h e g

Haute école de gestion
Genève

Designing an automated trading bot for strategic order execution

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Antonio ANTELO CORTEGANA

Conseiller au travail de Bachelor :

Grigorios ANAGNOSTOPOULOS, Adjoint scientifique HES

Genève, 19.09.2024

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre Bachelor en Informatique de Gestion.

L'étudiant a envoyé ce document par email à l'adresse remise par son directeur de mémoire afin qu'il l'analyse à l'aide du logiciel de détection de plagiat COMPILATIO.

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 19 septembre 2024

Antonio ANTELO CORTEGANA

Remerciements

First of all, I would like to express my sincere gratitude to my Bachelor's thesis supervisor, the Senior Research Associate Mr. Grigorios G. Anagnostopoulos who accepted to supervise my Bachelor's thesis. His expertise in Machine Learning, along with his insightful advice and book recommendations, greatly enhanced my understanding and the development of my work. I deeply appreciate his trust encouragement, and the time he dedicated to helping me succeed.

As English is not my native language, I found writing my Bachelor thesis quite challenging. Therefore, I would like to express my gratitude to my former colleague and friend, Mrs. Deborah Gottlieb, for the time she dedicated to reviewing my work. Her insightful comments on my English and her constructive feedback on my topic were incredibly helpful.

Lastly, I would also like to thank my family and close friends for always believing in me and for their constant support and encouragement, especially during the times when I struggled with motivation. A special thanks to my sister Natalia Antelo Cortegana for her endless patience in reviewing and providing feedback on my work multiple times.

I also utilized modern technologies to support my work. Github Copilot was used to assist in writing code efficiently, while ChatGPT provided valuable help in refining some of the English phrasing. These tools were employed thoughtfully, ensuring that the integrity of the work remained my own.

Résumé

This thesis explores the field of trading with the main objective of designing an automated trading bot that is theoretically able of executing strategic orders. The project uses a multilayer perceptron (MLP) neural network to predict future price movements of a stock, based on historical data. However, the model has not yet reached a level of accuracy sufficient for a deployment in real life.

The data used to train the model includes price history of the stock in the form of Open, High, Low and Close (OHLC) enriched by four technical indicators (EMA, MACD, RSI and VWAP) to improve prediction accuracy as much as possible. Although the project is limited to analysis and simulation on historical data, it provides a solid foundation for the creation of a trading bot. The thesis also includes a comparison with baseline strategies to analyze the model's performance in simulated trading.

The different sections cover a review of key trading concepts, including a part on Swiss legislation, as well as the development of an automated trading strategy.

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

1.1 Background

As society progressed, trade systems have significantly evolved, transitioning from the traditional act of bartering to the creation and development of financial markets. The rise of the internet has transformed the “landscape”, allowing individuals, known as retail traders, to access financial markets that were once exclusively available to institutional investors. This shift has democratized trading, making it more accessible to the general public.

Today, trading involves the buying and selling of various financial assets, such as currencies, stocks, commodities, and cryptocurrencies, all with the aim of generating profit. The trading environment can be particularly stressful, especially in volatile markets rife with uncertainties. Individuals employ various methods to make decisions about buying and selling. These methods can be influenced by a wide range of factors, including emotional influences. Therefore, effective management is essential in order to avoid any irrational decision-making.

1.2 Thesis goal

The primary aim of this thesis is to develop an automated trading bot designed to execute buy and sell orders based on predefined strategies. This system seeks to eliminate the emotional biases that often influence human trading decisions. Emotions such as fear and greed can significantly impair judgment, leading to suboptimal investment choices. To address this, the bot will employ a machine learning model known as a Multilayer Perceptron (MLP).

In this project, the bot aims to predict stock price movements—whether they will rise or fall, by analyzing historical data. This data includes key financial metrics such as Open, High, Low, and Close prices (OHLC), trading volume, and a selection of technical indicators. These indicators serve as inputs to the MLP model, which processes the data to uncover patterns not immediately apparent to human eyes. By leveraging this machine learning technique, the bot seeks to forecast future price movements and provide valuable insights into potential stock trends, enhancing trading accuracy and maximizing profit opportunities.

1.3 Thesis structure

This thesis is organized into nine chapters. It starts with a literature review on trading concepts (Chapter 2) and the development of artificial intelligence (Chapter 3). The

following chapters detail the development of the automated trading system (Chapter 4) and its methodology (Chapter 5), its implementation (Chapter 6), and a performance analysis (Chapter 7). Before concluding, this thesis will explore potential future developments and areas for improvement (Chapter 8) with a summary of key findings and future considerations (Chapter 9).

2. Literature review of trading: Main concepts

The purpose of this chapter is to present and explore the main concepts that must be taken into account in the context of our research for the Bachelor's thesis. Indeed, the development and conceptualization of a trading bot require the consideration of several complex parameters. This literature review aims to provide an overview of the theoretical foundations of trading, focusing on essential notions such as definition of trading, market types, buying/selling, the importance of trading psychology, the risk management, the legislation, and the technical indicators for trading strategies. These concepts will play a crucial role in the development of an efficient trading bot, tailored to the conditions of modern markets.

2.1 Definition of trading

The trading market has evolved considerably in the recent years. Historically, the first form of trade was through bartering. According to Kenton (2023), bartering was an act of trading goods or services between two or more parties without using money or any other form of currency (Kenton 2023). Essentially, bartering involved trading one person's goods or services for another person. Bartering is considered the oldest form of commerce in the world.

How did exactly bartering function?

To illustrate the act of bartering, we can consider a situation where Mr. A has 5 kilos of rice, which he assesses the value at CHF 35.00, but has not use for it. He could exchange it with Mr. B, who might need it. In return, Mr. B would need to offer a good or service valued at CHF 35.00, regardless of whether Mr. A needs it. This scenario is a simple illustration of how bartering worked in the past.

In recent years, trading has become widely accessible due to the internet and various online trading platforms that are available worldwide. This digital transformation has opened opportunities for new participants, known as retail traders, to buy and sell the same financial markets that were, until recently, exclusive to the financial elite.

What is the state of the trading market today?

As society progressed, trade systems evolved beyond bartering, leading to the creation and development of financial markets. The digital transformation has opened opportunities for new participants, known as retail traders, to access financial markets that were, until recently, exclusive to the financial elite The rise of the internet and online trading platforms has made trading widely accessible. The focus has shifted from

bartering goods to directly buying and selling assets such as currencies, stocks of public companies, commodities, cryptocurrencies, all with the aim of making a profit. This shift from bartering goods to trading assets not only represents a significant change in how trading is conducted, but also brings a need for a stronger surveillance. Consequently, trading activities are now strongly regulated by various governmental agencies. For example, in Switzerland, the Swiss Financial Market Supervisory Authority (FINMA) is responsible for overseeing these regulations.

Several types of markets exist where trading takes place, and a few of the most prominent are outlined below.

2.2 Market types

2.2.1 Forex

The forex market, short for Foreign Exchange, is the world's largest financial market, with a daily trading volume of 7.5 trillion US Dollars, according to a report from the Bank for International Settlements (Chen 2024).

This market includes large global banks, many trading companies, brokers and retail traders.

How does the forex market work?

The Forex market involves two currencies. For example, we can consider the EUR/USD pair, also known as "The Fiber". In this pair, the first currency, the euro, is referred to as the base currency, and the USD being the second currency, is the quote currency.

If a person wishes to trade this pair (wants to buy EUR/USD), the person would ask a bank or a broker for a quotation. We can suppose that today this pair EUR/USD is trading at 1.08, meaning to receive 1 EUR equals to 1.08 USD. Therefore, to buy one currency, I must sell the other currency. Taking the above example, it's like betting that the euro will appreciate against the US dollar.

The forex market is highly appreciated due to its liquidity. This liquidity allows traders to easily buy and sell currency pairs without impacting their price. Furthermore, the market is open 24 hours a day, five days a week, enabling investors to quickly react and respond to global economic events, regardless of the time or day.

2.2.2 Stock market

The stock market is a marketplace where shares of public companies are traded worldwide. These trades happen in formal marketplaces, like the New York Stock Exchange (NYSE) in the United States or the SIX Swiss Exchange located in Zurich,

Switzerland. Trades can also occur in less formal markets, known as Over-the-Counter (OTC) markets.

Unlike the NYSE, which is a centralized exchange, the OTC market is decentralized. Here, securities such as stocks, bonds, commodities are traded directly between parties without a formal exchange or intermediary. Due to its decentralized nature and absence of a centralized regulatory authority like the Securities and Exchange Commission (SEC) in the United States, the OTC market might offer less transparency. This can lead to higher risks for traders but also presents a potential for higher returns (Murphy 2024).

It is important to note that many countries have their own regulatory system to oversee these exchanges.

The stock market operates based on supply and demand. When we purchase a share of a company, we acquire a very small portion of ownership in that company. If the company performs well and generates significant profits, it will attract new investors. This increased interest can drive up the value of its shares, potentially leading to a rise in stock prices. The higher the demand, the lower the supply, which can cause the price to increase. On the other hand, if investors see that the company is facing some difficulties, the stock price might decrease.

Traders are making profits by buying shares at a lower price and selling them at a higher price. However, they can also lose money if the share's value decreases after they have bought it.

The stock market is an important actor in the global economy. It allows companies to raise funds by selling their shares in the market and traders have the opportunity to financially contribute to the success of the company (Gratton 2024).

2.2.3 Commodities

Beyond forex and stocks, commodity trading introduces about another kind of market that deals with everyday goods and raw materials that we all use in our daily lives. When we enjoy a cup of coffee in the morning, use our smartphones throughout the day, or fill our car's tank with gasoline, we are interacting with commodities. Indeed, commodity trading involves everyday goods, such as, and among others, agricultural products (corn, soybeans), precious metals (gold, platinum), and energy sources (oil ,electricity).These commodities are crucial for daily life and the global economy.(Capital.com 2017).

Trading commodities involves financial instruments like complex derivative securities such as futures contracts, options and swaps. These financial instruments allow traders

to speculate on future price movements or hedge their positions against potential risks (Fernando 2024a).

Commodity trading plays a crucial role in our global economy by ensuring the smooth transfer of goods from the producers to the consumers. Additionally, it opens the opportunities for investment and provides strategies for reducing risks by diversifying an investor's portfolio.

2.2.4 Cryptocurrency trading

The objective of cryptocurrency trading is similar to forex and stock markets: to buy assets at a low price and sell them at a higher price to generate profit, but with cryptocurrencies.

While traditional markets revolve around well-established currencies and goods, cryptocurrencies offer a more volatile and decentralized form of trade. Indeed, cryptocurrencies can be traded on various platforms, categorized as CEXs (Centralized Exchanges) or DEXs (Decentralized Exchanges).

Centralized Exchanges like Binance, Coinbase and Kraken act as intermediaries between buyers and sellers. In contrast, Decentralized Exchanges such as Uniswap and PancakeSwap operate without a centrale authority, enabling direct transactions between buyers and sellers (Ehrlich 2023).

Cryptocurrency trading is especially popular among the younger generation and might be due to its volatility. This volatility allows investors to potentially make significant profits quickly, however, it also increases the risk of equally rapid losses (Stilt, Gogol 2021).

In summary, there are many more types of markets: forex markets for currency, stock markets for company shares, commodities markets for raw materials, and cryptocurrency markets for digital assets. Nevertheless, an ideal strategy for every investor is to build a diversified portfolio to provide protection against market volatility. Diversification across these various asset classes can help to balance out and mitigate risk as they often react differently to economic events, ensuring long-term growth and protection against market downturns.

2.3 Buying/selling

Trading offers several ways to generate profits. The simplest and most well-known method involves buying an asset at a lower price and selling it at a higher one, pocketing the profits after deducting the broker commissions. This is called "going long" and it essentially means betting on the rise of our asset.

On the other hand, there is also an opposite strategy, known as “going short” or “short selling”, which essentially consists of opening a selling position. This technique, often complex for beginners, involves selling a stock that one does not own. It is very risky for all types of investors, especially novices, because the losses can be really significant.

For example, in a long position, the worst-case scenario for a trader is losing the entire initial investment. To simplify our explanation, we will exclude the broker fees. If Trader A goes long on Apple stock, anticipating that the stock will soon increase in value, so he buys 10 APPL shares at USD 170.00 each, for a total initial investment/position of USD 1’700.00. The worst scenario he could face is if the company Apple goes bankrupt and the stock price collapses to USD 0.00 per share, leading to a total loss of USD 1’700.00, which is the amount he initially put at stake.

Regarding the short position, we can imagine that Trader B predicts that Apple’s stock will soon decrease in value and opposite to Trader A, he decides to go short on it. He then sells 10 APPL shares that he does not own at USD 170.00 and just as Trader A he is creating a position of USD 1’700.00. However, unlike the Trader A, the maximum loss that he can encounter is theoretically unlimited. Indeed, if Apple’s stock keeps rising “to the moon” his losses will continue to increase until he closes his position. This could happen either by his choice or if the broker judges the position too risky and decides to close it for him (Kramer 2024).

“Going short” is a complex concept and how can we sell a stock that we do not own? This strategy can be better illustrated with a similar example without directly discussing trading.

My friend (the broker) lends me his car (representing the stock) for two weeks. Upon taking possession, the car’s value is estimated at CHF 20’000.00 (the position). Convinced that its value will drop by CHF 10’000.00 in the next two weeks, I decide to sell it on the market and cash in CHF 20’000.00. After two weeks pass, I am obligated to return the borrowed car to my friend, so I need go back to the market and repurchase it. Fortunately, the price has decreased as I anticipated, and I am able to buy it back at for only CHF 10’000.00, even though I initially sold it for CHF 20’000.00. I return the car to my friend, including a small amount of interest and I received a profit of CHF 10’000.00.

2.4 The importance of trading psychology

While strategies like “going long” or “going short” are key to trading, understanding trading psychology is also important. Trading psychology plays a fundamental role for

both professional and casual traders. To maintain long-term performance, traders must manage their emotions and behaviors based on what is happening in the market.

The trading world is a very stressful universe, especially in volatile markets with a lot of uncertainties. For this reason, every investor must work on managing their emotions. A lack of control over these emotions can lead us to making irrational decisions.

A common example of this is FOMO (Fear of Missing Out). It is a human feeling, which can push traders to make quick decisions. Recent events in the cryptocurrency world, where prices are soaring, show this phenomenon. Investors see an opportunity to make large profits in a short time, and as no one wants to miss the moment, they invest without fully assessing the risks (lborrmann 2022).

