FACULTY OF SCIENCES

Charting Wall Street: Detecting Common Trading Patterns and Assessing Their Relevance in Today's Automated Landscape

Rudolf Aelbrecht
Student number: 01601344

Supervisor: Prof. dr. Yvan Saeys
Counsellor: Arne Gevaert

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in de informatica

Academic year 2022-2023

GHENT UNIVERSITY

Summary

Technical analysis, especially the use of chart patterns, has been a cornerstone of trading strategies for decades. Traditional Patterns, like "head and shoulders" or "double top/bottom", are frequently referenced, yet empirical validations of their profitability remain limited. With the advent of machine learning, a new breed of 'Generic Patterns' has emerged, standing in contrast to these traditional forms.

The objective of this master dissertation was to discern the differences these patterns presented in comparison to random trading outcomes. To thoroughly evaluate both types of patterns, we compiled a dataset focused on S&P 500 companies spanning 1986 to 2022. Efficient data management was achieved through a tailored microservices architecture. For pattern detection, we tested a machine learning clustering approach (aiming to identify generic patterns) and a script-based method targeting well-established traditional patterns. The latter proved more effective. We assessed pattern performance using the Triple Barrier Method, supplemented by statistical analysis.

The results showed varied efficacy across patterns. For example, the double bottom pattern demonstrated potential, but the effect size was modest. Similarly, the double top, while different from random, showcased a minimal effect size and worse results. Classic patterns such as triple bottom, triple top, and head and shoulders struggled to display significant advantages in real-world trading. Surprisingly, the consistent performance of these patterns over years might hint at intrinsic limitations.

In conclusion, the findings suggest that traditional patterns display behavior distinct from random trading outcomes. However, the modest effect sizes raise questions about their realistic effectiveness in real-world trading scenarios. Integrating these patterns with other trading indicators might enhance their reliability. Furthermore, the research, through the development of custom tools and methodologies, lays a foundation for further exploration in this domain. The potential of advanced machine learning techniques in future studies offers a promising direction to deepen the understanding of trading patterns and their validity.

Samenvatting

Technische analyse, en in het bijzonder het gebruik van grafiekpatronen, vormt al decennialang een fundament van handelsstrategieën. Ondanks de populariteit van traditionele patronen zoals "head and shoulders" en "double top/bottom", ontbreekt het vaak aan grondige empirische validatie van hun effectiviteit. De recente opkomst van machine learning in deze context heeft echter geleid tot de introductie van 'Generic Patterns', die in constrast staan tot traditionele vormen.

Ons onderzoek heeft als doel de effectiviteit van deze patronen in kaart te brengen, met name in vergelijking met willekeurige handelsresultaten. Met behulp van een uitgebreide dataset van S&P 500-bedrijven uit de periode 1986-2022, hebben we zowel machine learning clustering als script-gebaseerde methoden ingezet voor patroondetectie. Uiteindelijk bleek de script-gebaseerde aanpak effectiever. De prestaties van de geïdentificeerde patronen werden geanalyseerd met de Triple Barrier Methode en statistische tests.

Interessant is dat patronen zoals 'double bottom' enig potentieel vertoonden, hoewel hun effectgrootte beperkt bleef. Anderzijds konden klassieke patronen zoals 'triple bottom', 'triple top' en 'head and shoulders' hun voordeel in de daadwerkelijke handelspraktijk niet aantonen. Bovendien suggereert de constante prestatie van deze patronen door de jaren heen mogelijke inherente beperkingen.

Als conclusie stellen we dat, hoewel traditionele patronen enig onderscheidend vermogen vertonen ten opzichte van willekeurige handelsactiviteiten, hun praktische effectiviteit beperkt is. Hun integratie met andere handelsindicatoren zou hun betrouwbaarheid kunnen vergroten. Daarnaast bieden de in dit onderzoek ontwikkelde tools en methodologieën een solide basis voor toekomstig onderzoek. Het verkennen van geavanceerde machine learning-technieken in vervolgstudies belooft een verdere verdieping van ons inzicht in de geldigheid en waarde van handelspatronen.

Charting Wall Street: Detecting Common Trading Patterns and Assessing Their Relevance in Today's Automated Landscape

Rudolf Aelbrecht

August 24, 2023

Abstract

Technical analysis, a cornerstone in trading strategies since the inception of stock markets, has seen increased interest with the advent of retail traders. The use of trading patterns, their efficiency and reliability, are continuous topics of debate. In this thesis, we compile a comprehensive dataset, extracting hundreds of symbols from companies listed in the historical S&P 500 index, to investigate the predictive power of common trading patterns. Utilizing a custom-built pipeline for processing stock data, we explore two distinctive methods for pattern detection: clustering and script-based approaches. The effectiveness of these patterns is assessed using label-based performance metrics, comparing their predictive accuracy to that of random trading. This comparative analysis serves to provide historical context to patterns and their usefulness overall.

1 Introduction

Since the dawn of the stock market, traders have strived to predict market movements, primarily through technical and fundamental analysis. Technical analysis, rooted in the study of historical price trends, believes that past market behavior can forecast future movements [1]. Fundamental analysis, on the other hand, focuses on evaluating a company's intrinsic quality, such as its financial health and industry position.

With the rise of online brokers, many newcomers, often less knowledgeable than seasoned investors, find solace in the apparent simplicity of technical analysis, especially in the backdrop of the growing interest in trading patterns. These patterns, enhanced by the accessibility to novel assets like cryptocurrencies and the advent of machine learning algorithms, are seen as effective tools. However, skepticism about the true effectiveness of technical analysis persists, with some arguing its reliability is no better than random chance [2].

Existing scholarly perspectives on technical analysis are divided. The validity of many studies can be questioned due to potential biases, limited dataset sizes, or lack of clarity in methodology [3]. To contribute clarity to this discourse, the current research work aims to evaluate trading patterns over several decades, employing advanced machine learning techniques. We address the efficacy of machine learning in identifying trading patterns, the predictive power of these patterns, and their evolving relevance in an era where machine learning is ubiquitous.

This research hopes to shed light on the utility of trading patterns, recommend optimal techniques for pattern identification, and facilitate future studies by creating tools for streamlined stock data processing, ultimately serving both novice and expert traders.

Section 2 delves into the realm of Technical Analysis, offering a summary of existing literature. Following this, Section 3, details the foundational steps we undertook as prerequisite work. In Section 4, we shed light on the approaches for detecting patterns, encompassing clustering-based and script-based detection methodologies. In Section 5, we discuss findings and in section 6 we present the conclusions.

2 Technical Analysis

Within the realm of technical analysis, chart patterns have maintained a recurring presence, highlighting their enduring fascination among certain traders and enthusiasts. Two principal categories emerge from the wealth of trading literature and guidebooks accumulated over past decades. Firstly, Traditional Patterns—often cited across a broad spectrum of non-academic trading books [5] are identified by their distinctive configurations, most notably the "head and shoulders" (Figure 1) and "double top/bottom" (Figure 2). Their widespread mention in non-academic resources contrasts with a noticeable lack of empirical studies evaluating their profitability. This divide between popular trading literature and empirical validation sets the stage for this thesis.

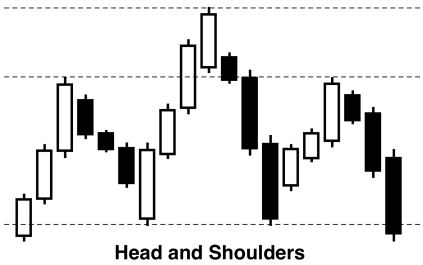


Figure 1: Head and shoulders pattern, characterized by a peak (head) flanked by two lower peaks (shoulders), indicating a potential trend reversal.

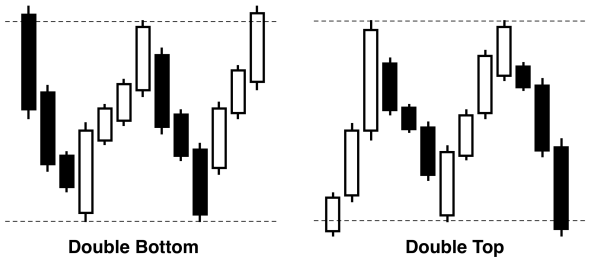


Figure 2: Double top and double bottom patterns, characterized by touching the same price range twice before definitively reversing the trend.

With the rise of advanced machine learning techniques, the discovery and categorization of Novel or 'Generic Patterns' have become more refined [6]. These patterns, although newer, have garnered attention due to their recurring appearances. It is worth noting that while some sources ascribe trading successes to these patterns, the limited empirical evidence from a few studies makes such claims demand careful consideration. Among the various machine learning methods employed, clustering serves as just one example of how these patterns have been systematically categorized and identified.

Subsequent literature has highlighted potential challenges in trading strategies [7]. Issues, such as neglecting transaction costs and the risk of overfitting, have been emphasized. These considerations stress the need for careful and precise strategy development. In this context, the Triple-Barrier Method stands out in algorithmic trading [8]. Rather than solely relying on price returns for labeling, this method integrates three key criteria: profit objectives, stop-loss levels, and a predefined time window. This multi-faceted approach to labeling makes it less dependent on raw price movements and offers a more nuanced perspective on potential trade outcomes. By doing so, it aims to provide a more robust foundation for algorithmic models, enhancing adaptability in various market conditions.

Building on the explored literature, this study aims to thoroughly examine the effectiveness of these patterns. While chart patterns are often emphasized, there's a clear lack of detailed research on the topic. It is important to point out that, based on current academic views, patterns should be interpreted with caution. However, given the widespread enthusiasm for patterns in online trading communities, this research seeks to determine if such caution is truly warranted and to add to the current knowledge of technical analysis.

3 Prerequisite Work

In order to contextualize the methodologies and findings presented in this research, this section provides

an overview of the primary data sources, the criteria for their selection, and the preliminary infrastructure development. By establishing this foundation, we aim to facilitate a comprehensive understanding of the subsequent research methodologies and results.

Data Collection and Preparation:

Initially, we sourced data from Binance [9] for cryptocurrencies. However, due to cryptocurrency's pronounced volatility and its limited historical depth, we chose not to pursue this further. The focus shifted to stock market data, specifically from Unicorn's EOD Historical Data [10], which offered daily candles spanning several decades. In the effort to acquire a comprehensive dataset for pattern detection, we curated about 500 symbols from a pool of 10,000 US-traded symbols, applying criteria such as market capitalization and historical length. The literature review indicated that many studies were limited in scope, prompting the broader approach. The dataset, which includes the top 250 companies each quarter from 1986 to 2022, became the foundational data for the research. However, it is imperative to note the potential survivorship bias [11], and any optimistic outcomes should be interpreted with this consideration.

Infrastructure Development:

While generalized platforms like Spark [12] offer extensive capabilities and flexibility, they sometimes fall short in delivering highly optimized performance for specific tasks. The decision to devise a custom pipeline for stock data processing arose from a desire for end-to-end control, allowing for hyper-optimization tailored to the unique requirements, particularly with the foundational emphasis on candle data and the derivative data computations it necessitates. Though traditional tools can be powerful, they might not provide the level of granular optimization that a custom-built solution can. Hence, the research leaned on a pipeline that did not just offer agility and insightful analysis, but also unparalleled control, enabling optimization and parallelization at every stage.

This pipeline leveraged a microservices architecture [13], each catering to a unique aspect of stock data (Figure 3):

1. **Bridge Service**: Integrates external stock data into the internal system.
2. **Candle Processor**: Handles candlestick data visualizing price movements.
3. **Indicator Processor**: Generates key market indicators from raw stock data.
4. **Event Processor**: Automates and oversees algorithmic stock analysis.

Recognizing the static nature of historical stock data, caching was integrated at each computation stage, optimizing performance. Data was segmented into blocks, inspired by the HTTP protocol's chunked data requests, ensuring efficient storage and retrieval. HTTP was the chosen communication protocol for its simplicity, though it demanded meticulous management due to its high request rate.

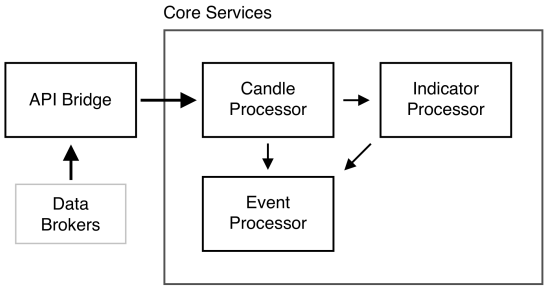


Figure 3: An overview of the services and their dataflow directions. Core services are exposed, and are used for visualizations and evaluating the patterns.

Visualization Development:

A custom visualization tool was crafted to tackle challenges in analyzing stock market data. This tool aimed for speed, flexibility, and reduced programming overhead. While open-source solutions like matplotlib offer a broad spectrum of functionalities, they sometimes are not optimized for in-depth time series analysis. Additionally, commercial and specialized tools often kept essential features behind paywalls or were challenging to master. Given these constraints, we chose to develop a tool using the React framework [14], ensuring fast data handling and rendering. It

should be emphasized that the intention was not to eclipse established platforms [15] but rather to create a specialized solution tailored for precise visualization of financial time-series data (Figure 4).

Figure 4: Double-top patterns being visualised.

4 Approach

Detecting Patterns:

The journey into pattern detection started with a machine learning-driven clustering technique. We believed that frequently occurring market patterns would stand out when clustered, especially in contrast to random or sporadic market movements. With this idea, we attempted to utilize Dynamic Time Warping (DTW) [6].

However, this approach was more intricate than we initially thought. Rather than isolating distinct recurring patterns, we encountered numerous high-frequency patterns. Many of these seemed to echo random market changes instead of clear trends. These issues, combined with the computational challenges of clustering, led us to rethink the strategy.

We shifted to a script-based detection method, focusing on established patterns often discussed in financial studies. We specifically emphasized the double bottom and double top patterns, choosing them for their regular appearance in market data and their straightforward descriptions.

The script-based approach came with its own set of challenges, primarily due to noisy data and unexpected anomalies. Yet, with the structured engineering of the pipeline, implementation was more straightforward. We were able to reliably identify patterns such as the double bottom, double top, triple bottom, triple top, and head and shoulders in the market data. Subsequent steps involved evaluating these patterns based on their occurrence and performance.

Evaluating Performance:

In determining the impact of specific events on price trajectories, we employed two methodologies: the Triple Barrier Method and the Hybrid Bucket-based Classifier.

The Triple Barrier Method (Figure 5) incorporates three barriers – two horizontal and one vertical – categorizing outcomes based on the first barrier breached. It accentuates: - Directionality in price movement. - Capture of sudden market downturns. - Independence from mere price values.

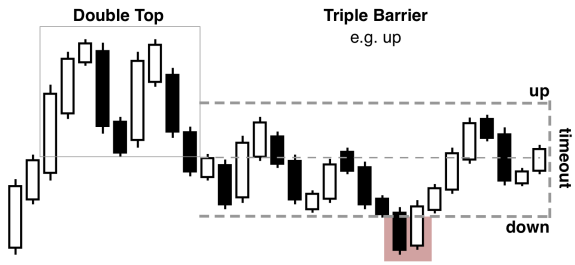


Figure 5: Triple barrier method. We broke the lower barrier, so this example would be labeled “down”.

However, the method’s vertical barrier might omit some price dynamics. Addressing this, we devised the Hybrid Bucket-based Classifier (Figure 6). This method divides outcomes into distinct ranges, capturing all dynamics, including those potentially overlooked by the Triple Barrier Method.

While the hybrid offers granular insights, for the purposes, the Triple Barrier Method’s directionality and comprehensive view make it the primary evaluation

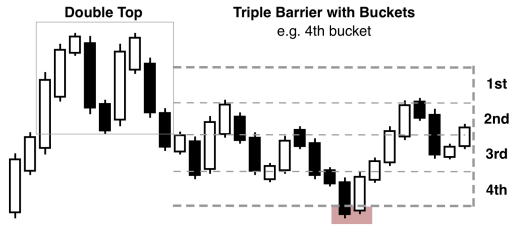


Figure 6: Bucket-based classifier. We broke the lower barrier, so we classify in the lowest/4th bucket.

technique, effectively discerning event-driven price movements.

Statistical Testing:

The primary objective is to validate that the observed patterns transcend random events and hold genuine predictive power. To achieve this, we used the following evaluation measures:

- **Success Rates**: We analyzed distinct patterns like double bottom, double top, among others, calculating their success using:

$$\text{Success Rate} = \frac{\text{Successful occurrences}}{\text{Total occurrences}}$$

  Comparisons were drawn against both main and random data to evaluate pattern reliability versus random instances.

- **Proportions Z-Test**: The z-test assessed the predictive capability of the patterns against random events. A notable z-score indicates patterns outperforming randomness, with p-values quantifying the confidence in these results.

- **Bonferroni Correction**: Given the multiple pattern evaluations, p-values were adjusted using the Bonferroni correction, ensuring the conclusions remain robust against potential Type I errors from multiple comparisons.

- **Effect Size**: Beyond mere significance, the real-world impact of observed differences was gauged using Cohen's h, offering a comprehensive view of the practical implications of the findings.

Through this multi-faceted statistical approach, we aimed to robustly ascertain and quantify the validity and impact of the identified patterns in trading scenarios.

5 Results

This section distills the findings from the examination of various trading pattern algorithms.

Random Pattern: Random (Figure 7) exhibited an increase in win rate in conjunction with the barrier's thresholds. This trend can be attributed to the stocks sourced from the S&P500 index, which historically has a rising trajectory. The statistical analysis confirmed that this increasing win rate can be explained by the inherent upward nature of the S&P500 index itself.

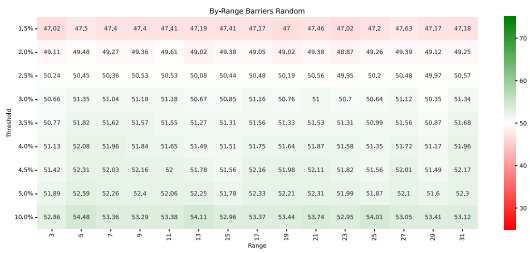


Figure 7: Random win rate

Double Bottom Pattern: The double bottom (Figure 8) consistently yielded a higher win rate than break-even, outperforming the random baseline. Although the difference was statistically significant after applying the Bonferroni correction, the practical effect size was small, with on average a Cohen's $h < 0.03$.

Double Top Pattern: The double top (Figure 9) pattern exhibited win rates often exceeding 50%, which was not what we expected, as we expected a trend reversal instead of a continuation. Its performance was statistically significant after the Bonferroni correction, but only up to a 4% threshold, indicating its effectiveness might diminish at higher thresholds.

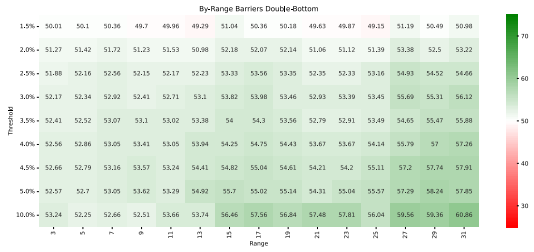


Figure 8: Double bottom win rate

The practical effect size was small, with on average a Cohen's $h < 0.02$.

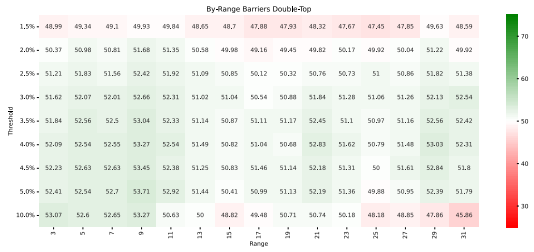


Figure 9: Double top win rate

Others (Triple Bottom, Triple Top, Head and Shoulders): The results did not find these patterns to provide consistent or statistically significant advantages after correcting for multiple comparisons. This might suggest their performance diverges from theoretical expectations when applied to real-world trading data.

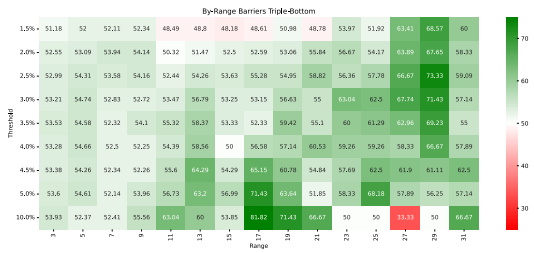


Figure 10: Triple bottom win rate

Temporal Analysis: An year-over-year analysis

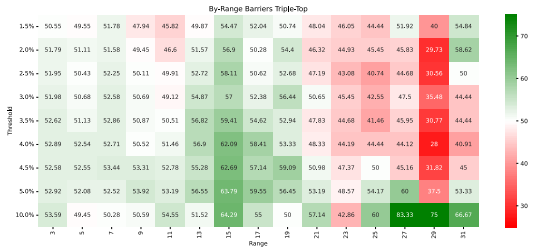


Figure 11: Triple top win rate

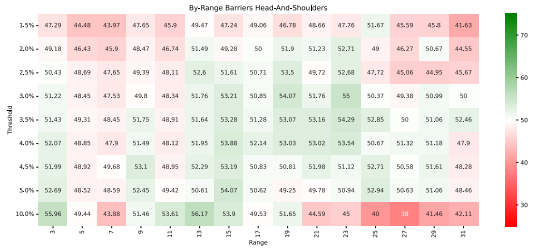


Figure 12: Head and shoulders win rate

revealed subtle trends in pattern performance. For instance, the double bottom pattern demonstrated a slight but consistent decline in its win rate over the years (Figure 13), albeit not reaching statistical significance. The double top and other patterns, when analyzed year-over-year, similarly exhibited trends that did not achieve statistical significance.

6 Conclusion

This study sought to gauge the efficacy of widely recognized financial market patterns. Initially leaning towards machine learning techniques, practical complexities led us to a scripted, algorithmic approach. Central to the work was an extensive software pipeline, facilitating the efficient analysis of stock datasets and a custom visualization tool, addressing challenges of data interaction.

Key findings include:

1. Classical patterns' effectiveness is questionable. For instance, the head and shoulders pattern

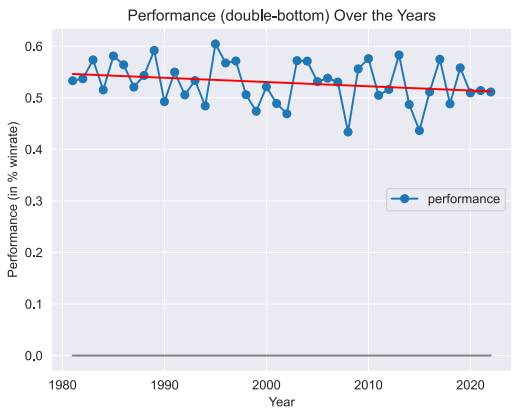


Figure 13: Year-over-year win rate for double bottom.

appears no different from random occurrences, and other patterns, like the double bottom, offer minimal effect sizes.

2. Patterns' performance remains largely consistent over time, despite the tech evolution and trading shifts. This constancy suggests a potential inherent limitation in their foundational efficacy.

In conclusion, while trading patterns have shown some utility, their consistent performance amidst technological advances raises doubts about their foundational effectiveness. Furthermore, literature suggests that relying solely on these patterns might be limiting. Combining them with other parameters, such as trends, could enhance their predictive accuracy. Exploring these combinations and diving deeper into machine learning offer promising avenues for future research.

7 References

1. Murphy JJ. Technical analysis of the financial markets: Study guide: A comprehensive guide to trading methods and applications. 2nd ed. Welwyn Garden City, England: NYIF; 1998.

2. Park CH, Irwin SH. What do we know about the profitability of technical analysis? Journal of Economic Surveys [Internet]. 2007;21(4):786–826. Available from: http://dx.doi.org/10.1111/j.1467-6419.2007.00519.x

3. Farias Nazário RT, Silva JL e, Sobreiro VA, Kimura H. A literature review of technical analysis on stock markets. The Quarterly review of economics and finance: journal of the Midwest Economics Association [Internet]. 2017;66:115–26. Available from: http://dx.doi.org/10.1016/j.qref.2017.01.014

4. Bulkowski TN. Trading classic chart patterns. Nashville, TN: John Wiley & Sons; 2002.

5. Murphy JJ. The visual investor: How to spot market trends. 2nd ed. Murphy JJ, editor. Chichester, England: John Wiley & Sons; 2009.

6. Tsinaslanidis P, Guijarro F. What makes trading strategies based on chart pattern recognition profitable? Expert systems [Internet]. 2021;38(5). Available from: http://dx.doi.org/10.1111/exsy.12596

7. Lento C, Gradojevic N. The profitability of technical trading rules: A combined signal approach. Journal of Applied Business Research (JABR) [Internet]. 2011;23(1). Available from: http://dx.doi.org/10.19030/jabr.v23i1.1405

8. Lopez de Prado M. Advances in financial machine learning. Nashville, TN: John Wiley & Sons; 2018.

9. Binance - cryptocurrency exchange for bitcoin, ethereum & altcoins [Internet]. Binance. Available from: https://www.binance.com

10. Prices EH, Api FD. Fundamental, EOD historical prices and financial data API [Internet]. Eodhd.com. Available from: https://eodhd.com

11. Kan R, Kirikos G, Halpern P, Andrew G. Biases in evaluating trading strategies [Internet]. Utoronto.ca. Available from: https://www-2.rotman.utoronto.ca/~kan/papers/mom1.pdf

12. Apache SparkTM - unified engine for large-scale data analytics [Internet]. Apache.org. Available from: https://spark.apache.org

13. Salah T, Jamal Zemerly M, Yeun CY, Al-Qutayri M, Al-Hammadi Y. The evolution of distributed systems towards microservices architecture. In: 2016 11th international conference for internet technology and secured transactions (ICITST). IEEE; 2016. p. 318–25.

14. React [Internet]. React.dev. Available from: https://react.dev

15. Free charting library by TradingView [Internet]. TradingView. Available from: https://www.tradingview.com/HTML5-stock-forex-bitcoin-charting-library/

Charting Wall Street: Detecting Common Trading Patterns and Assessing Their Relevance in Today's Automated Landscape

Rudolf Aelbrecht

August 24, 2023

Abstract

Technische analyse, een fundamenteel onderdeel van handelsstrategieën, heeft met de opkomst van particuliere handelaren toenemende belangstelling gewonnen. Het herkennen van handelspatronen, evenals hun efficiëntie en betrouwbaarheid, blijven centrale onderwerpen van discussie. In deze thesis hebben we een uitgebreide dataset samengesteld door honderden symbolen te extraheren van bedrijven die genoteerd staan op de historische S&P 500-index. Dit met als doel de voorspellende kracht van bekende handelspatronen te onderzoeken. Door gebruik te maken van een specifiek ontworpen proces voor het verwerken van beursgegevens, verkennen we twee verschillende methoden voor patroondetectie: clustering en script-gebaseerde benaderingen. De effectiviteit van deze patronen wordt beoordeeld met behulp van label-gebaseerde prestatie-indicatoren. Vervolgens wordt hun voorspellende nauwkeurigheid vergeleken met die van willekeurige momenten. Deze vergelijkende analyse verschaft inzicht in de historische context van handelspatronen en hun toepasbaarheid.

1 Introductie

Sinds de oprichting van de aandelenmarkt hebben handelaren getracht marktbewegingen te voorspellen, vooral door middel van technische en fundamentele analyse. Technische analyse, die zijn fundament vindt in de studie van historische prijstrends, is gebaseerd op de overtuiging dat vroegere marktprestaties toekomstige bewegingen kunnen aangeven [1]. Fundamentele analyse, daarentegen, focust zich op het beoordelen van de intrinsieke waarde van een bedrijf, zoals zijn financiële gezondheid en positie binnen de industrie.

Met de komst van online brokers wenden veel nieuwkomers, vaak minder geïnformeerd dan ervaren beleggers, zich tot de ogenschijnlijke eenvoud van technische analyse. Dit vooral gezien de toenemende interesse in handelspatronen. Deze patronen, versterkt door de beschikbaarheid van nieuwe activaklassen zoals cryptocurrencies en de opkomst van machine learning-algoritmes, worden beschouwd als effectieve instrumenten. Echter, twijfels over de daadwerkelijke effectiviteit van technische analyse blijven, met argumenten dat de betrouwbaarheid niet uitstijgt boven pure toeval [2].

Bestaande wetenschappelijke inzichten over technische analyse zijn verdeeld. Vele studies worden bekritiseerd vanwege mogelijke bias, beperkte datasetomvang of onduidelijke methodologie [3]. Met het doel bij te dragen aan dit debat, richt dit onderzoek zich op het evalueren van handelspatronen over meerdere decennia door gebruik te maken van geavanceerde machine learning-technieken. We onderzoeken: de effectiviteit van machine learning in het identificeren van handelspatronen, hun voorspellende kracht, en de verschuivende relevantie in een tijdperk waarin machine learning steeds dominanter wordt.

Dit onderzoek beoogt meer inzicht te geven in de bruikbaarheid van handelspatronen, aan te bevelen welke technieken het beste zijn voor patroonherkenning, en toekomstig onderzoek te stimuleren door tools te ontwikkelen voor een gestroomlijnde verwerking van beursgegevens. Dit alles ten dienste van zowel beginnende als ervaren handelaren.

In Sectie 2 verdiepen we ons in het domein van technische analyse en bieden we een overzicht van de huidige literatuur. Sectie 3 beschrijft de fundamentele stappen die we hebben ondernomen ter voorbereiding. Sectie 4 behandelt de benaderingen voor patroondetectie, waaronder zowel clustering-gebaseerde als script-gebaseerde methodologieën. In Sectie 5 bespreken we de resultaten en Sectie 6 sluit af met de conclusies.

2 Technische Analyse

Binnen het domein van technische analyse hebben grafiekpatronen een blijvende prominentie behouden, wat hun aanhoudende fascinatie onder handelaren en enthousiastelingen benadrukt. Uit de omvangrijke handelsliteratuur en handleidingen van de afgelopen decennia zijn er twee hoofdcategorieën te onderscheiden. De eerste categorie betreft de Traditionele Patronen - vaak aangehaald in tal van niet-academische handelsboeken [5]. Deze worden herkend aan hun kenmerkende vormen, zoals het "head and shoulders"-patroon (Figuur 1) en het "double top/bottom"-patroon (Figuur 2). De frequentie waarmee ze in populaire literatuur worden genoemd staat in schril contrast met het merkbare gebrek aan empirisch onderzoek naar hun winstgevendheid. Deze kloof tussen handelsliteratuur en wetenschappelijke validatie vormt de achtergrond voor deze thesis.

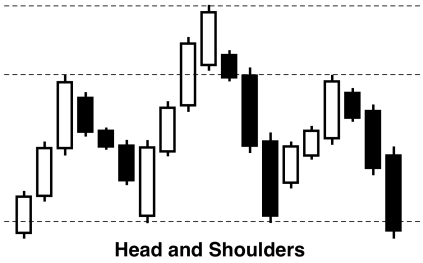


Figure 1: Head and shoulders pattern, characterized by a peak (head) flanked by two lower peaks (shoulders), indicating a potential trend reversal.

Met de vooruitgang in machine learning-technieken

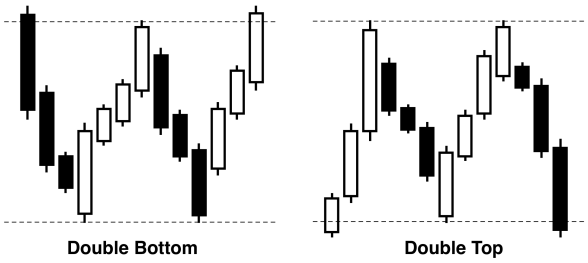


Figure 2: Double top and double bottom patterns, characterized by touching the same price range twice before definitively reversing the trend.

is er een nieuwe categorie handelspatronen geïntroduceerd, bekend als 'Generic Patterns' [6]. In tegenstelling tot traditionele patronen in technische analyse, ontstaan deze Generic Patterns rechtstreeks uit het machine learning-proces. Hoewel bepaalde bronnen hun handelssuccessen toeschrijven aan deze nieuwe patronen, suggereert het beperkte empirische bewijs dat zulke beweringen een grondige evaluatie behoeven. Clustering, als een van de machine learning-methoden, illustreert hoe deze Generic Patterns systematisch worden geïdentificeerd en gecategoriseerd.

