vedoucí práce:	doc. RNDr. Kamila Štekerová, Ph.D., MSc. Katedra informačních technologií

Zásady pro vypracování:

Cílem práce je popsat využití strojového učení v algoritmickém obchodování a vytvořit vlastní model.
Osnova:
1.Úvod
2.Toretická část
2.1. Principy algoritmického obchodování
2.2. Případové studie a příklady
3. Praktická část
3.1. Implementace
3.2. Vyhodnocení
4. Závěr

Seznam doporučené literatury:

Chan, E. P. (2013). Algorithmic Trading: Winning Strategies and Their Rationale.
Chan, E. (2008). Quantitative Trading: How to Build Your Own Algorithmic Trading Business.
Doe, J., & Smith, A. (2022). Machine learning and speed in high-frequency trading.
Smith, B. M. (2003). A History of the Global Stock Market: From Ancient Rome to Silicon Valley

Podpis studenta: Datum:

Podpis vedoucího práce: Datum:

© IS/STAG, Portál – Podklad kvalifikační práce , burlidm1, 1. května 2024 21:56