There is also the fear of being wrong and losing money, which can lead traders to hold a certain position. For example, if we have a buying position and the market does not go our way and continues to drop, our emotions might push us to keep the position open, convinced that the market will eventually rebound. An experienced trader would have set a stop loss, thinking that beyond this barrier, the trade becomes too risky, so they must close the position and accept the loss.

What are the solutions to mitigate this risk?

First, one must plan a strategy. Before taking a position, it is crucial to clearly define the following objectives: the entry point, the desired profit, and the case if the market is against us, set a stop loss with the maximum loss we are prepared to accept. Most importantly, once our objectives are well defined, we must not deviate from them at all. This will be the guiding principle to stay effective and not be influenced by emotions (Makarenko 2024).

Another good practice is also to create a trading journal to record all past transactions. This journal should include details like the date of the trade, side (buy/sell), quantity, ticker and the amount of profit or loss. This journal is an excellent tool for analyzing post-trade mistakes and thus improving future trades.

2.5 Risk management

In addition to the previous section, the risk management is an essential element for any trader aiming to protect investors from losing their capital. Risk management protects investors by preventing the loss of all their money and is a crucial pillar in trading that helps control and limit potential losses. We can imagine, even if a trader has closed a

good trade and generated a very good return, the trader is not safe from losing everything quickly if he is not having rigorous risk management.

For instance, a person with an account of CHF 1’000.00 should not take the same types of risks, nor commit the same proportion of ones capital in a single trade, as someone with an account of CHF 1’000’000.00. The consequences of a loss are much more severe for the smaller account in terms of the total capital percentage.

There is a well-known rule that many traders follow, the “one-percent rule” which simply states that no trader should lose more than 1% and, at most, 2% of their portfolio’s capital in a single trade (Kuepper 2023). Therefore, if the amount of capital in our portfolio is CHF 10’000.00, the maximum loss acceptable before closing the position is from CHF 100.00 to CHF 200.00 for a single trade.

2.6 Legislation

While risk management strategies are crucial for protecting capital, they must be applied within the framework of relevant financial legislation. Investors holding shares in Switzerland benefit from a relatively advantageous tax framework compared to other countries. The profits are tax-exempt for private investors, while professional investors are subject to different rules.

The distinction between a private and a professional investor is important. A person is categorized as a private investor if the money that has been invested in the stock market comes from another source of income, such as the salary from his main activity, or if the person does not live off his investments.

In Switzerland, during the tax declaration, the entire stock portfolio must be declared as wealth, and not as income. Therefore, while profits are exempt from taxes any dividends received must be declared as income. Switzerland levies a 35% withholding tax on the amount of dividends, which can be recovered in the next tax declaration if it is properly filled out (PostFinance 2023).

Swiss residents wishing to invest in the stock market must first go through a broker research session. One cannot open an account with just any international broker. Several points need to be considered besides the prices that are charged for the commissions, the spread, and any maintenance fees. One needs to be careful in the local regulations in Switzerland and the broker’s policy regarding international clients. Some brokers may choose not to accept Swiss residents due to compliance with the local regulations.

Indeed, the financial sector in Switzerland is heavily regulated and is overseen by the Swiss Financial Market Supervisory Authority (FINMA). Its role is to ensure that the various financial institutions operating in Switzerland follow its rules. These rules are in place to protect clients in the financial markets (FINMA [sans date]).

2.7 Technical indicators for trading strategies

2.7.1 Overview

While legislation offers the necessary protections for traders, understanding market behavior through the use of technical indicators is equally important. These indicators, ranging from trend and volume analysis to oscillators and volatility measures, allow traders to analyze market movements and make data-driven decisions. Let's dive into the key technical indicators that can support various trading strategies and enhance market analysis.

There are countless indicators designed to help us make decisions. However, it's important to understand that an indicator is not a magic potion that is always right. The markets are known to be very unstable and unpredictable, and no one can predict whether a stock will rise or fall. The indicators are there to support our analysis. However, there are not capable of confirming that a stock will increase our decrease in value.

There are many indicators available, some are offered by brokers, while others have been established by developers or amateurs who can share them for free or for a subscription fee.

As mentioned previously, there are so many technical indicators that these can be considered into different categories: trend, volume, volatility and oscillators. Each categories include various tools that are specific aspects of market analysis.

2.7.1.1 Trend Indicator:

Trend Indicators are very useful in helping to identify the general direction of the market. For example, the Moving Average (MA) (represented by a purple line) is a technical tool that can be used here. A flat moving average indicates us that the market is very neutral, while an ascending moving average suggests a bullish market, and a descending moving average indicates a bearish market (Mitchell 2024a).

Figure 1 Trend Indicator: Moving average

2.7.1.2 Volume Indicator:

Understanding transaction volume is a powerful indicator as it reveals the strength of buyers and sellers behind a trend. The On-Balance Volume (OBV), represented by a purple rectangle, is one of the indicators in this category. It accumulates the total volume of transactions over a given period, adding this volume to the total if the market finished higher, or subtracting it if it finishes lower (Mitchell 2024b). OBV helps traders assess whether a current trend has the strength to continue.

Figure 2 Volume Indicator: On-Balance Volume

2.7.1.3 Volatility Indicator:

Volatility indicators aid in understanding whether a market is stable or unstable by measuring the degree of volatility. A high volatility means that prices can rise or fall very rapidly within a very short period of time.

The Bollinger Bands are a technical tool within this category and consist of three lines. The middle line is the moving average, and the two other lines measure the standard deviation above and below this moving average. If the two lines are far from the moving average, it indicates a high market volatility, and if they are close together, it indicates that the market is calm (Groupe IG 2019).

Figure 3 Volatility Indicator: Bollinger Bands

2.7.1.4 Oscillator Indicator:

Oscillator indicators fluctuate between high and low values, which helps to take decision when market trends are unclear or when the prices show no significant upward or downward movement. They are often used for short-term trading (Chen 2021).

The Stochastic Oscillator is one of these oscillators. It measures the strength of a stock's price movement and can identify trend reversals. It compares a stock closing's price on a particular day to its price range between the highest and lowest price over a specific time period (Tradeciety 2023), (Hayes 2023).

On the right and left side of the graph, we can see the Stochastic above the 80 area. It suggests that market would reverse and price may go lower.

Figure 4 Oscillator Indicator: Stochastic Oscillator

The best approach in building a strategy often involves combining one or more of these technical indicators to see which configuration works with our style of trading to achieve the best results. However, using all available indicators at the same time is generally not recommended, as it can lead to confusion and mixed signals, making it very hard to make decisions.

2.7.2 Description and use of selected indicator

My strategy will be a combination of the following 4 technical indicators: Exponential Moving Average (EMA), Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD) and Volume Weighted Average Price (VWAP).

Each indicator works occasionally, although not all the time, so my strategy is to combine all four to try to maximize profits. However, there is a point to be careful about, because the more technical indicators we use, the harder it can be to enter a trade, even making it impossible, as my trading robot will have to wait for the market to meet all my entries criteria before opening a position.

I will define each of the indicators that I have selected and illustrate their mathematical formulas.

2.7.2.1 EMA

$$\text{EMA} = \text{Price today x}\left(\frac{2}{n+1}\right) + \text{EMA previous x}\left(1 - \frac{2}{n+1}\right)$$

$$n = \text{ number of period}$$

$$\left(\frac{2}{n+1}\right) = \text{is what is called the smoothing factor that gives more weight to recent prices}$$

This Exponential Moving Average (EMA) helps to identify buy and sell signals by analysing the stock prices over a defined period. It indicates the trend and is more accurate than the Simple Moving Average (SMA), because it responds more quickly to recent price changes, therefore it is often used for short-term trading by intraday traders (Capital.com [sans date]), (Strike 2023).

When the blue EMA line is below the stock price yet one can see the trend going upwards, it indicates a bullish market. Inversely, when the EMA line is above the stock price and trending downwards, a bearish market is indicated.

Figure 5 Selected Indicator: EMA

2.7.2.2 RSI

$$\mathrm{RSI} = 100 - \left(\frac{100}{1 + RS}\right)$$

$$\mathrm{RS} = \left(\frac{Average\ Gain}{Average\ Loss}\right)$$

(Wilson 2024)

The Relative Strength Index (RSI) is one of the oscillators category. It operates on a scale of 0 to 100 and is used to assess price movements. The RSI informs us if an asset is overbought when the line crosses above the 70 area, which could indicate that the price may soon decrease, or oversold when the line crosses below the 30 area, suggesting a potential price increase (Fernando 2024b).

Figure 6 Selected Indicator: RSI

2.7.2.3 MACD

$$\text{MACD} = \text{EMA}_{12} - \text{EMA}_{26}$$

$$\text{EMA}_{12} = \text{is the exponential moving average over 12 periods}$$

$$\text{EMA}_{26} = \text{is the exponential moving average over 26 periods}$$

$$\text{Signal Line} = \text{EMA}_{9}(\text{MACD})$$

$$\text{EMA}_{9} = \text{is the exponential moving average over 9 periods of the MACD line}$$

$$\text{Histogram} = \text{MACD} - \text{Signal Line}$$

The Moving Average Convergence Divergence (MACD) indicator is complementary to the EMA. Like the EMA, it provides buy and sell signals trough the MACD line (blue) and the Signal line (orange). When the MACD line crosses above the Signal line, it is a buy signal, conversely, when the MACD line crosses below the Signal line, it is a sell signal. The MACD can also help to identify trends and their reversals (Schlossberg 2024), (Dolan 2024).

Figure 7 Selected Indicator: MACD

This indicator consists of several elements shown in the image above. First, there are two lines. The MACD line, usually in blue, which is the difference between the EMAs calculated over 12 and 26 periods, and the Signal line is usually in orange, which is the 9 period EMA of the MACD line itself. Additionally, there is a histogram that visualizes the difference between the MACD and the Signal line. If the MACD is above the Signal line, the histogram is positive, otherwise it is negative.

2.7.2.4 VWAP

$$\text{VWAP} = \frac{\sum(P_i \times V_i)}{\sum V_i}$$

$$P_i = \text{average price} = (\text{High} + \text{Low} + \text{Close})/3$$

$$V_i = \text{current volume}$$

The VWAP (Volume Weighted Average Price) is the last trading indicator that I have chosen for my strategy. As its name suggests, the VWAP is the average price of a stock weighted by the total trading volume over the selected period. This indicator helps to buy or sell stocks or other assets at a price close to the average price. This means that if I sell above the VWAP line, I can be confident that I am selling the stock at a price higher than the average (CFI Team [sans date]), (Martin 2024)

Figure 8 Selected Indicator: VWAP

The VWAP can help to identify entries and exits points for the positions. If the price is located above the VWAP line in blue, it may signal a buying opportunity, because the price of the market is higher than the volume-weighted average, which indicates a buying pressure from the investors. Conversely, if the VWAP is above the price, it may indicates us a selling opportunity, as the price is below the average, showing a selling pressure (Fernando 2024c).

3. State of the art Artificial Intelligence & trading

In the chapter 2, we explored several key technical indicators—EMA, RSI, MACD, and VWAP—which traders use to analyze price movements and predict market trends. While these indicators offer valuable insights, trading success requires the ability to process and interpret large amounts of market data quickly and efficiently. This is where Artificial Intelligence (AI) comes into play. In the framework of my Bachelor thesis, I would like to complement my trading strategy by integrating the use of Machine Learning, specifically through the implementation of neural networks to predict future price movements and improve decision-making.

3.1 Artificial Intelligence's role in trading

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think, learn, and make decisions autonomously. While it cannot yet replicate human consciousness or self-awareness, current AI technologies, are capable of simulating human-like decision-making processes. Within the field of AI, Machine Learning (ML) and Deep Learning (DL) represent powerful methods that enable computers to learn and improve without explicit instructions.

Machine learning (ML) enables systems to autonomously learn and make decisions from data, as exemplified by Oracle's Autonomous Database, which automates data management and improves security while reducing human error (Oracle [sans date]).

Deep learning (DL) is a category of ML. It refers to models that are designed to learn from vast amounts of data through multiple layers of artificial neural networks. It mimics the human brain's ability to process complex data, which makes it highly effective in identifying non-linear relationships in stock prices and market trends (Holdsworth 2024).

In trading, ML and DL have become powerful tools for analyzing market trends, predicting price movements, and optimizing trading strategies. Unlike manual analysis, AI can process vast amounts of real-time data, identifying patterns and trends that would be difficult for human traders to detect (Bots 2024).

As a result, AI has transformed trading, from generating predictive models to executing trades, thereby minimizing human errors and emotions, which are often detrimental to successful trading.

3.2 Application of Artificial intelligence in trading firm

Today, AI is extensively used in the financial industry, with many firms integrating ML and DL into their trading strategies. A 2018 survey by BarclayHedge revealed that 56% of the participants used Artificial intelligence/Machine Learning to assist in making investments decisions, while approximately 2/3 relied on the technology to generate investment ideas. And additionally, a quarter of the survey participants reported using AI or ML for trade execution (IG 2022).

Firms such as Renaissance Technologies founded by James Simons with its famous Medallion Fund, has consistently outperformed the financial market with an average annual yield of 39.9% since his creation in 1988. Their success is largely granted to the integration of advanced quantitative models, leveraging large dataset with Machine Learning and Deep Learning algorithms to identify profitable patterns in market movements (Karlsson 2024).

This highly confidential and data-driven approach has set a new standard in algorithmic trading, inspiring many to explore the potential of AI in finance.

4. Development of an automated trading system

In the previous chapters, we explored the essential technical indicators—EMA, RSI, MACD, and VWAP—that help traders to make more rational decisions. While each of these indicators can be useful on its own, they don't always provide consistent or reliable signals when market conditions are volatile or unpredictable. Combining these indicators with a sophisticated decision-making model could increase the chances of success. This leads us to the development of an automated trading system that not only incorporates these indicators but also reduces the impact of human emotions on trading decisions.

4.1 Benefits using a trading bot

One of the primary reasons to use a trading bot, instead of manually executing trades based on the previously discussed indicators, is the emotional factor. As explained before, when we start trading with our own money, our personal funds, we are confronted with emotions that, without experience, can lead us to incur significant losses.

For example, we can imagine to buy 1 Bitcoin at 55,000 USDT, because we have heard that the market is bullish (meaning that is going up). Luckily, we were right, the next day the price increases by 5,000 USDT. An experienced person would have probably already secured their profits, yet the others might think: "Let's wait for another day as it could go up even more".

The following day, the price rises by another 2,000 USDT, reaching it at 62,000 USDT. At this point, although 7,000 USDT in profit is already very satisfactory, we hesitate to close the position, thinking that the price might continue to rise. Thus, our emotions urge us to stay instead of securing the gains already made.