Er is in de literatuur tevens aandacht voor uitdagingen die kunnen opduiken bij handelsstrategieën [7]. Zaken als het negeren van transactiekosten en het risico van overfitting zijn onder de aandacht gebracht. Deze aspecten benadrukken het belang van een zorgvuldige strategieontwikkeling. Een benadering die hier opvalt is de Triple-Barrier Methode in algoritmisch handelen [8]. Deze gaat verder dan enkel prijsrendementen voor labeling en omvat drie criteria: winstdoelstellingen, stop-loss niveaus en een vooraf vastgestelde tijdsduur. Door deze aanpak krijgt men een genuanceerder beeld van mogelijke handelsresultaten, waardoor modellen beter gegrond zijn en flexibeler kunnen inspelen op verschillende marktsituaties.

Gezien de bestaande literatuur heeft deze studie als doel de effectiviteit van deze patronen diepgaand te verkennen. Hoewel grafiekpatronen vaak worden benadrukt, is er een opmerkelijk tekort aan gedetailleerde studies hierover. Op basis van huidige

academische inzichten dient men voorzichtig te zijn met het interpreteren van patronen. Toch, gezien het grote enthousiasme in online handelsgemeenschappen, streeft dit onderzoek ernaar om te bepalen of dergelijke voorzichtigheid gerechtvaardigd is en zo een bijdrage te leveren aan de bestaande kennis over technische analyse.

3 Voorbereidend Werk

Deze sectie biedt een raamwerk om de methodologieën en resultaten van dit onderzoek te begrijpen. Het geeft een overzicht van de belangrijkste gegevensbronnen, de selectiecriteria en de ontwikkeling van de initiële infrastructuur. Door deze voorbereiding te schetsen, beogen we een helder inzicht te geven in de volgende onderzoeksmethodologieën en resultaten.

Data Verzameling en Voorbereiding:

Initieel werden gegevens van Binance [9] voor cryptocurrencies verzameld. Echter, gezien de opvallende volatiliteit en beperkte historische diepte van cryptocurrencies, besloten we onze focus te verleggen naar de aandelenmarkt. Hierbij vertrouwden we op gegevens van Unicorn's EOD Historical Data [10], die dagelijkse candles leverden over meerdere decennia. Bij het samenstellen van een uitgebreide dataset gericht op patroondetectie selecteerden we zorgvuldig ongeveer 500 symbolen uit een totaal van 10.000 in de VS verhandelde symbolen, met criteria zoals marktkapitalisatie en historische duur als leidraad. Ondanks dat eerdere studies vaak beperkt waren qua scope, kozen wij voor een uitgebreidere benadering. Onze dataset, die de top 250 bedrijven per kwartaal tussen 1986 en 2022 bevatte, diende als basis voor dit onderzoek. Het is echter essentieel om bewust te zijn van een mogelijke overlevingsbias [11], en optimistische resultaten dienen in dit licht te worden bekeken.

Infrastructuurontwikkeling:

Ondanks dat algemene platforms zoals Spark [12] breed inzetbaar en flexibel zijn, kunnen ze soms niet voldoen aan specifieke prestatiebehoeften. De keuze voor het ontwikkelen van een aangepaste pijplijn voor de verwerking van beursgegevens kwam voort uit de wens voor controle, wat resulteerde in een hyper-geoptimaliseerde aanpak, specifiek voor candlegegevens en daaruit afgeleide berekeningen. Onze aanpak berustte daarom op een pijplijn die niet enkel flexibiliteit en inzicht bood, maar ook controle, waarmee optimalisatie en parallelle verwerking op elk niveau werden gerealiseerd.

We hanteerden een microservices-architectuur [13], waarbij elke service specifiek was gericht op een bepaald facet van beursgegevens (Figuur 3):

1. **Bridge Service**: Integreert externe beursgegevens in het interne systeem.
2. **Candle Processor**: Verwerkt candlegegevens en visualiseert prijsbewegingen.
3. **Indicator Processor**: Genereert cruciale marktindicatoren vanuit de ruwe beursgegevens.
4. **Event Processor**: Stuurt en overziet geautomatiseerde algoritmische beursanalyses.

Omdat historische beursgegevens onveranderlijk zijn, werd caching toegepast op elk verwerkingsniveau om de prestaties te optimaliseren. We verdeelden gegevens in blokken, geïnspireerd door de 'chunked' data-aanvragen van het HTTP-protocol, om zo efficiënte opslag en snelle toegang te waarborgen. We kozen voor HTTP als communicatieprotocol vanwege de eenvoud, hoewel dit een uitgebreide cache strategie vereiste gezien de hoge request frequentie.

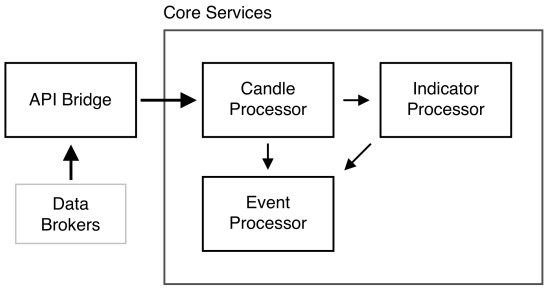


Figure 3: Een overzicht van de diensten en hun gegevensstroomrichtingen.

Visualisaties:

Voor de specifieke uitdagingen van beursgegevensanalyse is een op maat gemaakte visualisatietool ontworpen. We richtten ons hierbij op snelheid, flexibiliteit en het minimaliseren van programmeeroverhead. Ondanks dat opensource-oplossingen zoals matplotlib een breed spectrum aan functies aanbieden, zijn deze niet altijd geoptimaliseerd voor gedetailleerde tijdsreeksanalyses. Daarnaast merkten we op dat commerciële en gespecialiseerde tools vaak waardevolle functies vergrendelden achter betalingsmechanismen of een steile leercurve hadden. Mede door deze uitdagingen besloten we een eigen tool te ontwikkelen met het React-framework [14]. Dit framework stond garant voor vlotte gegevensverwerking en -weergave. Het is essentieel om te benadrukken dat ons doel niet was om gevestigde platforms zoals TradingView[15] te overtreffen, maar om een tool te creëren die specifiek was afgestemd op de nauwkeurige weergave van financiële tijdsreeksgegevens, zoals geïllustreerd in Figuur 4.

Figure 4: Double-top gevisualiseerd.

4 Aanpak

Detectie van Patronen:

Onze zoektocht naar het detecteren van marktpatronen startte met een clusteringsmethode gebaseerd op machine learning. We gingen ervan uit dat marktpatronen die frequent voorkwamen, duidelijk zichtbaar zouden zijn wanneer ze in clusters werden gebracht, vooral in contrast met sporadische of willekeurige marktbewegingen. Geïnspireerd door deze hypothese, pasten we Dynamic Time Warping (DTW) [6] toe.

Deze benadering bleek echter gecompliceerder dan we aanvankelijk hadden geanticipeerd. In plaats van specifieke, terugkerende patronen te identificeren, ontdekten we een overvloed aan hoogfrequente patronen. Veel van deze patronen reflecteerden willekeurige marktbewegingen eerder dan concrete trends. Door deze bevindingen en de computationele uitdagingen die met clustering gepaard gaan, besloten we onze strategie te herzien.

Als alternatief kozen we voor een script-gebaseerde detectiemethode, waarbij de focus lag op bekende patronen zoals beschreven in financiële literatuur. Specifiek hebben we ons gericht op de 'double bottom' en 'double top' patronen, vanwege hun frequente verschijning en hun relatief eenvoudige beschrijvende aard.

Deze script-gebaseerde aanpak kende weliswaar ook zijn uitdagingen, vooral door ruis in de gegevens en onvoorziene anomalieën. Desondanks bleek de implementatie, gezien de goed gestructureerde opzet van onze pijplijn, vergelijkbaar eenvoudiger. Hiermee konden we betrouwbaar patronen identificeren zoals 'double bottom', 'double top', 'triple bottom', 'triple top' en 'head and shoulders'. Onze volgende stappen bestonden uit het beoordelen van de frequentie en prestaties van deze geïdentificeerde patronen.

Prestatiebeoordeling:

Voor het meten van de invloed van specifieke marktgebeurtenissen op prijsfluctuaties, hebben we twee evaluatiemethoden aangenomen: de Triple Barrier Methode en de Hybrid Bucket-gebaseerde Classifier.

Triple Barrier Methode: Dit evaluatiemechanisme werkt met drie barrières - twee horizontale en één verticale. Afhankelijk van welke barrière het eerst wordt doorbroken, worden de uitkomsten geclassificeerd. Deze methode is nuttig omdat het: - De richting van prijsbewegingen benadrukt. - Plotselinge marktdalingen vangt. - Niet enkel gebaseerd is op absolute prijswaarden.

Een beperking van de Triple Barrier Methode is echter dat de verticale barrière soms bepaalde prijsdynamieken kan missen.

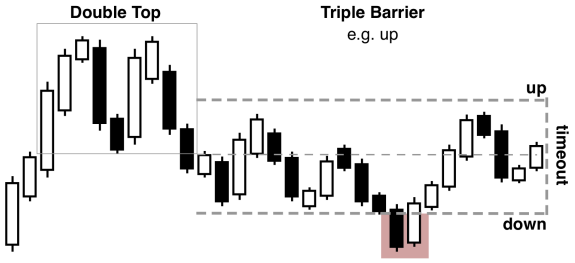


Figure 5: Triple Barrier Methode. Nadat de onderste barrière is doorbroken, labelen we dit voorbeeld als "neerwaarts".

Hybrid Bucket-gebaseerde Classifier: Om de bovengenoemde beperking aan te pakken, introduceerden we deze classifier. Deze benadering verdeelt de uitkomsten in verschillende bereiken en neemt alle prijsbewegingen in beschouwing, zelfs degenen die mogelijk niet worden gevangen door de Triple Barrier Methode.

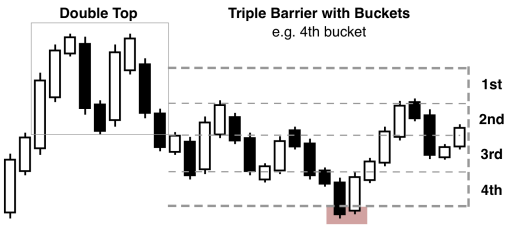


Figure 6: Bucket-gebaseerde classifier. Na het doorbreken van de onderste barrière classificeren we in het laagste/4e bucket.

Hoewel de Hybrid Bucket-gebaseerde Classifier een dieper inzicht geeft in prijsbewegingen, blijft de Triple Barrier Methode de voorkeursmethode vanwege zijn duidelijke richting en holistische blik. Het stelt ons in staat om prijsbewegingen die worden beïnvloed door specifieke gebeurtenissen, effectief te onderscheiden.

Statistische Testen:

Ons primaire doel was om aan te tonen dat de door ons geïdentificeerde patronen niet slechts willekeurige gebeurtenissen waren, maar daadwerkelijk beschikten over voorspellende waarde. We hebben verschillende technieken gebruikt om dit doel te bereiken:

- **Succesratio's**: Door het analyseren van diverse patronen zoals de double bottom, double top en anderen, hebben we hun succes gemeten met de volgende formule:

  $$\text{Succesratio} = \frac{\text{Aantal succesvolle gebeurtenissen}}{\text{Totaal aantal gebeurtenissen}}$$

  Hierbij hebben we onze bevindingen vergeleken met zowel echte als willekeurig gegenereerde data om de effectiviteit van de patronen ten opzichte van willekeurige scenario's te beoordelen.

- **Z-test van Verhoudingen**: Door de z-test toe te passen, hebben we de voorspellende kracht van de patronen tegenover willekeurige gebeurtenissen geëvalueerd. Een significante z-score duidt erop dat de patronen significant beter presteren dan willekeurige scenario's, en de bijbehorende p-waarden geven de betrouwbaarheid van deze uitkomsten weer.

- **Bonferroni-correctie**: Aangezien we meerdere patronen hebben geëvalueerd, gebruikten we de Bonferroni-correctie om p-waarden aan te passen. Dit zorgt ervoor dat onze bevindingen beschermd zijn tegen het risico op Type I-fouten die kunnen ontstaan door meerdere vergelijkingen.

- **Effectgrootte**: Naast het vaststellen van statistische significantie, hebben we ook de praktische relevantie van de waargenomen verschillen gemeten met Cohen's h. Dit biedt een dieper inzicht in de daadwerkelijke impact van onze bevindingen.

Door deze uitgebreide statistische methoden te combineren, streefden we ernaar de validiteit en het potentiële effect van de waargenomen handelspatronen op een grondige en kwantitatieve manier vast te stellen.

5 Resultaten

De onderstaande resultaten geven een overzicht van onze bevindingen met betrekking tot de prestaties

van diverse handelspatronen.

Willekeurig Patroon: Het onderzochte willekeurige patroon (Figuur 7) toonde een groeiend winstpercentage in verband met de barrièredrempels. Dit kan worden verklaard door de algemene opwaartse trend van de S&P500-index gedurende de onderzoeksperiode.

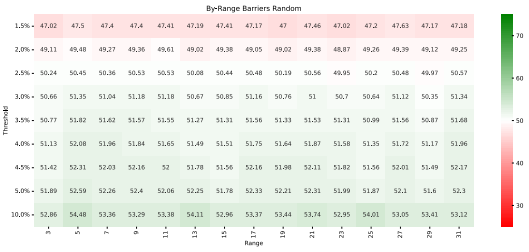


Figure 7: Random winstpercentage

Double Bottom Patroon: Dit patroon (Figuur 8) vertoonde consistent positieve resultaten ten opzichte van het break-even niveau, overtreffend het willekeurige patroon. Desondanks was de praktische impact ervan relatief beperkt, met een bescheiden effectgrootte volgens Cohen's $h < 0.03$.

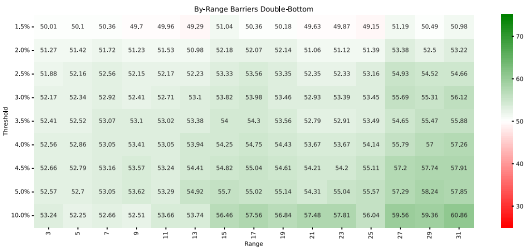


Figure 8: Double bottom winstpercentage

Double Top Patroon: Tegen onze verwachtingen in toonde dit patroon (Figuur 9) regelmatig een winstpercentage dat de 50% overtrof. Hoewel significant tot een drempel van 4%, wijst dit op een mogelijke afname van de effectiviteit bij hogere drempelwaarden.

Andere Patronen: Triple bottom, triple top, en head and shoulders leverden geen consistent of significant voordeel op na aanpassing voor meervoudige

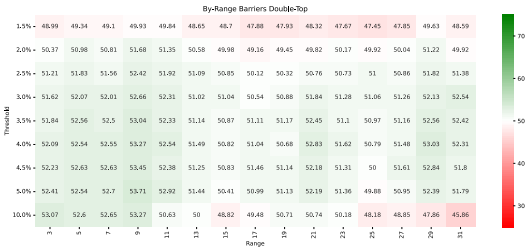


Figure 9: Double top winstpercentage

toetsing. Dit suggereert dat de praktische prestaties van deze patronen kunnen afwijken van hun theoretische verwachtingen.

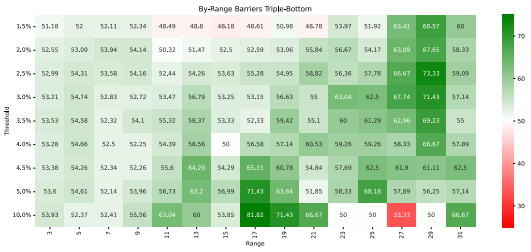


Figure 10: Triple bottom winstpercentage

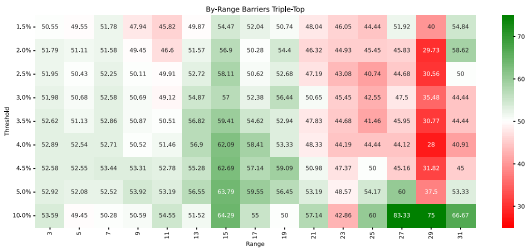


Figure 11: Triple top winstpercentage

Temporele Analyse: Een jaarlijkse analyse toonde aan dat bepaalde patronen, zoals het double bottom patroon, in de loop der tijd een lichte afname in winstpercentages vertoonden (Figuur 13), hoewel deze trend niet als significant werd beschouwd.

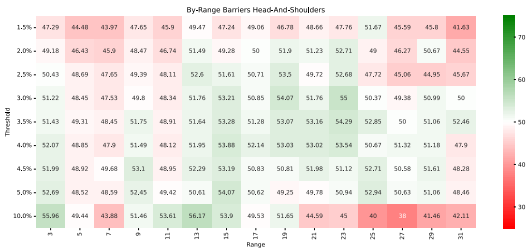


Figure 12: Head and shoulders winstpercentage

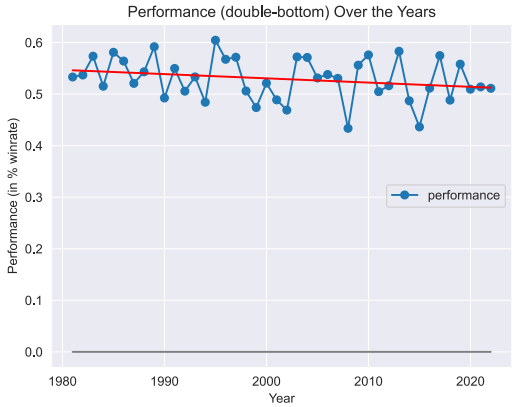


Figure 13: Year-over-year winstpercentage voor double bottom.

6 Conclusie

Onze studie richtte zich op de validatie van gangbare financiële marktpatronen. Hoewel we oorspronkelijk naar machine learning-technieken keken, bracht de praktische uitvoerbaarheid ons naar een scriptgebaseerde algoritmische benadering. De kern van dit onderzoek was een geavanceerde softwarepijplijn die grondige analyses van aandelen datasets mogelijk maakte, gecomplementeerd door een visualisatietool voor intuïtieve gegevensinteractie.

Enkele kernpunten uit onze bevindingen zijn:

1. De effectiviteit van traditionele patronen is niet overtuigend. Bijvoorbeeld, het head and shoulders patroon lijkt gelijkwaardig aan willekeur, en andere patronen zoals double bottom hebben slechts een minimale impact.

2. Patronen behouden grotendeels hun prestatieniveau ondanks technologische vooruitgang en veranderende handelslandschappen. Dit suggereert dat er mogelijk een inherente limiet is aan hun effectiviteit.

Samenvattend suggereren onze bevindingen dat traditionele handelspatronen op zichzelf een minimale impact hebben. Hoewel ze enige waarde kunnen hebben, blijft hun doeltreffendheid beperkt als ze geïsoleerd worden gebruikt. Er zijn aanwijzingen dat het integreren van deze patronen met aanvullende indicatoren, zoals markttrends, hun voorspellende potentieel kan versterken. Het verder onderzoeken en een diepere duik in machine learning-technieken belooft boeiende perspectieven voor vervolgonderzoek.

7 Bibliografie

1. Murphy JJ. Technical analysis of the financial markets: Study guide: A comprehensive guide to trading methods and applications. 2nd ed. Welwyn Garden City, England: NYIF; 1998.

2. Park CH, Irwin SH. What do we know about the profitability of technical analysis? Journal of Economic Surveys [Internet]. 2007;21(4):786–826. Available from: http://dx.doi.org/10.1111/j.1467-6419.2007.00519.x

3. Farias Nazário RT, Silva JL e, Sobreiro VA, Kimura H. A literature review of technical analysis on stock markets. The Quarterly review of economics and finance: journal of the Midwest Economics Association [Internet]. 2017;66:115–26. Available from: http://dx.doi.org/10.1016/j.qref.2017.01.014

4. Bulkowski TN. Trading classic chart patterns. Nashville, TN: John Wiley & Sons; 2002.

5. Murphy JJ. The visual investor: How to spot market trends. 2nd ed. Murphy JJ, editor. Chichester, England: John Wiley & Sons; 2009.

6. Tsinaslanidis P, Guijarro F. What makes trading strategies based on chart pattern recognition profitable? Expert systems [Internet]. 2021;38(5). Available from: http://dx.doi.org/10.1111/exsy.12596

7. Lento C, Gradojevic N. The profitability of technical trading rules: A combined signal approach. Journal of Applied Business Research (JABR) [Internet]. 2011;23(1). Available from: http://dx.doi.org/10.19030/jabr.v23i1.1405

8. Lopez de Prado M. Advances in financial machine learning. Nashville, TN: John Wiley & Sons; 2018.

9. Binance - cryptocurrency exchange for bitcoin, ethereum & altcoins [Internet]. Binance. Available from: https://www.binance.com

10. Prices EH, Api FD. Fundamental, EOD historical prices and financial data API [Internet]. Eodhd.com. Available from: https://eodhd.com

11. Kan R, Kirikos G, Halpern P, Andrew G. Biases in evaluating trading strategies [Internet]. Utoronto.ca. Available from: https://www-2.rotman.utoronto.ca/~kan/papers/mom1.pdf

12. Apache SparkTM - unified engine for large-scale data analytics [Internet]. Apache.org. Available from: https://spark.apache.org

13. Salah T, Jamal Zemerly M, Yeun CY, Al-Qutayri M, Al-Hammadi Y. The evolution of distributed systems towards microservices architecture. In: 2016 11th international conference for internet technology and secured transactions (ICITST). IEEE; 2016. p. 318–25.

14. React [Internet]. React.dev. Available from: https://react.dev

15. Free charting library by TradingView [Internet]. TradingView. Available from: https://www.tradingview.com/HTML5-stock-forex-bitcoin-charting-library/

Preface

I want to begin by thanking both Prof. Dr. Yvan Saeys and Arne Gevaert, who provided their expertise and guidance on this thesis and continued to support me throughout the year. Given that this thesis began as a proposal, I'm deeply grateful to have been given this opportunity to work on it.

I'd also like to thank my friends and family for their support: from cooking in the evenings to standing by me during good and buggy times.

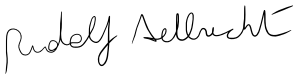
The subject of this thesis has been on my mind for years, but never had the skills or time to execute properly. Being able to write this thesis with the languages, tools, and knowledge I've built up over the years, truly feels like the high point of my academic journey.

This thesis delves into a substantial amount of economic terminology. However, explanations are available in the background section. While the foundation of this thesis is computer science, those from an economic background, possessing a basic grasp of programming, should be able to follow the majority of the content.

Toelating tot bruikleen

De auteur geeft de toelating deze masterproef voor consultatie beschikbaar te stellen en delen van de masterproef te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de bepalingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze masterproef.

Rudolf Aelbrecht, 24 Augustus 2023



Contents

Summary **ii**

Samenvatting **iii**

Extended Abstract **iv**

Preface **xx**

Toelating tot bruikleen **xxi**

1 Introduction **1**

2 Background **3**
2.1 Terminology 3
2.2 The Stock Market and Patterns 11

3 Literature Review **14**
3.1 Patterns of the Stock Market 14
3.2 Predictability of Patterns 17
3.3 Detecting Patterns 18
3.3.1 Traditional vs. Machine Learning Approaches 18
3.3.2 Approaches Within Machine Learning 18
3.3.3 Common Pitfalls 21
3.3.4 Insights from Reviewed Papers 22
3.3.5 Conclusion 23

4 Prerequisite Work **25**
4.1 Data Collection and Preparation 25
4.1.1 Data Sources 26
4.1.2 Data Selection 26
4.1.3 Data Preprocessing 28
4.2 Infrastructure development 28
4.2.1 Distributing work between micro-services 29
4.2.2 Formatting candles and derivatives 30
4.2.3 Passing data between micro-services 31
4.3 Visualisation development 34

4.3.1 Framework choice . . . 34
4.3.2 The chart component . . . 35
4.3.3 Fetching candles and indicators . . . 36
4.3.4 Plotting candles . . . 38
4.3.5 Pointer events . . . 39
4.3.6 Drawing of patterns . . . 40
4.3.7 Summary . . . 40

5 Processing **42**
5.1 Candle processor . . . 42
5.1.1 Data acquisition and storage . . . 43
5.1.2 Candle processing and transformation . . . 45
5.1.3 Candle blocks . . . 45
5.1.4 Transition candle blocks . . . 47
5.1.5 Splits . . . 52
5.1.6 Caching . . . 54
5.2 Indicator processor . . . 54
5.2.1 Definitions . . . 55
5.2.2 Expressions . . . 55
5.2.3 Candle Logistics . . . 56
5.2.4 Warmup . . . 57
5.2.5 Caching . . . 59
5.2.6 Transition blocks . . . 59
5.3 Event processor . . . 60
5.3.1 Event processor server . . . 60
5.3.2 Event Processing Client Framework . . . 61
5.3.3 Script Execution and Event Detection . . . 65

6 Detecting Patterns **70**
6.1 Approach 1: Clustering-based pattern detection . . . 70
6.1.1 Feature extraction using pattern detection . . . 71
6.1.2 Pattern formation using event sequences . . . 71
6.1.3 Calculating distances between patterns . . . 72
6.1.4 Bottom-up clustering . . . 73
6.1.5 Top-down clustering . . . 74
6.1.6 Optimisations . . . 74
6.1.7 Limitations and decision to change approach . . . 75
6.2 Approach 2: Script-based pattern detection . . . 76
6.2.1 Dealing with time warping . . . 77
6.2.2 Highs and lows detection . . . 78
6.2.3 Pattern detection based on highs and lows . . . 80

7 Results and discussion **86**
7.1 Overview of Approaches . . . 86
7.1.1 Approach 1: Clustering-based pattern detection . . . 86
7.1.2 Approach 2: Script-based pattern detection . . . 87
7.2 Methodologies and Metrics . . . 88

7.2.1 Labeling Methods: Triple Barrier vs Others 88
7.2.2 Selection and Justification of Performance Metric 91
7.2.3 Understanding the Timeout Limit 92
7.2.4 Event Frequency Analysis: Validating Validation Scripts 94
7.3 Data Attributes and Analysis 97
7.3.1 Skewness in Random Data 97
7.3.2 Statistical Analysis: Significance Testing 98
7.4 Performance Comparison and Interpretation 100
7.4.1 Performance Metrics and Comparison Tables 100
7.4.2 Observing Trends Over Time 105

8 Conclusions and Future Work **109**

A Appendix **112**
A.1 Dataset Information 120

References **121**

Chapter 1

Introduction

Since the inception of the stock market, traders consistently seek methods to anticipate and capitalize on market movements. They develop numerous strategies to help forecast these movements, with two prominent techniques being technical analysis and fundamental analysis.

Technical analysis involves visually recognizing trends and patterns in market data, often depicted in forms such as candlestick charts or other graphical representations [1]. Traders who utilize this method hold the belief that historical pricing and volume trends can forecast future market behavior [2]. Various approaches, including the application of mathematical and statistical methods, serve as key tools in this identification process, contributing to what is known as quantitative analysis.

Contrastingly, fundamental analysis focuses more on the intrinsic quality of the company, examining factors such as the company's financial health, industry position, and market competition. Fundamental analysts believe that through careful evaluation of a company's fundamentals, they can estimate the intrinsic value of the company's stock and thus find opportunities where the current market price does not reflect this value [3].

The emergence of online brokers [4] significantly reduces the barriers to entry for individual retail traders. These newcomers may lack the depth of financial knowledge compared to more experienced investors. As a result, many find it simpler to begin by identifying patterns in candle charts through technical analysis rather than interpreting complex quarterly reports and other fundamental aspects of a company.

In recent years, the interest in trading patterns has significantly grown. This surge has been propelled by the accessibility to assets like cryptocurrencies [5], an endless generation of content on the subject, and the evolution of machine learning algorithms [6], providing innovative ways to identify and analyze patterns.

Using patterns that have historically proven effective without the need for additional research appears like an obvious strategy to many retail traders. However, there is a considerable amount of criticism suggesting that relying solely on technical analysis equates to a coin toss, as patterns often encompass the move they are trying to predict. Additionally, it is part of human nature to perceive patterns where there are none. Therefore, it is challenging to assert whether technical analysis is effective, and further research is required [7].

Public opinions on technical analysis are mixed, and so is the perspective in scholarly literature. While some studies claim that technical analysis is a valid strategy, numerous others argue against its viability. An additional difficulty lies in verifying these claims, as studies often fall short in terms of the size of their dataset, disclosure of data leakage, and providing a clear explanation of their methodology. These gaps in the literature may introduce biases into their findings, complicating the process of drawing definitive conclusions [8].

To address these limitations, we examine trading patterns over the past several decades on a large set of stocks by identifying common patterns, assessing their prevalence, and evaluating their predictive power. The questions we therefore aim to answer are (1) How can we more effectively identify common trading patterns using advanced machine learning techniques? (2) Do these patterns have any predictive power? (3) Has the predictive power changed over time with the prevalence of machine learning?

By answering these questions, we aim to evaluate the effectiveness of patterns in trading strategies, determine the optimal approach for identifying patterns, and lay the groundwork for future research by building tools that simplify the processing of stock data. By addressing the limitations in existing literature, this research contributes to the ongoing debate on whether trading patterns are a useful tool in trading strategies for both retail traders and advanced traders alike.

The structure of the thesis is as follows:

- In Chapter 2, we provide background information to give more context to the workings of financial markets and their terminology.
- In Chapter 3, we discuss existing literature and its conclusions.
- In Chapter 4, we explain the origin of the data, its limitations, and the necessary processing.
- In Chapter 5, we offer a detailed overview of how data is processed to make it fast and accessible for analysis.
- In Chapter 6, we present the methodology for identifying patterns using clustering and algorithmic approaches.
- In Chapter 7, we present the results from the identification and evaluation chapters.
- In Chapter 8, we conclude and discuss future work.

Chapter 2

Background

In this chapter, we will provide an overview of the key concepts and terminology used in financial markets. This information is intended for readers that have no or little knowledge about the stock market and its terminology, to better understand the context and significance of the work being done in this thesis.

First, we will introduce all the basic terminology associated with the stock market. In the second section, we will explain the importance of trading patterns and their role in day-to-day trading strategies.

2.1 Terminology

Trading and investing

Trading and investing often interchange in discussions about financial markets; however, they do not share the same meaning [9]. Trading typically involves the buying and selling of financial assets, like stocks, bonds, or commodities, with the aim of profiting from short-term price movements.

On the other hand, investing focuses on long-term strategies where an individual buys and holds an asset for an extended period. In long-term strategies, individuals hope to see assets appreciate over time, reducing the risk associated with short-term fluctuations by spending time in the market rather than timing the market. The focus is on profit through a gradual increase in the asset's value rather than short-term price fluctuations.

Different individuals use both strategies in different financial markets. More often, traders analyze patterns and movements in assets to decide whether to buy or sell. Meanwhile, investors concentrate on the fundamentals of an asset, including its financial health, earnings, revenue growth, and dividend yield, among other factors.

As we analyze trading patterns centered around market movements, we take the perspective of a trader.

Stock market, stock exchanges, and market indices

The stock market is a marketplace where individuals and institutions can buy or sell publicly

traded companies. Companies sell shares to raise capital, and as the value of a company increases, so does the price of its stocks, resulting in a return for the buyer of the shares. The price of a share is determined by the last transaction price, often speculating more on a company's future performance than its present situation.