However, the next day, the price drops sharply to 45,000 USDT. Not only have we lost the previous gains, but our trade is at loss. Faced with this, we have two choices available: either we wait and hope that the price will recover or we accept the loss of 10,000 USDT. Often, our instinct pushes us to wait in the hope for a market rebound. However, if the price continues to fall, we could end up taking even greater loss.

4.1.1 The importance of an automated trading bot

The scenario above is common and reflects the dilemmas that many traders, professional or not, are often faced with: waiting in hope that luck will turn my way next day. This is one of the main reasons why I decided to build my trading bot.

The main argument for using a trading bot developing by us or purchased from someone is that it lacks emotions. It will only execute what it has been programmed to do. If a

position is profitable, it will automatically close the position according to the parameters that have been defined beforehand, without wondering if the price could rise further. Similarly, if our trade is losing, it will not wait for a potential rebound. It will close the position based on the present criteria.

In brief, an automated trading bot will adhere strictly to the predefined strategy and parameters, making rational decisions based solely on data. This prevents the common mistakes of hesitating to take profits or holding onto a losing position in the hope that the market will reverse.

4.2 Overview of the trading strategy

My trading strategy is based on the use of a multilayer perceptron (MLP) neural network, which we will be discussed further in the upcoming chapters about the Methodology (Chapter 5) and the Implementation (Chapter 6). This neural network will be tasked with making automated trading decisions. The bot will analyse historical price data, specifically the Open, High, Low and Close (OHLC), along with the following four technical indicators: EMA, RSI, MACD and VWAP, which were covered in the previous chapter. This combination will help us to determine when to open and close the positions in the market.

4.2.1 Order execution workflow

The MLP neural network is trained on historical data, with the objective of predicting the probability that the price of our asset will rise or fall in the next period. The idea is as each new data point is received by the model, it will analyse the different prices (OHLC) and calculate the four indicators to assess the probability of a price movement.

- Buy signal

  If my model predicts with a probability higher than a certain threshold, for example 51**%**, that the price will increase in the next period, my bot should open a long position, which means buying the asset.

- Sell signal

  Conversely, if the model predicts that the price will decrease, the bot could open a short position, which involves selling the asset, commonly known as shorting, as the trend is going down. However, to keep the example simple, we have excluded this scenario.

Once there is an open position, my closing strategy is based on several scenarios.

- Trend reversal

  If in the following periods, my MLP model starts predicting a trend reversal, for example, from an upward trend to a downward trend or vice versa, my bot will close the position to secure gains or minimize losses.

- Stop loss

  Upon opening the position, we must set a stop loss order in advance to limit significant losses if the market moves against our position. It will automatically execute once our position crosses the maximum loss threshold that we have defined beforehand (2%).

- Take profit

  This is closely similar from the stop loss, however here we will define our profit targets to close the position. For example, if we are in a buying position, we will instruct it to close the position once we have reached a profit of 5%.

4.2.2 Explanation of the strategy choice

While indicators such as RSI, MACD, VWAP and EMA can sometimes be effective in identifying trading opportunities, these indicators do not always work and can produce unreliable signals. Each indicator has its own strengths and weaknesses, and their effectiveness can vary depending on market conditions.

My approach is therefore to combine these four indicators with historical price data and integrating them into a multilayer perceptron (MLP) neural network to leverage and maximize their strengths. The purpose of the MLP model is to detect complex patterns that are difficult for a human to spot it. This allows more robust predictions rather than relying on a single indicator.

4.3 Introduction to Alpaca API: features & benefits

For market data collection, construction, testing and execution of my automated orders, I opted for a free solution offered by an American broker. Therefore, I decided to open an account with Alpaca Securities LLC, which provides the necessary tools to successfully carry out my trading bot project.

Alpaca is a broker regulated by the Securities and Exchange Commission (SEC) and the Financial Industry Regulatory Authority (FINRA). The SEC is a U.S. federal agency similar to FINMA in Switzerland, with the mission to protect investors, maintain orderly and efficient financial markets. The FINRA, is an independent nongovernmental organization operating under the supervision of the SEC, responsible for regulating exchange members to protect investors from abuses (Manning 2023).

I also chose Alpaca, because it offers detailed documentation that helps developers to test and implement their trading strategies. For this project, I opened a paper trading account. This type of account allows me to simulate market transactions without risking real money, which is an essential safety measure in the testing phase of my bot (Alpaca [sans date]), (Nasli 2024).

4.4 Choice of python and libraries

I selected Python as language for the development of my automated trading bot for several reasons.

First of all, since I already have a foundation and I am familiar with this language, I found it very intuitive and accessible for my project. Additionally, one of Python's main strengths is its large online community, which provides a wealth of free or really affordable educational resources, training and forums. This support is a key advantage for quickly solving technical problems that we may face.

Another strength of this language is its wide range of specialized libraries which are perfectly suited to my needs in trading and machine learning to successfully complete my project. For example, the Alpaca libraries provided by my broker Alpaca Market, I was able to execute REST API requests that allowed me to retrieve the price history of a stock and I also have the possibility to place buy or sell orders directly from the code without using the user interface.

For computing my technical indicators, I use the Pandas_ta library which allows for simple and efficient data manipulation and analysis.

For backtesting my strategies, there is the Backtrader library which offers well detailed summaries. However, for this project I finally opted for a more educational and flexible approach by developing my own backtesting system to better fine tune each step of the process.

Finally, for the Machine Learning part and the creation of my neural network, I am using TensorFlow. It is a very powerful and practical tool for creating, testing and improving my MLP (Multi-Layer Perceptron).

5. Methodology

To build the bot, I plan for the neural networks to analyze the historical data of stocks to identify any patterns in price movements that might be difficult for humans to see.

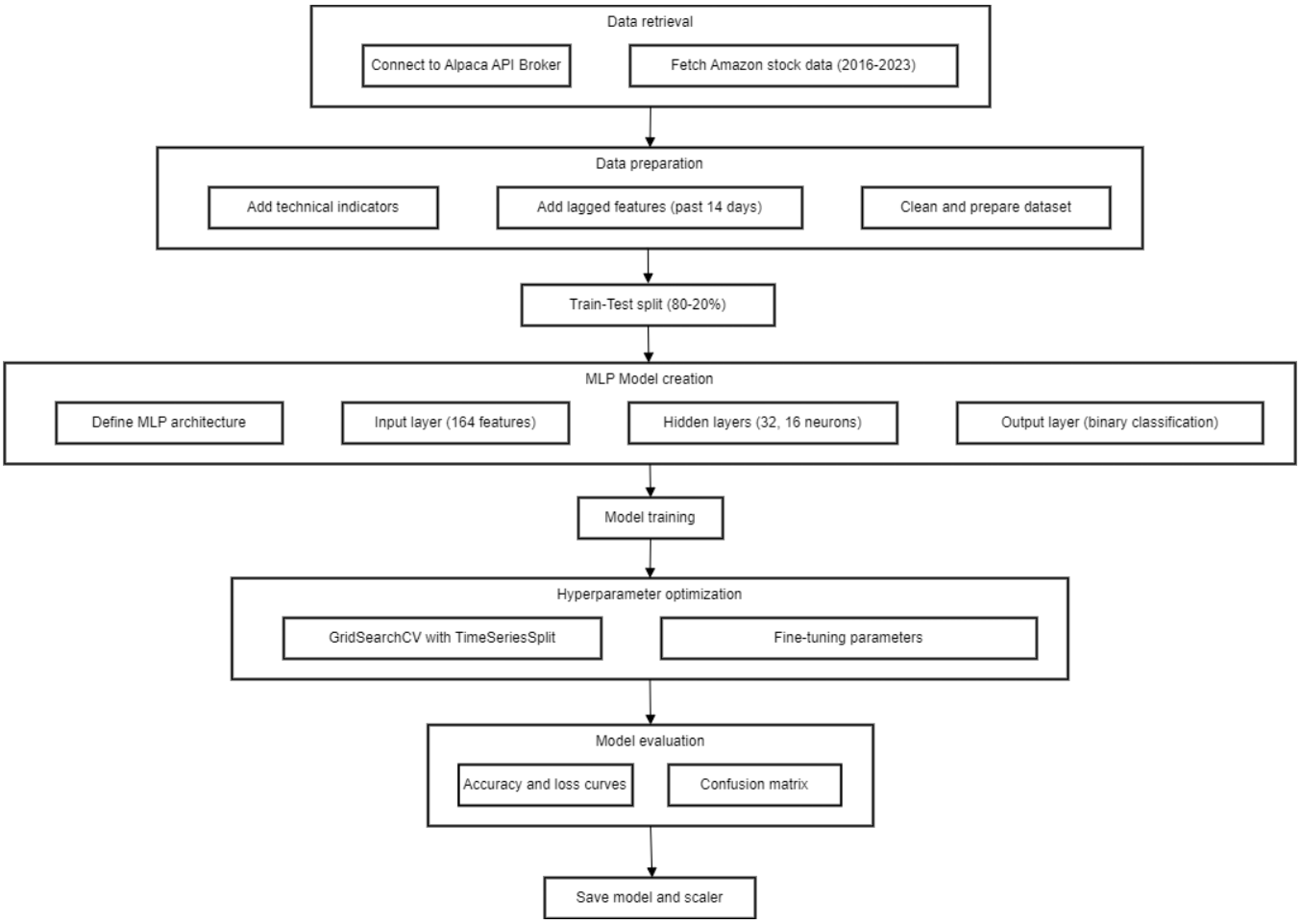
Regarding the integration of neural networks, I intend to perform predictive analysis to forecast future price movement based on historical data such as the Open, High, Low and Close price (OHLC), the volume and four technical indicators. The type of neural network used is called Multilayer Perceptron (MLP).

A MLP is a type of neural network consisting of multiple layers of neurons: an input layer, one or more hidden layers and an output layer. It processes the input data through all the layers, applying an activation function to capture complex non-linear patterns in the data. In this case, the MLP will perform a binary classification to predict whether the stock price will rise or fall. The output will have two possible values: 1, indicating a price rise, or 0 indicating a price fall (Jalswal 2024).

Although more complex models like Long Short-Term Memory (LSTM) are often used for time-series data, the MLP serves as a robust starting point for machine learning in trading. It is particularly effective in combining multiple data inputs, including the technical indicators we discussed in the previous chapter, to make more informed trading decisions.

A high-level overview of the methodology used in this project can be seen in the flowchart below. It summarizes the key steps, from data retrieval to saving the final model.

Figure 9 Flowchart: Methodology



5.1 Machine Learning implementation

The code is written in Python using a Jupyter Notebook. This environment offers greater flexibility to run specific sections of the code without having to rerun the entire program after each modification. This increases efficiency and reduces the consumption of the computer resources during the project development.

This project is structured into three main stages:

- Data preparation
- Model creation
- Evaluation

5.1.1 Data preparation

The historical stock price data is retrieved directly from the Alpaca broker API. For this project, we have chosen to use data from the company Amazon. This choice is not random, as Amazon is one of the largest market capitalizations in the world with

continuous growth. It is a pioneer in e-commerce, cloud computing through Amazon Web Services (AWS) and streaming services with Prime Video. This factors have made it popular among investors (Plumier 2024).

Technically, the data retrieval is done through the function get_historical_data(). This function takes as parameters, the stock ticker, as well as the start and end dates for the desired period. The code then executes a GET request to the Alpaca Market API using the StockBarsRequest() method provided by the broker's library. This request returns a table with the following information: timestamps, symbol, open, high, low, close, volume, trade_count and vwap.

As mentioned earlier, the prediction model requires technical indicators to support its decision-making. This is why a second function named add_technical_indicators() was created to enrich the previously retrieved data. This function adds additional columns to the initial table by calculating various technical indicators, including EMA, RSI, MACD, MACD_Signal, MACD_Hist and VWAP. These are calculated using the Pandas_ta library.

Once these indicators have been added, the data is cleaned of any missing or null values to ensure a high-quality dataset.

To visualize this, the last function returns the table below:

Table 1 Historical Data: Amazon

1-10 1729 rows × 14 columns

timestamp	symbol	open	hi…	low	close	volume	trade_count	vwap	EMA	RSI	MACD	MACD_Signal	MACD_Hist	VWAP
2016-02-22 …	AMZN	542.20	560.650	541.0800	559.50	5230007.0	54851.0	555.320915	541.339281	52.439651	-22.844101	-34.338385	11.494284	553.743333
2016-02-23 …	AMZN	555.55	556.910	545.3300	552.94	4169735.0	51619.0	552.254102	542.444112	50.554230	-19.496145	-31.369937	11.873792	551.726667
2016-02-24 …	AMZN	545.75	554.270	533.1501	554.04	6375305.0	69749.0	543.038047	543.548482	50.873193	-16.563175	-28.408585	11.845410	547.153367
2016-02-25 …	AMZN	555.52	559.390	545.2946	555.15	4638270.0	51282.0	551.804941	544.653388	51.215181	-13.987963	-25.524460	11.536497	553.278200
2016-02-26 …	AMZN	560.12	562.500	553.1700	555.23	4998390.0	51822.0	557.459482	545.660685	51.241526	-11.804558	-22.780480	10.975922	556.966667
2016-02-29 …	AMZN	554.00	564.810	552.5100	552.52	4464476.0	43141.0	557.174650	546.313953	50.251553	-10.175571	-20.259498	10.083927	556.613333
2016-03-01 …	AMZN	556.29	579.250	556.0000	579.04	5215861.0	61730.0	570.612024	549.430719	58.667158	-6.667782	-17.541155	10.873373	571.430000
2016-03-02 …	AMZN	581.75	585.000	573.7000	580.21	4708510.0	59260.0	579.046557	552.362079	58.996708	-3.750195	-14.782963	11.032768	579.636667
2016-03-03 …	AMZN	577.96	579.866	573.1100	577.49	2834087.0	36011.0	576.623840	554.755215	57.842091	-1.638579	-12.154086	10.515507	576.822000
2016-03-04 …	AMZN	581.07	581.400	571.0650	575.14	3556731.0	44022.0	576.511250	556.696623	56.807662	-0.152970	-9.753863	9.600893	575.868333

At this stage, there is only one final step to complete the preparation of the dataset Currently, each instance contains information for just a single day. It is clear that the model will struggle to effectively predict future price movements based only on the current day. This is why we have to enrich the data with a history of the previous 14 days. This allows the model to better understand past trends within a single instance.

The addition of extra historical data is done through the function add_lagged_features(), whose purposes is to generate new columns for each of the previous day. For Example,

we now have 14 versions of the column open (open_1, open_2, … open_14), as well as the other price columns and the technical indicators.

Once these steps have been completed, we call the prepare_data() function, which aims to organize and structure the dataset used to train the model.

This function is divided into three steps:

- Exclude unnecessary columns
- Define the target variable
- Split the data

5.1.1.1 Exclude unnecessary columns

Columns such as timestamp, ticker, trade_count and the vwap (which was provided directly from the broker's StockBarsRequest() method and not from the pandas_ta library) are excluded, because they are not required for predicting price movements.

5.1.1.2 Define the target variable

The aim of the model is to predict whether the closing price on the next day will be higher or lower than the closing price on the current day. Thus, the target variable (y) is a binary output. It takes the value of 1 if the price goes up or 0 if the price goes down.

5.1.1.3 Split de data

The prepare_data() function conclude with the split of the data into two subsets. The first subset is used for model training and validation and it represents 80% of the dataset. The second subset, which represents the remaining 20% of the dataset, is used as the test set. The model must be tested on data it has never seen during the training phase.

Finally, it is important to note that this split is performed without using the shuffle mode. This preserves the chronological order, which is crucial for time series data.

5.1.2 Machine Learning model creation

As mentioned previously, the model chosen to predict the future price movements of Amazon stock is a Multi-Layer Perceptron (MLP) artificial neural network. This MLP model is a common architecture used for binary classification problems (Brownlee 2022).

The process is structured into three main steps:

- Create the MLP model
- Train and evaluate the model
- Optimize the hyperparameters

5.1.2.1 Create the MLP model

The neural network is built using the TensorFlow library and its Keras API. The create_model() function allows us to generate a default model based on purely arbitrary parameters that will serve as a baseline before the optimization with GridSearchCV.

The default parameters are as follows:

Table 2 MLP model : Default parameters

Number of neurons	32
Learning rate	0.001
Dropout rates	0.3
L2 regularization	0.001
Number of epochs	100

The four main layers of the model are:

Input layer: It is sized according to the number of features in the dataset. For this project, we have 164 columns representing price history (Open, High, Low and Close), volume, the four technical indicators, as well as the entire historical data for the previous 14 days.

Hidden layer: There are two hidden layers, the first one is made up of 32 neurons, while the second contains 16. Each layer is followed by a ReLu (Rectified Linear Unit) activation function, which is one of the most popular in deep learning (Krishnamurthy 2024).

The ReLu function's goal is to filter the output values of each hidden layer by transforming negative values into zero, but keeping positive values unchanged. This allows the model to better capture complex relationships in the data while ensuring efficient training by avoiding negative gradients, which could slow down the learning (Keldenich 2021a).

Dropout layers: Between each hidden layer, there is a layer called Dropout. In the baseline model, it is defined with a rate of 0.3. This means that 30% of neurons are randomly deactivated at each iteration. These layers reduce overfitting and help to better generalize the model to new data (Keldenich 2021b).

Output layer: Finally, there is the output layer, whose goal is to predict whether the price will rise or fall. This is a binary variable since there are only two possible outputs. This layer uses the sigmoïd activation function., which is well suited for our project because it returns a value between 0 and 1. An output value close to 1 indicates a high probability of a price increase, while a value close to 0 indicates a probability of a price decrease (Sharma 2022).

5.1.2.2 Train and evaluate the model

Once our initial MLP model has been created with the baseline parameters, we then need to train it on historical data. This step is important, because it allows the model's internal weights to be adjusted to improve its predictions.

The weights in a neural network are parameters that evaluate the influence of each neuron on the final output. With each pass through the network, the weights are updated based on the errors made by the model. This better captures the relationships between the input data and the expected outcomes (GeeksForGeeks 2023).

5.1.2.2.1 Split the dataset

To perform the training, we first need to split the data. In the Python code, this split is done using the train_test_split() function from the library scikit-learn. The data is divided into two parts:

- **Training data**: The data used to train the model represents 80% of the total dataset. It is used to enable the model to learn and adjust its parameters.
- **Test data**: The test part therefore represents 20% and has been never shown to the model during the training phase. This helps to evaluate how well the model generalizes.

5.1.2.2.2 Normalize data

In our dataset, we can see that the values between the different columns can have very different scales, for example the data between prices and volumes. To avoid biasing the learning process, these data are normalized, meaning all the data are put on the same scale. This is done using the StandardScaler class from scikit-learn.

5.1.2.2.3 Train the model

The model is trained using the Adam optimization algorithm, which is one of the most popular in machine learning. This optimizer adjusts the model weights by minimizing the loss function called binary_crossentropy. We use this loss function, because it corresponds perfectly to our binary classification problem of predicting whether the price will go up or down (Vishwakarma 2023) (Godoy 2022).

5.1.2.3 Optimize hyperparameters

To improve the model in order to get it more efficient, we need to readjust its parameters. This process is called fine-tuning. It can be done manually by adjusting parameter combinations, or we can use a Python library to automate the process.

5.1.2.3.1 GridSearchCV

The GridSearchCV tool assists in testing multiple parameter combinations in order to identify the most effective ones to optimize and get the best performing model. This is done by creating a param_grid variable, which is a type of dictionary containing all possible combinations.

Table 3 GridSearchCV: Hyperparameter combinations

Learning rate	0.001	0.01
Number of neurons	32	64
Dropout rate	0.3	0.4
L2 regularization	0.001	0.01
Batch size	32	64
Number of epochs	50	100

5.1.2.3.2 Cross-validation

Cross-validation is a crucial method for evaluating model performance while trying to avoid overfitting. It involves splitting the dataset, in this case, into 4 folds. The model is then trained on some of these subsets and evaluated on the others.

For our model, we use cross-validation with the TimeSeriesSplit library, which is specifically designed to handle time-series data. The advantage of this type of cross-validation is that it takes the chronological order into account, ensuring that the data used for training is always located before the data used for validation. This contrasts with other traditional cross-validation methods, which randomly shuffle the dataset (L 2023).

By combining GridSearchCV with TimeSeriesSplit, we are trying to optimize the hyperparameters as much as possible. After performing this combination, the best parameters are :

Table 4 GridSearchCV: Best parameters

Number of neurons	32
Learning rate	0.01
Dropout rates	0.3
L2 regularization	0.01
Number of epochs	50
Batch size	32

5.1.3 Evaluation of results

After training and optimizing the model on the historical data of Amazon stock from 2016 to 2023, the best validation accuracy achieved was 0.5350. While this is not a remarkable result, it is slightly better than random guessing. The model was retrained using the best parameters found, with an additional of early stopping technique to further prevent overfitting and ensure the model does not simply memorize the training data. Although GridSearchCV suggested 50 epochs, the early stopping tool halted the training after 10 epochs as no further improvement in validation accuracy was observed (Bhattbhatt 2024).

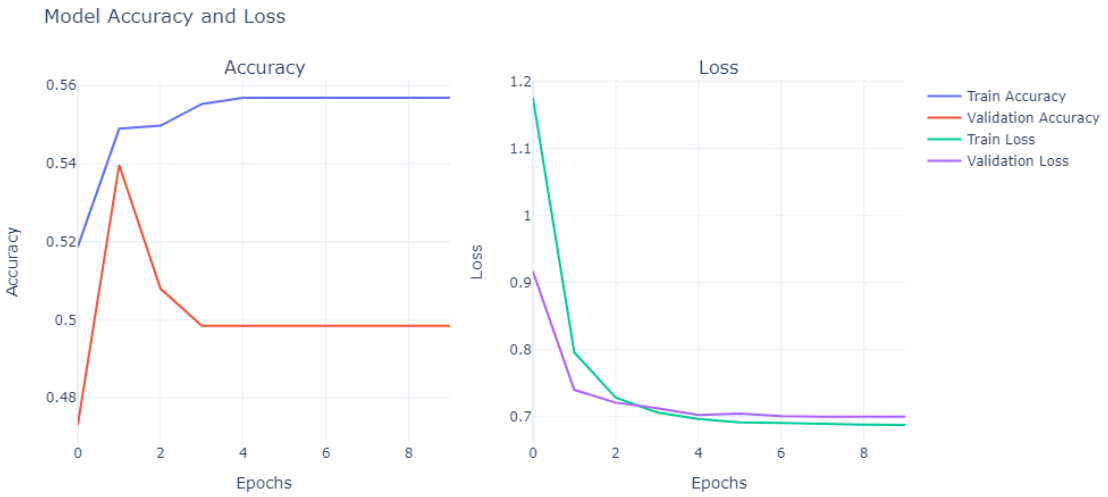
The performance of the model can be visualized through the accuracy and loss curves, as well as the confusion matrix.

- Accuracy and loss curves

  Figure 9 represents two graphs, the first on the left shows the evolution of accuracy during both training and validation phases. The blue line represents the accuracy on the training data, which regularly increases and reaches almost 0.56. However, the validation accuracy shown by the orange line shows a high at approximately 0.54 and then stabilizes around 0.50. This indicates that the model quickly reaches its best performance on new data, which might suggest some overfitting since the validation accuracy stops improving after 3 epochs.

On the right, the loss curves provide additional information into the model's training process. The training loss, represented by the green line, decreases rapidly during first epochs, which indicates that it is learning effectively on the train set. However, the validation loss represented by the purple line flattens after a few epochs, which shows that the model stops improving on new data after a certain point.

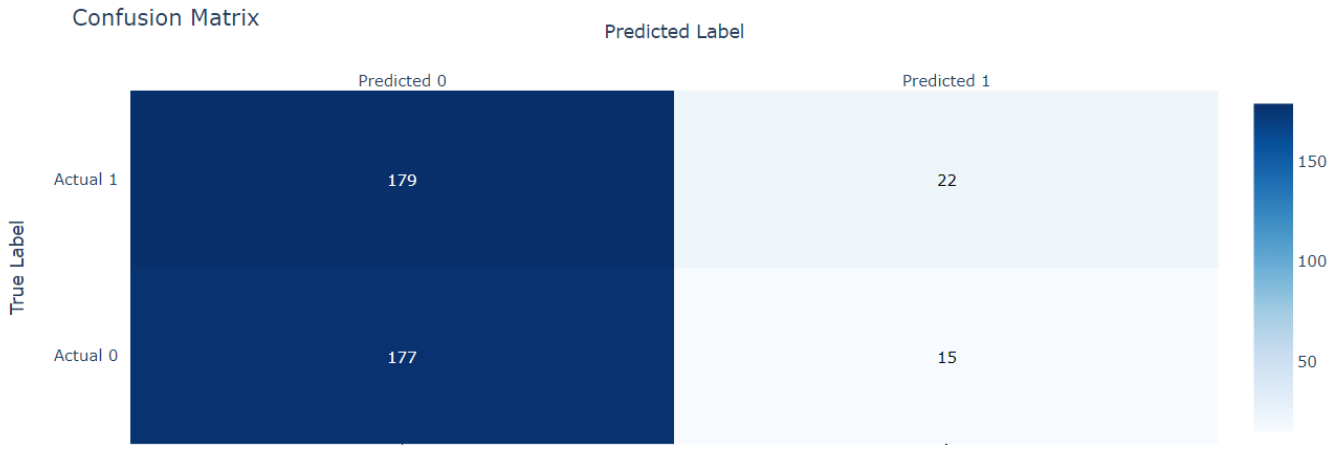
Figure 10 Best model: Accuracy and loss curves



- Confusion matrix

  The confusion matrix provides a detailed view at the model's predictions. It shows that the model performs better at identifying price movements going up, correctly predicting 179 out of 201 actual upwards movements (True positives). However, it struggles with downward predictions, as seen in the high number of false negatives – 177 out of 192 cases where the model predicted an increase when the price actually decreased.

Figure 11 Best model: Confusion matrix



This imbalance between the correct predictions of upward versus downward movements highlights a key limitation of the model. Although it predicts upward trend fairly well, its difficulty in predicting downward movements accurately could result in poor trading decisions.

6. Implementation of baseline and MLP strategies

As we move from the methodology outlined in chapter 5 to the practical application of our trading strategies, it is essential to understand the foundational approaches and their implementation. In this chapter, we will explore three baseline trading strategies that I created which will serve as benchmarks for evaluating the performance of our Multi-Layer Perceptron (MLP) model in predicting stock prices.

Each strategy, including the MLP model, will start with an initial capital of $10,000. Although it is generally inadvisable to commit the entire capital to a single trade without diversifying, for simplicity, each trade (whether opening or closing a position) will use the full available capital. Please note that they will only take long positions (buying) and will not open in any short positions. This means that the profits are generated only when stock prices rise.

Additionally, to ensure a realistic simulation, we have included considerations for slippage and commission fees:

- Slippage: This refers to the difference between the expected price and the actual price at which a trade is executed. Slippage often occurs in volatile market conditions or with low liquidity, where prices fluctuate rapidly. For this project, slippage is set at 1%, meaning we will buy 1% above the market price and sell 1% below the market price (Hayes 2024).

- Commissions: To reflect real-world trading conditions, a commission of $5 per trade (buy or sell) is applied. These fees are factored into the net returns of each strategy.

6.1 Strategy: buy and hold

The buy and hold strategy involves purchasing stocks at the beginning of the simulation period and holding them until the end. No additional transactions are made during this period. This passive strategy incurs only a 5 USD fee at the initial purchase, with no further trading costs.

6.2 Strategy: buy and sell randomly

This strategy simulates a trading approach where buy and sell decisions are made purely at random. Additionally, a parameter defines how frequently the trading bot operates throughout the year. For this simulation, the bot trades on 50% of the trading days. The goal is to evaluate whether a strategy with no decision-making basis can still generate profits.

6.3 Strategy: based on technical indicators

This strategy employs a straightforward application of technical indicators, similar to those used by the MLP model. It relies on the RSI, MACD, EMA, and VWAP indicators to determine when to open and close positions.

A long position is opened when signals indicate a bullish market, such as when the RSI is below 30 (indicating an oversold market), the MACD is above its signal line, or the stock price exceeds the EMA or VWAP, with a general condition that the market is in an upward trend.

Conversely, positions are closed upon detecting bearish signals, such as when the RSI exceeds 70 (indicating overbought market), the MACD falls below its signal line, or the stock price drops below the EMA or VWAP. If any of these conditions are met and the number of shares available is greater than 1, the position is liquidated.

6.4 Strategy: based on the MLP model

The final strategy utilizes a Multi-Layer Perceptron (MLP) model, a machine learning algorithm designed to predict stock price movements. Each day, the model forecasts whether the stock price will rise or fall the following day, and these predictions guide buying or selling decisions. When the model predicts a price increase, a long position is opened. Conversely, if the model predicts a price decrease, the shares available are sold, and the position is closed.

To complement the model's predictions, a risk management system is implemented with take profit and stop loss mechanisms. After opening a position, if the stock price rises and reaches a predefined profit threshold of 5%, the position is automatically closed to secure gains. Similarly, if the model's prediction is incorrect and the market moves against the position, the stop loss is triggered (total position liquidation) upon reaching a predefined loss threshold of 2% to limit potential losses.