Stock exchanges are organizations that list and facilitate the buying and selling of stocks, providing a marketplace for buyers and sellers. Today, this marketplace is completely digital. Well-known stock exchanges include the NASDAQ, NYSE, and Euronext. However, for this thesis, we focus on stocks from the NYSE.

Market indices represent a specific category of stocks, often measuring performance. These indices commonly group stocks in a certain sector or based on shared traits, like the market value of a company. The most well-known market indices include the S&P500, featuring the 500 largest stocks in the United States, and the DJIA, which contains 30 large publicly traded companies. These lists are curated and not exact representations of the largest US companies, but they provide a good benchmark for the stock market's performance. Often, beating one of these indices is considered challenging, and thus they provide a reliable benchmark to compare an investor's portfolio performance.

Stock splits

A stock split is a corporate action where a company increases the number of its outstanding shares by dividing each existing share. For instance, in a 2-for-1 stock split, each share of the company's stock is divided into two. This action effectively doubles the number of shares while simultaneously halving the price of each share. Importantly, a stock split does not change the company's total market value, as the increase in shares is offset by the decrease in price.

Stock splits are often carried out by companies with a high share price to make their shares more affordable and attractive to retail investors. For instance, if a company's shares are trading at $200, a 2-for-1 stock split would reduce the share price to $100. Post-split, investors can buy twice as many shares for the same total investment.

Technical analysis and quantitative analysis

Technical analysis involves the use of charts, volume movements, and other tools to anticipate future price movements [2]. It assumes market behavior repeats in the form of trends or patterns. Technical analysis frequently bases itself on indicators derived from market price or volume or patterns plotted on the candle chart.

Quantitative analysis employs mathematical and statistical models to analyze financial data [10]. It differs from technical analysis as it fully relies on algorithms to identify relationships and patterns in data. These algorithms can often be complex and reveal patterns that are not easily observable through traditional analysis.

Both technical and quantitative analysis complement each other but can also be used independently or combined to propose different trading strategies. However, due to the advanced mathematical and computational skills required for quantitative analysis, it is less popular among retail traders.

Support and resistance

Support and resistance are concepts that identify potential price levels in the market where

significant buying or selling pressure might exist [11]. When the price rises but gets stuck at a certain level, this level is termed a resistance level. When the price falls but does not go below a point due to buying pressure, this level is known as a support level (Figure 2.1).

Figure 2.1: Green zones represent resistance, due to selling pressure. Red zones represent support due to buying pressure. These are not exact lines, and the levels are not strictly horizontal.

Using technical analysis, it is possible to speculate about these levels in advance, allowing a trader to buy in zones just above the buying pressure, optimizing the price at which they enter the market.

Candlesticks

Candlestick charts often serve to show the historical price movements of assets. They display the open, high, low, and close price for a given time interval, represented as a candle. By using candles, more information can be presented without the need to plot multiple series.

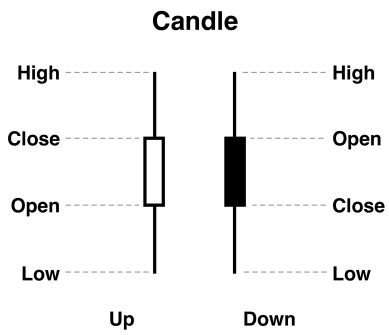


Figure 2.2: A candle represents 5 datapoints, the color of the body represent whether the candle closed higher or lower than its open price.

Candlesticks are depicted with a wick, representing the range between the lower and upper

prices, and a body that indicates the opening and closing prices (Figure 2.2). While the choice of colors for the candles can vary, they are commonly represented as green (or white) and red (or black). In this context, a white candlestick indicates a closing price above its opening price, akin to a green candle. Conversely, a black candlestick signifies a closing price below its opening, similar to the traditional red candle.

In this thesis, candlestick charts are presented using a black and white color scheme. This choice is made for several reasons: to ensure aesthetic consistency across all figures, and importantly, to avoid the emotional connotations tied to colors like red and green. Red, often associated with losses or downturns, and green, linked with gains or positive movement, can introduce biases in interpretation. By using a neutral black and white scheme, we circumvent these associations, emphasizing that patterns can manifest in either direction and that a downward movement does not inherently signify a loss. Within this framework, a white candlestick indicates a closing price above its opening, whereas a black candlestick signifies a closing price below its opening.

Trading patterns

1. **Candle Formations**: Within the framework of trading wisdom, specific candlestick formations are often suggested to represent indicators of short-term trader sentiment. These patterns can emerge from a single candle or a short sequence of consecutive candles. For example, formations such as the Bullish Engulfing, Hammer, Three White Soldiers, and Morning Doji Star are frequently discussed as indicative of certain market behaviors. The Hammer is commonly viewed as a sign of a potential bullish turn, while the Morning Doji Star might indicate growing positivity during a downtrend, hinting at a possible upward move. It's crucial to note that such interpretations greatly rely on the surrounding context of previous price movements (Figure 2.3).

2. **Series of Candles (Technical Analysis Patterns)**: Patterns over a series of candles are common in trading literature. Often relying on trendlines, support, or resistance levels to define their shape, these patterns provide insights into potential future price directions.

   - **Double Bottom/Top**: This pattern bounces twice from a support (for Double Bottom) or resistance (for Double Top) line, signaling a potential reversal (Figure 2.5). The double bottom indicates a bullish reversal following a downtrend, whereas the double top suggests a bearish reversal after an uptrend. Variants include the triple bottom/top.

   - **Head and Shoulders**: Indicating potential reversals, this pattern consists of three peaks: a higher peak (head) flanked by two lower peaks (shoulders) (Figure 2.4). The key is a return to the baseline after each peak, signaling a reversal from the prevailing trend.

   - **Others**: Other notable examples of patterns in trading literature include Trendlines, which connect price highs or lows to indicate support or resistance levels; Wedges, which hint at narrowing price movements and potential reversals; and Flags, short-lived patterns that suggest continuation after a significant price move and consolidation.

3. **Generic Patterns from Machine Learning**: Beyond traditional technical analy-

sis, machine learning provides tools to identify and predict price movements based on patterns that might not be evident to the human eye. By feeding historical price and trading volume data into algorithms, machine learning can discern recurring patterns and make predictions. Unlike traditional patterns which have established names and definitions, machine learning might identify 'generic' patterns based purely on mathematical relationships in the data. These patterns do not have set names or narratives but can be crucial for predictive analytics, especially when dealing with large and complex datasets.

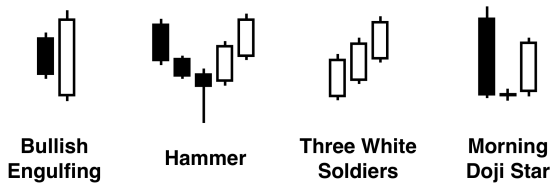


Figure 2.3: Candle patterns based on a short sequence of candles.

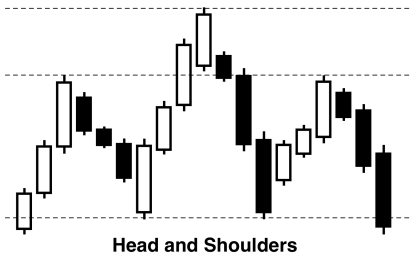


Figure 2.4: Head and shoulders pattern, characterized by a peak (head) flanked by two lower peaks (shoulders), indicating a potential trend reversal.

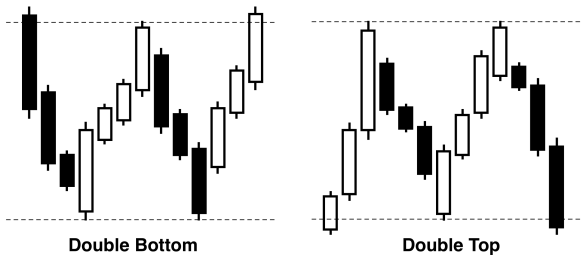


Figure 2.5: Double top and double bottom patterns, characterized by touching the same price range twice before definitively reversing the trend.

Technical indicators

Technical indicators are mathematical calculations based on an asset's price and, at times, its volume. They are used in technical analysis to provide insights into future price movements. Depending on the type of technical indicator, it can be plotted on top of the candle chart or presented as a separate chart.

A common example of a technical indicator is moving averages, which represent the average price calculated over a specific amount of time. These can be used to identify trend changes or

potential support and resistance levels in the market. One particular type of moving average, known as the Simple Moving Average (SMA), is calculated based on a set number of recent candles.

Figure 2.6: An example of the SMA.

The Exponential Moving Average (EMA) is a type of moving average [12] that gives more weight to recent price data (Figure 2.7), which makes it respond more significantly to price changes than a simple moving average. It is often used to gauge short-term trends, providing traders with a quicker signal for buy and sell opportunities.

Another technical indicator is the Relative Strength Index (RSI), which measures the speed and change of price movements and is often used to identify overbought or oversold conditions in a market [13]. An RSI value above 70 suggests a potential overbought condition (indicating a potential price drop), whereas an RSI below 30 indicates a potential oversold condition (suggesting a potential price increase).

Bollinger Bands (BB) is another notable technical indicator, which is based on moving averages and standard deviations. It consists of an upper band, lower band, and a middle band (usually the simple moving average) [14]. The bands expand and contract based on market volatility and can be used to identify potential reversal points or overbought/oversold conditions. Since BB relies on other indicators like moving averages, it provides a layered analytical approach, reflecting both trend and volatility information.

Figure 2.7: Two Exponential Moving Averages (EMA) being drawn alongside candles. Each EMA has a certain range for which it calculates the moving average, which is in this case 12 and 50 candles.

Figure 2.8: RSI showing whether a stock is overbought on a scale from 0 to 100.

Figure 2.9: BB representing the upper and lower band, showing a number based on the standard deviation.

2.2 The Stock Market and Patterns

A brief history of the stock market

The history of the stock market dates back to the 17th century with the establishment of the first modern stock exchange in Amsterdam. At that point, the market primarily traded shares of the Dutch East India Company. As stock markets developed globally, individuals began to invest, aiding the growth of the industrial revolution. The New York Stock Exchange (NYSE), founded in 1792 in the United States, became increasingly accessible with technological advancements such as the telegraph [15].

Throughout the 20th century, the stock market sees significant volatility and speculation. Multiple market crashes during this period expose the inherent instability of financial markets, challenging their perceived role as the economy's pillars [16]. Despite these crashes, the stock market continues to be a crucial part of the global financial infrastructure, enabling millions to invest in publicly traded companies and other investment opportunities. The advent of online trading platforms and other financial technologies further democratizes market access, allowing individuals to invest with any amount of capital [4].

However, the rise of automated trading introduces new challenges in predicting market movements. As high-frequency trading and algorithmic strategies become dominant in the market, it does not necessarily make traditional patterns and trends less reliable; rather, it challenges the fairness of the market, potentially contributing to increased price volatility and decreased effectiveness of price discovery based on firm fundamentals [17]. This shift leads to increased market complexity and emphasizes the need for investors to carefully consider their strategies before participating in the market.

The use of technology in modern trading is unavoidable, offering faster and more efficient trade execution, improved market liquidity, and increased price transparency [18]. However, it also receives criticism for potentially contributing to market volatility and for creating an uneven playing field. High-frequency traders with access to advanced technology and market data may have an advantage over smaller investors. Regardless, technology has democratized the trading landscape, affording retail traders access to tools and information that were once the exclusive province of institutions.

Technology transforms the way that trading is conducted in financial markets, and its continued evolution will likely shape the industry's future.

Market participants

Many participants populate the financial markets, ranging from large-scale institutions investing on behalf of their clients to individuals who trade in their spare time. For clarity, we can categorize these participants into two primary groups:

- Retail traders: These are individual investors often not associated with trading at any professional level. They typically invest smaller amounts of capital.
- Institutional investors: These entities invest large sums, often have easier access to market information, and include pension funds, mutual funds, insurance companies, and endowments.

Other significant participants, like banks, exist, though they are not typically classified as

institutional investors. Banks play an essential role in financial markets, often engaging in large-scale, long-term investments using their own capital. Additionally, banks provide various services such as issuing securities to brokers or retail traders.

Despite the dominance of large-scale institutional investors and high-frequency traders in the market, small traders, often referred to as retail traders, can still find opportunities to make profits. There are several reasons for this:

- Agility: Retail traders are typically more flexible and faster to react to market changes. They do not have the burden of moving large volumes of shares, which can be time-consuming and impact the market price.
- Technology and Information Access: In today's digital age, retail traders have access to the same high-quality information and analysis tools that were once only available to institutional investors. This levels the playing field, enabling retail traders to make informed decisions based on real-time data and sophisticated analysis tools.
- Diversity of Strategies: Unlike large funds, retail traders are not tied to a single investment strategy. They have the freedom to explore different strategies, markets, and timeframes that might be unsuitable for larger players due to their size and regulations.
- Niche Market Opportunities: Retail traders can often exploit opportunities in smaller, less liquid markets that might be overlooked by larger players.

While it is crucial to acknowledge the inherent risks involved in trading financial markets and the need for skill, knowledge, and discipline to achieve success, it is important to recognize that retail traders still possess a chance, albeit small, to surpass benchmarks. Although their size is comparatively modest, they play a substantial and expanding role within the market ecosystem.

The concept of market efficiency and the Efficient Market Hypothesis

The Efficient Market Hypothesis (EMH) suggests that all available information is incorporated into asset prices in an efficient market, so they always reflect their "fair value." [19]. The EMH describes three levels of market efficiency:

- Weak form efficiency: This level suggests that current prices include all past trading information. As a result, any analysis would not give an investor an advantage over others, and techniques like technical analysis would not consistently yield profit.
- Semi-strong form efficiency: At this level, all public information, not just past trading data, is reflected in asset prices. Therefore, neither technical nor fundamental analysis would consistently provide a competitive edge.
- Strong form efficiency: This is the highest level, where all information - both public and private - is factored into asset prices. Consequently, no investor could gain an advantage over others, even with insider information.

The EMH calls into question the effectiveness of trading patterns. If markets are efficient, and all information is already reflected in asset prices, then trading patterns should not provide any useful insights for predicting future prices. If such patterns did exist and were spotted by traders, the ensuing trades would quickly cancel out the pattern by adjusting the price of the asset.

However, the EMH is not without its critics [20]. There are instances of market anomalies and significant mispricing, suggesting that markets might not always be perfectly efficient. As

a result, some people believe that patterns can form, can be spotted, and can potentially be used to predict future price movements.

In summary, the effectiveness and profitability of using patterns in trading largely hinge on one's belief in the degree of market efficiency. This topic is the subject of ongoing debate and research, and it plays a key role in many trading strategies.

The psychology of trading and behavioral finance

The psychology of trading and behavioral finance offers a counter-perspective to the Efficient Market Hypothesis. This perspective suggests that cognitive biases and herd behavior might manifest as patterns in asset prices [21].

Traders and investors, like all individuals, are influenced by cognitive biases—systematic errors in thinking that impact the decisions and judgments they make [22]. For instance, traders might align with a trend due to the "herd mentality" bias, a psychological phenomenon where individuals tend to follow the actions of a larger group, irrespective of the rationality of those actions [23]. Such biases can intensify price movements, resulting in observable market patterns.

Another cognitive trap is the "confirmation bias." Here, traders might lean towards information that aligns with their existing beliefs, overlooking data that might challenge them [24]. This tendency could lead traders to remain in a position based on their belief in a market rebound or exit a position early due to fears of a downturn. Such behavioral inclinations might be reflected in asset price patterns.

The myriad of such psychological phenomena gave birth to behavioral finance, a discipline that delves into how cognitive biases shape financial markets. Unlike traditional finance theories that bank on rational decision-making, behavioral finance underscores the undeniable influence of psychological factors on market dynamics.

Due to cognitive biases and herd behavior, patterns might emerge and persist [22]. This is often evident in speculative markets like cryptocurrencies or in volatile sectors like biotech, where price shifts often seem more rooted in sentiment and speculation than in concrete financial metrics [25]. In these arenas, traders who believe they discern these patterns might see them as indicators of potential market movement. However, it is essential to recognize that such patterns are unpredictable, and their mere presence does not guarantee trading outcomes.

In conclusion, the psychology of trading and behavioral finance offers insights into why and how patterns might appear in financial markets, enriching the understanding and adding depth to trading and investment discussions.

Chapter 3

Literature Review

Technical analysis, a method predicting future price movements based on historical price and volume data, has been extensively discussed in both academic and applied financial literature. One of the core components of technical analysis is the use of chart patterns. These patterns, which appear as specific geometrical configurations on price charts, are theorized to be indicative of potential future market directions. Given the extensive claims about these patterns, this chapter seeks to critically assess the literature surrounding them.

While a vast array of patterns is identified and utilized by analysts and traders, this review will concentrate on a select few that consistently appear in foundational literature: the double bottom, double top, and head and shoulders patterns. For instance, such patterns have been comprehensively discussed in Bulkowski's "Trading Classic Chart Patterns"[26].

A critical objective of this chapter is to explore the empirical validity of these patterns. The research questions guiding this inquiry include: What evidence exists in academic literature supporting the predictive validity of these patterns? How are these patterns being detected in research, and with what outcomes?

By narrowing the focus to these foundational patterns, this chapter aims to provide an understanding of their place within the broader framework of technical analysis.

3.1 Patterns of the Stock Market

Despite the debates on its efficacy, it cannot be denied that a significant body of literature has been dedicated to the domain of technical analysis over the decades. This section undertakes a review of the most popular books on technical analysis, summarizing their thoughts on patterns in general, with a specific focus on their interpretations of double bottom/top and head and shoulders patterns. These particular patterns were chosen as they are discussed in every book in this category of literature. The aim is not to validate or refute the principles of technical analysis but to understand the sentiment and insights expressed by the authors in a very condensed form, given the vast scope of these comprehensive works.

Getting Started in Technical Analysis (1999) [27]

"Getting Started in Technical Analysis" introduces key technical patterns in market analysis,

such as the 'double tops and bottoms' and the 'head and shoulders.' These patterns are known for their potential to indicate trend reversals, especially if they appear after significant price movements.

'Double tops and bottoms' are described as two peaks or troughs, potentially signaling a trend reversal after a significant trend, though they may vary in depth and require careful observation. The 'head and shoulders' formation is notable for its three-peak structure, where penetrating the 'neckline' is considered a significant trend reversal indicator.

The book emphasizes the importance of preceding price actions and warns against prematurely identifying these patterns without considering the broader context, underscoring that patterns can be predictive but not in isolation. When correctly identified and in the right context, these patterns can be powerful tools for traders. However, misidentification can lead to misguided trading decisions, so patience, context, and precision are essential.

Encyclopedia of Chart Patterns (2005) [28]

The "Encyclopedia of Chart Patterns" is a comprehensive guide to various chart patterns in markets, including the pattern known as the Adam & Adam Double Bottom, which signals bullish trend reversals.

The Adam & Adam Double Bottom pattern is characterized by two sharp V-shaped troughs roughly at the same price level. This formation, particularly with specific volume characteristics, can be a robust indicator of a bullish trend reversal. The encyclopedia emphasizes that declining volume trends or heavier volume on the left bottom tend to make the pattern more reliable, adding a layer of validation beyond price.

It also notes the importance of waiting for a clear breakout above the resistance level formed between the two troughs before making a trading decision, as this patience can prevent false alarms and enhance trading accuracy.

The Encyclopedia recognizes that while patterns like the Adam & Adam Double Bottom are valuable predictors, they are not without fail. Attention to volume nuances and waiting for a breakout can increase the success rate of trades based on this pattern.

How to Make Money in Stocks (2002) [29]

"How to Make Money in Stocks" explores the details of various stock patterns, discussing the "double bottom" and "head and shoulders" formations among others.

The double bottom pattern, likened to the letter "W", consists of two troughs, with the second undercutting the first. This act is seen as crucial, shaking out weak investors and paving the way for a potential uptrend. The book praises the head and shoulders pattern for its reliability in predicting stock peaks, but critiques the head and shoulders bottom pattern, suggesting it lacks a strong prior uptrend, necessary for dominant market leaders.

Emphasizing the importance of discernment in pattern recognition, the book advises traders to be judicious in their interpretations, highlighting that understanding the nuances can make the difference in trading success. While acknowledging certain patterns as more reliable, it stresses the importance of traders' ability to discern subtleties, suggesting that patterns can be indicative but must be interpreted with care.

Technical Analysis of Stock Trends (1987) [30]

This foundational book on technical analysis serves as a standard reference for traders and analysts, explaining the principles and patterns of the field and updating them to align with modern financial markets.

The "double bottom" and "double top" are described as mirror reflections of one another, acting as reversal patterns that signal potential trend changes. The Head-and-Shoulders pattern is explored, emphasizing the difference in formation between Tops and Bottoms, with the latter often more drawn out and rounded. Volume dynamics are highlighted as essential in validating a pattern's authenticity; a breakout, whether upward or downward, is often coupled with increased volume, lending support to the potential trend change.

Emphasizing that patterns alone are not sufficient, the book asserts that accompanying volume is vital for confirming a pattern's reliability, making a pattern without the correct volume dynamics less trustworthy.

The Art and Science of Technical Analysis (2012) [31]

"The Art and Science of Technical Analysis" explores the blend of empirical and intuitive aspects of technical analysis, emphasizing the importance of both methodical and perceptual understanding for predicting market trends.

Patterns are described as more than mere chart figures; they symbolize underlying market sentiments and weave a complex picture of market movements. Significant deviations from a trend may signal market recalibrations or impending changes, but the book also acknowledges times when markets act unpredictably, with patterns becoming confusing. This complexity underscores the market's inherent unpredictability.

While recognizing patterns' role in providing insights into market direction, the book encourages traders to balance reliance on patterns with adaptability to market complexities, promoting a more nuanced approach.

The Visual Investor (2009) [32]

"The Visual Investor" serves as a guide to interpreting stock charts and understanding the implications of visual patterns for investment decisions.

The book distinguishes between a pullback, a temporary reversal that goes against the prevailing trend, and a genuine double top, a reversal pattern where the price stops rising and starts descending. This distinction is marked by the breaking of a sequence of rising peaks and troughs.

Triple tops and bottoms are explained as extensions of double top and bottom patterns but are viewed as stronger reversal signals due to their repeated testing of resistance or support levels. The widely recognized Head and Shoulders pattern is discussed, with particular attention on volume dynamics, typically diminishing as the pattern progresses and spiking during the breakout.

"The Visual Investor" emphasizes the importance of recognizing patterns and their nuances, debunking common misconceptions, and accentuating the role of volume in validation, suggesting that patterns alone may not always be definitive.

Trading Classic Chart Patterns (2002)

"Trading Classic Chart Patterns" provides an in-depth examination of common chart formations and their variations, focusing on the subtleties that can refine interpretations.

The book describes variations of the double top pattern based on the peaks' profiles: Adam & Adam, Adam & Eve, Eve & Adam, and Eve & Eve. These variations detail the characteristics of the peaks, with "Adam" being sharp and jagged and "Eve" more rounded, aiding in a more precise understanding of the pattern.

The Head-and-Shoulders pattern is analyzed thoroughly, stressing the importance of symmetry and the slope of the neckline in determining the strength and direction of a potential breakout or breakdown.

Emphasizing the need for clarity in pattern recognition, the book warns against premature or over-interpretation, insisting on meeting specific criteria to validate a pattern.

Summary

Upon review of the aforementioned texts on technical analysis, several consistent themes emerge. Primarily, the authors highlight that price patterns like the double bottom/top and head and shoulders are more than mere graphical configurations; they encapsulate market dynamics and reflect underlying sentiments and trends. While acknowledged as potentially powerful predictive indicators, these patterns necessitate nuanced interpretation, taking into account factors such as volume dynamics and broader market trends. There's a recurrent emphasis across these works on the indispensable nature of contextual analysis, cautioning against isolating patterns from their broader market contexts. They advocate for recognition of patterns within a comprehensive understanding of market intricacies, underscoring the importance of discernment and adaptability in utilizing these tools for financial decision-making.

3.2 Predictability of Patterns

Technical analysis has historically played a pivotal role in forecasting price movements, but its efficacy remains a subject of ongoing debate. A comprehensive study "A literature review of technical analysis on stock markets" [33] examined the state of the field, offering insights that delineate both its strengths and existing gaps.

Economic Context: One noteworthy observation is the disproportionate focus on advanced economies. Of the 85 articles surveyed, 53 pertained to stock markets in advanced countries, leaving a mere 23 to address developing economies. This imbalance reveals a discernible gap in the literature, indicating a dearth of explorations into the efficacy of technical analysis within developing or emerging economies, such as the BRICS nations.

Methodological Divergences: The methodologies deployed in these studies further underscore the disparities in scholarly focus. Among the surveyed works, chart patterns were scarcely represented, making appearances in a mere 2 of the 85 papers. This indicates a potential avenue for future research, particularly in examining whether the integration of chart patterns with advanced pattern recognition algorithms might enhance the forecast of abnormal returns.

Operational Tools: A closer inspection of the analytical tools revealed a pronounced inclination towards moving averages and statistical analyses. However, a conspicuous gap exists in

merging technical analysis with modern analytical methodologies. The potential for integrating technical analysis with tools such as machine learning remains largely unexplored and presents a promising avenue for future studies.

Profitability, Predictability, and Risk are three foundational aspects of technical analysis, with risk-adjusted return being a key component. Risk-adjusted return is a measure that takes into account both the profit and the level of risk involved in achieving that profit, providing a more comprehensive view of an investment's performance. Many studies support the benefits of technical analysis, but the omission of risk-adjusted returns in some research raises concerns. By neglecting this critical measure, the studies may present an incomplete picture of the investment's true potential and thus cast doubt on the validity of their findings. Additionally, the potential for publication bias may lead to an overrepresentation of studies confirming the effectiveness of technical patterns, thereby skewering the overall perception of the utility of technical analysis.

In summary, while there exists a robust body of literature attesting to the merits of technical analysis, this survey highlights pressing gaps in the field. The exploration and empirical validation of foundational chart patterns remain untapped areas.

3.3 Detecting Patterns

3.3.1 Traditional vs. Machine Learning Approaches

Technical analysis, often known as "charting," has been a long-standing method in financial practices. It involves studying price chart patterns to predict future prices. However, one of the main criticisms of this method is its subjectivity, as interpretations can differ among users.

The paper "Foundations of Technical Analysis" [34] addresses this issue by suggesting a more structured and automated way to recognize technical patterns. The researchers use nonparametric kernel regression, a modern statistical method, to spot patterns in price time series. This move towards automation is inspired by successes in other areas like fingerprint and face recognition, where advanced statistical methods are used.

The study makes several key points. Firstly, while some classic patterns like the head and shoulders are informative, they might not be the most efficient. There could be other "optimal patterns" better suited to find specific patterns in financial data. However, it is essential to note that the best patterns for finding unusual data might not be the same as those for making the most money in trading.

In summary, while traditional technical analysis has its place and has shown value over the years, combining it with machine learning and updated statistical methods could be the way forward. This combination could make technical analysis more accurate and possibly more profitable.

3.3.2 Approaches Within Machine Learning

3.3.2.1 Dynamic Time Warping

Dynamic Time Warping (DTW) serves as a sophisticated algorithm for assessing similarity between temporal sequences, which may vary in length. Within the financial sector, its

utility has been recognized for analyzing sequences in financial time series, such as stock price movements.

In the study titled "What makes trading strategies based on chart pattern recognition profitable?" [35], the primary focus was on the development of a trading system that leverages DTW to identify generic patterns that have historically shown profitability. Contrary to traditional trading approaches that are anchored to predetermined technical patterns, this methodology adopted a more expansive view. The research encompassed an analysis of 560 NYSE stocks, meticulously evaluating the system's efficacy in relation to factors such as pattern length and the defined take-profit and stop-loss levels.

The principal takeaways from the research encompass several distinct insights. First, the study showcased the viability of trading systems that expand beyond traditional boundaries, identifying any chart pattern historically tied to profitability rather than relying solely on predetermined technical patterns. Next, the research established a clear correlation between the length of the pattern and the efficacy of the trading system, suggesting that more elongated patterns might serve as more robust indicators. Additionally, the 'take-profit' metric was found to be vital in enhancing system performance, whereas the 'stop-loss' metric had a more subdued impact. Finally, the system, after adjusting for transaction costs, generated modest but consistent returns, surpassing conventional benchmarks when the optimal parameters were applied. Together, these findings underline the innovative application of Dynamic Time Warping (DTW) in trading strategies, illustrating the potential of this novel approach.

To summarize, the research accentuates the potential of integrating dynamic time warping in the formulation of trading strategies. With the recognition of any historically profitable pattern and not solely relying on established ones, the study presents a novel approach.

3.3.2.2 Neural Networks and Deep Learning

The study "Stock Price Prediction Incorporating Market Style Clustering" [36] presents a methodical examination of the influence of market styles on stock price prediction. Recognizing the distinct behaviors of stocks under different market conditions, the research emphasizes the integration of these styles into forecasting models for better accuracy.

To explore this, the research followed a systematic approach:

Data Segregation: Stock time series data were divided into various window lengths, processed with technical indicators and news sentiment features.

Clustering: Hierarchical clustering was applied to categorize these windows into market styles, with a focus on optimizing window lengths and the number of styles.

Distance Measurement: A new distance metric was developed to identify recurring patterns within the established market styles.

Prediction Framework: Based on the identified market styles, a forecasting model was formulated to predict stock price trajectories.

Drawing on five years of data from the Hong Kong Stock Exchange, including stock prices and related news, the study uncovered several key insights. Firstly, the consistent presence of market styles across various time windows suggests their utility in enhancing stock forecasting

models. Secondly, the research demonstrated the ability of a market style from one day to predict the style for the next day. With the integration of market styles, there was a significant improvement in prediction accuracy, with gains of up to 9% over baseline models. Lastly, results from backtesting highlighted potential profit increases when integrating market styles into trading signals, further emphasizing the value of this approach.

In conclusion, the research effectively demonstrates the potential of market styles in refining stock price predictions. The outlined methodology, from data processing to clustering and its integration into forecasting models, provides a foundation for future research. This could include a more detailed exploration of market styles using diverse data sources and understanding the inherent properties of market style clustering.

3.3.2.3 Other Machine Learning Techniques

The paper "Clustering of time series data - a survey" [37] consolidates extensive research on time series clustering, emphasizing its applicability and versatility across numerous domains. This systematic review investigates different methodologies and assumptions researchers have adopted when clustering time series data and the inherent challenges and potential research directions.

From the review, the following insights are drawn:

Data Assumptions and Limitations:

- A majority of the research predominantly deals with stationary time series, with few exceptions.
- Notable works address specific assumptions, such as discrete-valued time series data, data errors, or unevenly sampled data.
- A gap is evident in studies involving multivariate time series with diverse lengths for each variable.

Multivariate Time Series Studies:

- Certain research either overlooks cross-correlation between variables or assumes conditional independence.

Complementary Techniques:

- Methods like change-point detection, can pinpoint the beginning of time series, aiding in subsequent analysis.

Differentiating Time Series from Static Data Clustering:

- The distinction primarily rests on deriving suitable similarity or dissimilarity measures, which is crucial for clustering time series data effectively.

Research Opportunities:

- Modern clustering techniques, such as genetic algorithms, remain underutilized.
- There's potential in integrating both unsupervised clustering and supervised classification approaches.
- While ensembles of clustering methods might offer superior results, challenges like inconsistent labeling persist.

- A clear need emerges for algorithms designed expressly for time series clustering, especially as most scaled-up techniques cater predominantly to static data.

Temporal Knowledge Discovery:

- Despite its significance, time series clustering appears underrepresented in existing surveys, highlighting the need for more exhaustive research.