The following chapter will focus on the analysis of the results obtained and the interpretation of the performances.

7. Analysis of results

7.1 Performance review

In this chapter, we will conduct an in-depth evaluation of the performance of the different strategies implemented during the simulated period from 1st January 2024 to 31 August 2024. The objective is to assess whether our MLP model proves to be effective in generating profits compared to others baseline strategies.

These strategies are evaluated based on the following performance indicators:

- Gross return: The total portfolio value before commissions are deducted.
- Net return: The return after deducting the commissions.
- Hit ratio: This measures the percentage of winning trades, calculated from the total number of trades executed.
- Risk-Reward ratio: This ratio compares the average profit from winning trades to the average loss from losing trades (Skorokhod 2024).
- Profit factor: The last metric that compares the total amount of gains realized to the total losses incurred (Skorokhod 2024).

Here are the results obtained for each strategy:

Table 5 Performance: Results strategies

Strategies	Gross return	Fees	Net return	Hit ratio	Risk-reward ratio	Profit factor
Buy and Hold	$ 10,380.32	$ 5.00	$ 10,375.82	53.33%	0.93	1.08
Random Buy/Sell	$ 7,656.12	$ 140.00	$ 7,516.12	6.67%	1.69	0.39
Technical Indicators	$ 9,087.92	$ 75.00	$ 9,012.92	36.97%	0.77	0.84
Neural Network	**$ 8,940.77**	**$ 80.00**	**$ 8,860.77**	**4.17%**	**0.64**	**0.23**

To summarize the results in the table, the strategy buy and hold stands out as the most successful, generating the highest net return of USD 10.375.82 and a hit ratio of 53.33%

despite having a relatively low risk-reward ratio (0.93) and a modest profit factor (1.08). This can confirm that a passive approach can still yield strong returns.

The buy and sell strategy randomly performed the worst, with a low net return and profit factor, highlighting its inconsistency despite a relatively high risk-reward ratio.

The technical indicators strategy showed more consistency than the random approach, but its modest risk-reward ratio and profit factor indicate it struggled to outperform the market.

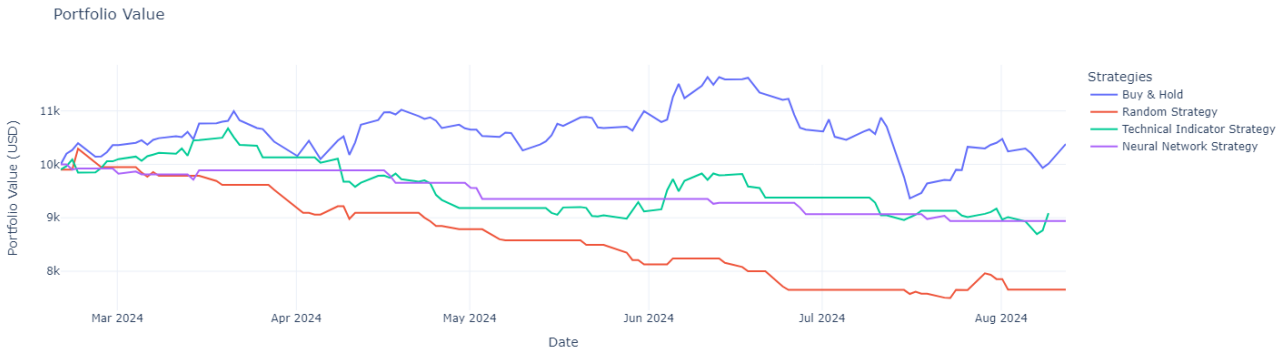
Finally, our neural network strategy showed mixed results, with a net return of USD 8,860.77 and a low hit ratio (4.17%). Its risk-reward ratio (0.64) and profit factor (0.23) indicate that the model faced difficulties in consistently identifying profitable opportunities.

7.2 Interpretation of results

In this section, we interpret the results of the four strategies using the graphs for visualizations. These graphs are the comparison for the portfolio value for each strategy, the buy and sell points and the distribution of daily return, allowing for a deeper understanding of their performance.

7.2.1 Portfolio value evolution

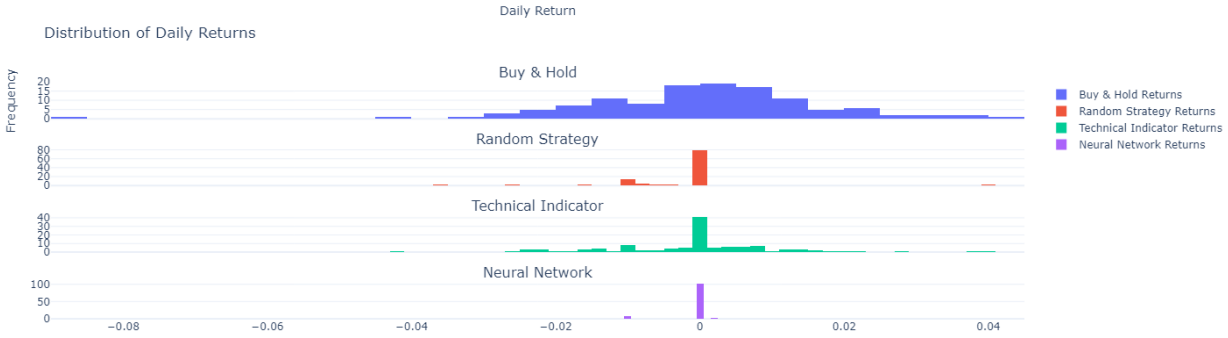
Figure 12 Results: Portfolio value



In the figure above, it is evident that the Buy and hold strategy performed the best over the simulation period, demonstrating stead and growth. In contrast, the random buy and sell strategy experienced the most significant and rapid decline in portfolio value. The neural network strategy, represented by the purple line, shows a gradual and consistent decrease throughout the period, likely due to its suboptimal predictions that struggled to identify profitable opportunities. However, it displayed more stability compared to the strategy based solely on technical indicators, which have shown greater volatility.

7.2.2 Distribution of daily returns

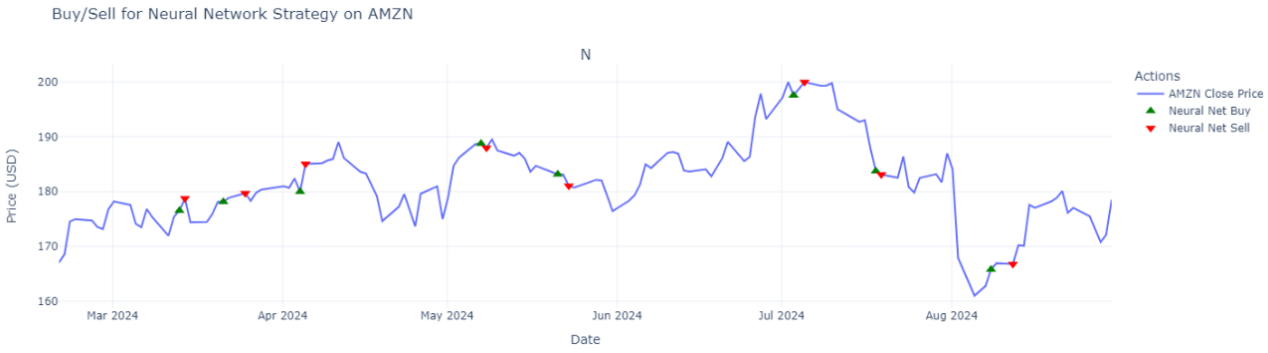
Figure 13 Results: Distribution of daily returns



In the distribution of daily returns, the histogram shows how much the returns vary for each strategy. The neural network strategy has a wider range of returns, with some losses, meaning its more volatile. This suggests that while the model occasionally captures good opportunities, it also faces significant downside risks, which explains the lower overall returns. In contrast, the buy and hold strategy has a much more consistent return distribution, reinforcing its stable performance.

7.2.3 Buy and sell points for the Neural network strategy

Figure 14 Results: Buy and sell points Neural network strategy



This chart shows that the model executed trades regularly throughout the simulation period. The model tended to take profit during short-term upward movements and closed positions when it predicted a price decline. However, the strong drawdown in June and August suggest that the model failed to anticipate the major downward trend, which likely contributed to its overall underperformance. Despite some poor predictions, the model did manage to close positions at profitable moments in certain cases. It shows that there is significant room for improvement.

8. Prospective developments

Currently, the bot uses a neural network model based solely on technical data, such as historical price, volume and four technical indicators discussed in the previous chapters. One of the primary improvements would be to incorporate a fundamental analysis approach that would complement the existing technical analysis. This could involve analyzing the financial health of the company by examining balance sheets, price-to-earnings (P/E) ratios, earnings per share (EPS) and financial reports. (Segal 2024). Adding this additional data could provide a better view of the company's value and might increase the accuracy of the predictions.

Another enhancement would be the integration of sentiment analysis. This would involve analyzing social media, financial news and some trading forums like reddit to capture the overall sentiment of the market. By evaluating the current mood of investors, the model could make more informed trading decisions (Equiti 2023).

Finally, a major improvement would be to explore more advanced Machine Learning models beyond the MLP, which has served as a solid introduction to the field of Machine Learning for me. The next step would be to implement a Recurrent Neural Network (RNN), specifically a Long Short-Term Memory (LSTM) model, which is better suited for forecasting stock prices due to its ability to retain temporal relationships over longer sequences (Chauhan 2023).

9. Conclusion

To conclude my Bachelor's project on the creation of an automated trading bot, I would like to share the feeling that accompanied me throughout this journey. First of all, it is important to recognize that this field, where the main target is to generate profits consistently, represents a major challenge that not everyone can easily overcome. It is quite clear that if it were easy, everyone with a minimum of motivation and a computer would already be a millionaire today.

This short experience not only gave me a taste of the challenges and problems we can face in the algorithm trading, but it also triggered a strong interest in the field of Machine Learning and the many projects and issues that this discipline can solve. Throughout this project, the code was tested and revised multiple times to address these challenges. Although this process required several iterations, it has been a valuable learning experience and represents a solid foundation for future improvement.

Indeed, I have realized that the integration of both technical and fundamental analysis, as well as the addition of sentiment analysis, could greatly enhance the accuracy and performance of the trading model. In order to develop an effective trading bot, it involves to take into account numerous factors and different aspects, which makes it far from easy but really fascinating to explore.

Despite the modest results of my trading bot, I remain motivated and plan to deepen my knowledge in Machine Learning and Deep Learning in order to improve the model I created. I look forward to pursuing a career in the future in this rapidly growing field, which is highly valued by companies.

Bibliographie

ALPACA, [sans date]. About Alpaca - Documentation | Alpaca. [en ligne]. [sans date]. Disponible à l'adresse : https://alpaca.markets/deprecated/docs/about-us/ [consulté le 13 août 2024].

BHATTBHATT, Vrunda, 2024. A Step-by-Step Guide to Early Stopping in TensorFlow and PyTorch. Medium [en ligne]. 24 janvier 2024. Disponible à l'adresse : https://medium.com/@vrunda.bhattbhatt/a-step-by-step-guide-to-early-stopping-in-tensorflow-and-pytorch-59c1e3d0e376 [consulté le 19 septembre 2024].

BOTS, Quantum Bots-Cutting Edge Trading, 2024. The Future of Finance: AI and Machine Learning in Trading. Medium [en ligne]. 11 mai 2024. Disponible à l'adresse : https://medium.com/@quantumbots/the-future-of-finance-ai-and-machine-learning-in-trading-38254930b251 [consulté le 18 septembre 2024].

BROWNLEE, Jason, 2022. When to Use MLP, CNN, and RNN Neural Networks. MachineLearningMastery.com [en ligne]. 11 août 2022. Disponible à l'adresse : https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/ [consulté le 5 septembre 2024].

CAPITAL.COM, [sans date]. EMA Trading Strategy: How to Use EMA in Trading. [en ligne]. [sans date]. Disponible à l'adresse : https://capital.com/exponential-moving-average [consulté le 13 août 2024].

CAPITAL.COM, 2017. What is a commodity? [en ligne]. 24 août 2017. Disponible à l'adresse : https://www.youtube.com/watch?v=aF0CH1OxivA [consulté le 12 mai 2024].

CFI TEAM, [sans date]. Volume Weighted Adjusted Price (VWAP). Corporate Finance Institute [en ligne]. [sans date]. Disponible à l'adresse : https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/volume-weighted-adjusted-price-vwap/ [consulté le 13 août 2024].

CHAUHAN, Prajjwal, 2023. Stock Prediction and Forecasting Using LSTM(Long-Short-Term-Memory). Medium [en ligne]. 8 juillet 2023. Disponible à l'adresse : https://medium.com/@prajjwalchauhan94017/stock-prediction-and-forecasting-using-lstm-long-short-term-memory-9ff56625de73 [consulté le 18 septembre 2024].

CHEN, James, 2021. Oscillator: What They Are and How They Work. Investopedia [en ligne]. 4 mars 2021. Disponible à l'adresse : https://www.investopedia.com/terms/o/oscillator.asp [consulté le 12 mai 2024].

CHEN, James, 2024. How Do You Start Forex Trading? A Beginner's Guide. Investopedia [en ligne]. 1 avril 2024. Disponible à l'adresse : https://www.investopedia.com/articles/forex/11/why-trade-forex.asp [consulté le 12 mai 2024].

DOLAN, Brian, 2024. What Is MACD? Investopedia [en ligne]. 16 septembre 2024. Disponible à l'adresse : https://www.investopedia.com/terms/m/macd.asp [consulté le 19 septembre 2024].

EHRLICH, Steven, 2023. Crypto Exchanges: What Investors Need To Know. Forbes [en ligne]. 8 mai 2023. Disponible à l'adresse : https://www.forbes.com/sites/digital-assets/article/crypto-exchanges-what-investors-need-to-know/ [consulté le 12 mai 2024].

EQUITI, 2023. Introduction to Sentiment Analysis. Equiti Default [en ligne]. 5 septembre 2023. Disponible à l'adresse : https://www.equiti.com/sc-en/education/market-analysis/sentiment-analysis-101/ [consulté le 18 septembre 2024].

FERNANDO, Jason, 2024a. What Are Commodities and Understanding Their Role in the Stock Market. Investopedia [en ligne]. 23 janvier 2024. Disponible à l'adresse : https://www.investopedia.com/terms/c/commodity.asp [consulté le 12 mai 2024].