In summary, these papers provide an overview of methodologies, challenges, and potentials in clustering time series data. These findings can be applied when implementing a possible machine learning approach that makes use of stock data represented as time series.

3.3.3 Common Pitfalls

When integrating machine learning into the technical pattern detection landscape, it is pivotal to navigate pitfalls and intricacies:

1. **Overfitting on Historical Data:**
   - Machine learning models trained excessively on past patterns can become too tailored to historical nuances, compromising their ability to generalize and predict future patterns effectively.
2. **Blind Reliance on Established Algorithms:**
   - Just as established patterns might not guarantee results, solely leaning on traditional algorithms can neglect potential advances in the machine learning domain.
3. **Ignoring Feature Engineering:**
   - The success of machine learning in pattern detection is closely tied to the quality of features used. Overlooking the importance of relevant feature extraction and engineering can lead to suboptimal performance.
4. **Data Inconsistencies and Anomalies:**
   - Inconsistent or anomalous data can mislead algorithms, leading to incorrect pattern identifications. Continuous data validation and preprocessing are essential.
5. **Static Model Approaches:**
   - Markets evolve, and so should machine learning models. Relying on static models without periodic updates and refinements can cause them to become outdated over time.
6. **Over-reliance on Single Metrics:**
   - While accuracy might seem a primary goal, other metrics like precision, recall, or F1-score could provide better insights, especially in imbalanced datasets commonly seen in financial time series.
7. **Neglecting Model Interpretability:**

- Especially in financial domains, understanding why a model makes a decision can be as crucial as the decision itself. Ensuring models are interpretable can aid in trusting and refining the patterns they detect.

In summary, while machine learning offers promising tools for pattern detection in financial data, it is imperative to approach the task with a comprehensive understanding of both the data's intricacies and the nuances of algorithmic modeling.

3.3.4 Insights from Reviewed Papers

3.3.4.1 Generic vs Specific Patterns

In the corpus of literature examined for this thesis, chart patterns frequently surface, underscoring their importance in trading strategies. Broadly, these patterns can be categorized into two main groups:

1. **Traditional Patterns (Sourced from Established Literature)**

   Originating from academic research and established trading guidelines, these patterns represent configurations that have been consistently observed over time. Common examples include:

   - *Head and Shoulders,* indicative of a trend reversal.
   - *Double Top/Bottom,* signifying a reversal of the current trend.
   - *Other patterns,* presented in various forms like "diamonds" and "triangles", each with specific market implications.

   The reliability of these patterns is attributed to their historical validation. However, rigorous academic investigations into these patterns have been limited, with only a select number of papers examining their profitability.

2. **Novel Patterns (Identified through Machine Learning Techniques)**

   With continued research into financial markets, new techniques for pattern detection are introduced, challenging traditional notions of patterns that have been documented for decades. Methods such as Dynamic Time Warping (DTW) and various machine learning techniques are instrumental in identifying these patterns. Among this category are the so-called 'Generic Patterns'. Unlike traditional patterns, these do not have established definitions but are recognized based on their recurrent and profitable manifestations in financial markets.

To summarize, traditional patterns offer a historical foundation, while the discovery of novel patterns, facilitated by tools like DTW and diverse machine learning approaches, indicates a promising avenue for future research and exploration.

3.3.4.2 Evaluating Trading Strategies

In the realm of finance, the efficacy of trading strategies can be affected by various factors ranging from transaction costs to the intricacies of algorithm design. When developing and evaluating these strategies, it is imperative to consider both the tangible costs of trading and the methodologies employed for data labeling in algorithm-driven strategies. This section delves into the significance of transaction costs as highlighted by "The Profitability

Of Technical Trading Rules: A Combined Signal Approach," [38] and further explores the innovative labeling techniques, especially the Triple-Barrier Method from the book "Advances in Financial Machine Learning," [39] which are pivotal in assessing the performance of trading algorithms and strategies.

Transaction Costs and Their Role in Strategy Evaluation:

Transaction costs, often underestimated, can drastically shift the profitability landscape of trading strategies. As revealed in "The Profitability Of Technical Trading Rules: A Combined Signal Approach," even established technical trading rules may falter when these costs are factored in. For indices such as the S&P/TSX 300 Index and NASDAQ Composite Index, only certain rules like the moving average cross-over maintained profitability post the inclusion of transaction costs. The study further emphasized the potential of a combined rule approach, which often surpassed individual rules and consistently outperformed the buy-and-hold strategy across assessed indices.

It is noteworthy that these observations may be influenced by external factors like market events within the data's time span (1995-2004). Therefore, extending the study across diverse and longer durations can provide a more comprehensive understanding of the strategy's robustness in different scenarios.

Advanced Labeling Techniques in Trading Algorithms:

In the realm of algorithmic trading, especially when using machine learning, the way data points are labeled is crucial. A standout technique for this task is the Triple-Barrier Method. This method labels data based on three barriers: two horizontal barriers set by profit-taking and stop-loss limits, and a vertical barrier determined by the time passed since the start of a position. What makes this method unique is its path-dependent nature. This means it requires examining the entire sequence of market movements from when a position is opened until one of the barriers is touched. By doing so, the algorithm can be finely attuned to market shifts. We discuss this method in more detail in Chapter 8.

Additionally, this method aids in determining both the direction (or side) and the magnitude (or size) of a trade. In situations where the direction of a trade is unclear, the algorithm can be adapted to predict both the direction and size simultaneously, optimizing its performance in diverse market conditions.

Conclusion:

When constructing or assessing a trading strategy, it is clear that there's no one-size-fits-all approach to evaluating performance. The landscape is broad, with considerations ranging from transaction costs to the intricacies of algorithmic methodologies. Novel techniques, such as the Triple-Barrier Method, bring fresh perspectives to the table to labeling in the context of machine learning. These evolving methods underscore the importance of remaining adaptable and open to diverse approaches.

3.3.5 Conclusion

Technical analysis has long been a tool for predicting market movements. Highlighted by patterns like double tops, bottoms, and the head and shoulders, these patterns give insights

into market feelings but need careful interpretation. A key lesson from these sources is to look at patterns as part of the bigger market story.

However, the actual power of technical analysis remains a topic of discussion. Silva, Sobreiro, and Kimura's 2017 review shows this, pointing out that most research focuses on developed countries, often leaving out the developing ones. Also, while many studies use tools like moving averages, not many have explored the potential of newer methods like machine learning. The limited studies on chart patterns and the absence of risk-adjusted returns in some research show there's more work to be done.

Modern techniques, such as statistical tools and machine learning, are enhancing traditional chart analysis. The paper "Foundations of Technical Analysis" recommends integrating classic charting methods with machine learning to identify patterns previously overlooked by conventional approaches.

Dynamic Time Warping (DTW) has shown its value, especially in research about profitable chart-based trading strategies. In another direction, a study on the Hong Kong Stock Exchange reveals the importance of understanding market styles for better stock prediction. The HSMM-CP model stands out for its skill in identifying complex financial patterns.

The paper "Clustering of time series data" provides a comprehensive review of time series clustering methodologies, highlighting different data handling and modeling assumptions across studies. It underscores the unique challenges of time series data, introduces complementary techniques, and points to potential areas of future research, emphasizing the need for specialized algorithms tailored for such data.

However, mixing machine learning with financial pattern spotting has its challenges. Overfitting, where models become too focused on past data, is a major concern. Other issues include depending too much on old algorithms and unexpected data changes. As finance continues to embrace machine learning, being aware of these challenges is important.

An interesting tool in the world of machine learning-driven trading is the Triple-Barrier Method. This approach uses three criteria to label data: profit levels, stop-loss levels, and how long a position has been open. What makes it stand out is its focus on the entire market movement from when a trade starts until a certain event happens. It helps algorithms adjust to market shifts, improving trade predictions.

To conclude, the literature highlights the importance of chart patterns, both old and new. Traditional patterns have a history of trust, while new patterns discovered through modern tools point to fresh opportunities. When looking at trading strategies, it is crucial to consider everything from transaction costs to how data is labeled. These findings will be instrumental in subsequent chapters as we delve into the detection of patterns employing machine learning techniques.

Chapter 4

Prerequisite Work

To focus solely on the detection part of finding patterns in stock data, a significant amount of prerequisite work is needed. This foundational work improves the efficiency and speed of data retrieval, where speed refers to the quick access and processing of data. Running a local platform eliminates the constant need to contact APIs for data retrieval and allows us to concentrate on making detection methods easily explainable, without worrying excessively about the noise or complexity inherent in the data [40]. This chapter covers many aspects of this foundational work, including collection, processing, tool development, and visualisation.

All the services described in this chapter are developed in the Go programming language [41], chosen for its ability to easily parallelize work and provide simple memory management through its performant garbage collector. This approach helps save a significant amount of time where performance would usually come at the cost of debugging, and vice versa.

The infrastructure itself is developed around the idea of a microservices architecture. A web of interdependent microservices allows for the separation of each role in the preprocessing pipeline into its own service. By adopting this pipeline approach, we significantly reduce the risk of fallout, as this allows for the completion and testing of one service without impacting the rest of the project. In real-world scenarios, such an approach has already been observed to help reduce factors like technical debt and improve development velocity over time [42]. This underlines the practical relevance of the decision to adopt microservices in the context of this thesis.[1]

4.1 Data Collection and Preparation

In the analysis of patterns, we exclusively utilize stocks and their splits. This approach aligns with the concept of technical analysis, which focuses solely on the visual aspect of a chart. However, despite the core analysis relying solely on stock prices, we incorporate a company's market capitalization as part of its fundamentals when selecting the dataset. This consideration ensures a more nuanced selection process, reflecting the company's overall value

[1] All source code, tools, and software projects associated with this chapter are publicly available on GitHub at https://github.com/charting-wall-street.

in the market. In this section, we will explore the collection, selection, and preprocessing of the raw data.

4.1.1 Data Sources

During the development of the tools, two data sources are utilized. One data source is EOD Historical Data [43], which provides the necessary information for stocks, while the other is Binance [44], which offers candle data for cryptocurrencies.

Initially, cryptocurrency data is used for the development process due to its high availability, high resolution, and cost-free access. Binance provides a well-documented API that enables the retrieval of data from the start of its inception. However, this data, due to its inherent volatility, may pose challenges in reliably detecting patterns. Nonetheless, it should be noted that higher volatility does not necessarily imply fewer returns [45]. In fact, the heightened volatility might suggest that patterns could manifest more prominently and could be more effective. Additionally, a significant amount of historical data is required to assess the feasibility of patterns on large time frames, which is lacking in the cryptocurrency market due to its relative infancy. Therefore, while it might be that patterns actually work better with this higher volatility, it is a subject that warrants deeper exploration. Consequently, the decision is made not to explore this dataset further in the current context, but it remains an avenue for potential future research.

To acquire stock market data, many turn to publicly available resources. However, in the experience, these were found to be both impractical and time-consuming to extract, particularly given the limitations many services impose on the amount of historical data accessible. While it is entirely feasible to construct a similar dataset at no cost from online resources, and evidence suggests this can yield data of nearly the same quality [46], in the context, turning to a third-party service was more expedient. To this end, we chose EOD Historical Data, a paid platform offered by Unicorn. This platform granted us access to decades of historical stock data and company fundamentals, streamlining the data collection process without any additional effort on the part.

The stock data includes end-of-day (EOD) information, such as candles, splits, and historical company fundamentals. Candles and splits are used to calculate stock prices in the charts, while company fundamentals assist in determining the worth of companies. Leveraging these resources, a comprehensive dataset is obtained, enabling accurate pattern detection in stock market data. This dataset forms the foundation for conducting analyses and investigations in the research project.

4.1.2 Data Selection

Numerous scientific papers generally assess their trading strategies using a limited set of stocks in a limited amount of time. The selection of these stocks is often influenced by their contemporary popularity or historical significance, leading to inherent bias [47]. A majority of these chosen stocks are typically successful companies that have weathered various economic trials over the past decades. As a result, investing in such stocks has historically resulted in profits due to their average upward trajectory.

Apple exemplifies a stock that is frequently utilized in these evaluations. Its trajectory of

success is unparalleled, recently underscored by its achievement of surpassing the 3 trillion dollar valuation mark, making it the most valuable company in the world at this point in time. However, it is crucial to recognize that very few corporations have attained or sustained such monumental success as Apple. By focusing algorithms predominantly on stocks like Apple, there's an inadvertent bias introduced, favoring consistently stable and prosperous companies that appear unlikely to experience downturns in the foreseeable future.

To diminish this bias, the goal is to generate a comprehensive catalogue of publicly traded stocks. This method permits the recognition of patterns in both small and large corporations. Still, the selection procedure is not entirely free from bias. To exclude graphs with excessive noise, companies with a market capitalization below $200 million at the time of writing this paper are left out. Stocks with a long history are also given precedence to facilitate a more comprehensive evaluation. As a result, the dataset naturally encompasses stocks mainly linked to successful companies. Nonetheless, it is believed that this approach offers a more representative view of the market, and trading strategies with robust risk management can still exploit patterns observed in high-value company stocks, potentially yielding superior results than when relying on a restricted dataset.

While excluding smaller companies, it is impractical to test the algorithm on every single stock that has ever existed. Including all US-traded symbols would result in a list exceeding 10,000 symbols spanning various kinds of stocks, funds, and financial assets. Since the primary focus lies on common stocks, symbols that do not represent common stocks are filtered out. Moreover, derivative stocks, often denoted by letters like X or U, are excluded to further refine the list.

After reviewing random stocks from the residual list, it becomes clear that many of these are small firms with charts that lack value due to low trading volume or limited historical data. To tackle this issue, the list is refined to only include NYSE symbols, thus eliminating smaller companies. This measure trims the list down to roughly 6,000 publicly traded symbols.

A list of 6,000 stocks still proves unwieldy for thorough evaluation and may include smaller companies with unusual charts resulting from irregular splits or occurrences like share creation to uphold specific price levels. The primary interest lies in patterns emerging from market trading activities. To further streamline the list, market capitalization data for each company is collected for each year of their existence. This information is used to prune the list based on company valuation by choosing the 250 largest companies for each quarter.

The final filtering criterion pertains to the duration of existence. Several companies in the dataset might have only operated for a few years, making their long-term performance evaluation challenging. To address this issue, any company that does not boast at least 20 years of historical data is discarded. This refinement procedure yields a final list of approximately 500 symbols[2], which are used as the dataset for pattern detection.

While the data quality has been broadly evaluated, there might still be some gaps in the completeness of the dataset. Particularly, data points at the beginning of each stock's timeline may lack the same depth as more recent data due to the digitization process. However, since

[2]The final list of symbols used, together with the last retrieval date, can be found in the appendix. Since we cut off our statistics at the end of 2022, the exact retrieval date should not impact the reproducibility of the study.

the objective is to evaluate patterns over a significant timeframe, these points are included to gain insights into historical patterns.

Moreover, it is crucial to recognize that the chosen selection of the dataset introduces a potential bias known as survivorship bias [48]. By focusing on stocks that have a long history of existence, stocks that may have ceased trading or experienced substantial declines are unintentionally excluded. As a result, the dataset primarily consists of stocks that have persisted and flourished over time. This survivorship bias should be considered when interpreting the results and generalizing them to the broader stock market.

In conclusion, the methodology involves choosing the 250 largest companies each quarter from 1986 to 2022 and filtering out companies with a market capitalization below $200 million or less than 20 years of historical data. This stringent filtering process culminates in a dataset comprising just over 500 symbols, serving as the bedrock for detecting patterns in the research. By focusing on substantial and well-established companies, the aim is to procure a comprehensive and representative dataset for accurate pattern analysis.

4.1.3 Data Preprocessing

Upon data download, the processing phase commences. The sparsity of stock market data is notable, as it lacks night data, weekends, or holidays. To streamline the analysis, a focus is placed exclusively on daily candles, thereby circumventing complexities associated with trading day start and end times. In order to facilitate indexing and plotting, any gaps arising from closed market days are filled with empty candles, clearly labeled as missing. As a result, all data is organized and stored in an array, enabling convenient access through timestamp-based indexing.

As stocks may have distinct start times, loading an entire company's historical data into memory for every instance of its requirement can impose substantial overhead. To mitigate this challenge, the data is partitioned into blocks, with each block comprising 5000 candles. These blocks can be stored separately, either on disk or in memory, as individual data entries. Although storing a few megabytes of historical data in this manner may appear excessive, it proves essential for efficient access to assets when working with minute data or simultaneously processing multiple stocks.

4.2 Infrastructure development

Rather than relying on existing scripts or available pipelining tools such as Apache Spark [49], Apache Kafka [50], and traditional ETL (Extract, Transform, Load) solutions, the choice has been made to develop a specialized pipeline for processing stock data. These aforementioned tools, while powerful and versatile, may introduce complexities and overheads not necessarily optimized for the specific use-case. This decision is driven by the requirement to swiftly obtain the essential data for the subsequent analysis stages of the research. The custom pipeline has been meticulously designed to optimize performance through the effective utilization of caching and processing methods, which will be elaborated upon in the forthcoming section.

4.2.1 Distributing work between micro-services

A microservices architecture is employed to distribute the workload across specialized services [51]. This approach enables quick modifications and restarts of individual services without losing any in-memory data stored by other services. Moreover, it facilitates streamlined testing, as services can be promptly patched and restarted in the event of a crash. This flexibility is crucial for rapid evaluation of new features and components intended for integration into the pipeline.

The pipeline consists of three main services and one helper service (Figure 4.1):

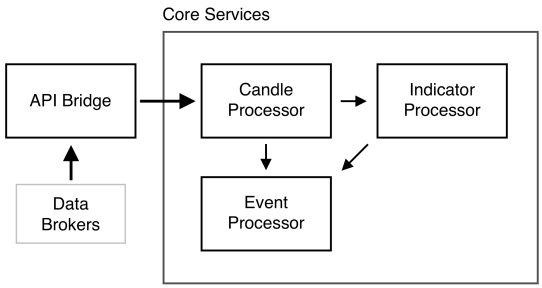


Figure 4.1: An overview of the services and their dataflow directions. Core services are exposed, and are used for visualizations and evaluating the patterns.

1. Bridge Service: Acts as an interface between external APIs and internal services.
2. Candle Processor: Stores and processes candle data, providing it in multiple time intervals.
3. Indicator Processor: Processes candles to generate commonly used market indicators.
4. Event Processor: Manages and executes algorithm scripts, eliminating the need for manual startup.

During the development of the candle and indicator services, it became evident that storing intermediate results would play a pivotal role, especially given the challenges posed by microservice data management, including ensuring data consistency and efficiently querying database states [52]. Initially, caching was implemented to store blocks of candles loaded from disk. However, due to the static nature of historical data, caching could be safely incorporated at each stage of derivative calculation, optimizing performance.

To optimize caching, we employ a medium-sized cache that follows the least-recently used principle. This cache efficiently stores a variety of data, such as raw blocks of candles, blocks of candles calculated for different time intervals, and blocks of indicators, along with their respective parameters and exchange metadata. By including frequently used metadata, we can more easily determine the start and end dates when needed.

To further enhance efficiency, service requests for data are designed to avoid redundant requests. When a service makes a request, it prevents other threads from making the same request concurrently. Once the request is fulfilled, the data is retained in memory to minimize the overhead of redundant data requests. Reducing the number of data exchanges significantly improves performance, as the primary overhead in calculating algorithms and indicators lies in data exchange.

For instance, caches are implemented at both endpoints, minimizing traffic and reducing overhead associated with data requests beyond the initial request. This approach is inspired by the practices employed in web caching, where strategic document caching is essential to decrease download times and reduce network traffic [53]. In much the same way as web caching policies are designed to adapt to changing workloads, the caching strategy in this research has been tailored to respond to dynamic demands and characteristics of the stock data workload.

4.2.2 Formatting candles and derivatives

The initial concept of dividing the dataset into blocks was inspired by the common practice of chunked data requests over HTTP, aiming to optimize both storage and retrieval processes. By partitioning data into blocks, it aligns with the typical nature of web requests, where data is fetched or sent in batches rather than as continuous streams. Consequently, creating blocks consisting of 5,000 candles not only streamlines the storage and retrieval processes but also results in a more manageable payload size. This smaller, block-sized payload can then be efficiently sent over HTTP to the web client, allowing for smoother rendering of each candle on a canvas.

The expansion to include indicators not only leverages the existing candle drawing logic but also the block-based data structuring. When we structure the indicators using consistent blocks of 5,000 candles with the same time offset, we gain the advantage of flexibility in interval scaling. Essentially, if each block represents a 1-minute interval, combining five such blocks gives a 5-minute interval, and further merging results in extended intervals like 10 minutes, 30 minutes, and so forth. This block merging approach offers a systematic method for scaling, but it comes with a limitation: it does not perfectly align with natural time divisions like months or years, as these are not strictly based on fixed-length periods.

While the block merging approach offers systematic flexibility, using Unix timestamps as the starting point introduces its own set of complexities. Data predating the Unix epoch can result in negative timestamps. When these timestamps are divided by the number of candles to determine the block index, any value between -1 and 0 gets rounded to 0, courtesy of integer division logic. This rounding forces us to implement extra checks. A potential remedy might be to introduce a large offset to the Unix timestamp. Though such a deviation from standard practice might not be conventional, its utility is evident and may merit exploration in future iterations or platform reworks.

In the endeavor to maintain consistency across all Go projects, we established distinct packages specifically for containers related to candles and indicators. This structure not only ensures easy access to universal utility functions across services but also refines the operational flow. With this, both candle and indicator blocks can seamlessly represent data from diverse assets. We've incorporated serialization functions directly into the package, enhancing the efficiency of data storage and retrieval, whether from local storage or over the network. For data types, we followed conventions outlined in both the Binance API and EOD API, which employ $float64$ types for most fields, except for timestamps and indices. This adherence simplifies the data integration process, ensuring compatibility and coherence with established data formats.

The adoption of candle blocks significantly enhances the speed of calculating indicators and working with candle data. By eliminating this bottleneck and standardizing the structure in the pipeline, we can concentrate on the actual processing of the data.

4.2.3 Passing data between micro-services

This thesis describes several microservices that employ a consistent data exchange structure. The services utilize HTTP requests for interaction, both from client applications to services and between services themselves. The decision to use HTTP, rather than alternative technologies such as web frameworks, was motivated by the desire to avoid complexities associated with data exchange. Specifically, the choice of HTTP was driven by the availability of visualization and debugging tools that support this protocol.

The visualization tool, which is an integral part of the system, also relies on HTTP, enabling the straightforward implementation of a fetcher to retrieve data from the services. This facilitates result verification for each service without requiring substantial additional effort to implement receiver and sender logic. However, it should be noted that the tools developed in this thesis are not suitable for production environments where service endpoints may be publicly accessible. This limitation arises from the fact that the services were not evaluated for security and lack measures to prevent abuse.

To handle HTTP requests efficiently, two publicly available libraries were utilized. The first library, gorilla/mux [54], serves as a request router, allowing developers to define and handle complex routing patterns and URL parameters in web applications. The second library, urfave/negroni [55], acts as middleware, providing a modular and user-friendly framework for managing HTTP middleware functions. The gorilla/mux library is employed to configure endpoints and handle tasks such as Cross-Origin Resource Sharing (CORS) required for communication between the visualization tool and the services. Negroni, on the other hand, is used primarily for error recovery to prevent server crashes caused by panics in request routines.

To simplify data exchange, structs were created and shared between services through centralized packages containing their definitions. This approach enables straightforward encoding and decoding of data streams based on the known struct being passed. Depending on the "Accept" header in the HTTP request, the response is sent in either JSON or binary format. The binary format was introduced to eliminate the overhead associated with JSON decoding [56]. By default, services communicate with each other using binary encoding, while the visualization tool exclusively uses JSON since it serves as an auxiliary tool where performance can be sacrificed. The adoption of binary encoding reduces the overhead introduced by JSON encoding.

In most cases, Go's gob package [57] is utilized for binary encoding. This package offers the capability to encode and decode values in a compact binary format, which facilitates data serialization for transmission over networks or for file storage. When encoding candles using the gob package, the overhead is reduced to mere single milliseconds, as opposed to multiple milliseconds. However, an inherent inefficiency emerges when one needs to extract specific data points: the entire block must be retrieved each time. This process, when repeated multiple times to extract a few candles from different blocks, underscores the fact that the gob package, while efficient, still introduces discernible overhead. To address this inefficiency, a custom binary encoder and decoder were developed. It is important to highlight that binary encoding is solely used for encoding candle and indicator blocks, as they represent the majority of the data exchanged between services.

The binary encoding structure for candle blocks is illustrated in the following table:

Offset (bytes)	Encoded Data
0	Number of Candles (uint64, big endian)
8	Candle 1 - Open (float64, big endian)
16	Candle 1 - High (float64, big endian)
24	Candle 1 - Low (float64, big endian)
32	Candle 1 - Close (float64, big endian)
40	Candle 1 - Volume (float64, big endian)
48	Candle 1 - TakerVolume (float64, big endian)
56	Candle 1 - NumberOfTrades (uint64, big endian)
64	Candle 1 - Time (uint64, big endian)
72	Candle 1 - Missing (uint8)
...	...
8 + cSize*(i-1)	Candle i - Open (float64, big endian)
16 + cSize*(i-1)	Candle i - High (float64, big endian)
24 + cSize*(i-1)	Candle i - Low (float64, big endian)
32 + cSize*(i-1)	Candle i - Close (float64, big endian)
40 + cSize*(i-1)	Candle i - Volume (float64, big endian)
48 + cSize*(i-1)	Candle i - TakerVolume (float64, big endian)
56 + cSize*(i-1)	Candle i - NumberOfTrades (uint64, big endian)
64 + cSize*(i-1)	Candle i - Time (uint64, big endian)
72 + cSize*(i-1)	Candle i - Missing (uint8)
...	...
8 + cSize*(n-1)	Candle n - Open (float64, big endian)
16 + cSize*(n-1)	Candle n - High (float64, big endian)
24 + cSize*(n-1)	Candle n - Low (float64, big endian)
32 + cSize*(n-1)	Candle n - Close (float64, big endian)
40 + cSize*(n-1)	Candle n - Volume (float64, big endian)
48 + cSize*(n-1)	Candle n - TakerVolume (float64, big endian)
56 + cSize*(n-1)	Candle n - NumberOfTrades (uint64, big endian)
64 + cSize*(n-1)	Candle n - Time (uint64, big endian)
72 + cSize*(n-1)	Candle n - Missing (uint8)
Offset after Candles	Meta Data (serialized using gob)

In testing, when requesting data from the candle service where the requested payload is cached, the average response times were as follows: 25ms using JSON, 12ms using gob, and 5ms using binary encoding (evaluated using Go's built-in benchmarking toolkit).

Further optimization efforts to reduce the remaining 5ms would require significant investment. Within this remaining time, approximately 2ms can be attributed to server response time, 2ms to payload reading, and 1ms to the decoding algorithm. The binary encoding and decoding technique employed in this thesis significantly reduced overhead time, as hundreds of requests are executed per second when requesting results at the end of the pipeline. Although additional optimization may be possible, such efforts are beyond the scope of this thesis.

A similar approach to encoding and decoding data is used for indicators. However, since indicator blocks are exchanged less frequently, a detailed discussion is omitted. The payload

structure for indicators is as follows:

Offset (bytes)	Encoded Data
0-7	Number of series (uint64)
8 - n	Series Data (variable)
	— Series Key
	— Series Kind
	— Series Axis
	— Series Values (Repeated IndicatorValue)
	—— Value (float64)
	—— Missing (bool)
n+1 - end	Meta Data (gob encoded IndicatorMeta)

Although binary encoding effectively reduces overhead, it poses challenges when expanding data structures with additional information. As evident from the encoding table, gob encoding is still used for the metadata section, as the finality of the metadata and the potential for changes remained uncertain. Therefore, binary encoding was selectively implemented where it provided the most significant benefits. In all other data exchanges, JSON or gob encoding is still employed, as developing a custom encoder for each scenario would not be justifiable given the time investment required for implementation and testing.

The use of HTTP requests and the optimizations applied to encoding and decoding processes expose certain system limits, particularly related to file descriptors. When payloads are requested but not properly closed, the corresponding connections remain open for a certain period. As the system makes hundreds to thousands of requests per second, this issue becomes problematic as file descriptors accumulate, waiting for subsequent requests. It was observed that these open TCP connections are not reused when multiple requests are performed in parallel, leading to undesired consequences. The official documentation of the http library states:

"If the returned error is nil, the Response will contain a non-nil Body which the user is expected to close. If the Body is not both read to EOF and closed, the Client's underlying RoundTripper (typically Transport) may not be able to re-use a persistent TCP connection to the server for a subsequent 'keep-alive' request" [58].

To address potential issues with persistent TCP connections, it is essential to ensure that the request body is drained before closure. Neglecting this order can lead to numerous redundant persistent TCP connections.

```go
// Drain any remaining data in the response body
if _, err = io.Copy(io.Discard, resp.Body); err != nil {
    log.Fatal(err)
}
// Close the response body, which should also close the TCP connection
if err = resp.Body.Close(); err != nil {
    log.Fatal(err)
}
```

By addressing file descriptor issues and optimizing JSON serialization overhead, the system overhead has been notably reduced. This optimization paves the way for scaling the pipeline without facing data exchange constraints. Caching, by eliminating repetitive data fetches, further boosts the efficiency of the pipeline. The upcoming chapter on processing delves deeper into caching mechanisms.

To wrap up this section, the detailed exposition of the data exchange methodologies underscores a commitment to both efficiency and adaptability. By carefully selecting encoding techniques and streamlining system interactions, this framework not only ensures the effective support of the subsequent stages of this research but also provides valuable groundwork for real-world scenarios involving the analysis of stock market data.

4.3 Visualisation development

In this section, we present the development process of a custom visualization tool tailored to address the challenges associated with efficiently analyzing candle data, indicators, and patterns in the stock market. The primary objective behind this endeavor was to create a tool that offers improved speed and flexibility while minimizing the programming effort required.

While tools like matplotlib are versatile and widely-used, they were not specifically designed for extensive time series analysis, which can pose challenges when seeking a higher level of interactivity for effective visualization. Advanced platforms such as DXCharts [59] and TradingView [60], although offering more specialized features, come with their own limitations. These tools, often part of larger web platforms, can be behind paywalls and necessitate considerable integration efforts, requiring users to familiarize themselves thoroughly with their APIs and mechanics. Additionally, intermediate solutions may present constraints, such as limiting the number of indicators or the volume of data that can be concurrently visualized, potentially inhibiting a comprehensive analysis of the diverse data patterns we've uncovered.

To overcome these limitations, we decided to develop the visualization tool specifically tailored to the research objectives (Figure 4.2). The focus during the development process was on achieving efficient handling and rendering of large datasets. Through the strategic selection and implementation of algorithms and data structures, we ensured that the tool is capable of efficiently processing and visualizing the extensive amounts of data encountered in the stock market.

In the subsequent sections, the intricacies of the tool's development will be detailed. The aim was to strike a balance between simplicity and functionality, resulting in a tool that stands as a valuable resource for in-depth analysis of stock market data. This tool enables users to efficiently navigate various candle patterns, indicators, and associated trends. Recognizing the limited resources available for visualizing stock market data, the exploration seeks to provide a methodological approach for those in the research community. This can serve as a useful guide for individuals who are in search of a practical solution to visualize and comprehend financial data.

4.3.1 Framework choice

The decision to utilize the React framework [61] for the visualization tool was guided by two key considerations. Firstly, React's component-based architecture enabled us to create

Figure 4.2: Visualization tool. Showing candles for symbol AAPL. Cursor shows the price and time at that point.

interactive user interfaces by leveraging reusable components, reducing the effort required for UI development. This approach facilitated code reusability and streamlined the overall development process.