FERNANDO, Jason, 2024b. Relative Strength Index (RSI) Indicator Explained With Formula. Investopedia [en ligne]. 10 avril 2024. Disponible à l'adresse : https://www.investopedia.com/terms/r/rsi.asp [consulté le 12 mai 2024].

FERNANDO, Jason, 2024c. Volume-Weighted Average Price (VWAP): Definition and Calculation. Investopedia [en ligne]. 12 juillet 2024. Disponible à l'adresse : https://www.investopedia.com/terms/v/vwap.asp [consulté le 13 août 2024].

FINMA, Eidgenössische Finanzmarktaufsicht, [sans date]. La FINMA, une autorité de surveillance indépendante. Eidgenössische Finanzmarktaufsicht FINMA [en ligne]. [sans date]. Disponible à l'adresse : https://www.finma.ch/fr/finma/tout-sur-la-finma/ [consulté le 12 mai 2024].

GEEKSFORGEEKS, 2023. Weights and Bias in Neural Networks. GeeksforGeeks [en ligne]. 11 octobre 2023. Disponible à l'adresse : https://www.geeksforgeeks.org/the-role-of-weights-and-bias-in-neural-networks/ [consulté le 13 septembre 2024].

GODOY, Daniel, 2022. Understanding binary cross-entropy / log loss: a visual explanation. Medium [en ligne]. 10 juillet 2022. Disponible à l'adresse : https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a [consulté le 13 septembre 2024].

GRATTON, Peter, 2024. What Is the Stock Market and How Does it Work? Investopedia [en ligne]. 12 avril 2024. Disponible à l'adresse : https://www.investopedia.com/terms/s/stockmarket.asp [consulté le 12 mai 2024].

GROUPE IG, 2019. Le trading à l'aide des bandes de Bollinger. IG [en ligne]. 24 janvier 2019. Disponible à l'adresse : https://www.ig.com/fr-ch/strategies-de-trading/le-trading--a-l_aide-des-bandes-de-bollinger-190124 [consulté le 12 mai 2024].

HAYES, Adam, 2023. Stochastic Oscillator: What It Is, How It Works, How To Calculate. Investopedia [en ligne]. 30 septembre 2023. Disponible à l'adresse : https://www.investopedia.com/terms/s/stochasticoscillator.asp [consulté le 13 août 2024].

HOLDSWORTH, Jim, 2024. What Is Deep Learning? | IBM. [en ligne]. 17 juin 2024. Disponible à l'adresse : https://www.ibm.com/topics/deep-learning [consulté le 18 septembre 2024].

IG, 2022. How can AI and ML drive hedge funds? IG [en ligne]. 24 novembre 2022. Disponible à l'adresse : https://www.ig.com/en-ch/prime/insights/articles/artificial-intelligence-machine-learning-hedge-funds-220901 [consulté le 18 septembre 2024].

JALSWAL, Sejal, 2024. Multilayer Perceptrons in Machine Learning: A Comprehensive Guide. [en ligne]. 28 juillet 2024. Disponible à l'adresse : https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning [consulté le 19 septembre 2024].

KARLSSON, Kasper, 2024. Renaissance Technologies and The Medallion Fund. [en ligne]. 21 mars 2024. Disponible à l'adresse : https://quartr.com/insights/company-research/renaissance-technologies-and-the-medallion-fund [consulté le 18 septembre 2024].

KELDENICH, Tom, 2021a. Fonction d'Activation: Comment elle fonctionne ? - Meilleur Guide. [en ligne]. 8 février 2021. Disponible à l'adresse : https://inside-machinelearning.com/fonction-dactivation-comment-ca-marche-une-explication-simple/ [consulté le 13 septembre 2024].

KELDENICH, Tom, 2021b. Le Dropout c'est quoi ? Deep Learning Explication Rapide. [en ligne]. 11 juillet 2021. Disponible à l'adresse : https://inside-machinelearning.com/le-dropout-cest-quoi-deep-learning-explication-rapide/ [consulté le 13 septembre 2024].

KENTON, Will, 2023. Barter (or Bartering) Definition, Uses, and Example. Investopedia [en ligne]. 29 septembre 2023. Disponible à l'adresse : https://www.investopedia.com/terms/b/barter.asp [consulté le 12 mai 2024].

KRAMER, Leslie, 2024. Long Position vs. Short Position: What's the Difference? Investopedia [en ligne]. 16 février 2024. Disponible à l'adresse : https://www.investopedia.com/ask/answers/100314/whats-difference-between-long-and-short-position-market.asp [consulté le 12 mai 2024].

KRISHNAMURTHY, Bharath, 2024. ReLU Activation Function Explained. Built In [en ligne]. 26 février 2024. Disponible à l'adresse : https://builtin.com/machine-learning/relu-activation-function [consulté le 5 septembre 2024].

KUEPPER, Justin, 2023. Risk Management Techniques for Active Traders. Investopedia [en ligne]. 12 juin 2023. Disponible à l'adresse : https://www.investopedia.com/articles/trading/09/risk-management.asp [consulté le 12 mai 2024].

L, Gen David, 2023. Five Methods for Data Splitting in Machine Learning. Medium [en ligne]. 2 décembre 2023. Disponible à l'adresse : https://medium.com/@tubelwj/five-methods-for-data-splitting-in-machine-learning-27baa50908ed [consulté le 19 septembre 2024].

LBORRMANN, 2022. Qu'est-ce que la psychologie du trader ? Production lynxbroker.fr [en ligne]. 3 mai 2022. Disponible à l'adresse : https://www.lynxbroker.fr/bourse/trading/la-psychologie-du-trader/la-psychologie-du-trader/ [consulté le 12 mai 2024].

MAKARENKO, Vitaly, 2024. Quadcode - What is Risk Management in Trading, and How Does It Work? Quadcode [en ligne]. 1 mai 2024. Disponible à l'adresse : https://quadcode.com/blog/what-is-risk-management-in-trading-and-how-does-it-work [consulté le 19 septembre 2024].

MANNING, Liz, 2023. Financial Industry Regulatory Authority (FINRA) Definition. Investopedia [en ligne]. 4 août 2023. Disponible à l'adresse : https://www.investopedia.com/terms/f/finra.asp [consulté le 13 août 2024].

MARTIN, Vincent, 2024. Volume Weighted Average Price (VWAP) Definition. Investing.com [en ligne]. 16 mai 2024. Disponible à l'adresse : https://www.investing.com/academy/analysis/vwap-formula/ [consulté le 13 août 2024].

MITCHELL, Cory, 2024a. Trend Trading: The 4 Most Common Indicators. Investopedia [en ligne]. 29 février 2024. Disponible à l'adresse : https://www.investopedia.com/articles/active-trading/041814/four-most-commonlyused-indicators-trend-trading.asp [consulté le 12 mai 2024].

MITCHELL, Cory, 2024b. How to Use Stock Volume to Improve Your Trading. Investopedia [en ligne]. 23 février 2024. Disponible à l'adresse : https://www.investopedia.com/articles/technical/02/010702.asp [consulté le 12 mai 2024].

MURPHY, Chris B., 2024. Over-the-Counter (OTC) Markets: Trading and Securities. Investopedia [en ligne]. 5 mars 2024. Disponible à l'adresse : https://www.investopedia.com/terms/o/otc.asp [consulté le 12 mai 2024].

NASLI, Adam, 2024. Alpaca Trading Review 2024 - Pros & Cons. [en ligne]. 12 août 2024. Disponible à l'adresse : https://brokerchooser.com/broker-reviews/alpaca-trading-review [consulté le 13 août 2024].

ORACLE, [sans date]. Intelligence Artificielle, Machine Learning, Deep Learning. Oracle France [en ligne]. [sans date]. Disponible à l'adresse : https://www.oracle.com/fr/database/deep-learning-intelligence-artificielle/ [consulté le 18 septembre 2024].

PLUMIER, Isabelle, 2024. Action Amazon ▷ Dernières actualités | Cours, analyses & graphiques. Production lynxbroker.fr [en ligne]. 15 août 2024. Disponible à l'adresse : https://www.lynxbroker.fr/bourse/cours-bourse/actions/action-amazon/ [consulté le 12 septembre 2024].

POSTFINANCE, 2023. Déclarer des actions en Suisse. PostFinance [en ligne]. 10 mars 2023. Disponible à l'adresse : https://www.postfinance.ch/fr/blog/placer-de-largent-mode-demploi/declarer-les-actions-en-suisse.html [consulté le 12 mai 2024].

SCHLOSSBERG, Boris, 2024. How to Trade the MACD. Investopedia [en ligne]. 27 juillet 2024. Disponible à l'adresse : https://www.investopedia.com/articles/forex/05/macddiverge.asp [consulté le 13 août 2024].

SEGAL, Troy, 2024. Fundamental Analysis: Principles, Types, and How to Use It. Investopedia [en ligne]. 21 juin 2024. Disponible à l'adresse : https://www.investopedia.com/terms/f/fundamentalanalysis.asp [consulté le 18 septembre 2024].

SHARMA, Sagar, 2022. Activation Functions in Neural Networks. Medium [en ligne]. 20 novembre 2022. Disponible à l'adresse : https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 [consulté le 13 septembre 2024].

SKOROKHOD, Yuliia, 2024. Métriques Clés pour évaluer le Trading. JustMarkets [en ligne]. 18 juillet 2024. Disponible à l'adresse : https://justmarkets.pro/fr/trading-articles/learning/key-metrics-to-evaluate-trading [consulté le 19 septembre 2024].

STILT et GOGOL, Frank, 2021. Study: 94% of Crypto Buyers are Gen Z/Millennial, but Gen X is Outspending Them. Stilt [en ligne]. 16 mars 2021. Disponible à l'adresse : https://www.stilt.com/data/vast-majority-crypto-buyers-millennials-gen-z/ [consulté le 12 mai 2024].

STRIKE, 2023. What is Exponential Moving Average (EMA)? Definition, Formula. Strike [en ligne]. 26 septembre 2023. Disponible à l'adresse : https://www.strike.money/technical-analysis/ema [consulté le 13 août 2024].

TRADECIETY, 2023. How To Use The Stochastic Indicator Step By Step. [en ligne]. 17 janvier 2023. Disponible à l'adresse : https://tradeciety.com/how-to-use-the-stochastic-indicator [consulté le 13 août 2024].

VISHWAKARMA, Neha, 2023. What is Adam Optimizer? Analytics Vidhya [en ligne]. 29 septembre 2023. Disponible à l'adresse : https://www.analyticsvidhya.com/blog/2023/09/what-is-adam-optimizer/ [consulté le 13 septembre 2024].

WILSON, Hannah, 2024. Relative Strength Index (RSI): Calculation, Uses. Investing.com [en ligne]. 17 juin 2024. Disponible à l'adresse : https://www.investing.com/academy/analysis/relative-strength-index-definition/ [consulté le 13 août 2024].

Appendix 1: neural_network_model.ipynb

Note: To execute the code, a .env file is used where the API key and secret key for logging into the Alpaca broker account are stored. However, for security reasons, this file has not been shared.

```python
import os
from dotenv import load_dotenv
import pandas as pd
import pandas_ta as ta
import numpy as np
import random
import tensorflow as tf
from datetime import datetime
from alpaca.data.historical import StockHistoricalDataClient
from alpaca.data.requests import StockBarsRequest
from alpaca.data.timeframe import TimeFrame
from scikeras.wrappers import KerasClassifier
from sklearn.model_selection import GridSearchCV, train_test_split, TimeSeriesSplit
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score, roc_curve, confusion_matrix, ConfusionMatrixDisplay
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import EarlyStopping
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.figure_factory as ff
import matplotlib.pyplot as plt
```

[56]

```python
# Load environment variables from the file .env
load_dotenv()
# Get the API key and the secret from the .env
API_KEY = os.getenv("APCA_API_KEY_ID")
API_SECRET = os.getenv("APCA_API_SECRET_KEY")
# Connect to the broker Alpaca with the ID
data_client = StockHistoricalDataClient(API_KEY, API_SECRET)
```

[57]

```python
# Set seeds to 42 for reproducibility with the randoms parameters
SEED = 42
random.seed(SEED)
np.random.seed(SEED)
tf.random.set_seed(SEED)
```

[58]

```python
def get_historical_data(symbol, start_date, end_date):
    # Request the data from the broker
    request_params = StockBarsRequest(
        symbol_or_symbols=symbol,
        start=datetime.strptime(start_date, '%Y-%m-%d'),
        end=datetime.strptime(end_date, '%Y-%m-%d'),
        timeframe=TimeFrame.Day
    )
    # Fetch the data as a dataframe
    bars = data_client.get_stock_bars(request_params).df
    # Reindex the DF with only the timestamp column
    bars = bars.reset_index()
    bars['timestamp'] = pd.to_datetime(bars['timestamp'], errors='coerce')
    bars.set_index('timestamp', inplace=True)
    return bars
```

[59]

```python
# Add technical indicators
def add_technical_indicators(df):
    df["EMA"] = ta.ema(df["close"], length=20) # length = periods
    df["RSI"] = ta.rsi(df["close"], length=14)
    macd = ta.macd(df["close"], fast=12, slow=26, signal=9)
    df["MACD"] = macd["MACD_12_26_9"]
    df["MACD_Signal"] = macd["MACDs_12_26_9"]
    df["MACD_Hist"] = macd["MACDh_12_26_9"]
    df["VWAP"] = ta.vwap(df["high"], df["low"], df["close"], df["volume"])

    df.dropna(inplace=True) # Drop the rows with values as null
    return df
```

[60]

{} Code | M↓ Markdown

```python
# Retrieve historial data
bars = get_historical_data("AMZN", "2016-01-01", "2023-12-31")
bars.index = bars.index.tz_convert(None)  # Remove timezone information from the index
bars_with_indicators = add_technical_indicators(bars)
```

[61]

```python
# Add 14 columns of previous for the same instance (row in the df)
def add_lagged_features(df, num_days=14):
    lagged_data = {}
    for i in range(1, num_days + 1):
        for column in ["open", "high", "low", "close", "volume", "EMA", "RSI", "MACD", "MACD_Signal", "MACD_Hist", "VWAP"]:
            lagged_data[f"{column}_{i}"] = df[column].shift(i)

    # Create a new DataFrame with the lagged columns
    lagged_df = pd.DataFrame(lagged_data)

    # Concatenate the original DataFrame with the lagged DataFrame and ensuring no duplication
    df = pd.concat([df, lagged_df], axis=1)

    # Remove the rows that have no enough lagged data
    df = df.iloc[num_days:]  # Skip the first num_days rows
    return df

bars_with_lagged_features = add_lagged_features(bars_with_indicators)
def prepare_data(df):
    # Columns to exclude from the features
    excluded_columns = ["timestamp", "symbol", "trade_count", "vwap", "close"]

    feature_columns = [column for column in df.columns if column not in excluded_columns]

    # Input features (X) and target variable (y)
    x = df[feature_columns].values # Features columns without the target (close)

    # Target variable (y) : 1 if the next closing price is higher than the current day, otherwise 0
    y = (df["close"].shift(-1) > df["close"]).astype(int).values

    # Split data into training and test sets (80% train, 20% test) without shuffling
    x_train_val, x_test, y_train_val, y_test = train_test_split(x, y, test_size=0.2, shuffle=False)
    return x_train_val, x_test, y_train_val, y_test

x_train_val, x_test, y_train_val, y_test = prepare_data(bars_with_lagged_features)
```

```
# Constants for the default model
NUM_NEURONS = 32 # of each hidden layer
LEARNING_RATE = 0.001 # LR for the optimizer
DROPOUT_RATES = 0.3 # Dropout to avoid overfitting
L2_REGULARIZATION = 0.001 # L2 to reduce overfitting
NUM_EPOCHS = 100
```

```
# Normalize the data
scaler = StandardScaler()
x_train_val = scaler.fit_transform(x_train_val) #fit on the train data
x_test = scaler.transform(x_test) #Transform test data with the same scaler

# Generate model with dynamic parameters
def create_model(learning_rate=LEARNING_RATE, num_neurons=NUM_NEURONS, dropout_rate=DROPOUT_RATES, l2_reg=L2_REGULARIZATION):
    model = Sequential([
        Input(shape=(x_train_val.shape[1],)),
        Dense(num_neurons, activation='relu', kernel_regularizer=l2(l2_reg)),
        Dropout(dropout_rate),
        Dense(num_neurons // 2, activation='relu', kernel_regularizer=l2(l2_reg)),
        Dense(1, activation='sigmoid') # binary classification
    ])
    # Compile the model with Adam optimizer and binary crossentropy for the  loss function
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

```
# Early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
[66]
```

```
# Hyperparamater for the grid search
param_grid = {
    'model__learning_rate': [0.001, 0.01],
    'model__num_neurons': [32, 64],
    'model__dropout_rate': [0.3, 0.4],
    'model__l2_reg': [0.001, 0.01],
    'batch_size': [32],
    'epochs': [50, 100]
}
# GridSearch with TimeSeriesSplit
model = KerasClassifier(model=create_model, verbose=0)
tscv = TimeSeriesSplit(n_splits=4)
# Create a GridSearchCV object
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=tscv, scoring='accuracy', return_train_score=True, n_jobs=-1)
# Fit the model using GridSearch with the early stopping
grid_result = grid.fit(x_train_val, y_train_val, callbacks=[early_stopping])
#grid_result = grid.fit(x_train_val, y_train_val)

# Get the results for the best model
results = grid_result.cv_results_

print(f"Best parameters: {grid_result.best_params_}")
print(f"Best validation accuracy: {grid_result.best_score_:.4f}")
```

```
# Retrain the model with the best parameters
best_model = Sequential()
best_model.add(Input(shape=(x_train_val.shape[1],)))
best_model.add(Dense(grid_result.best_params_['model__num_neurons'], activation='relu', kernel_regularizer=l2(grid_result
 .best_params_['model__l2_reg'])))
best_model.add(Dropout(grid_result.best_params_['model__dropout_rate']))
best_model.add(Dense((grid_result.best_params_['model__num_neurons']) // 2, activation='relu', kernel_regularizer=l2
 (grid_result.best_params_['model__l2_reg'])))
best_model.add(Dense(1, activation='sigmoid'))

best_model.compile(optimizer=Adam(learning_rate=grid_result.best_params_['model__learning_rate']), loss='binary_crossentropy',
 metrics=['accuracy'])
history = best_model.fit(x_train_val, y_train_val, validation_split=0.2, epochs=grid_result.best_params_['epochs'],
 batch_size=grid_result.best_params_['batch_size'], callbacks=[early_stopping])
```

```python
def plot_accuracy_loss(history):
    fig = make_subplots(rows=1, cols=2, subplot_titles=("Accuracy", "Loss"))
    # Accuracy Plot
    fig.add_trace(go.Scatter(x=list(range(len(history.history['accuracy']))), y=history.history['accuracy'], mode='lines',
     name='Train Accuracy'), row=1, col=1)
    fig.add_trace(go.Scatter(x=list(range(len(history.history['val_accuracy']))), y=history.history['val_accuracy'],
     mode='lines', name='Validation Accuracy'), row=1, col=1)
    # Loss Plot
    fig.add_trace(go.Scatter(x=list(range(len(history.history['loss']))), y=history.history['loss'], mode='lines', name='Train
     Loss'), row=1, col=2)
    fig.add_trace(go.Scatter(x=list(range(len(history.history['val_loss']))), y=history.history['val_loss'], mode='lines',
     name='Validation Loss'), row=1, col=2)
    # Update Layout for x/y axis
    fig.update_xaxes(title_text="Epochs", row=1, col=1)
    fig.update_xaxes(title_text="Epochs", row=1, col=2)
    fig.update_yaxes(title_text="Accuracy", row=1, col=1)
    fig.update_yaxes(title_text="Loss", row=1, col=2)
    fig.update_layout(title_text="Model Accuracy and Loss", height=500, width=1000, template="plotly_white")
    fig.show()
# Show
plot_accuracy_loss(history)
```

```python
def plot_roc_auc(X_val, y_val, model):
    # Predict the probabilities for the validation data
    y_pred_proba = model.predict(X_val).ravel()
    # Compute the false positive rate (fpr) and true positive rate (tpr) for ROC curve
    fpr, tpr, _ = roc_curve(y_val, y_pred_proba)
    # Compute the area under the ROC curve (AUC)
    auc = roc_auc_score(y_val, y_pred_proba)
    #Generate figure
    fig = go.Figure()
    # Plot curve
    fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', name=f'ROC Curve (AUC = {auc:.4f})'))
    # Plot random classifier
    fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], mode='lines', name='Random Classifier', line=dict(dash='dash')))
    # Update Layout
    fig.update_layout(title="ROC-AUC Curve", xaxis_title="False Positive Rate", yaxis_title="True Positive Rate",
     template="plotly_white")
    fig.show()
#Show
plot_roc_auc(x_test, y_test, best_model)
```

```python
def plot_confusion_matrix(X_val, y_val, model):
    # Predict labels for the validation set with threshold at 0.5, because binary classification
    y_pred = (model.predict(X_val) > 0.5).astype(int)
    # Compute matrix
    cm = confusion_matrix(y_val, y_pred)
    # Confusion Matrix Heatmap
    fig = ff.create_annotated_heatmap(cm, x=['Predicted 0', 'Predicted 1'], y=['Actual 0', 'Actual 1'], colorscale='Blues',
     showscale=True)
    # Update Layout
    fig.update_layout(title_text="Confusion Matrix", xaxis_title="Predicted Label", yaxis_title="True Label",
     template="plotly_white")
    fig.show()

# Show
plot_confusion_matrix(x_test, y_test, best_model)
```

```python
# Save the model
best_model.save('my_neural_network_model.keras')
```
[130]

```python
# Save the scaler
import joblib
joblib.dump(scaler, 'scaler.pkl')
```

Appendix 2: strategies_evaluation.ipynb

```
import os
from dotenv import load_dotenv
import numpy as np
import pandas as pd
import pandas_ta as ta
from datetime import datetime
from alpaca.data.historical import StockHistoricalDataClient
from alpaca.data.requests import StockBarsRequest
from alpaca.data.timeframe import TimeFrame
import random
import plotly.graph_objs as go
from plotly.subplots import make_subplots
from tensorflow.keras.models import load_model
import joblib
[104]
```

```
# Load environment variables from the file .env
load_dotenv()
API_KEY = os.getenv("APCA_API_KEY_ID")
API_SECRET = os.getenv("APCA_API_SECRET_KEY")
data_client = StockHistoricalDataClient(API_KEY, API_SECRET)
# Constants
SEED = 42
COMMISSION = 5
SLIPPAGE = 0.01
RISK_FREE_RATE = 0.01 # For sharp ratio
[105]
```

```
def get_historical_data(symbol, start_date, end_date):
    request_params = StockBarsRequest(
        symbol_or_symbols=symbol,
        start=datetime.strptime(start_date, '%Y-%m-%d'),
        end=datetime.strptime(end_date, '%Y-%m-%d'),
        timeframe=TimeFrame.Day
    )
    bars = data_client.get_stock_bars(request_params).df
    bars = bars.reset_index()
    bars['timestamp'] = pd.to_datetime(bars['timestamp'], errors='coerce')
    bars.set_index('timestamp', inplace=True)
    return bars
[106]
```

```
def add_technical_indicators(df):
    df["EMA"] = ta.ema(df["close"], length=20)
    df["RSI"] = ta.rsi(df["close"], length=14)
    macd = ta.macd(df["close"], fast=12, slow=26, signal=9)
    df["MACD"] = macd["MACD_12_26_9"]
    df["MACD_Signal"] = macd["MACDs_12_26_9"]
    df["MACD_Hist"] = macd["MACDh_12_26_9"]
    df["VWAP"] = ta.vwap(df["high"], df["low"], df["close"], df["volume"])
    df.dropna(inplace=True) #pour delete les lignes avec des valeurs null
    return df
[107]
```

```
#Sharp ratio
def calculate_sharpe_ratio(daily_returns):
    excess_daily_returns = daily_returns - RISK_FREE_RATE / 252 # = 252 jours de trading
    return np.mean(excess_daily_returns) / np.std(excess_daily_returns)
```

```
def calculate_risk_reward_ratio(daily_returns):
    profits = daily_returns[daily_returns > 0]
    losses = daily_returns[daily_returns < 0]
    if len(losses) > 0:
        return np.mean(profits) / abs(np.mean(losses))
    else:
        return np.inf  # If no losses, return infinity
```
[109]

```
def calculate_profit_factor(daily_returns):
    profits = daily_returns[daily_returns > 0]
    losses = daily_returns[daily_returns < 0]
    if len(losses) > 0:
        return sum(profits) / abs(sum(losses))
    else:
        return np.inf  # If no losses, return infinity
```
[110]

```
def calculate_hit_ratio(daily_returns):
    positive_returns = daily_returns[daily_returns > 0]
    return len(positive_returns) / len(daily_returns)
```
[111]

```
bars = get_historical_data("AMZN", "2024-01-01", "2024-08-31")
bars.index = bars.index.tz_convert(None)
bars_with_indicators = add_technical_indicators(bars)
def strategy_buy_n_hold(df):
    cash_available = 10000 # Initial cash available
    nb_shares = 0
    total_commission = 0
    # 1st buy with the cash available with the initial price
    initial_price = df["close"].iloc[0]
    nb_shares = cash_available / initial_price
    cash_available = 0
    total_commission += COMMISSION # 5$ commission fee
    # Final value of the pfolio with last price
    final_price = df["close"].iloc[-1]
    final_pfolio_value = nb_shares * final_price
    total_gross_return = final_pfolio_value
    # Net return
    total_net_return = total_gross_return - total_commission
    #Daily pfolio for the graph
    daily_pfolio_value = [cash_available + nb_shares * df["close"].iloc[i] for i in range(len(df))]
    return total_gross_return, total_commission, total_net_return, daily_pfolio_value
```

```
def strategy_buy_n_sell_randomly(df, trade_percentage=50):
    random.seed(SEED)  # Ensure same results with the random
    cash_available = 10000
    nb_shares = 0
    total_commission = 0
    pfolio_value = 0
    daily_portfolio_value = []
    buy_dates = []
    sell_dates = []
    # Number of active trading days
    total_days = len(df)
    trade_days = int((trade_percentage / 100) * total_days) # Percentage of active trading days

    # Check if trade_days doesn't exceed total_days
    if trade_days > total_days:
        trade_days = total_days
    elif trade_days <= 0:
        raise ValueError("Percentage of active trade days should be positive")

    # Generate random trading within the range
    trading_days = sorted(random.sample(range(0, total_days), trade_days))

    last_action = None # To avoid 2 buys consecutives or selling imediately

    for index in range(0, len(df)):
        if index not in trading_days: # Skip non-trading days
            daily_portfolio_value.append(pfolio_value)
            continue

        action = random.choice(["buy", "sell"])
        price = df["close"].iloc[index]
```

```
        if action == "buy" and cash_available > 0 and last_action != "buy": # Avoid consecutive buys
            price_with_slippage = price * (1 + SLIPPAGE)
            nb_shares = cash_available / price_with_slippage
            total_commission += COMMISSION
            cash_available = 0
            buy_dates.append(df.index[index])
            last_action = "buy"

        elif action == "sell" and nb_shares > 0 and last_action != "sell": # Avoid consecutive sells
            price_with_slippage = price * (1 - SLIPPAGE)
            cash_available = nb_shares * price_with_slippage
            nb_shares = 0
            total_commission += COMMISSION
            sell_dates.append(df.index[index]) #pour le graphique
            last_action = "sell"
        # Update pfolio value
        pfolio_value = nb_shares * df["close"].iloc[index] + cash_available
        daily_portfolio_value.append(pfolio_value)
    # Final return
    total_gross_return = (cash_available + nb_shares * df["close"].iloc[-1])
    total_net_return = total_gross_return - total_commission
    total_portfolio_value = daily_portfolio_value
    # Daily return and sharpe ratio
    daily_portfolio_value = pd.Series(daily_portfolio_value).pct_change().dropna()
    sharp_ratio = calculate_sharpe_ratio(daily_portfolio_value)

    return total_gross_return, total_commission, total_net_return, total_portfolio_value, sharp_ratio, buy_dates, sell_dates
```

```
def strategy_technical_indicator(df):
    cash_available = 10000
    nb_shares = 0
    position_value = 0
    daily_portfolio_value = []
    total_commission = 0
    buy_dates = []
    sell_dates = []
    for index in range(1, len(df)):
        price = df["close"].iloc[index]

        # Identify general trend with moving avg of 20 days
        trend =  df["close"].iloc[index] > df["EMA"].iloc[index] # Bullish trend if current price is > than EMA

        # If statement to open a long position
        if (df["RSI"].iloc[index] < 30 or
            df["MACD"].iloc[index] > df["MACD_Signal"].iloc[index] or
            df["close"].iloc[index] > df["EMA"].iloc[index] or
            df["close"].iloc[index] > df["VWAP"].iloc[index] and
            trend):  # => Only buy in a bullish trend

            if cash_available > 0:
                price_with_slippage = price * (1 + SLIPPAGE)
                nb_shares = cash_available / price_with_slippage
                total_commission += COMMISSION
                cash_available = 0
                buy_dates.append(df.index[index])