Secondly, the familiarity and proficiency with React played a significant role in the decision-making process. Drawing upon the existing knowledge, we were able to effectively execute the development process and leverage React's inherent advantages to meet the specific requirements.

React's support for real-time changes during programming, facilitated through mechanisms such as reconciliation [62] and efficient rendering optimizations, proved invaluable. This functionality allowed for instant visibility into updates, minimizing the need for manual refreshes or rebuilds. Consequently, this real-time feedback greatly enhanced the development workflow and productivity.

In summary, the decision to utilize the React framework played a crucial role in the development of the visualization tool. React's emphasis on efficient rendering and the ability to quickly set up the tool were key factors in choosing it as a framework.

4.3.2 The chart component

At the core of the visualization tool is the development of a chart component, which serves as the foundation for the charting functionality. This chart component receives a range of parameters, such as asset type, interval, list of indicators, and other relevant data. By leveraging this component and its flexible parameterization, we are able to generate charts for diverse assets and customize their display based on the specific parameters provided.

It is important to note that the focus in this thesis is primarily on the development and

functionality of the core chart component within the visualization tool. As such, we will not delve into the details of the UI components and their implementation. While UI components play an integral role in the overall user experience, their design and specific functionalities fall outside the scope of this thesis and are not directly relevant to the research objectives. The aim is to provide a comprehensive understanding of the chart component's structure, data handling, and rendering capabilities, which are essential for detecting patterns in the stock market.

At this stage of development, the chart component was implemented as a standard React component, utilizing an empty canvas element to visualize the chart data. React's built-in state variables were initially used to track and manage the component's variables. However, as the model structure stored a significant amount of data, relying solely on React's state management led to memory issues. The sheer volume of data, including candle and indicator data, as well as parameters related to chart position and zoom levels, posed challenges in terms of efficient memory usage within React's state management system. Consequently, we needed to explore alternative approaches to effectively handle the substantial data stored in the model structure.

To effectively manage the substantial data volume and overcome the limitations of React's state management, we implemented a separate model variable as a storage mechanism. This model variable served as a centralized repository for all the variables previously stored as state variables within the chart component.

To ensure synchronization between the model and the parameter changes, we leveraged React's effect hooks [63]. When a variable within the chart component changed, React initiated updates. We captured these updates using the effect handlers provided by React. Subsequently, we used these updates to modify the corresponding variables within the model. This synchronization allowed us to efficiently handle the extensive data volume while also effectively managing memory.

Integrating the model variable ensured that any changes in the model were promptly reflected in the UI. Moreover, by updating only the internal state of the chart component while keeping the rest of the UI updates within React's state management, we adhered to the principles of pure components, as recommended by React's guidelines [64]. This method streamlined the development of an interactive interface, allowing users to effortlessly select symbols and indicators without any performance hindrances.

4.3.3 Fetching candles and indicators

The charting process in the visualization tool commenced by prioritizing the retrieval and display of candle data. Upon the creation of the charting component, we implemented an event loop specifically designed to handle the data-fetching process. This event loop operated within the context of a React effect hook, granting us the flexibility to interrupt the process as needed. In the event that the chart component was destroyed, the event loop would be promptly canceled, ensuring that any pending fetch requests were discarded and preventing unnecessary resource consumption.

Within each iteration of the fetch loop, we identified the required data to be fetched. If any data was determined to be missing, we initiated the fetch process and stored the retrieved data

in the model. Importantly, React remained unaware of these updates, thereby eliminating unnecessary updates to the Document Object Model (DOM). As a result, the canvas, which we had full control over, stood as the singular component within the application that relied upon this fetched data. The chart component is the sole component that relies on the fetched data, and its independent management of the data, separate from React's state management, does not hinder its functionality.

To efficiently manage the data for candle visualization, the system maintains two blocks of candles in memory at all times. Each block consists of 5,000 candles, ensuring that the displayed data never exceeds this limit. We allocate space in the model for these two candle blocks and record the corresponding start and end times. As users adjust the timeline, causing one block to move out of view and another to come into view, we seamlessly swap the data blocks and fetch the new block as required (Figure 4.3). If only one block is visible, we retain it in the first slot of the model. An example of this boundary in blocks can be seen in Figure 4.4.

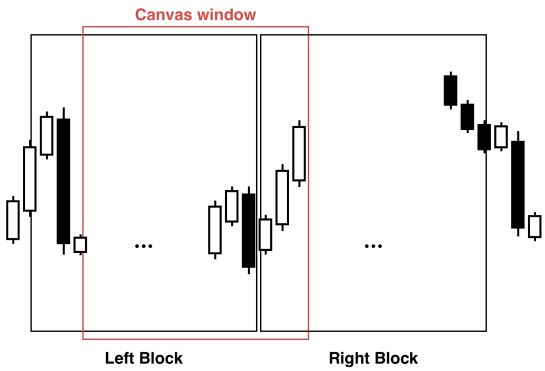


Figure 4.3: Example of how candle blocks can show a snippet of both blocks. The canvas window can at most be the width of a block, so only two blocks are ever needed in memory.

To mitigate excessive server requests caused by rapid symbol switches or chart movements, we implemented a serialized update mechanism for charts. This mechanism ensures that we refrain from fetching another block while one is already being fetched, thereby preventing flooding of requests. Moreover, we introduced a delay mechanism to avoid overwhelming the server with repeated requests in case of temporary unavailability.

The same approach is applied to fetching indicators, incorporating the additional complexity of plotting multiple indicators on a single chart. To handle this, we utilize a map data structure to store the fetched blocks for each indicator. The map is organized based on a unique key generated by concatenating the indicator identifier and its parameters, ensuring a distinct identifier for each indicator. Although this approach may not be recommended in a production environment, we retain all values in the mapping without eviction. However, given that we store a maximum of two blocks per indicator combination, the memory usage remains insignificant for casual chart usage. When transitioning between assets, the indicator blocks are cleared, freeing up memory resources.

Figure 4.4: Example of a boundary in the candle blocks.

4.3.4 Plotting candles

In addition to storing candle and indicator blocks in memory, the model also includes variables for the corresponding start and end times, as well as price bounds. These price bounds are calculated based on the candle data blocks currently stored in memory. If users choose to modify these bounds, they become decoupled from the candle blocks, allowing for independent adjustments.

With a buffer of two blocks that ensure the presence of all candles within the requested interval, we proceed to plot each candle within these blocks while excluding those that fall outside the bounds stored in the model. Determining the first and last visible candles is straightforward since the candle blocks are indexed as arrays. By converting the timestamps to corresponding indices within the blocks, we precisely locate the position of each candle on the chart.

The process of rendering each candle hinges on screen dimensions. The screen width helps define the width of each candle, affecting the portrayal of its body and wick. Additionally, the screen is segmented into slots, each representing a place for a visible candle. The conversion of time positions to slot indices ensures the accuracy of candle placements. The x-coordinate calculation translates time data to screen space. For the y-coordinate, we adapt the candle's price data to the screen's coordinate system, normalizing it to the range of visible prices and the screen height.

Utilizing the calculated x and y coordinates, we are able to determine the precise positions of all candle and indicator points on the chart. To facilitate smooth rendering, we leverage the requestAnimationFrame feature available in most browsers. By passing a function to this feature, the browser autonomously manages the animation loop, rendering frames as needed. This approach ensures that frames are only drawn when the window is visible and adheres to

the device's frame rate.

When we no longer require the chart rendering, such as when we destroy the canvas, we can stop the rendering of frames by utilizing the cancel callback provided by the effect hook. This is similar to how we managed the fetching loop, ensuring efficient resource allocation and preventing unnecessary rendering when not needed.

Using the same process of plotting as candles, indicators are represented by a single data point that corresponds to a specific time and price. Therefore, the drawing of indicators follows an analogous procedure to that of candles. By applying the calculated x and y coordinates, we can accurately position indicator points on the chart, aligning them with their corresponding time and price values. In addition to price-based indicators, there are non-price indicators plotted at the bottom of the chart. These indicators represent independent values and are positioned using the same procedure based on specific time points. However, their y-values are normalized relative to the data within the corresponding indicator block.

With these advancements, we have successfully implemented a basic renderer capable of accurately rendering candles and indicators (Figure 4.5). This renderer allows for the visualization of price movements and indicator trends, providing valuable insights into the analyzed data.

Figure 4.5: Candles being drawn alongside indicators.

4.3.5 Pointer events

To handle pointer events, we utilize the onMouseMove event connected to the canvas element, updating a model variable with mouse-related information on each mouse movement. This enables us to track mouse events and their coordinates accurately. We prioritize implementing UI elements using React and leverage this setup for user interaction, allowing manual chart adjustment by storing the initial position when the mouse is pressed.

For price and time bounds calculations, we employ procedures outlined in Algorithms 1 and 2, respectively. The `movePriceAxisManual` function calculates the new price ranges based on a given vertical movement (deltaY), while the `moveTimeAxis` function adjusts the time range based on a given horizontal movement (deltaX).

Algorithm 1 Manual Price Axis Movement

```
1: procedure movePriceAxisManual(δY)
2:     rangeDelta ← upperPriceRange − lowerPriceRange
3:     step ← rangeDelta/chartHeight
4:     priceLowerRange += δY × step
5:     priceUpperRange += δY × step
6: end procedure
```

Algorithm 2 Time Axis Movement

```
1: procedure moveTimeAxis(δX)
2:     timeStep ← (upperTimeRange − lowerTimeRange)/chartWidth
3:     lowerTimeRange −= δX × timeStep
4:     upperTimeRange −= δX × timeStep
5: end procedure
```

When auto-price adjustment is enabled, we determine the minimum and maximum values of the lows and highs of all visible candles.

4.3.6 Drawing of patterns

In the thesis, we delve into the detection of patterns represented by data containing price and time positions. These patterns are referred to as "events" and are plotted as visual elements such as lines, points, or text. Similar to retrieving candle data, we obtain these event structures and store them in the model of the chart. Instead of organizing them into blocks, we receive a sparse list of events, each containing information about shapes, lines, or labels based on price. Leveraging the understanding of handling candle data, we efficiently determine the screen positions for these event elements. Once the candles and indicators are plotted, we proceed to plot the event shapes onto the chart, showcasing patterns alongside the underlying data (see Figure 4.6).

4.3.7 Summary

The utilization of the visualization tool provides a means to efficiently validate the output of the services. By visualizing the data, we can promptly identify any deviations from expected results and take appropriate actions. The tool's local execution allows us to stream a large volume of real-time candle data without being restricted by the limitations often imposed by online platforms, which may have restrictions on the number of items that can be plotted.

Regarding memory consumption, the tool demonstrates a low amount of resource usage. In Firefox, it only consumes 118MB of system memory when idle, with the two blocks of candles in memory accounting for a mere 1.2MB (Figure 4.7). Although the feature set is minimal

Figure 4.6: Double-top patterns being visualised.

compared to commercial tools like TradingView, the total memory consumption is significantly lower.

Retained Size (Bytes)		Shallow Size (Bytes)		Node
1 366 840	5%	8 248	0%	objects › Object @ 0x116d197a2a50
1 283 536	5%	152	0%	objects › Object @ 0xf80028f21c0
1 282 856	5%	120	0%	objects › Object @ 0xf8002874718
1 282 568	5%	120	0%	objects › Object @ 0xa7b09960f20
641 168	2%	40	0%	objects › Object @ 0x305be4075128
641 144	2%	40	0%	objects › Object @ 0xf80028f35b0
183 064	1%	40	0%	objects › Object @ 0x116d19797ad8
155 536	1%	56	0%	objects › Object @ 0xf800287b3f8
155 536	1%	56	0%	objects › Object @ 0x273f6c5a64d8
117 752	0%	32 792	0%	objects › Object @ 0x116d19798b38
91 680	0%	2 104	0%	objects › Object @ 0x1a78ba65a970
65 272	0%	16 408	0%	objects › Object @ 0x116d19798b50

Figure 4.7: Memory snapshot of objects when displaying two blocks on screen. The 1.2MB is the chart model structure, with each of the 641KB children being a block of candles.

It is important to emphasize that the tool is not intended to replace online platforms or other interactive applications. Instead, it serves as a valuable tool for visualizing results generated through algorithmic approaches. For tasks such as technical analysis (TA), we recommend utilizing online tools or dedicated interactive platforms that are specifically designed for user interaction. These platforms are better suited to meet user needs, offer more comprehensive features, and may include automation functionalities or proprietary code requirements.

Chapter 5

Processing

This chapter focuses on the processing of data retrieved from data brokers through the API bridge. It examines various services in detail and explores how they transform a series of candles into structured blocks of candles and indicators. These transformed data can then be readily accessed via an HTTP request.

Furthermore, caching techniques and the concept of generating transition candles are discussed as means to decrease computational requirements and streamline the complexity within the pipeline.

5.1 Candle processor

The data in this context consists solely of candles, making it crucial to seamlessly provide this data to other services. All services within the pattern processing pipeline make calls to either an intermediate service or directly to the candle service in order to acquire the required data. This positions the candle service as the focal point and the most crucial link in the chain of services.

While all services utilize Go for implementation, the ease with which we could leverage multi-threading in the language was a key factor in this choice, particularly when referencing its design principles [65]. Among these services, the candle service stands out as it particularly benefits from Go's multi-threading features. At any given time, the candle service might be tasked with managing up to 20 simultaneous requests, each querying for distinct candles and time intervals. Operating autonomously from other services, the constraints of the candle service are primarily determined by disk I/O and CPU speed. Given the minimal computational load, this implementation allows for the streaming of thousands of candle blocks every second.

The candle service assumes multiple responsibilities. Firstly, it retrieves and verifies all symbols that require tracking. Subsequently, it maintains up-to-date data for these symbols and offers it in multiple time intervals. Additionally, as services also need information on data availability, the candle service includes the start date of the first candle block in its response. Consequently, all this metadata can be obtained in a single request.

5.1.1 Data acquisition and storage

When the service is initiated, an audit is performed to determine if any data requires updating. The audit process includes downloading any missing candle blocks. The duration of the initial startup time is minimal if the service was offline for a short period. In such cases, the missed blocks from the offline period to the restart are fetched. The feature of retrieving minute data for cryptocurrencies has been successfully implemented on Binance [66]. However, for research experiments focusing on stock data, the live updating feature for the stock market link has not been further developed. If the algorithm detection is employed in production, it becomes necessary to minimize API requests by continuously updating the candle data. This allows for realistic market pattern calculations to be performed every minute.

Even if real-time data is used, it is not relevant to the research objectives. Only daily candles are utilized, eliminating the need to keep the service online continuously.

Once the audit retrieves a candle from the bridge between the data brokers and the candle service, a collection of arbitrary candles is obtained. The only specification is that the candles should begin at a specific point in time. This starting point is chosen based on the size of the candle blocks. The collection of candles is then divided into candle blocks, and the subsequent data in the sequence is requested starting from the end of the last fully retrieved candle block. Some services allow retrieving all candles for the symbol's entire history in a single request, while others impose a limit of 1000 candles. Consequently, multiple requests are necessary to fill a single block. The auditor is responsible for requesting and filling blocks with candles received from the bridge. If the received candles are insufficient to fill a single block, the auditor needs to request multiple times based on the last received candles. In cases where enough candles are returned to fill multiple blocks, they need to be divided accordingly. Additionally, the auditor should account for rate limiting as described in the exchange information of each broker.

Each time a block is retrieved, it is stored on the disk. Once the audit is complete, the web server can read these files from the disk. This approach minimizes memory usage during retrieval and maintains a clear separation between the audit code and the server code. When a block is saved to disk, it is serialized using Go's gob package [57], which simplifies the reading and writing of Go structs. Gob is chosen for its speed and convenience in encoding and decoding data in Go.

A JSON meta file accompanies each binary file, providing basic information. During the audit, the meta files are parsed to determine which blocks have already been fetched and the data they contain. If a block has not been fully retrieved, it is deleted and requested again. This ensures that the service starts with the most up-to-date information.

The storage method for the files on disk is not a significant consideration. Since there is one file for every 5000 candles, even in the worst-case scenario, it poses no issues for the operating system or the auditor. It is even feasible to run the service with a remotely mounted share to store its data. By minimizing I/O operations through design, maintaining a large collection of stocks in this manner incurs only a slight overhead.

To maintain up-to-date data, the same audit code is executed every minute. This approach reduces the number of API requests needed to retrieve a full block. Only the last missing candles are requested, and active candle blocks are stored in memory until they are completed

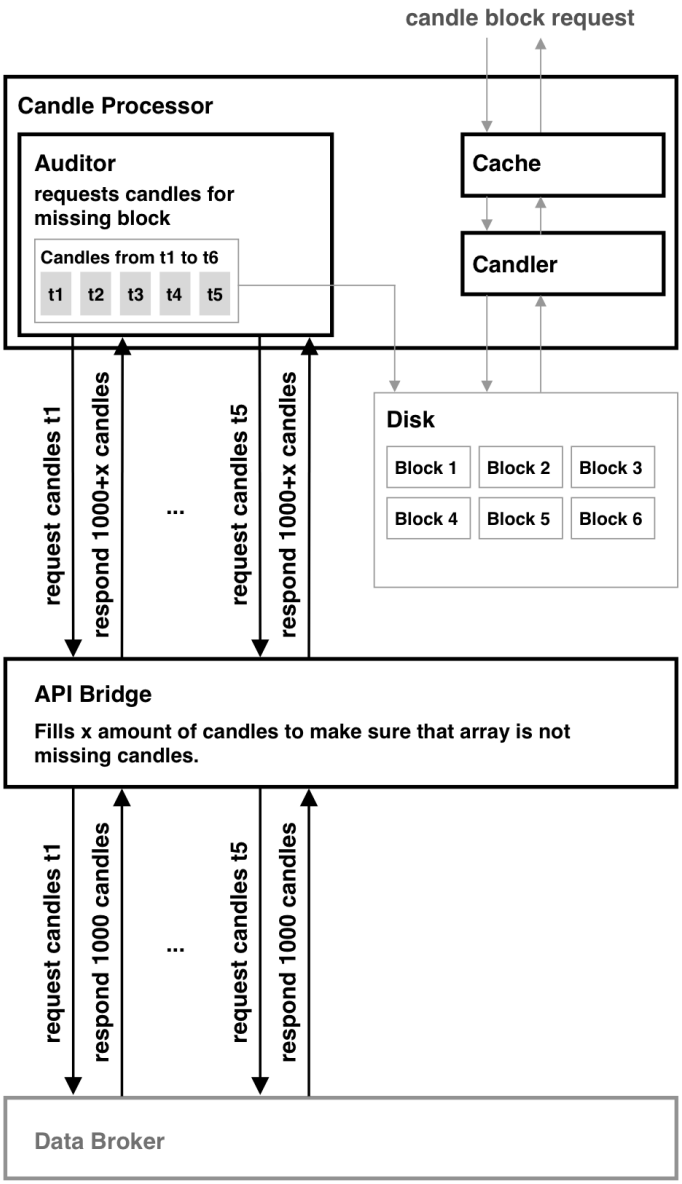


Figure 5.1: An overview of how the auditor collects candles from the bridge until it can fill one block. It then stores that block to disk, which is then available to the functions responsible for creating derivative candle blocks, called “candler”.

and then written to disk.

5.1.2 Candle processing and transformation

The candles stored on disk are referred to as "Primitive Candles" because they serve as the foundation for calculating other intervals. For example, if the service requires hourly candles, one option would be to fetch them from the connected data brokers. However, this would increase the number of requests, which is subject to rate limitations. Instead, we derive any candle block with an interval larger than the Primitive Candle set.

Initially, we worked with minute data, and therefore the Primitive Candle sets were stored accordingly. However, in the case of stock data, the most historical information is only available as daily candles. This led us to introduce support for multiple "resolutions." In the context of candle blocks, resolution indicates the interval of the Primitive Candle set they are based on. This allows us to retrieve daily candles dating back decades while also utilizing minute data when necessary, as long as the required Primitive Candles are available.

5.1.3 Candle blocks

Now that we have the Primitive candles, the focus shifts to calculating derivatives. When a candle is retrieved with the same interval and resolution as the ones stored on disk, the existing data from disk is forwarded. However, in all other cases, the requested candle block is calculated by recursively computing all intermediate candle blocks.

To optimize cache performance and reuse calculated derivative blocks, we selected a mapping that prioritizes the most commonly used candle intervals. By frequently using the same intervals, we enhance cache performance when subsequent candles are requested based on previously calculated derivative blocks. A portion of the mapping we settled on is as follows:

Interval	Dependency
Interval3m	Interval1m
Interval5m	Interval1m
Interval15m	Interval5m
Interval30m	Interval15m
Interval45m	Interval15m
Interval1h	Interval15m
Interval2h	Interval1h
Interval4h	Interval1h
...	...

With an understanding of how to calculate derivatives, we can now examine Algorithm 4, which recursively fetches candles. This operation is based on the `mergeCandles` function (see Algorithm 3) that combines the information of two candles. By determining the candle ratio between the two candle blocks based on the interval, we merge the appropriate number of candles to create a single representative candle for the larger interval.

Algorithm 3 Merge Candles

```
procedure mergeCandles(src, dst)
    if src.Missing then
        return
    end if
    if dst.Missing then
        dst.Missing ← false
        dst.Open ← src.Open
        dst.Low ← src.Low
    end if
    if src.High > dst.High then
        dst.High ← src.High
    end if
    if src.Low < dst.Low then
        dst.Low ← src.Low
    end if
    dst.Close ← src.Close
    dst.Volume += src.Volume
    dst.TakerVolume += src.TakerVolume
    dst.NumberOfTrades += src.NumberOfTrades
end procedure
```

Algorithm 4 Fetch Candles

```
1: procedure FetchCandles(symbol, block, interval)
2:     exchangeInfo ← GetExchangeInfo(symbol)
3:     smallestResolution ← −1
4:     for each resolution in exchangeInfo.AvailableResolutions do
5:         if resolution = interval then
6:             return GetPrimitiveSet(symbol, block, interval).Candles
7:         end if
8:         if smallestResolution = −1 or resolution < smallestResolution then
9:             smallestResolution ← resolution
10:        end if
11:    end for
12:    subInterval ← IntervalMap[interval]
13:    subNumber ← interval / subInterval
14:    startTime ← BlockToUnix(block, interval)
15:    candles ← create empty array of size 5000
16:    lastTime ← startTime − interval
17:    for i = 0 to length of candles do
18:        lastTime += interval
19:        candles[i].Missing ← true
20:        candles[i].Time ← lastTime
21:    end for
22:    for i = 0 to subNumber − 1 do
23:        subBlock ← UnixToBlock(startTime, subInterval) + i
24:        subSet ← FetchCandles(symbol, subBlock, subInterval)
25:        for j = 0 to length of subSet do
26:            src ← subSet[j]
27:            dst ← candles[(src.Time − startTime) / interval]
28:            mergeCandles(src, dst)
29:        end for
30:    end for
31:    return candles
32: end procedure
```

5.1.4 Transition candle blocks

Evaluating a strategy or algorithm at extended time intervals can sometimes lead to skewed perceptions. This distortion might be more pronounced when viewing current charts. A potential misinterpretation arises due to indicators often representing only a singular point in time. In contrast, the inherent design of candles addresses this issue. Unlike indicators, candles visually encapsulate the entire movement range within their body and wicks. This design ensures a comprehensive overview of market movements, even when the candle is still in the process of forming.

To facilitate accurate calculations with an indicator, the concept of a transition block becomes crucial. Transition blocks are applicable to both candles and indicators. However, since indicators do not differentiate between block types and simply evaluate them based on the

provided candles, the emphasis here is on explaining candle transition blocks.

In this context, evaluation involves processing the indicator as a transition indicator, incrementally, at each step throughout the simulation. Importantly, the resolution of the transition block aligns with the simulation's step size. If the simulation steps are minute-by-minute, for instance, the transition block is requested in minute resolution for the chosen interval, such as an hour.

This methodology guarantees an accurate representation of the indicator alongside its corresponding candle representation at each specific step, regardless of the broader timeframes used for the candles and indicators. By employing an indicator that evaluates the transition blocks, calculations retain their accuracy and representation within simulations, even when using larger intervals, like hourly ones, for candles and indicators.

As a result, it becomes a straightforward process to evaluate each step of the received transition block corresponding to each simulation step. This approach optimizes the process, making it highly efficient for any chosen step size in the simulation.

Transition candle blocks consist of candles associated with each timestamp of the resolution interval, which refers to the period on which the primitive candles are based. Another perspective is that transition blocks represent the values of the requested higher resolution interval (for example, 1 hour) by sampling values from a lower resolution (such as 1 minute). This method enables tracking of a candle's fluctuations over time (Figure 5.2).

Figure 5.3 depicts the original chart with standard candles, while Figure 5.4 showcases a chart using transition candles. The transformation is notably visible when portraying an interval at high resolution. However, when selecting an interval based on a resolution with a smaller ratio (Figure 5.5), the effect is less evident (Figure 5.6).

The acquisition of these transition blocks involves progressing through the primitive candles that constitute the candles of the larger interval being requested. With each time step, the current candle merges with the previous one, and the result is stored. Upon completing the duration of a single candle at the requested interval, the transition candle resets to the candle in the subsequent step.

In most situations, the only requirement is the set of candles through which the process steps. However, a transition block might occasionally commence at an interval that begins midway through a candle of that interval. As a result, the preceding candle block is also retrieved, which allows for the initialization of the candle. Given that the mapping from candle block to transition block is 1:1, no further information will be required.

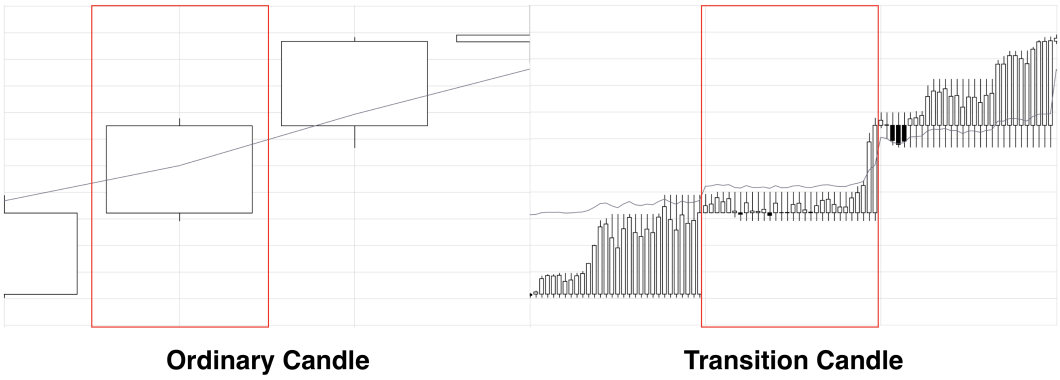


Figure 5.2: An example of the same candle being shown in the red box. On the left side we cannot see how the indicator line was a resistance point for the price movement, while the transition candles clearly show that there was resistance which got broken.

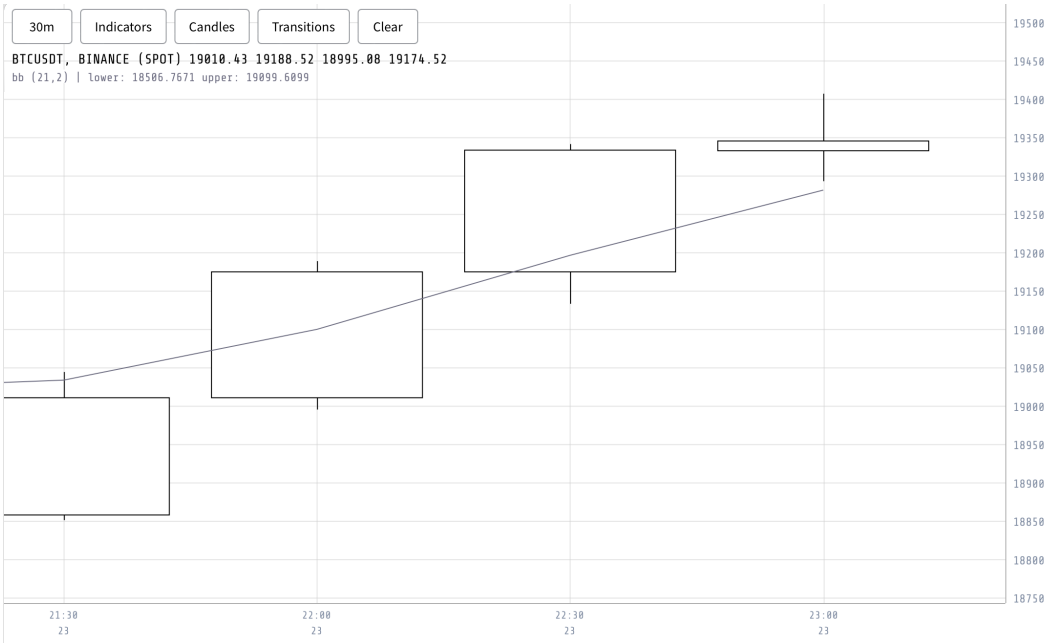


Figure 5.3: Candles representing the price for BTCUSDT. The indicator line is the upper Bollinger band.

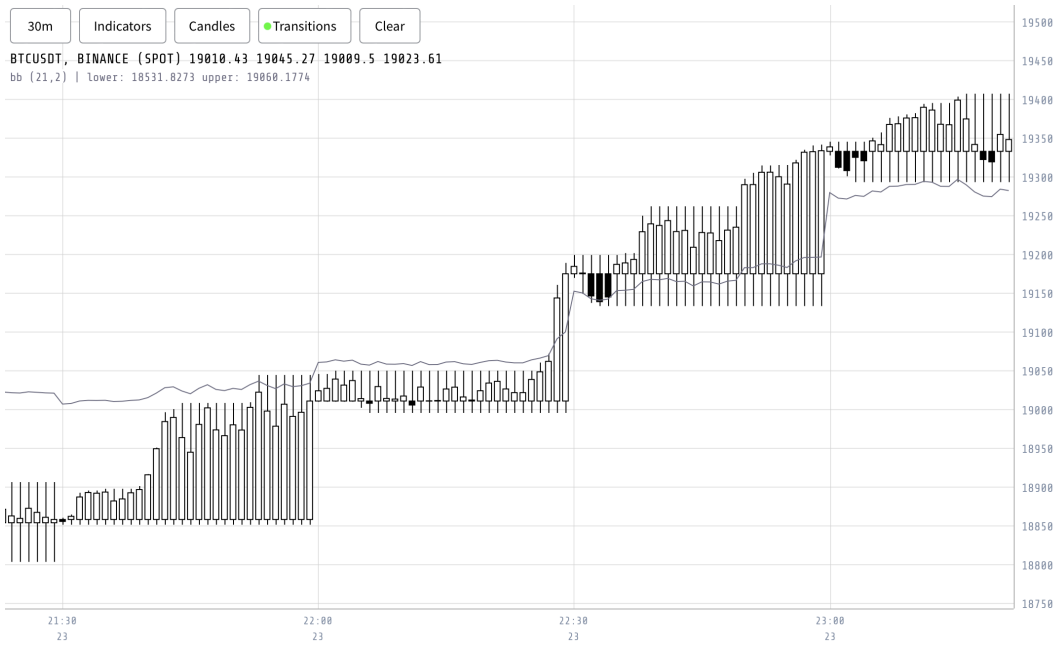


Figure 5.4: Transition representing the change in price per candle for BTCUSDT. The indicator is also calculated accounting for the transition candles.

Figure 5.5: Candles representing the price for AAPL. The indicator line is the exponential moving average (EMA).

Figure 5.6: Transition representing the change in price per candle for AAPL. The indicator is also calculated accounting for the transition candles.