        # If statement to close the long position
        elif (df["RSI"].iloc[index] > 70 or
              df["MACD"].iloc[index] < df["MACD_Signal"].iloc[index] or
              df["close"].iloc[index] < df["EMA"].iloc[index] or
              if nb_shares > 0:
                  price_with_slippage = price * (1 - SLIPPAGE)
                  cash_available = nb_shares * price_with_slippage
                  total_commission += COMMISSION
                  nb_shares = 0
                  sell_dates.append(df.index[index]) #pour le graphique

         position_value = nb_shares * df["close"].iloc[index] + cash_available
         daily_portfolio_value.append(position_value)

    total_gross_return = (cash_available + nb_shares * df["close"].iloc[-1])
    total_net_return = total_gross_return - total_commission
    total_portfolio_value = daily_portfolio_value
    daily_portfolio_value = pd.Series(daily_portfolio_value).pct_change().dropna()
    sharp_ratio = calculate_sharpe_ratio(daily_portfolio_value)

    return total_gross_return, total_commission, total_net_return, total_portfolio_value, sharp_ratio, buy_dates, sell_dates
```

```
def add_lagged_features(df, num_days=14):
    lagged_data = {}
    for i in range(1, num_days + 1):
        for column in ["open", "high", "low", "close", "volume", "EMA", "RSI", "MACD", "MACD_Signal", "MACD_Hist", "VWAP"]:
            lagged_data[f"{column}_{i}"] = df[column].shift(i)
    lagged_df = pd.DataFrame(lagged_data)
    df = pd.concat([df, lagged_df], axis=1)
    df = df.iloc[num_days:]
    return df

bars_with_lagged_features = add_lagged_features(bars_with_indicators)
[116]
```

```
# Load the ML model
best_model = load_model("my_neural_network_model.keras")
'''
# Reset optimizer with same param when the model was created. It's not necessary !
#best_model.compile(optimizer=Adam(learning_rate=BEST_PARAMS['learning_rate']), loss='binary_crossentropy',
 metrics=['accuracy'])
'''
# Load scaler
scaler = joblib.load("scaler.pkl")
```

```
# Prepare columns for the model (with 14 days of hist data)
features_columns = []
for i in range(1, 15):
    for column in ["open", "high", "low", "close", "volume", "EMA", "RSI", "MACD", "MACD_Signal", "MACD_Hist", "VWAP"]:
        features_columns.append(f"{column}_{i}")


excluded_columns = ["timestamp", "symbol", "trade_count", "vwap", "close"]
features_columns = [col for col in bars_with_lagged_features.columns if col not in excluded_columns]

# Extract data and normalize it with the scaler
x_new_data = bars_with_lagged_features[features_columns].values
x_new_data_scaled = scaler.transform(x_new_data)
[118]
```

```
# Predict the buy/sell signals
predictions = best_model.predict(x_new_data_scaled)
# Convert predictions into classes 0 or 1
predicted_classes = (predictions > 0.5).astype(int)
# Adjust bars to have the same length as the predicted_classes
bars_with_indicators = bars_with_indicators.iloc[-len(predicted_classes):].copy()
def strategy_neural_network(df, predictions, profit_taking=0.05, stop_loss=0.2):
    cash_available = 10000
    nb_shares = 0
    total_commission = 0
    daily_portfolio_value = []
    buy_dates = []
    sell_dates = []
    purchase_price = None  # Store purchase price to calculate P/L

    # Loop thru each day with the predictions
    for index in range(len(predictions)):
        price = df["close"].iloc[index]

        # Buy if model predicts price increase
        if predictions[index] == 1 and cash_available > 0:
            print(f"Achat effectué à {price} le {df.index[index]}")
            price_with_slippage = price * (1 + SLIPPAGE)
            nb_shares = cash_available / price_with_slippage
            total_commission += COMMISSION
            cash_available = 0
            purchase_price = price_with_slippage
            buy_dates.append(df.index[index])

        # Sell if model predicts price decrease, if profit reaches target or if it catches the stop loss
        elif nb_shares > 0:
            # Calculate potenital P/L
            price_with_slippage = price * (1 - SLIPPAGE)
            potential_profit = (price_with_slippage - purchase_price) / purchase_price
            potential_loss = (purchase_price - price_with_slippage) / purchase_price
```

```
            if predictions[index] == 0 or potential_profit >= profit_taking or potential_loss >= stop_loss:
                if potential_loss >= stop_loss:
                    print(f"Stop-loss triggered at {price} on {df.index[index]} with a loss of {potential_loss * 100:.2f}%")
                elif potential_profit >= profit_taking:
                    print(f"Sold at {price} on {df.index[index]} with a profit of {potential_profit * 100:.2f}%")
                else:
                    print(f"Sold at {price} on {df.index[index]} du to model price decrease prediction")

                cash_available = nb_shares * price_with_slippage
                total_commission += COMMISSION
                nb_shares = 0
                sell_dates.append(df.index[index])
                purchase_price = None  # Reset price after selling

        position_value = nb_shares * df["close"].iloc[index] + cash_available
        daily_portfolio_value.append(position_value)

    # Calculate total returns
    total_gross_return = (cash_available + nb_shares * df["close"].iloc[-1])
    total_net_return = total_gross_return - total_commission
    total_portfolio_value = daily_portfolio_value

    # Calculate daily returns and Sharpe ratio
    daily_portfolio_value_series = pd.Series(daily_portfolio_value).pct_change().dropna()
    sharpe_ratio = calculate_sharpe_ratio(daily_portfolio_value_series)

    return total_gross_return, total_commission, total_net_return, total_portfolio_value, sharpe_ratio, buy_dates, sell_dates
```

```
buy_hold_gross, buy_hold_commissions, buy_hold_net, buy_hold_daily  = strategy_buy_n_hold(bars_with_indicators)
random_gross, random_commissions, random_net, random_daily, random_sharpe, random_buy_dates, random_sell_dates =
 strategy_buy_n_sell_randomly(bars_with_indicators)
technical_gross, technical_commissions, technical_net, technical_daily, technical_sharpe, technical_buy_dates,
 technical_sell_dates = strategy_technical_indicator(bars_with_indicators)
neural_net_gross, neural_net_commissions, neural_net_net, neural_net_daily, neural_net_sharpe, neural_net_buy_dates,
 neural_net_sell_dates = strategy_neural_network(bars_with_indicators, predicted_classes, profit_taking=0.05, stop_loss=0.02)
```

```
fig = make_subplots(specs=[[{"secondary_y": False}]])

fig.add_trace(
    go.Scatter(x=bars.index, y=buy_hold_daily, mode='lines', name='Buy & Hold'),
)

fig.add_trace(
    go.Scatter(x=bars.index, y=random_daily, mode='lines', name='Random Strategy'),
)

fig.add_trace(
    go.Scatter(x=bars.index, y=technical_daily, mode='lines', name='Technical Indicator Strategy'),
)

fig.add_trace(
    go.Scatter(x=bars.index, y=neural_net_daily, mode='lines', name='Neural Network Strategy'),
)

fig.update_layout(
    title="Portfolio Value",
    xaxis_title="Date",
    yaxis_title="Portfolio Value (USD)",
    legend_title="Strategies",
    template="plotly_white",
)

fig.show()
```

```
fig_technical = make_subplots(rows=1, cols=1, shared_xaxes=True, subplot_titles=("Technical Indicator Strategy"))

# Technical Indicator Strategy
fig_technical.add_trace(
    go.Scatter(x=bars.index, y=bars['close'], mode='lines', name='AMZN Close Price'),
    row=1, col=1
)
fig_technical.add_trace(
    go.Scatter(x=technical_buy_dates, y=bars.loc[technical_buy_dates]['close'],
               mode='markers', marker=dict(symbol='triangle-up', color='green', size=10), name='Technical Buy'),
    row=1, col=1
)
fig_technical.add_trace(
    go.Scatter(x=technical_sell_dates, y=bars.loc[technical_sell_dates]['close'],
               mode='markers', marker=dict(symbol='triangle-down', color='red', size=10), name='Technical Sell'),
    row=1, col=1
)

fig_technical.update_layout(
    title="Buy/Sell for Technical Indicator Strategy on AMZN",
    xaxis_title="Date",
    yaxis_title="Price (USD)",
    legend_title="Actions",
    template="plotly_white",
)

fig_technical.show()
```

```
fig_random = make_subplots(rows=1, cols=1, shared_xaxes=True, subplot_titles=("Random Strategy"))

# Random Strategy
fig_random.add_trace(
    go.Scatter(x=bars.index, y=bars['close'], mode='lines', name='AMZN Close Price'),
    row=1, col=1
)
fig_random.add_trace(
    go.Scatter(x=random_buy_dates, y=bars.loc[random_buy_dates]['close'],
               mode='markers', marker=dict(symbol='triangle-up', color='green', size=10), name='Random Buy'),
    row=1, col=1
)
fig_random.add_trace(
    go.Scatter(x=random_sell_dates, y=bars.loc[random_sell_dates]['close'],
               mode='markers', marker=dict(symbol='triangle-down', color='red', size=10), name='Random Sell'),
    row=1, col=1
)

fig_random.update_layout(
    title="Buy/Sell for Random Strategy on AMZN",
    xaxis_title="Date",
    yaxis_title="Price (USD)",
    legend_title="Actions",
    template="plotly_white",
)

fig_random.show()
```

```
fig_nn = make_subplots(rows=1, cols=1, shared_xaxes=True, subplot_titles=("Neural Network Strategy"))

# Neural Network Strategy
fig_nn.add_trace(
    go.Scatter(x=bars.index, y=bars['close'], mode='lines', name='AMZN Close Price'),
    row=1, col=1
)
fig_nn.add_trace(
    go.Scatter(x=neural_net_buy_dates, y=bars.loc[neural_net_buy_dates]['close'],
               mode='markers', marker=dict(symbol='triangle-up', color='green', size=10), name='Neural Net Buy'),
    row=1, col=1
)
fig_nn.add_trace(
    go.Scatter(x=neural_net_sell_dates, y=bars.loc[neural_net_sell_dates]['close'],
               mode='markers', marker=dict(symbol='triangle-down', color='red', size=10), name='Neural Net Sell'),
    row=1, col=1
)

fig_nn.update_layout(
    title="Buy/Sell for Neural Network Strategy on AMZN",
    xaxis_title="Date",
    yaxis_title="Price (USD)",
    legend_title="Actions",
    template="plotly_white",
)

fig_nn.show()
```

```
# Create daily return for each strategy                                    12 34
buy_hold_returns = pd.Series(buy_hold_daily).pct_change().dropna()
random_returns = pd.Series(random_daily).pct_change().dropna()
technical_returns = pd.Series(technical_daily).pct_change().dropna()
neural_net_returns = pd.Series(neural_net_daily).pct_change().dropna()

fig = make_subplots(rows=4, cols=1, shared_xaxes=True, subplot_titles=("Buy & Hold", "Random Strategy", "Technical Indicator",
 "Neural Network"))

fig.add_trace(
    go.Histogram(x=buy_hold_returns, nbinsx=50, name='Buy & Hold Returns'),
    row=1, col=1
)

fig.add_trace(
    go.Histogram(x=random_returns, nbinsx=50, name='Random Strategy Returns'),
    row=2, col=1
)

fig.add_trace(
    go.Histogram(x=technical_returns, nbinsx=50, name='Technical Indicator Returns'),
    row=3, col=1
)

fig.add_trace(
    go.Histogram(x=neural_net_returns, nbinsx=50, name='Neural Network Returns'),
    row=4, col=1
)

fig.update_layout(
    title="Distribution of Daily Returns",
    xaxis_title="Daily Return",
    yaxis_title="Frequency",
    template="plotly_white",
```

```
fig.show()
```

```
#Hit ratio
buy_hold_hit_ratio = calculate_hit_ratio(buy_hold_returns)
random_hit_ratio = calculate_hit_ratio(random_returns)
technical_hit_ratio = calculate_hit_ratio(technical_returns)
neural_net_hit_ratio = calculate_hit_ratio(neural_net_returns)
```

[127]

```
# Calculate risk-reward ratio and profit factor for each strategy
buy_hold_risk_reward = calculate_risk_reward_ratio(buy_hold_returns)
buy_hold_profit_factor = calculate_profit_factor(buy_hold_returns)

random_risk_reward = calculate_risk_reward_ratio(random_returns)
random_profit_factor = calculate_profit_factor(random_returns)

technical_risk_reward = calculate_risk_reward_ratio(technical_returns)
technical_profit_factor = calculate_profit_factor(technical_returns)

neural_net_risk_reward = calculate_risk_reward_ratio(neural_net_returns)
neural_net_profit_factor = calculate_profit_factor(neural_net_returns)
```

12 32

```
# Print results in the console
print(f"Buy and Hold Strategy:")
print(f" - Gross Return: ${buy_hold_gross:.2f}")
print(f" - Total Commissions: ${buy_hold_commissions:.2f}")
print(f" - Net Return: ${buy_hold_net:.2f}")
print(f" - Hit Ratio: {buy_hold_hit_ratio:.2%}")
print(f" - Risk-Reward Ratio: {buy_hold_risk_reward:.2f}")
print(f" - Profit Factor: {buy_hold_profit_factor:.2f}\n")

print(f"Random Strategy:")
print(f" - Gross Return: ${random_gross:.2f}")
print(f" - Total Commissions: ${random_commissions:.2f}")
print(f" - Net Return: ${random_net:.2f}")
print(f" - Hit Ratio: {random_hit_ratio:.2%}")
print(f" - Risk-Reward Ratio: {random_risk_reward:.2f}")
print(f" - Profit Factor: {random_profit_factor:.2f}\n")

print(f"Technical Indicator Strategy:")
print(f" - Gross Return: ${technical_gross:.2f}")
print(f" - Total Commissions: ${technical_commissions:.2f}")
print(f" - Net Return: ${technical_net:.2f}")
print(f" - Hit Ratio - Technical Indicator: {technical_hit_ratio:.2%}")
print(f" - Risk-Reward Ratio: {technical_risk_reward:.2f}")
print(f" - Profit Factor: {technical_profit_factor:.2f}\n")

print(f"Neural Network  Strategy:")
print(f" - Gross Return: ${neural_net_gross:.2f}")
print(f" - Total Commissions: ${neural_net_commissions:.2f}")
print(f" - Net Return: ${neural_net_net:.2f}")
print(f" - Hit Ratio - Neural Network: {neural_net_hit_ratio:.2%}")
print(f" - Risk-Reward Ratio: {neural_net_risk_reward:.2f}")
print(f" - Profit Factor: {neural_net_profit_factor:.2f}\n")
```