When services need an exact state of a candle at a specific time, they can request the transition candle block for that duration, as depicted in Algorithm 5. This gives the exact state of the candle at that moment. In simulations, this approach offers significant advantages: operations only utilize transition blocks, negating the need to account for changes in candles or indicators over time. Thus, any employed indicator or candle-set in the pipeline will be the transition block in use.

Algorithm 5 Fetch Transition Candles

```
procedure FetchTransition(symbol, block, interval, resolution)
    if interval = resolution then
        return primitiveSet(symbol, block, interval)
    end if
    prevBlock ← GetPrimitiveSet(symbol, block − 1, resolution)
    currBlock ← GetPrimitiveSet(symbol, block, resolution)
    blockStartTime ← currBlock.UnixFirst()
    firstCandleTime ← blockStartTime / interval × interval
    stepIndex ← blockStartTime / interval
    stepCandle ← create new Candle with Time = firstCandleTime and Missing = true
    for i = prevBlock.Index(firstCandleTime) to 5000 do
        c ← prevBlock.AtIndex(i)
        mergeCandles(c, stepCandle)
        stepCandle.Time ← c.Time
    end for
    candles ← create empty array of size 5000
    for i = 0 to length of currBlock.Candles do
        c ← currBlock.Candles[i]
        nextIndex ← c.Time / interval
        if nextIndex ≠ stepIndex then
            stepIndex ← nextIndex
            stepCandle ← create new Candle with Time = c.Time and Missing = true
        end if
        mergeCandles(c, stepCandle)
        stepCandle.Time ← c.Time
        candles[i] ← create copy of stepCandle
    end for
    return candles
end procedure
```

Transition blocks may appear as a minor addition to the platform, but they proved crucial in debugging indicators. An added advantage is that they significantly simplify much of the simulation code used in later stages of the project.

5.1.5 Splits

Stock splits serve a crucial function in financial data analysis, especially in the realm of algorithmic trading. At a fundamental level, these splits are essential for maintaining the

continuity and coherence of time series data. When a stock undergoes a split, whether it is a 2-for-1 or any other ratio, the stock's price is adjusted accordingly. Without accounting for this adjustment, the time series would exhibit sudden drops or jumps, corresponding to the split events. For instance, in a 2-for-1 split, a stock priced at $100 would suddenly appear as $50 in the time series, even though there has not been an actual decrease in the stock's value. Such abrupt shifts can introduce anomalies that mask genuine price movements, making it challenging to discern legitimate patterns or trends in the data. These artificial disruptions could mislead algorithms, leading to incorrect interpretations and potentially erroneous trading decisions.

In addition to ensuring the consistency and integrity of financial data, stock splits also introduce unique patterns surrounding the split events themselves [67]. Observing these patterns can offer insights into potential predictable price behaviors. For example, a stock might experience heightened trading activity or volatility leading up to and following a split [68]. Recognizing and understanding these patterns can be invaluable for algorithmic trading, providing opportunities to generate trading signals based on anticipated market reactions to split announcements and implementations.

The algorithm for adjusting candlestick data for splits is detailed in Algorithm 6. This iterative process ensures that the stock price data is consistently adjusted for splits, providing a more reliable basis for analysis and visualization.

Algorithm 6 Adjust Candlestick Data for Splits

```
for each candle in candleArray do
    for each split in reversed splitArray do
        if candle.time < split.time then
            candle.Open /= split.ratio
            candle.High /= split.ratio
            candle.Low /= split.ratio
            candle.Close /= split.ratio
        else
            Move to next split
        end if
    end for
end for
```

To summarize, adjusting for stock splits is a critical preparatory step for any robust financial analysis or algorithmic trading strategy. It ensures the consistency and reliability of the price data, while also offering opportunities for pattern recognition and event generation around split events. However, the process of adjusting for splits does add an overhead to data processing, which necessitates the exploration of efficient data handling methods. With this in mind, we can now explore caching techniques as a means to improve the performance and scalability of the trading system, while also eliminating the necessity of recalculating these splits on multiple occasions.

5.1.6 Caching

The code incorporates a cache implemented with the Ristretto library [69]. The cache size can be determined according to needs, but it is found that sizes exceeding 1GB perform quite satisfactorily. In this implementation, a 3GB size is chosen, as this is the amount of spare memory available. This memory remains in use and is only evicted or subjected to garbage collection when new requests are made.

Estimating the true memory footprint of complex data structures can be challenging in many programming languages, especially when they involve pointers. These pointers typically reference the location of data rather than the data itself, leading to potential underestimations of actual memory usage. In the context of the Go application, a specified cost per item stored in the cache is required. To address this, the size of one Candle struct is determined to be 65 bytes. Multiplying this by 5000 gives the estimated size for a set of candles. Adding some buffer for additional overhead and metadata, a rounded value of 360,512 bytes is determined. This suggests that the cache can accommodate approximately 8000 such items.

Since historical data remains static, it can be safely stored in cache without any timeout. However, there are instances where incomplete candle blocks are stored. To account for potential real-time use of the algorithm, a time-to-live is added to incomplete blocks. Given that algorithms typically complete in a few seconds, the time-to-live is set to 10 seconds. If dealing with second-level data, the time-to-live could theoretically be reduced to as low as one second.

From an algorithm's perspective, all data is cached. Once the service receives an initial request, all subsequent requests receive immediate responses. This functionality enables near real-time previews of indicators and algorithm results by minimizing the overhead of calculating transition and intermediate blocks for each request.

5.2 Indicator processor

Indicators serve as vital tools for discerning market trends. This chapter delves into the operation of a specialized service responsible for generating indicators based on the processed candle data from the previous chapter. Consequently, it is feasible to compute popular indicators, such as the Exponential Moving Average (EMA) and the Relative Strength Index (RSI).

The primary objective of this service is to consolidate all indicator logic into a singular service, enabling instant output for a requested indicator in real-time. Given that indicators can be requested with a range of parameters, advanced caching techniques are employed to streamline the retrieval and processing of candles.

Indicators are usually expressed as simple equations when looking at how they are calculated in the real world. These expressions can be stateful or stateless, which means a stateful expression could use the previous value. The Exponential Moving Average (EMA) serves as an example of such a stateful expression:

EMA = Closing price x multiplier + EMA (previous day) x (1-multiplier)

While this appears straightforward in theory, it requires access to the complete history of the

dataset for its calculation. Since the operations are performed on candles, alternative methods for approximating the full history must be explored for use in a stateful indicator, to avoid the necessity of always employing the entire dataset.

Handling stateless expressions is relatively simpler and may not require the complexity introduced by this service. However, the service does facilitate inward and outward caching of data to hasten even basic calculations when viewed in the project's scope. The Simple Moving Average (SMA) serves as an example of a stateless expression:

SMA = (A1+A2+...+An)/n

Lastly, some indicators are based on other indicators. Bollinger Bands (BB), for instance, rely on the Simple Moving Average (SMA). This necessitates the capability to calculate indicators recursively, and strive to prevent redundant calculations.

5.2.1 Definitions

Indicator definitions often involve intricate mathematical expressions that are not easily distilled into a few lines of code. The intricacy becomes more pronounced when one indicator relies on another, complicating their translation into conventional scripting languages.

The approach to handling these challenges involves a systematic stepping mechanism through each time interval, be it a minute, hour, or any other defined period. By progressing one step at a time, we're able to calculate the value of an indicator for each moment. Ideally, if the calculation at each step is of linear complexity, the entire process should be achievable in linear time.

To simplify and standardize the representation of these indicators, we've abstracted the calculation logic behind interfaces. The primary structure for this purpose uses a "Moment" type. This "Moment" acts as a container for current data, the preceding data value in the series, and any associated parameters. By passing this "Moment" to the respective indicator's function, we aim to represent a broad spectrum of indicators comprehensively.

For instance, a complex indicator like the Exponential Moving Average (EMA) requires an initialization function. We cannot simply start calculations from an assumed 'beginning of time,' as traditional mathematical formulas might suggest. On the other hand, Bollinger Bands represent an example of an indicator dependent on another, specifically the Simple Moving Average.

However, some advanced indicators, like diagonal resistance lines, may not fit this stepwise, series-generation model due to their unique requirements. While we've built the system to accommodate a wide range of indicators, it is essential to understand that some complexities extend beyond the boundaries set by the interfaces. Additionally, while the indicators are designed to be efficient and integrated within the service, they are not directly used for pattern detection due to the nuances in how patterns and indicators differ in their computations.

5.2.2 Expressions

Each indicator definition is structured as a series linked to a specific key. Many indicators, like the Bollinger Bands with its distinct top and bottom bands, comprise multiple series.

Calculating all these values is essential. Given that most standard indicators maintain separate formulas for each series, they can be efficiently computed concurrently.

Algorithm 7 Simple Moving Average Procedure

```
procedure SimpleMovingAverage
    definition ← create new Definition
    definition.Series ← create new map to store series of indicators into
    sma ← create new series
    sma.Step ←
    function smaStep(p, params)
        size ← params[0]
        acc ← 0.0
        points ← 0
        for i ← 0 to size − 1 do
            c ← p.FromLast(i)
            if c.Missing then
                continue
            end if
            acc ← acc + c.Close
            points ← points + 1
        end for
        if points = 0 then
            return 0, false
        end if
        return acc/float64(points), true
    end function
    definition.Series["sma"] ← sma
    return definition
end procedure
```

In Algorithm 7, the Definition object contains everything needed to compute the SMA for a specified series. The *sma.Step* function determines the simple moving average over a designated period (as defined by size). It aggregates the closing prices (*c.Close*) from the most recent size points in the series and computes the average. Missing data points are excluded. The derived series of simple moving averages is then housed under *definition.Series*["*sma*"].

Ultimately, this approach enables the efficient computation of prevalent indicators and their corresponding series, leveraging the advantages of concurrent processing.

5.2.3 Candle Logistics

The Moment structure encapsulates the necessary candle sets for indicator block calculation, as well as a caching mechanism for data retrieval during the indicator computation. Since indicators align with the intervals they're calculated for, there exists a direct one-to-one mapping with the specified candle block. Indicators might require previous candles, necessitating the retrieval of the preceding block. Following the acquisition of these blocks, the computation commences.

The pseudocode representation of the Moment interface:

```
INTERFACE Moment
    METHOD Time() RETURNS integer
    METHOD IsMissing() RETURNS boolean
    METHOD IsTransition() RETURNS boolean
    METHOD Last() RETURNS Candle
    METHOD FromLast(index: integer) RETURNS Candle
    METHOD AtTime(timestamp: integer) RETURNS Candle
    METHOD Indicator(name: string, params: list of integer) RETURNS MomentIndicator
```

Dependencies of an indicator are unknown initially, hence they are fetched at runtime. A cache stores any indicators requested to calculate the desired indicator. To retrieve two blocks, the transition block is requested on top of the two earlier blocks. The historical blocks are necessary since the indicator can retrieve past candles, and candle blocks represent only the current time step.

When an indicator is requested, the same functions processing a request are utilized. These functions run recursively until all dependencies are calculated and then returned to the caller. Each result returned to the caller is stored in a local cache by an ID that represents the indicator.

Indicators are retrieved from the cache at every step of evaluating an ID. The largest overhead in calculating indicators was the actual hashing of the indicators. Many approaches were tested, but the most straightforward option of converting the parameters to strings and concatenating them with the indicator ID was the fastest to index the cache.

Another aspect to consider is that candles can be missing due to market closure or data unavailability. As algorithms are generally assumed to be continuous, this needs to be abstracted. Therefore, when stepping through the candles of a block for evaluation, it is first ensured that the candle is not missing. This largely eliminates the need to deal with them when calculating a candle and is only necessary when an indicator requires historical candles which might be missing. Depending on the expression for that indicator, the missing candles could be considered in the final result.

5.2.4 Warmup

While the warm-up methods significantly improve the starting conditions for indicators, it is worth addressing the resultant accuracy of the indicators. By approximating the initial candle using heuristics or separate expressions, the accuracy of the indicators during the warm-up phase might deviate from a scenario where the entire history is available. However, as the indicators are computed over subsequent data points and more real data becomes available, this deviation tends to diminish.

The influence of the initial conditions wanes as more data points are processed, and the effect of this warm-up deviation on overall analytical results is typically minimal, especially for longer time series. Although we did not conduct an in-depth analysis of the warm-up's effect on accuracy, this approach involved using an extensive warm-up period. While this might have slightly impacted performance, the effect was negligible. The primary goal was to ensure accurate results, and we believe this extensive warm-up period achieves that. Furthermore,

the provided global parameter for fine-tuning accuracy allows users to adjust the extent of the warm-up period, offering a balance between computational efficiency and desired accuracy.

Certain indicators operate under the assumption that the entire history is accessible and initiate from a previous value. Given the block structure at work and the unavailability of the full history, heuristics are employed to approximate the initial candle without actually having access to it. One method of procuring the first candle could be to simply retrieve the candle from the previous block, but this implies that each time a block is retrieved, all preceding blocks would need to be calculated – a time-consuming process even with caching.

As an alternative, separate expressions are used to compute the starting value of the stepping expression. Warm-up methods, which are frequently described in indicator definitions, enable this process. By utilizing the previous block for warm-up, a near perfect approximation of the initial starting value can be achieved. While this might require substantial computational power, it offers considerable improvement over calculating previous indicator blocks or attempting to guess the first value. Given that some expressions might not necessitate a long warm-up period, a global parameter is provided to fine-tune the accuracy. One such approach to handling this is illustrated in Algorithm 8, which demonstrates the computation of an Exponential Moving Average (EMA).

Algorithm 8 Exponential Moving Average Calculation

```
procedure ExponentialMovingAverage
    definition ← create new Definition
    definition.Series ← create new map of string to SeriesDefinition
    ema ← create new SeriesDefinition
    function ema.Init(p, params)
        length ← params[0]
        alpha ← 2.0/(float64(length) + 1.0)
        ema ← p.FromLast(warmUpLength).Close
        for i ← warmUpLength + 1 to 0 do
            if p.FromLast(i).Missing then
                continue
            end if
            ema ← p.FromLast(i).Close × alpha + ema × (1.0 − alpha)
        end for
        return ema
    end function
    function ema.Step(p, prev, params)
        length ← params[0]
        c ← p.Last()
        k ← 2.0/(float64(length) + 1.0)
        return c.Close × k + prev × (1.0 − k), true
    end function
    definition.Series["ema"] ← ema
    return definition
end procedure
```

In contrast to many indicators, there are specific ones that inherently defy this method of warm-up. A prime example is the VWAP (Volume Weighted Average Price). VWAP gives the average price a security has traded at throughout the day, based on both volume and price. As it requires ongoing computations over a trading day, to compute an accurate VWAP, one would ideally need continuous data from the inception of the security's trading. In the context of this service which operates on daily candles, this implies that VWAP will inherently exhibit jumps between these blocks of data. The nature of this indicator means that traditional warm-up methods would not suffice, as you'd essentially be trying to back-calculate from the dawn of the security's existence. Thus, it is imperative to understand and accept that for indicators like VWAP, discrepancies between blocks are unavoidable and must be taken into consideration when they are used.

5.2.5 Caching

In the context of caching, we apply the same approach as employed in caching candle blocks within the candle service. However, we utilize a smaller cache size, typically around 2GB, as indicators are not reused as frequently as candle blocks. Nonetheless, incorporating a cache remains crucial for achieving the optimal performance observed in the pipeline.

In contrast to candle sets, indicators involve an additional parameter to consider. We refer to this parameter as the "list of parameters" associated with a specific indicator. As previously mentioned in the context of caching, this introduces a slight increase in complexity as we need to distinguish results based on different parameters. However, aside from this distinction, it does not pose any significant challenges.

5.2.6 Transition blocks

Transition blocks can also be requested, similar to candle sets. As previously mentioned, stepping through the transition blocks occurs in the same manner as stepping through the candles of a typical block. The distinguishing factor is that while the candle block can be used to retrieve historical candles, the transition candles cannot facilitate this historical lookup. Hence, it remains necessary to retrieve the current and previous candle blocks. When utilizing the previous value in an expression, it is also necessary to use the previous value from the reference of a candle; meaning, this is the last value at the point in time that the last candle closed. This enables observation of how the market tends to anticipate candles interacting with indicators, and how an indicator might "touch" a candle, but when observing transition candles, it may be evident that contact never occurred.

In conclusion, the indicator processor is a pivotal tool within the pipeline, enabling the rapid generation of a diverse range of indicator blocks with varying parameters. By implementing caching when responding to clients and when fetching candle blocks, overhead can be effectively minimized when an indicator is required within a service. This efficiency allows for the generation of thousands of indicator blocks per second. Use of these indicators will be seen in the pattern detection methods within subsequent chapters. Furthermore, getting the basics right with this service is essential as it lays the groundwork for more advanced results calculations, which are categorized as algorithms.

5.3 Event processor

The last daemon in the pipeline is the event processor. In order to do the actual pattern extraction, we needed to make it part of the pipeline. Since we already have the tools built in previous, we bundled these tools together in a package that can be used to write algorithm scripts. These algorithm scripts can then be compiled and spawned as workers by the algorithm daemon, which caches the results for each script, and takes away the hassle of having to build, run and cache programs in the actual experimentation phase.

We opted to consider each algorithm script as its own program, which implements the same package that contains all the necessary logic to be run by the daemon. This means that as someone writing and reviewing scripts, all the actual logistics of data can be left behind. We can access any candle block and any interval block by using an interface which hides all the logistics.

This way we can create easily explainable scripts that can detect patterns and run them on hundreds of scenarios in a few seconds times. Without the work of the previous services in the pipeline, this was simply not possible. Since we are also not limited by any constraints imposed by third-party platform, we can write any scripts without constraint.

When comparing to indicators, algorithms could be expressed as stateful variants of the expressions that were being evaluated in the indicator processor. Indicators are much more optimisation as they are short and stateless expressions which output their data in the same block format as candles. This makes it easy to display indicator data alongside candle data.

When working with algorithms, we opted to use events instead of mapping candle points. This means that we can store much more information per event compared to a data point used in an indicator. As this data is also sparse, it can still be transferred quite quickly and displayed in the visualisation tool. These events also include a lot of data that we can use to make better visualisations that are not possible with conventional indicators.

In the next section, we will explain how we ended up with the framework used to calculate the algorithms. We will first have a look at the server structure used to spawn the worker processes. Next, we will explain how the algorithm helper package is structured, and the logic behind it. Finally, we will show some examples of the actual algorithm scripts that are being used.

5.3.1 Event processor server

Initially, we went with a queue approach where a client would be able to request a job, which would be the calculation of an algorithm. The client could then check up on this job, and once the job would be finished, the client would be able to request the result for said job. This, however, involved a lot of coordinating between the client and the service, and also between the service and the worker. The initial reason for taking this approach was because algorithms took a long time to calculate. But with the many optimizations, the actual time it took the worker to evaluate a script was less than the time it took for all the overhead communications to take place. We, therefore, went back on this approach in favor of simple synchronous requests.

When a client requests the results for an algorithm, we calculate all the events from the start

of the requested asset's history till the end. The service waits for the worker to finish and forwards its results back to the client. If the worker is not yet running, it will be compiled and started. Once the worker has started, it will stay running until the termination of the service.

In most cases, keeping the worker running is fine. But when making changes or debugging one of the algorithm's scripts, we would like to request the results for the new version of the script. We, therefore, add a special parameter that can be passed to indicate to the service that it should stop the existing worker, build it again, and start it. The difficulty with this is that both the worker and service will accept and run multiple requests in parallel, meaning that we cannot just stop the worker at any time without risking data loss.

To coordinate restarts, we send a termination signal to the worker and hold any incoming requests for that worker. The worker will now enter its termination state. Once it has finished all running requests, it will gracefully terminate, and with a successful exit signal, the service will know that it can safely rebuild and restart it. Since Go makes it possible to easily implement thread-safe variables, we can easily coordinate this termination and restart using channels to indicate termination, and locks to make sure no additional requests are passed when the server is restarting.

We are now able to build and run any script and restart it safely at any time. This makes it possible for another service to simply request the events of an algorithm by sending a simple HTTP request. It not only simplifies the final experimentation but also makes it possible to use algorithm results in other algorithm scripts.

Ideally, we do not want to have any caching steps at the end of the pipeline, and therefore implement all possible caching on the server. In order to make memory management concerning the cache of this result as convenient as possible, we cache all the results in the event processor. Since we know the number of events and how much an event will take up in memory, we can make a good guess for each result that we store in the cache. This way, we can effectively store recent results for a long time to make sure the experiments do not request a result that needs to be computed twice. The cache is set up in the exact same way as for the candle service, so we will not discuss it any further here.

5.3.2 Event Processing Client Framework

In order to make it possible to easily run, and work on different algorithms, we implemented a package that is used as the backbone for a running script. We will refer to this package as "algocore". Besides including all the logic to run scripts, it also includes all types that are used to reply to requests. With this, we can add this as a dependency to the event processor, and easily decode any replies.

Go refers to dependencies as modules, and each module can include any number of packages. In the case, the algocore package contains a few useful packages:

- algo: Contains a set of structs to contain the results of an algorithms output.
- env: Contains all interfaces and helper functions used to provide all interfaces needed for a script.
- kiosk: Provides an interface to retrieve data from any of the daemons.
- ritmic: Contains the web server logic, used to launch the script when running headless.

kiosk

The key components in this module is the kiosk package. As this package is responsible for providing access to all the services, and returning them in the appropriate structure, it contains functions to access any candle blocks, indicator blocks, and algorithm events. It also provides an extra flag to retrieve the transition variant for any block.

When building the kiosk package, we started from the existing work that was done in the indicator processor, and reused a lot of the logic that was already implemented there. We did repackage them in order to make them more reusable. In order to make it easy for any service that might need to retrieve data from the pipeline, we created a series of data supplier structs. The data suppliers provide a robust interface that uses all the caching logic that was discussed in 5.2.5 to reduce the number of outgoing requests. Since indicators were only able to retrieve and cache candle blocks, the logic has been expanded to include indicator blocks and algorithm events.

As a reminder, the caching logic in 5.2.5 aims to minimize outgoing requests and reduce data retrieval response time. We achieve this by caching previously fetched data like candle blocks, indicator blocks, and algorithm events. This caching mechanism speeds up the access to frequently queried data, eliminating redundant requests and thus cutting down overhead.

The supplier logic is structured as:

```
INTERFACE MarketSupplier
    METHOD Algorithm(name: string, params: array of float64) RETURNS AlgorithmSupplier
    METHOD Interval(interval: integer) RETURNS IntervalSupplier
    METHOD Price() RETURNS float64
    METHOD Time() RETURNS integer

INTERFACE IntervalSupplier
    METHOD Candle() RETURNS Candle
    METHOD FromLast(offset: integer) RETURNS Candle
    METHOD ToTimeStamp(index: integer) RETURNS integer
    METHOD ToIndex(timeStamp: integer) RETURNS integer
    METHOD Indicator(name: string, params: array of integer) RETURNS IndicatorSupplier

INTERFACE IndicatorSupplier
    METHOD Value() RETURNS float64
    METHOD Exists() RETURNS boolean
    METHOD Series(key: string) RETURNS float64

INTERFACE AlgorithmSupplier
    METHOD Events() RETURNS array of Event
    METHOD PastEvents() RETURNS array of Event
    METHOD LastEvents() RETURNS array of Event
    METHOD HasEvents() RETURNS boolean
```

This structure facilitates requests for any candle, indicator, and computed algorithms.

We earlier mentioned that caching is done in each service, with the goal of being able to simply

request any data, and not having to worry about introducing too much overhead. For cases where the suppliers are overkill, or cases that do not work on a block basis, e.g., the validation scripts, we provide an alternative. We provide getter functions to access data for any service. These getters will execute the http request, and return it in the correct type. On top of that, since we have designed a encoding for blocks to binary, we can make sure that the transfer is done as efficiently as possible.

With the kiosk package we can not effectively feed any data that we need to the scripts. On top of that, this is also the way that we retrieve data for the evaluation scripts, as those are also written in Go, the final payoff was that we did not have to write any logic for logistics in retrieving data.

env

The environment package contains all the suppliers that we mentioned before. On top of that we also provide an interface for the memory and parameters. This way these interfaces could be reused tot potentially expand the algorithm logic to other purposes like strategy evaluation.

The memory and parameters interfaces are essential components of the env package, as they allow for more structured and controlled access to the script's state and input parameters. The memory interface makes it easy to store and retrieve the script's state, while the parameters' interface ensures that input parameters are accessed and parsed without the need for error handling.

The parameters are defined in advance, and can thus be retrieved by their key. If a parameter is missing, then this will make the program crash. This avoids any accidental parameter usages that were not intended with the use of indices, and also makes it convenient to retrieve parameters in the correct type.

Given that the scripts maintain a state, there's a necessity for a storage mechanism for this state. In the design, we've adopted a simple approach: the script is permitted to store any struct it desires. This stored struct can be retrieved in subsequent steps, and its type can be defined by the script. If there's a need for the script to update the state, it has two options. It can either push the modified struct back into memory or utilize a pointer to the struct, ensuring updates are reflected directly in the stored data.

The pseudocode representation of the storage mechanism:

```
INTERFACE Parameters
    METHOD Get(key: string) RETURNS float64
    METHOD GetInt(key: string) RETURNS integer

INTERFACE Memory
    METHOD Store(data: any type)
    METHOD Read() RETURNS any type
```

Through this structure, we empower scripts with the flexibility to manage their state seamlessly, be it reading, storing, or updating.

ritmic

The actual running of the scripts happens with the ritmic package. This package simply

accepts the step function for the algorithm, and sets up the server which is on stand-by for any simulation requests. Since debugging is hard once the script is running in the daemonized environment, we also provide testing function, in order to evaluate the stepping function without the need to start a server.

When talking about a set of parameters, we refer to this set as a scenario. The ritmic server is able to evaluate multiple scenarios in parallel, but sequentially evaluates each scenario to avoid any data leakage. The evaluation of scripts goes as follows:

1. Create a provider for all scenarios using the kiosk's Provider factory, provided the symbol and resolution as input.
2. Iterates through each block of candle data.
3. For each block, retrieve the current and previous block using the data provider that we created earlier.
4. Iterates through 5000 candles, skipping the ones with missing data.
5. Create a MarketSupplier for the current time instance using both blocks.
6. Iterates through all the scenarios and for each scenario:
7. Retrieve the memory associated with the scenario.
8. Creates a result handler for storing the results.
9. Evaluate the trading script using the stepper function

The primary method for storing results in the design is the use of nested maps. This implementation detail remains concealed from the script. Instead, the script interacts with the provided interface, thereby allowing optimizations in the result-gathering process and mitigating potential issues like unintentional cheating if a structure were to be passed to the step function directly.

The pseudocode representation:

```
INTERFACE ResultSet
    METHOD GetSymbolResultSet(symbol: string) RETURNS SymbolResultSet

INTERFACE SymbolResultSet
    METHOD GetScenarioSet(index: integer) RETURNS ScenarioSet

INTERFACE ScenarioSet
    METHOD GetEvent(index: integer) RETURNS Event
    METHOD GetParameters() RETURNS list of float64

INTERFACE Event
    METHOD AddSegment(segment: SegmentAnnotation) RETURNS Event
    METHOD AddPoint(point: PointAnnotation) RETURNS Event
    METHOD AddEvent(event: Event, label: string) RETURNS Event
    METHOD SetColor(color: string) RETURNS Event
    METHOD SetIcon(icon: string) RETURNS Event
    METHOD SetPrice(price: float64) RETURNS Event
    METHOD SetTime(ts: integer) RETURNS Event

INTERFACE ResultHandler (available for the step function)
```

```
METHOD NewEvent(label: string) RETURNS Event
```

Once the results are compiled, the system encodes them in JSON format and sends it back to the requester. In case of execution errors, the web server will attempt to catch and return the error. However, for grave errors, such as trying to access negative time indices, the program will terminate abruptly, since such scenarios should never transpire.

In summary, the algocore package serves as the payoff of the preprocessing work. It gives a simple interface to all data related to technical analysis. By having caching at every step in the pipeline it makes it possible to have extremely fast replies, and thus finally possible to evaluate multiple scenarios which would never be as quick using online services, or simple databases.

The evaluation process in the ritmic package is structured to ensure efficient execution of the algorithm scripts. By evaluating multiple scenarios in parallel and sequentially evaluating each scenario, the process minimizes the potential for data leakage and ensures that each scenario is evaluated independently. By taking these precautions in advance, we can say with confidence that chances of accidental leakages are second to none. This structure also allows for a more organized flow of data, as these parallel requests have a better chances of generating cache hits.

5.3.3 Script Execution and Event Detection

In the previous chapter, we discussed how the pipeline is set up. Now that we have a near finished pipeline, it is time to make use of it by writing scripts that can be evaluated. Since these scripts are tightly integrated, these can be evaluated much faster than if they run as part of a large monolith program.

These scripts are critical to the methodology, as they are the gateway to quickly processing market in minutes which would otherwise take hours. The benefit of the fast processing time is that we can quickly iterate the results, and test on a wider range of hyperparameters. We will have a brief overview on how these scripts work, and explain them in detail in their respective methodology chapter.

Since we need features to do any sort of pattern extracting, these scripts will provide the means to do so. Scripts produce temporal events, and these events can be used to determine if a pattern has happened or not. A collection of events should represent a patterns. As seen in the literature, these patterns are often a collection of two or three signals, which then has a followup market movement.

As explained in the previous section, we employ the interfaces provided by the algocore package to primarily access candle and indicator data. This is, of course, the least of what is expected, as any third party platform often also gives the ability to evaluate scripts. What we can do, on top of using market data, is to use other script events in other scripts.

Since the pipeline is able to instantly provide near real-time results, with a bit of overhead, we can use many scripts as dependencies to make each script focus on one task and make it easy to validate. This way we can create complexity out of simplicity with the thought of keeping everything as explainable as possible.

The events that each script produces depend much on what to script is trying to achieve. In

order to accommodate all types of events, the interface is used to create them as described in 5.3.2. Once the algorithm instance responds to the daemon, these are structured in an interface understood by all services which depend on the script results.

The pseudocode representation of the Event structure:

```
STRUCTURE Event
    FIELD CreatedOn: integer
    FIELD Time: integer
    FIELD Price: real number
    FIELD Label: string
    FIELD Icon: string
    FIELD Color: string
    FIELD Annotations: AnnotationCollection
```

An event happens at a given time and price. The script author can choose the time and price, since a script could predict an event, or mark one that happens in the past. The CreatedOn variable is, however, off limits for the author, since this variable is used to mask off the list of events when this is provided to another script. This way we can avoid any leaking of events that have future knowledge, but still make it able to show events that could happen in the future, like predicting tops.

In order to differentiate between events, we use a simple label field, which can then be used to label events like “uptrend” or “downtrend”.

Fields “icon”, “color” and “annotations” are used here to make visualization and validation easier. The color and icon fields are self-explanatory. The annotation field is used to store “sub-events”, which contain extra information that we can use to validate an algorithm. In the interface that we specified, one can pass an event directly to these annotations, this way we can see if the right values of the dependency-algorithm are used when we evaluate the results of this algorithm.

These visualisation variables can be fully omitted in a production workflow, and would significantly reduce the computational and memory overhead; that is introduced by storing all this extra metadata.

Instead of thinking about the market represented by candles and their price, we can start thinking about the market as a series of temporal events. As opposed to simplifying the data by using methods like fourier transforms, we give meaning to price structures in the market.

In the methodology we will use a series of scripts, one set of scripts is used to extract features for pattern clustering, the other set is used to detect the patterns directly. These scripts have different goals, and therefore are differently implemented.

- Clustering scripts: Take a common event based on an indicator to produce a market signal. Has a time and place.
- Pattern scripts: Use the market price to determine technical analysis-like features, which can be used in combination to extract patterns.

Clustering scripts are therefore extensions of indicators. Whilst pattern scripts are, although simple, a more advanced technique of extracting events from the market. These pattern scripts also return data with some annotations to clearly visualize the pattern that is detected.

As explained in the previous chapter (5.3.2), the scripts are always based on a stepper evaluation function. The scripts will always be evaluated over the whole history of a given asset. We call this function every step. Every step is defined by the requested interval for the algorithm. If the algorithm runs on an asset, it evaluates that asset's resolution, which is the lowest possible interval for that asset. This means that the algorithm has full control over what indicators it would like to use. The idea behind this is that the user of the script should pass all the parameters as part of the params, which can then be used to pass any intervals.

The types of the variables have been explained in the previous section. The MarketSupplier supplies data from other services, the ResultHandler is used to output events, Memory is used to store temporary data, and Parameters are used to access parameters passed to the script representing a given scenario.

Since we use the whole history, there is no point in having a warmup. The algorithm knows and can account for the first value in the asset's history, and thus no warmup logic is needed. Pattern scripts do have some warmup logic, since they look for structures in a series of candles, they first need to parse a few candles, and therefore will only work after some candles have been stepped through.

In order to specify a script, how long things like these histories need to be, we use parameters. These parameters are always represented as floats, and can then be requested from within the script as an int, or a float.

This provides enough information for filling in unknowns as candle intervals, indicator parameters, or ranges in which to look for features. Since these parameters can represent floats, it could be used in an advanced case to evaluate trading strategies and pass parameters for profits and loss thresholds. In this thesis, however, we will exclusively use it to determine events in the context of patterns.

In the context of detecting Exponential Moving Average (EMA) crosses, one would specify the two ranges (such as 12 and 50) for which to detect crosses and the interval (in this case, the daily interval of 86400 seconds) on which to perform the detection. By passing these three parameters (12.0, 50.0, 86400), the algorithm can identify EMA crosses based on the specified ranges and an interval.

Depending on the type of function, it will need to have some memory. In all the scripts, we have to use memory, in order to know what state we are, so we can produce appropriate events. Initializing and loading memory could look as follows:

- Define a structure *Store* that keeps track of the last EMA (Exponential Moving Average) cross state:
  - This structure has a state field *State* used to represent the last state of the EMA lines, initially set to undefined.
- Declare a variable named M of type *Store*.
- Attempt to read from memory, and store the result in M.
- If M is null (the memory read did not return any value):
  - Initialize M as a new instance of *Store*.
- Do any changes, related to the state of EMA, like changing *State*.
- After this of the candle step, we push the value of M back to memory.

Given the structure of a script, one should be able to implement any algorithm. We then

request the events for those algorithms from the clustering and manual pattern evaluators. In the case of clustering, we can use these events directly as features for the clustering; these clustering algorithms have no need for any other data.

The evaluator which we use to check how well manual detection and clustering patterns are, can use the algocore package to request candles and the events which is used to see how well they perform.

In cases the market data is missing; just as with indicators, we will not step through that datapoint. This does not mean that we can ignore missing data in the script. We still have to check if data is missing when requesting candles from the data provider, as this provides any historical candle which could not exist. The same is true for indicators, which might only be available after its warmup period.

If we only use the current candle, then this does make it easier to script, since we can always assume that it is present. As for the evaluation, we also have to make sure that when retrieving the candle for which this event was created actually exists, as it is possible for events to be created in the future.

Failing to account for missing data can have significant consequences on the accuracy of output events, leading to wrong results. It is crucial to pay careful attention to missing candles, as there is no reliable method to detect their usage

The pipeline is designed to pass small blocks of data between services. The same cannot be said for algorithm's result sets. These results can be quite large when parsing events at a high resolution, or in cases where many events are detected. In order to avoid any bottlenecks, avoiding generating many events is the best cause of action. When every two candles produce an event, then a script is a bad fit for generating those events, and one should consider an approach by using indicators.

By using the metadata provided for each event, we can make sure that visualizing many events is done without impacting framerate in the visualization tool. This way we can omit events that are not visible to the user in a given block.

Another factor that involves performance is the use of the memory in a script. Since this memory is passed with a pointer, one should not be concerned when trying to store a lot of data in these structs. We will always evaluate algorithm linearly and not multithreading when stepping through a single scenario. This makes sure that we can use large structs in memory, which could be expanded to more advanced tasks like machine learning in each evaluation step and storing the model in memory.

With the established method for quickly generating and visualizing these events, we can check the accuracy of well-known events, such as EMA crosses and RSI thresholds. By comparing the charts and event placements with trusted platforms like TradingView, we can see if the events match what's typically expected from the literature.

It is important to mention that while the initial checks align with these third-party platforms, we have not carried out a full, detailed verification. The approach is based on the idea that if there were any issues, they'd likely show up as we use the system, giving us a real-world way to test its accuracy.

With this capability to pull out events from market data, the pipeline is now complete. We'll go into more detail about these events in the next methodology section, where we'll discuss the scripts we used.

Chapter 6

Detecting Patterns

This chapter delves into the heart of the exploration: pattern detection. We'll be examining two key techniques in detail: clustering and manual pattern detection.

In Section 6.1, we explore the potential of clustering for automated pattern detection within market data. We will delve into the methodology for extracting relevant features from the market and the means to interpret them. Since patterns in financial data are challenging to define within a Euclidean space, we will examine various existing techniques for clustering that can effectively handle this complexity. Specifically, this section will shed light on how unsupervised machine learning techniques can unearth patterns in market data that might otherwise remain obscured or undetected."

Section 6.2 shifts focus towards a traditional approach to pattern detection. We will explain the algorithms used to detect patterns, by using the pipeline that we have built in previous chapters. Since many patterns exist, we try to detect the most common patterns, by using highs and lows as a basis. This section sheds light on the standard patterns that traders often look for when starting with trading with patterns.

6.1 Approach 1: Clustering-based pattern detection

The initial focus of the pattern detection efforts involved employing a clustering-based approach. The motivation behind this decision stemmed from the recognition that manual pattern detection can be a labor-intensive task, requiring significant effort and time. Consequently, we sought an alternative method that would leverage machine learning techniques to automatically detect patterns. This approach aimed to streamline the pattern detection process and improve efficiency.

The foundation of the clustering approach revolved around grouping patterns together. The underlying premise was that if the market exhibits recurring patterns, these patterns should occur more frequently than random patterns. By utilizing distance formulas such as dynamic time warping (DTW), we aimed to identify these recurring patterns regardless of their temporal intervals. Moreover, we expected that these patterns would exhibit a coherent merging pattern in terms of their occurrence count, facilitating effective clustering.

6.1.1 Feature extraction using pattern detection

In the processing chapter, a processor was developed to generate events that will serve as features for the clustering algorithm. Event detection is performed using a dedicated script, which identifies specific events that are subsequently divided into windows. Each window represents a pattern, and the goal is to merge these patterns to identify common patterns. The main objective of the clustering algorithm is to identify common patterns within a set of random patterns.

To initiate the event generation process, a detector was created to identify fundamental events commonly used in trading. These events include ema-cross, bolling-band hits, and rsi lower and upper band hits. Each event can be detected by implementing a corresponding event detector that checks for specific conditions. For instance, in the case of ema-cross, the event detector would identify instances where a longer ema crosses a shorter ema, labeling it as an up-cross or down-cross depending on the context (Figure 6.1).

Figure 6.1: Green and red triangles represent up and downtrends, based on the crossing of the EMA indicator lines. The lines are drawn as indicators, and the triangles are output from the detector.s

6.1.2 Pattern formation using event sequences

With the available events, they are partitioned into windows, where each window consists of a specific number of sequences. In this example, there are four sequences for each type of indicator, and the events within these sequences are denoted as integers. The timeline of events is divided into a predetermined number of days corresponding to the window size, with an overlap of a specified number of days to ensure comprehensive coverage. Positive market events are represented by the integer 1, negative events by -1, and the absence of an event by 0. This results in straightforward sequences of data that can be efficiently processed by

machine learning algorithms, utilizing this abstract data to draw meaningful conclusions.

The decision to represent events as integers stems from their significant performance improvements when employing the distance algorithm. Although alternatives, such as using floats, were considered, the choice was made to represent events as integers. By utilizing floats, it would be possible to merge algorithms and assign float values to events, indicating a reduced level of certainty when events are combined. However, this would incur additional computational time and increase the complexity of the distance algorithm. While it simplifies the clustering process, it introduces additional overhead.

An alternative approach was contemplated, considering the utilization of floats to encode more data within the sequences instead of discrete events. This additional data could include information such as the distance between exponential moving average (ema) lines to anticipate crosses or the proximity to a new line of the relative strength index (rsi). However, this approach would involve dealing with continuous data, while the intention is to work specifically with events. Consequently, this idea was swiftly dismissed. It is worth noting that such an approach would also complicate data management, as the current implementation employs compressed sparse structures for efficient storage of data on disk. This allows for reduced storage size when handling a large number of patterns. Moreover, when reading this sparse data, empty sequences can be promptly identified, eliminating the need for comparing such sequences, as two empty sequences are known to have a distance of zero. This optimization proves significant as empty sequences occur frequently.

Careful consideration should be given to the selection of events. With knowledge gained from the second approach, there is now a better understanding of which events could be employed successfully. Notably, the consideration of highs and lows, as well as the intersections of price with support and resistance lines, may prove useful in detecting meaningful patterns. While common indicators were initially explored, their effectiveness was limited. It is clear that expecting substantial results solely based on these indicators was overly optimistic.

6.1.3 Calculating distances between patterns

The cornerstone of clustering revolves around computing distances between patterns. As previously explained, these patterns are composed of multiple event sequences, which necessitates a suitable metric for quantifying their dissimilarity. Since we are operating in a non-Euclidean space, the options for selecting distance algorithms are limited. Dynamic Time Warping (DTW), Edit Distance, and Longest Common Subsequence are the most commonly used algorithms for comparing events within this space. Among these options, DTW aligns most closely with the objectives of this thesis, although additional exploration is needed to definitively determine the optimal algorithm.

In the implementation, a variant of Dynamic Time Warping is utilized to minimize memory usage during distance calculations. As each pattern consists of sequences of the same length, memory consumption can be quantified as n^2. However, by leveraging Hirschberg's algorithm, memory usage is reduced to $2 \times n$, resulting in a significant enhancement in efficiency.

DTW offers more meaningful distances between events. Nonetheless, there persist performance issues worth noting. In a bid to boost comparison efficiency, we incorporate a window within the distance formula. This addition ensures temporal alignment of events when matching

patterns. For each event series, distances between patterns are calculated (for instance, comparing the ema-crosses of pattern A with the sequence in pattern B), and the distances of all event series are subsequently aggregated.

The implementation differs from traditional DTW in several aspects:

1. Skipping Empty Sequences: The algorithm features a capacity to skip over empty event sequences in patterns, a function not present in traditional DTW.
2. Edit Distance with Window Constraint: The implementation uses an edit distance calculation similar to DTW but incorporates a window constraint to limit computations within a specific window around the diagonal.
3. Use of Float32 for Distances: The code employs the float32 data type for storing and manipulating sequence distances instead of different numeric types.
4. Existence of Zero Distance Function: The implementation includes a function to check if the total distance between two patterns is zero, a function not commonly found in traditional DTW.
5. Cumulative Pattern Distance Calculation: In our context, patterns are defined based on multi-sequence data. Unlike traditional DTW, which compares only two individual sequences, our implementation calculates the total edit distance between corresponding event sequences in two patterns. This cumulative calculation across multiple sequences adds a layer of complexity, distinguishing it from the standard, single-sequence comparison approach of DTW.

The optimal window size is determined to be 3, as it provides satisfactory performance for the algorithm. However, further experimentation is necessary to identify the optimal window size in a more practical setting.

Additional context regarding other variables is required for a comprehensive understanding of their roles and functions in the implementation.

6.1.4 Bottom-up clustering

In the research, the initial approach employed is a bottom-up or agglomerative strategy for clustering. This approach treats each pattern as a leaf node and iteratively merges nodes to form parent nodes in a hierarchical manner, aiming to identify distinct trading patterns observed in the market.

It is worth noting that these patterns exist within a non-Euclidean space, as opposed to a conventional Euclidean space. To calculate the dissimilarity between patterns, a critical requirement for effective clustering, non-Euclidean distance algorithms discussed earlier in the research are utilized. The computational time primarily revolves around the distance calculations between each pair of nodes. As these calculations are applied to the entire hierarchical tree, the available computational resources are quickly depleted.

Despite concerted efforts to streamline the clustering process and reduce the number of patterns, the outcomes are unsatisfactory. The attempts to enhance efficiency do not yield significant reductions in computational time or substantial improvements in performance. Further details regarding these optimization techniques, along with their limitations and challenges, can be found in Section 6.1.7.

6.1.5 Top-down clustering

In the top-down clustering approach, we adopt a strategy similar to K-Means Clustering. The process begins by initializing a certain number of centroids, which are representative patterns, based on a random selection from the available patterns. Each remaining pattern is then assigned to the nearest centroid based on their similarity. This assignment process is iteratively repeated for each centroid, optimizing the clustering arrangement.

To efficiently distribute the workload across multiple threads, the newly created child centroids are added to a job list that is processed by a number of worker threads. This parallelization allows for faster computation and scalability when dealing with large datasets.

Ideally, to achieve an optimal split, it would be desirable to calculate distances between all patterns. However, this can be computationally expensive. Therefore, a heuristic approach is employed instead. It involves randomly selecting a pattern and using its most distant neighbor to split the cluster. By utilizing this approximation method, we can achieve reasonably good results while minimizing computational costs.

Once the split has been made, distances are calculated for both resulting clusters. The patterns closest to each cluster are then assigned to their respective sets. This process of splitting and assigning patterns is repeated until the desired clustering arrangement is obtained.

In certain cases, there may be patterns that could fit better within a different cluster or should be placed under the same parent cluster. To address these potential errors, an audit-like procedure can be conducted when the clustering has reached a sufficient number of levels. This involves comparing all patterns with each other to identify potential alternative merges, ensuring that the clustering arrangement is refined and accurate.

6.1.6 Optimisations

Given that the total number of patterns in the dataset is unknown, and the objective is to discover new patterns rather than pre-existing ones, we initially attempt to run the algorithm on the entire dataset. However, this approach proves impractical due to the complexity of handling a large number of patterns, which amounted to around 500k+ with our initial parameters of a window size of 14 days and an overlap of 2 days. While we are uncertain about the quantity of patterns of a specific type, it is necessary to discover a significant number of instances of a given pattern to obtain meaningful results. This presents a dilemma where we must make a trade-off between reducing the number of assets or reducing the number of samples, which would also decrease the chances of finding patterns.

Merging identical patterns:

To address the challenge of reducing the number of extracted patterns from the complete dataset, the most effective approach is to merge patterns with a distance of zero. This merging process is performed recursively and can be executed using multiple threads, resulting in a substantial reduction in the number of patterns. During this process, the patterns are converted into nodes, and an associated count is maintained to ensure the preservation of this metric.

Sampling patterns:

To improve memory consumption, another approach involves sampling patterns from the source dataset. By selecting a predetermined ratio of desired samples relative to the total number of samples, we can reduce the working set of patterns. However, it is important to note the reduction in the number of unique patterns. Nevertheless, the resulting groups after merging, as explained in the previous paragraph, remain comparable to the number of nodes.

Unique pattern identifiers:

An attempted optimization that proves unsuccessful in practice is assigning a unique identifier to each pattern using a hashing algorithm. The aim is to simplify the merging of equivalent patterns. However, due to the temporal warping of patterns, accounting for this aspect in the hashing process is deemed infeasible, leading to the abandonment of this optimization approach.

Summary:

A combination of merging identical patterns and sampling proves effective in reducing the number of patterns to a manageable working set, enhancing computational efficiency. However, the effects of sample size should be further explored to determine an appropriate ratio.

Despite the optimizations applied to clustering time, the process remains insufficiently fast for efficient hyperparameter optimization. The identified list of parameters includes:

- Number of indicators.
- Window size of the pattern.
- Overlap between windows.
- Resolution of the candles (e.g., 1h vs 30m).
- Parameters for specific indicators.
- Correlation between indicators.
- Interval of indicators.
- Window size for distance calculations.
- Given the extensive nature of these parameters, a highly efficient clustering approach would be preferable to facilitate the optimization of hyperparameters. This efficient approach would enable the identification of useful patterns and the reduction of noise in the patterns discovered through educated guesswork. By efficiently optimizing these hyperparameters, we can enhance the effectiveness of pattern identification and analysis in the context of the research.

6.1.7 Limitations and decision to change approach

Despite initial efforts to identify patterns, challenges are encountered that hinder successful implementation. Contrary to initial expectations, a multitude of patterns with high occurrences are discovered rather than a limited set of common patterns. This observation prompts a reevaluation of the methodological approach, recognizing that the primary objective is to effectively identify and comprehend trading patterns.

Both the bottom-up and top-down clustering approaches pose difficulties in terms of computation time and memory usage, rendering clustering-based detection impractical for the specific aims. Notably, the extracted patterns lack discernible and interpretable structures, appearing more random in nature rather than indicative of meaningful market behaviors (see Figure 6.2 for an example of a pattern obtained through top-down clustering).

In light of these challenges, and to maintain focus on the central research question regarding the utility of trading patterns, a decision is made to explore alternative methods for pattern detection. The hope is that these alternative approaches provide greater clarity regarding the appearance and behavior of trading patterns in the market, potentially yielding valuable insights that may inform a future reconsideration of the clustering approach.

```
60467735c9c11e3cb4812dfc67ca0f40 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ 1
d6be4dbc547c02479c4ca65cdfe24aeb _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
f2d4187073c52fcd2e11ae814ccc90b2 _ _ _ _ 1 _ _ _ _ _ _ _ _ _ -1 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

Figure 6.2: A pattern as a result of top-down clustering. It has 192 occurrences in 52584 total samples. No clear structure can be seen.

The underlying philosophy centers around enhancing pattern detection and visualization accessibility for retail traders. While the identified limitations could potentially be addressed by allocating additional resources to improve the performance of the clustering algorithm, such a solution would inadvertently restrict access to this innovative approach, as it would only be viable for individuals or organizations equipped with the necessary resources. The proposed alternative pattern detection method ensures broader accessibility by being compatible with standard computer systems, enabling real-time pattern detection without imposing significant resource requirements.

6.2 Approach 2: Script-based pattern detection

A second approach to detecting algorithms in the stock market is by trying to detect the algorithms programmatically. This is in contrast with the previous approach, in which we wanted to detect the algorithms automatically. When detecting the algorithms automatically, we did not have a clear view on what the outcome might be, so working in reverse means that we know what we are trying to detect, and have to write a detector starting from the shape of the pattern we are trying to find.

Since there are many patterns in the market, we have selected a few patterns to detect. The most basic patterns that we are trying to find is the double bottom and double top pattern. From the literature, one would expect a move after these patterns, in the case of a double bottom, this would be an upwards move, and vice versa for the double top pattern.

When we have a double top detector, we can easily adapt this detector to detect double bottom patterns. With a bit more complexity, we can also identify triple top and triple bottom patterns, as well as head-and-shoulder patterns. We will explain how we can detect those patterns easily with the finished pipeline.

Once we have a good detector, we can start validating the occurrences of these patterns on the whole of the dataset and validate whether these patterns have any significant predicting

power, and thus answer the question of this thesis.

6.2.1 Dealing with time warping

Before we embark on detecting patterns in the stock market, it is crucial to understand how patterns might appear from the perspective of different horizons. Traders that are using a large horizon will use large time intervals to base their decisions on, and vice-versa for traders with a small horizon, which will use smaller time intervals. Using smaller horizons often introduces more noise and volatile movements, which directly translates into more pattern occurrences. Detecting more patterns does not mean better, since signals are harder to interpret at smaller time intervals.

Detecting patterns using a clustering technique avoids this problem of temporality, with dynamic time warping making it possible to merge patterns that might have been warped in time. The clustering detector would produce results, regardless of the interval of the candles. One can think of DTW detecting patterns regardless of the granularity of the candles, where the granularity is how many candles of a specific interval (in the clustering examples, daily candles) contribute to a given pattern. A higher granularity would therefore mean that a pattern would consist of many candles (see Figure 6.3).

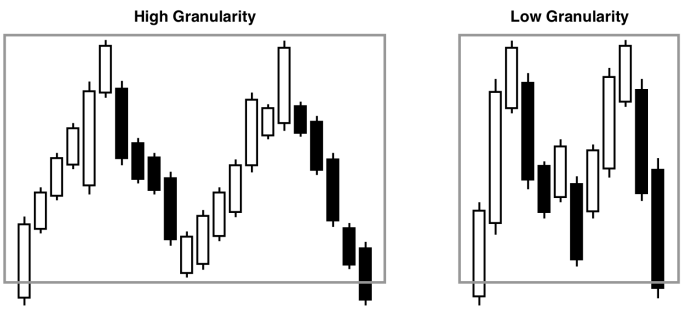


Figure 6.3: A double top pattern with different granularity

When detecting patterns manually, we need to be able to detect patterns with any granularity. One way of achieving this is to run the pattern detector at different intervals (e.g., daily, 3 daily, weekly, biweekly, etc. . .). Another way is to use hyperparameters in the algorithms to detect patterns with more or less granularity. This way we can always use the same interval for the candles, but are still able to detect patterns with different granularity.

To summarize, just as we exclusively use daily candles in the clustering algorithms, we will also exclusively use daily candles in the manual pattern detection. This removes the need of evaluating multiple intervals. In order to still find candles in larger time frames, we will also introduce this concept of granularity to the manual pattern detector.

We explicitly state the interval of candles in the following sections. If there is nothing mentioned regarding the interval, it is the daily candle interval that is being used.

6.2.2 Highs and lows detection

In order to detect all the common patterns, we need to identify the highs and lows. These points are considered resistance and support prices, prices at which buyers or sellers often place orders, and are the basis for lines that are commonly used in technical analysis.

Highs and lows are nothing more than local maxima and minima detection. We use a certain window size, and try to find the maxima and minima based on the price of candles. Since candles have 4 data points, we will use the lower price of the candle for the minima and the higher price for the maxima.

Say we have a window of seven, then checking if a given candle is a maxima is just as simple as looking if the three candles to its left and three candles to its right have high prices that are lower. The same logic applies to finding the minima. An example of this can be seen in Figure 6.4.

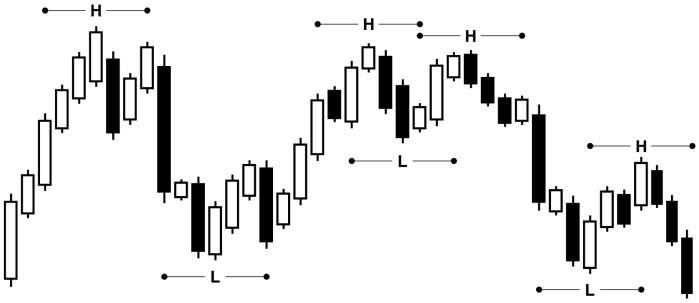


Figure 6.4: Highs and lows with a range of 3

To implement this algorithm, we make use of a first-in-first-out stack (FiFo). Each time the algorithm steps through one candle, we push the candle to the stack. If the stack is full, then the oldest candle will be removed. To find the high and low candles, we take the middle candle of the stack, which will be the candidate. If all other candles in the stack are lower, then we can mark the candidate as low. If all other candles in the stack are higher, then we mark the candidate as high.

Since we cannot look into the future, we cannot determine if the most recent candle will be the top. That means, depending on the range of the algorithm, the highs and lows will lag more as the range increases. Choosing a higher range, however, means that the highs and lows will be stronger indicators that could produce more reliable signals. We will evaluate the effect of range in the chapter discussing the results.

With the high and low events extracted, we can implement all the basic pattern detectors that we have planned to detect. Since these detectors only use these events as their input, the granularity of the patterns will depend fully on the range that is set for the highs and lows detection. For instance, if one uses a large range for detecting highs and lows, it will directly translate to patterns being detected over a larger timeframe, and thus have a higher granularity.

By sweeping the range of the high and low detector, we can detect all patterns in all temporal

spaces. We therefore solve the concern of not being able to detect all the candles when we were limited to one interval. Since the pipeline is fast enough to generate these events in an instant, as long as ranges are within reasonable bounds, there should be no concern about performance.

A minimum value for the range would be 1, which would make every other candle a possible candidate for a high and low event. This, however, introduces a lot of noise candles, and we recommend using a minimum range of three to reduce the noise. The complexity of detecting a pattern will be $O(r * h)$, with range r and history of an asset h, which means that the speed of calculating these events depends on the range.

There is no theoretical upper limit to the range, but the higher the range, the less relevant the events become due to their lagging nature. For example, if you would take a range of 100, then you would only know after 100 candles if you had reached the top, although it could have some use, the correlation will probably be insignificant. The largest range we used for testing was 40 days, we do not recommend any higher, as this produces very few usable events. We will visualize the number of events in relation to this parameter in the result chapter.

In the testing of this script, we noticed a lot of bad patterns marked as caused by ties in candle prices. When candles have the same highs or lows, as an effect of bad data quality (Figure 6.5) or rare market circumstances (Figure 6.6), they must be omitted from the result. Failing to omit these results could introduce a lot of worthless events that generate incorrect patterns in algorithms that use highs and lows as their basis.

Figure 6.5: Candle ties because of a lack of data accuracy. Old stock data is sometimes not accurate enough for daily candles, so the weekly range is spread over daily candles.

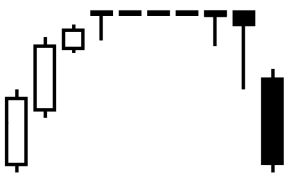


Figure 6.6: Candle ties because of rare market circumstances. A lot of selling pressures caused a halt in price movement.

We can simply ignore ties, by making sure that the high or low does not share a price with any other candle in its stack. This is sufficient for stock data. For currencies that use more

precision, we would add a threshold, but since this is outside the scope of the thesis, we will not discuss it further.

Now that we have highs and lows, we should be able to detect most patterns that are based on horizontal resistance/support lines. These include: head and shoulders, inverse head and shoulders, double top, double bottom, triple top, and triple bottom.

6.2.3 Pattern detection based on highs and lows

With the highs and lows, we can start detecting the most basic of trading patterns. We will be discussing double bottom, double top, triple bottom, triple top and heads-and-shoulders. These can be identified just by using high and low data points and the neighboring candles. All these trading patterns have very similar shapes, and can therefore be identified with almost the exact same algorithms. We will first discuss double bottom, and then explain the changes needed to detect any of the related patterns.

Double Bottom

Double bottom pattern is a common pattern used to identify a trend reversal (Figure 6.7). It consists of two bottom points, one top, and entry and an exit. These make up a "W" shape. The bottom points form a support line, and the top forms a resistance line. At the support line, it is assumed that there are a lot of buyers, and hence the price will not go any lower. The same can be said for the resistance line. Once the resistance is broken, meaning that the price jumps above that level, the trend is predicted to continue upwards.

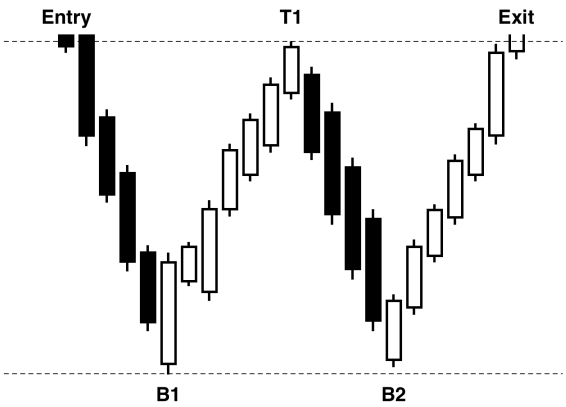


Figure 6.7: Double Bottom Pattern

We will start by abbreviating a couple of words in the explanation:

- $B1$: The first bottom of the pattern.
- $B2$: The second bottom of the pattern.
- $T1$: The first and only top of the pattern.
- $Entry$: The first candle of the pattern, and the first candle touching the resistance line to the left of B1.
- $Exit$: The last candle of the pattern, and the first candle touching the resistance line to the right of B2.

The double bottom pattern can thus be described as the following sequence: $Entry \rightarrow B1 \rightarrow T1 \rightarrow B2 \rightarrow Exit$, with no intermittent events. Since price data is noisy, and far from exact, we also introduce a *threshold* parameter to allow some margin when determining the support and resistance line (Figure 6.8). *threshold* describes how much percent $B1$ can differ from $B2$, and how much *Entry* and *Exit* can differ from $T1$. We took a *threshold* value of 1%, as making it much larger will start to produce wrong pattern results (Figure 6.9).

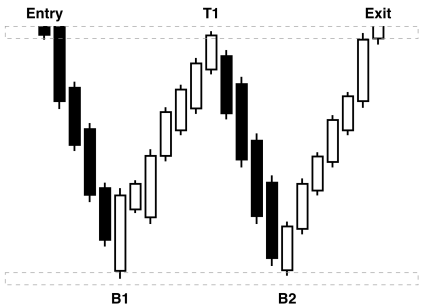


Figure 6.8: A threshold box is used to allow some variation between tops and bottoms.

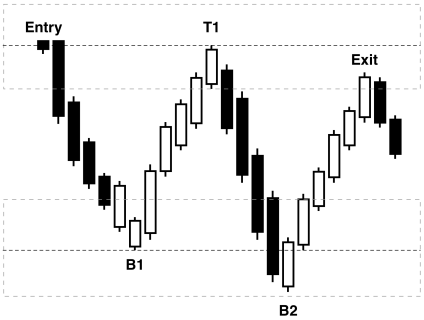


Figure 6.9: Since this threshold is too large, this pattern is more likely a falling-wedge than a double bottom pattern.

We will first start by detecting the sequence *bottom*, *top* and *bottom*. We allocate a small stack in the detector's memory. The size of this stack is 3 items, and it will contain the last 3 events that we parse from the highs and lows detector. If these events are the sequence that we are looking for, then we can start looking for *Entry* and *Exit*.

Even if there is no high or low event, we will still attempt to find a new *Entry* and *Exit*. *Entry* can be known at any time if we have matched the top and bottom sequence, but *Exit* might not be known if that candle is still in the future. The full detection script looks as follows:

1. Push any high and low event to the stack of 3 items
2. If the stack does not match $bottom \rightarrow top \rightarrow bottom$, we exit
3. If $B1$ and $B2$ differ more than *threshold*, we exit
4. Iterate all candles to the left of $B1$, if one is lower than $B1$ we exit, if one is higher than $T1 - threshold$ it is *Entry*

5. Iterate all candles to the right of $B2$, if one is lower than $B2$ we exit, if one is higher than $T2 - threshold$ it is $Exit$
6. Create an event with $Entry$, $B1$, $T1$, $B2$, $Exit$

If during the iterate step, we run out of candles, we exit. Missing candles are skipped at any time. Once we exit, the algorithm daemon will go to the next candle, and the process starts again from step 1.

When looking at the results, we identified some problems with patterns being recognized in areas with anomalies. This is an effect of the highs and low algorithm being conservative in detecting highs and lows. A missed high or low can cause us to match the wrong shape as a pattern (Figure 6.10). We add an extra rule that whilst looking for the $Entry$ and $Exit$, we are not allowed to go higher than $T1 - threshold$.

Figure 6.10: A sudden spike in price, that is not detected as a top, since it has multiple tops sharing the same price.

The pattern gets added to the output events at the moment that we have identified the exit. This means that the patterns can only be evaluated from the moment the timestamp of the exit candle.

Double top

Double-top pattern is the complement of the double bottom-pattern (Figure 6.11). It consists of two tops, one bottom, and associated entry and exit points, forming an "M" shape. The tops act as resistance levels, while the bottom acts as a support level. The logic for this algorithm is exactly the same as for double bottom, and can be implemented, keeping in mind that lower, higher, top and bottom are to be switched around.

Triple Bottom

The triple bottom pattern is an extension of the double bottom pattern (Figure 6.12. It introduces an extra top $T2$ and bottom $B3$. Instead of searching for the exit starting from $B2$, we start from $B3$. All the other logic remains the same as for double bottom. Because the initial steps of detecting the pattern are equivalent with the double bottom detector, one will always find a double bottom pattern where a triple bottom pattern is. The third trough will not contain a double bottom pattern, and thus could prove triple bottom to still be useful in these exceptions.

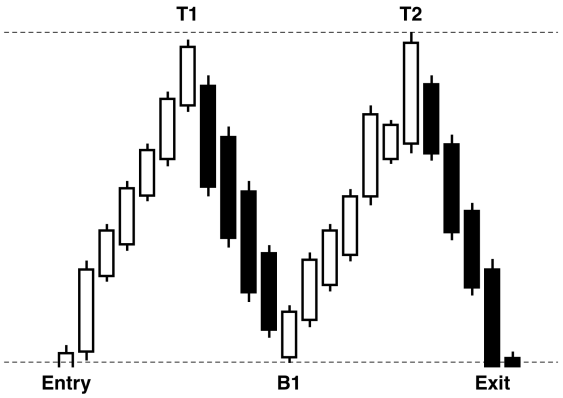


Figure 6.11: Double top Pattern

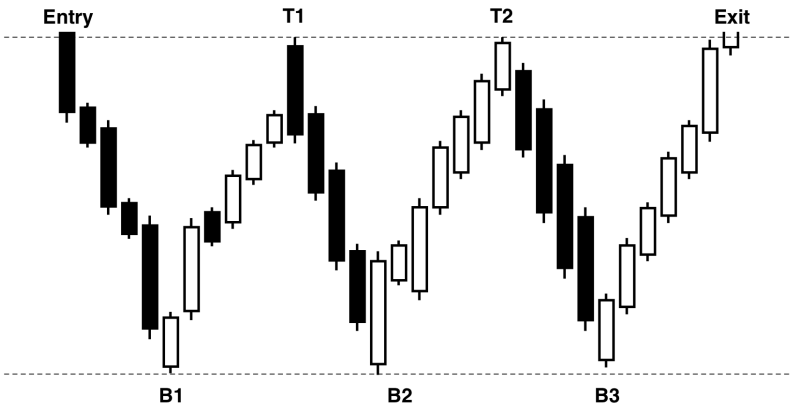


Figure 6.12: Triple bottom Pattern

Triple Top

This is the complement for triple bottom (Figure 6.13). We introduce $T3$ and $B2$, and look for the *Exit* starting from $T3$.

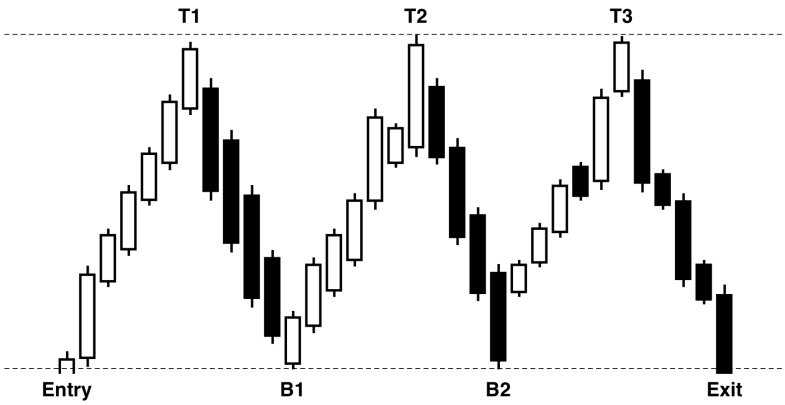


Figure 6.13: Triple Top Pattern

Head and shoulders

Head and shoulders start from the logic of triple top (Figure 6.14). We add one extra condition: $T2$ has to be some significant threshold S above $T1$ and $T3$, otherwise we exit. The algorithm would look as follows:

1. Push any high and low event to the stack of 5 items
2. If the stack does not match $top \rightarrow bottom \rightarrow top \rightarrow bottom \rightarrow top$, we exit
3. If $T1$ and $T3$ differ more than *threshold*, we exit
4. If $T2$ is not higher than $T1 + S$, we exit
5. Iterate all candles to the left of $T1$, if one is higher than $T1$ we exit, if one is lower than $B1 - threshold$ it is *Entry*
6. Iterate all candles to the right of $T3$, if one is higher than $T3$ we exit, if one is lower than $B2 - threshold$ it is *Exit*
7. Create an event with *Entry*, $T1$, $B1$, $T2$, $B2$, $T3$, *Exit*

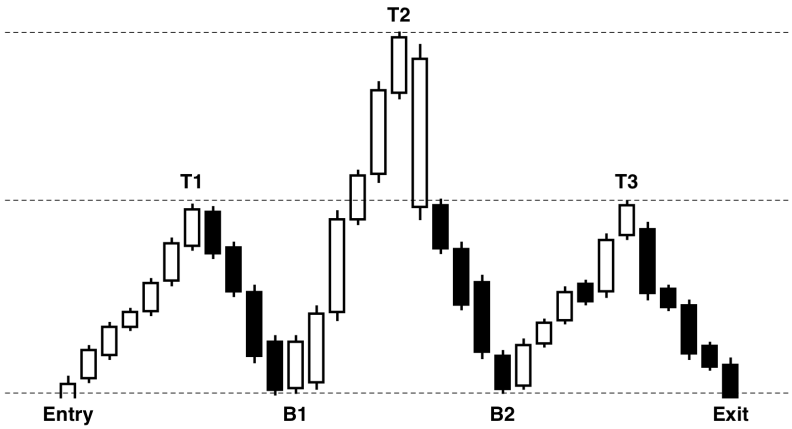


Figure 6.14: Head and Shoulders Pattern

Chapter 7

Results and discussion

The evaluation of trading patterns can be approached in multiple ways. One method involves incorporating them into a trading strategy. However, there's a prevailing doubt about whether trading patterns alone can be an effective strategy. It is challenging to evaluate their influence on price movement without bias. This research aims to determine if these patterns indeed have a significant impact on future price movements, especially when compared to random points in time.

Financial markets are often characterized by randomness, but with a slight tendency to move upward over time. If one were to pick many random time points, the average future movement from these points would be expected to be a little above zero. If trading patterns are detected, one might expect a noticeable departure from this average. But the extent of this expected departure remains a subject of debate. To address this, this thesis examines time points associated with trading patterns and compares them to those chosen randomly.

In the research, we refer to these time points as "events." We'll discuss both "pattern events" and "random events." A substantial difference between the average outcomes of pattern events and random events would warrant a closer look at existing research on the topic. We'll then see if these findings align or contradict previous claims about the utility and predictability of trading patterns.

7.1 Overview of Approaches

7.1.1 Approach 1: Clustering-based pattern detection

The preliminary focus of the investigation involved a clustering-based technique for pattern detection. This method was initially chosen to counteract the labor-intensive nature of manual pattern detection by employing machine learning algorithms to identify trading patterns automatically. The fundamental premise behind this approach was the idea that recurring patterns in the market should be more frequent than random ones, and therefore, could be effectively grouped together.

Dynamic time warping (DTW) was employed as a tool to identify these recurrent patterns, independent of their temporal intervals. This method was designed to streamline the pattern

detection process and enhance efficiency, on the assumption that these recurring patterns would display a consistent merging pattern in their frequency of occurrence.

Nevertheless, this initial approach was met with significant challenges that impeded its successful execution. Contrary to the original assumptions, the clustering method yielded numerous high-occurrence patterns rather than a select group of common ones, which required a reassessment of the methodological strategy.

Both the bottom-up and top-down clustering strategies posed computational and memory usage challenges, making this method impractical for the aims. Furthermore, the patterns identified via this approach lacked clear, interpretable structures and appeared more akin to random noise than meaningful market behaviors.

In response to these challenges and in pursuit of the central research question concerning the effectiveness of trading patterns, we decided to explore alternative pattern detection techniques. These new approaches aimed to provide a more nuanced understanding of the behavior and appearance of trading patterns in the market, potentially offering valuable insights that might inform a future reconsideration of the clustering method.

7.1.2 Approach 2: Script-based pattern detection

In the second approach to detecting stock market patterns, a script-based detection method was utilized. Unlike the first method which relied on automatic detection, this approach specifically targeted known patterns using tailored scripts.

The choice of patterns focused specifically on the double bottom and double top patterns, as these are often associated with reversal signals in the market. These patterns were chosen for three primary reasons: their frequent occurrence in market data, their simple and discernible structure, and the prevailing consensus in market literature that they typically indicate a reversal in price movements, such as a change from a downtrend to an uptrend, as highlighted in the literature review.

Once the script-based detector was operational, the dataset was scanned for these patterns. This process allowed for an assessment of their predictive strength, thus addressing this thesis's main question - the efficacy and relevance of these trading patterns.

Findings indicated that detection accuracy can fluctuate based on market horizons. Specifically, shorter horizons may introduce additional noise and volatility, leading to an uptick in pattern occurrences. Yet, even with these potential hurdles, the script-based approach consistently identified patterns within the market data.

Central to the script-based method is the identification of market highs and lows – crucial for pinpointing patterns. While the technique faced challenges, such as integrating various data sources and managing anomalies like tied candles (candles that have identical bodies and wicks, often due to data gaps being filled with duplicated candles from earlier periods), it effectively sidestepped these issues, preventing the incorporation of incorrect patterns.

In summary, the script-based method successfully detected five prevalent trading patterns: double bottom, double top, triple bottom, triple top, and heads-and-shoulders. An in-depth analysis of these patterns' predictive abilities is the focus of the upcoming sections of this thesis.

7.2 Methodologies and Metrics

7.2.1 Labeling Methods: Triple Barrier vs Others

This thesis does not focus on trading strategies, where metrics such as risk-adjusted returns and maximum drawdowns would be of high interest. However, we need a method to gauge the impact of a specific event. Existing evaluation techniques used in trading strategies can be repurposed for our needs by treating each event like a trade. In this case, we limit the focus to understanding whether the events significantly differ from what would be expected by chance, without the need to consider extra factors like risk and fees, which are usually important in trading strategies.

Among the techniques used in trading strategy evaluations, the fixed-time horizon method (Figure 7.1) is often used due to its simplicity. This method involves making a trade at a specific time and then evaluating its performance after a set period (e.g., one week). Despite its widespread use as a performance measure for trades, it is not ideal for evaluating trading models due to several reasons:

1. Market volatility can distort statistical properties

Market volatility can lead to significant price changes. Consequently, times of high volatility can lead to larger wins and losses. If the evaluation period overlaps with such volatile periods, it may distort the results.

2. It does not reflect real-world trading conditions

In real-world trading, stop-losses are used to limit losses for a trade. If the market suddenly drops but quickly recovers, the evaluation will not be affected, even though the trade might have resulted in a loss. This can result in underestimating losses.

3. It does not provide information about trade direction

This method calculates average returns over a set period but does not provide any information about the direction of a trade.

4. It is limited in granularity as it requires checking multiple horizons

To capture all possible scenarios, one needs to consider all horizons. If only a single horizon is used, key information could be overlooked.

The fixed-time horizon method, despite its simplicity and popularity, has limitations when it comes to evaluating trading models. As a result, we have chosen not to use this method in the analysis.

To address many of the issues prevalent in the fixed-time horizon method, we propose the implementation of the triple barrier method. This method incorporates three barriers: two horizontal (upper and lower) and one vertical (Figure 7.2). The horizontal barriers are set based on a specified threshold, functioning as stop-loss and take-profit limits. The vertical barrier is determined by a time interval. This enhancement over the fixed-time horizon method, which solely employs a time-based vertical barrier, introduces two supplementary horizontal barriers. These additional barriers offer increased insights into the trajectory of the event, thereby allowing us to classify the outcomes of patterns into one of three categories, based on the first barrier breached.

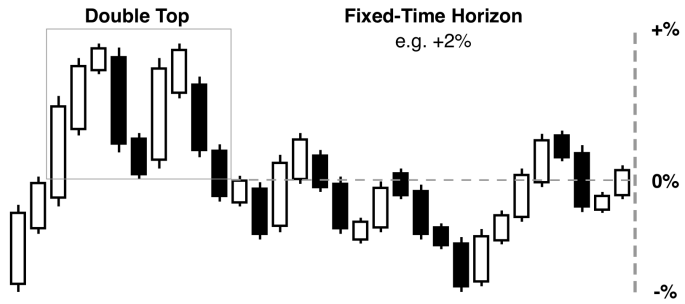


Figure 7.1: Fixed-time horizon evaluation. The low was not captured since we only evaluate at the vertical barrier.

Employing the triple barrier method yields categorical or discrete labels, thereby providing a clearer distinction between random and pattern occurrences compared to percentage-based returns. While the analysis does not make use of the fixed-time horizon method, we recognise that it is essential for the findings to be comparable with other research using this method.

The triple barrier method addresses and resolves several challenges associated with the fixed-time horizon, as detailed below:

1. **Independence from Price Values:** Instead of relying solely on raw price values, the method uses the direction of the price change. This approach offers a less biased representation, as it does not get overly influenced by the magnitude of price changes, but rather focuses on their directional trend.

2. **Accounting for Market Crashes:** With the lower horizontal barrier in place, sudden market drops (which often lead to stop activations in traditional trading) are aptly accounted for. This barrier essentially acts as a safety net, capturing sharp negative price swings.

3. **Incorporation of Directionality:** Unlike the fixed-time horizon method, which mainly focuses on magnitude or average returns, the triple barrier method includes directionality. This feature informs whether the price escalated upwards or took a downward trajectory.

The addition of directionality is particularly crucial. It is not just about discerning whether an event has influenced the price; it is also about understanding *how* it influenced it. This is where the concept of directionality shines. Instead of merely calculating the net positive or negative returns, directionality dives deeper, providing a comprehensive perspective of price behaviors throughout the event's duration.

In unpredictable and volatile markets, prices can shift dramatically and swiftly. In such scenarios, directionality becomes even more paramount. By focusing solely on returns or price values, one might mistake these quick, intense price swings as mere market noise. But with the directionality feature of the triple barrier method, these fluctuations are interpreted in the context of the event, revealing whether they are genuine reactions to the event or just market anomalies.

In essence, integrating directionality does not just offer a binary view (impact or no impact).

Instead, it provides a nuanced understanding, elucidating the specific direction of the influence, thereby deepening the comprehension of how events sway price movements.

One challenge that persists with the triple barrier method, similar to the fixed-time horizon method, is the need to evaluate across multiple time horizons. In real-world scenarios, holding a position indefinitely is unrealistic due to associated costs such as fees. However, by delivering data across multiple horizons, we can provide sufficient insights for those seeking to incorporate patterns into their existing trading strategies. To enable robust comparisons, we will also report the proportional differences and effect sizes in the findings.

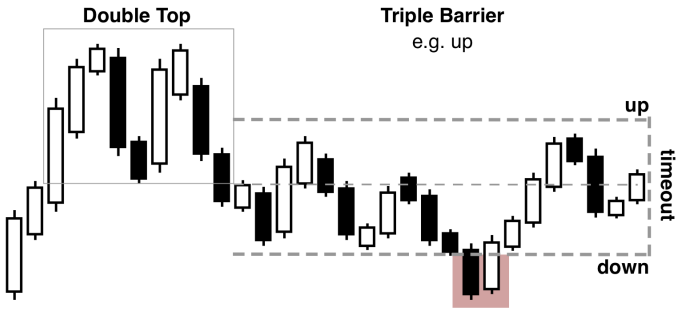


Figure 7.2: Triple barrier method. We broke the lower barrier, so this example would be labeled "down".

One aspect of the triple barrier method that we must account for is that the vertical barrier might be omitting some price dynamics. In order to evaluate the effect with timed-out trades, we will create a hybrid of the triple barrier method and a bucket-based classifier (Figure 7.3).

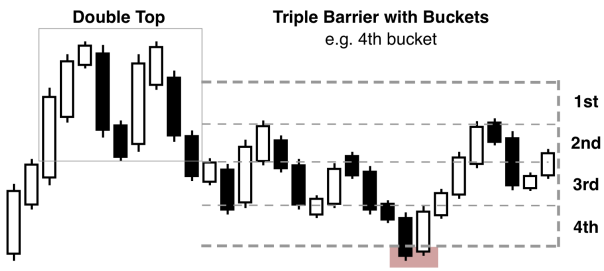


Figure 7.3: Bucket-based classifier. We broke the lower barrier, so we classify in the lowest/4th bucket.

One consideration while employing the triple barrier method is the potential omission of certain price dynamics due to the presence of the vertical barrier. To ensure an effective analysis of trades that reach the time-out phase, we introduce a hybrid approach that combines the triple barrier method and a bucket-based classifier (Figure 7.3).

The bucket-based classifier, diverging from the other methods, does not rely on predetermined barriers or a fixed time interval. Instead, we divide outcomes based on a specified number of buckets, each representing a distinct range of outcomes. For instance, if we use a threshold of

5%, then a bucket would range from -5 to -2.5, -2.5 to 0, 0 to 2.5, and 2.5 to 5. This allocation system ensures that all trades find their place within a bucket.

The outcomes of the triple barrier method that strike the top or bottom barrier are allocated to the highest or lowest bucket, respectively. When an outcome hits the vertical barrier, we categorize the event into a bucket based on the nature of its exit. This approach allows us to capture the price dynamics potentially overlooked by the triple barrier method, while still leveraging the clear directionality it provides.

Using this hybrid bucket classifier helps circumvent the limitations of the fixed-time horizon method, while preserving the realistic assessment capabilities of the triple barrier method. We present the findings predominantly using the triple barrier method, given its comprehensive informational value. Outcomes from the bucket-based classifier will also be included for comparison, but will be discussed in detail only when they provide unique insights.

However, it is worth noting that while the bucket-based classifier enables a finer granularity of outcome classification, the benefits it offers over the triple barrier method are not as substantial as one might expect. As we will detail in future sections, the increased effort required to store and interpret the bucket-based data does not necessarily yield proportionally higher benefits. Therefore, for practical purposes, the triple barrier method still remains the primary evaluation technique.

With these two evaluation techniques - the triple barrier method and the hybrid bucket-based classifier - we are now well-equipped to label outcomes for both pattern and random events. These labels provide us with a concrete foundation to make a statistical comparison, thereby helping us understand if a pattern indeed has any substantial influence on future price movements.

7.2.2 Selection and Justification of Performance Metric

In this thesis, we assess performance using a simple yet informative metric: the win rate. This is a commonly used performance metric in pattern-based trading and provides us with a clear understanding of the strategy's effectiveness.

The win rate serves as a performance metric, calculated by the ratio of successful trades to the total trades made. When no trades are executed, the win rate defaults to 0, indicating inactivity. A win rate of 50 represents a break-even scenario—equal counts of wins and losses. Any score surpassing 50 highlights a majority of wins, while an impeccable score of 100 signifies every trade being successful.

We can formalize the calculation of the win rate using the Performance function outlined in Algorithm 9.

Algorithm 9 Calculate Win Rate

```
function PERFORMANCE(wins, losses)
    trades ← wins + losses
    if trades = 0 then
        return 0
    end if
    winRate ← wins/trades
    return winRate
end function
```

Performance metrics like the win rate are quintessential for evaluating trading strategies such as the triple barrier method and the hybrid-bucket approach. In the case of the former, emphasis is placed on the frequency at which the upper and lower barriers are breached. Conversely, the hybrid-bucket approach prioritizes the final placements of trades in specific buckets, especially the extremities.

Despite its simplicity, the win rate provides us with a valuable measure of the pattern detection strategies' success. However, it is important to note that this win rate does not directly translate into potential profits. A high win rate does not necessarily yield high returns, especially if the profit per win is small. Consequently, the results should be interpreted as indicative of performance, not direct estimates of potential profits.

Moreover, we consciously avoid discussions of returns or profit percentages in the results section for two main reasons. Firstly, a trading strategy based solely on patterns is generally suboptimal. Secondly, return rates could be misleading due to the aforementioned considerations.

The subsequent sections will delve into the detailed performance of individual patterns and random events, with a comparative analysis illustrating the difference between the two. These insights will feed into the final conclusions on the implications of using patterns in trading strategies.

7.2.3 Understanding the Timeout Limit

The timeout limit plays a vital role in adding a sense of realism to the trading simulations. In real-world scenarios, trades often cannot run indefinitely due to constraints such as margin calls and trading fees. Incorporating a timeout limit not only enhances the realism of the simulations, but it also enables us to extrapolate the average duration of a trade. This, in turn, informs us about the frequency at which a trade could be practically made given the observed win rate.

In addition to pursuing realism, the timeout limit plays an integral role in the reliability and the interpretability of the results generated by the Triple Barrier method. When trades are stopped due to reaching the timeout limit, they are excluded from the final results of the evaluation. If the timeout limit is set too low, a large number of trades may be stopped prematurely, resulting in a significant portion of the data being omitted from the final results. This omission could potentially skew the results and lead to misleading conclusions about the efficacy of trading patterns.

On the other hand, if the timeout limit is set too high, the trades may not reflect practical

trading scenarios, thereby reducing the real-world applicability of the results. When the timeout limit is excessively high, nearly all trades reach completion, thereby diluting the differentiating power of the Triple Barrier method.

In this thesis, we carefully evaluated the impact of different timeout limits. To ensure a comprehensive analysis, we adopted a linear range of values as the timeout limit. The selection of these ranges was informed by the principle of balancing between an adequate sample size and the degree of realism the timeout limit imparts.

As the timeout limit decreases, trades are more likely to be prematurely terminated, leading to an increased rate of stopped trades. Conversely, as the timeout limit increases, trades have a higher chance of reaching completion, reducing the differential effect of varying the timeout limit.

To systematically understand the interplay between the timeout limit and the trade outcomes, we plotted the trade sample size as a function of time, condensing the various thresholds into a single continuum (see Figure 7.4). This allowed us to observe the trends more comprehensively.

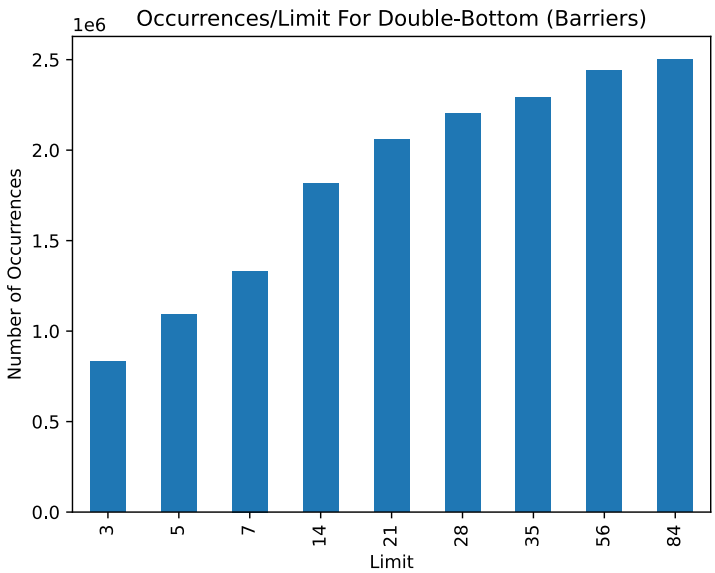


Figure 7.4: The occurrences for the barrier evaluator in function of timeout limit.

The resulting plot, specifically for the Triple Barrier method, exhibited a logarithmic relationship. This was anticipated as the number of patterns increased with the extension of the timeout limit. A critical observation was the substantial increment in the number of occurrences when transitioning from a 7-day to a 14-day timeout limit. Given that the 14-day limit accounted for more than half of the observed patterns, it was deemed a suitable choice for subsequent analyses.

A comparative evaluation of the 7-day and 14-day timeout limits indicated a marginal difference in the derived win rates (see Figure 7.5). Furthermore, when examining the win rates between the 14-day and 21-day limits (refer to Figure 7.6), the disparity was found to be even smaller. These observations provided further validation of the decision to adopt the 14-day limit,

suggesting that it was a sufficiently representative for this thesis's objectives. Thus, the analysis highlights the critical role of the timeout limit in assessing trade performance and how the choice of limit can influence the interpretation of results.

Figure 7.5: Difference in win rate for double bottom with timeout limit 7 and 14. A positive value is limit 14 having a higher win rate than limit 7.

In contrast to the Triple Barrier method, the bucket-based evaluator does not discard any trades; instead, it simply assigns them to their respective buckets regardless of whether they timed out. This results in a consistent number of trades across all timeout limits (see Figure 7.7). While at first glance this might seem advantageous, it also poses a challenge in determining an appropriate timeout limit. This information is crucial when applying the Triple Barrier method in real-world scenarios, as it informs us about the practicable range for the timeout limit.

It is important to remember, however, that these observations are heavily dependent on the hyperparameters employed in the simulations. For instance, if the time granularity of the data were changed from daily to hourly candles, the suitable timeout limit would also likely decrease correspondingly. Therefore, while the findings provide some general guidance, the choice of timeout limit must be tailored to the specificities of each trading strategy and data set.

7.2.4 Event Frequency Analysis: Validating Validation Scripts

Before conducting the performance analysis of patterns, the discussion initially focuses on the parameter employed for detecting highs and lows. This parameter, termed as 'range', serves as the sliding window for identifying local minima and maxima. Through testing, it becomes evident that this range parameter significantly impacts the types of patterns detected. No single value manifests as universally optimal for the detection of all patterns. Therefore, to understand the variety and number of patterns, an iteration over a set of values is performed

Figure 7.6: Difference in win rate for double bottom with timeout limit 14 and 21. A positive value is limit 21 having a higher win rate than limit 14.

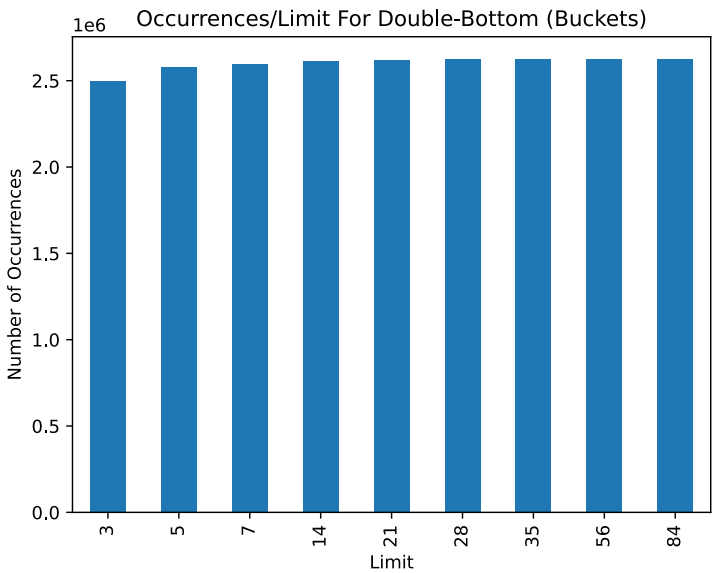


Figure 7.7: The occurrences for the bucket-based evaluator in function of timeout limit.

to ascertain the patterns detected for each range parameter.

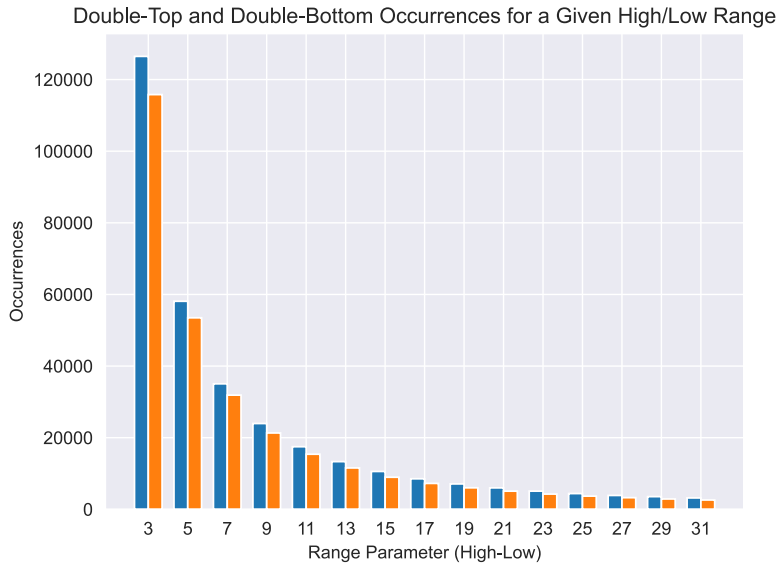


Figure 7.8: Occurrences of pattern for each range parameter.

Figure 7.8 presents an exponential decline in pattern detection as the step size (or range value) increases. The chart spans a range from step size 3 to step size 31.

At the onset, with a step size of 3, there's a peak in pattern detection with around 120k samples. Yet, as we progress, each increment in the step size by 2 leads to a substantial drop in detected patterns: 60k at a step size of 5, 37k at a step size of 7, and so forth.

To elucidate this trend, let's delve into the mechanics:

1. **Persistence of Minima and Maxima**: "Minima" and "maxima" denote the lowest and highest points within a specific data window. As the size of this window (the range) adjusts, these low and high points can remain consistent. In other words, when varying the range, the same patterns formed by these points may be identified in multiple range configurations.

2. **Recurring Patterns**: Given the persistence of certain minima and maxima across range adjustments, some patterns may reappear across different step sizes. These patterns are not confined to a specific range; they can manifest across a spectrum of configurations.

3. **Impact of Broader Ranges**: While the recurrence of some patterns can be observed, an expansive range may miss out on detecting more intricate or short-term patterns. These patterns, clearer within a confined window, might become less pronounced or altogether omitted in a wider scope. This omission becomes more prevalent as the step size increases, contributing to the exponential drop in detected patterns.

In essence, as the window broadens (with larger step sizes), while some patterns persist, others get overshadowed or missed, leading to the overall decline in pattern occurrences as depicted in Figure 7.8.

Informed by these findings, the selected parameters for the final evaluation are set at 3, 5, 7, 9, 11, 13, and 15. The forthcoming analysis and tables are based on these range parameters. It is also noted that although larger range values can expand the detection scope, they simultaneously risk diminishing the predictive power. The process of identifying highs and lows inherently serves as a lagging indicator, thereby introducing a lag equivalent to $range/2$. Consequently, an extensive sliding window might render a pattern obsolete by the time it is detected, a phenomenon that is accentuated in the subsequent results.

Given these considerations, the range is capped at 15, and the results for each range are consolidated in the result table. The final values are discussed in the context of the time limit and threshold range for the triple barrier method.

The discussion then transitions to the examination of the heatmaps and results generated from the detected patterns.

7.3 Data Attributes and Analysis

7.3.1 Skewness in Random Data

Before delving into the analysis of the results, it is imperative to confirm the randomness of the dataset. An initial observation revealed a concerning skew in profitability, even in what was termed as "random data." This unexpected profitability raised questions about the genuine randomness of the data. Therefore, the primary objective is to validate the true randomness of this dataset to ensure the credibility of all subsequent analyses.

Examining Average Returns

To confirm the randomness of the dataset, we examined the average return for each candle, which is computed as:

$$\text{Average Return} = \frac{Close - Open}{Open}$$

From the calculations, the average of all returns indicated a profit of 0.03528% per candle. Ideally, in a completely random setting, this figure should be zero. However, given that the dataset includes stocks with historically strong performances in the S&P 500, a slight deviation can be understood.

Utilizing Validator Scripts and Strategies

We further substantiated the findings with an auxiliary validator script. This script makes use of outputs from the Triple Barrier and bucket-based methods. The focus here was not on the typical performance metrics of the Triple Barrier method. Instead, we concentrated on the return per run to determine the percentage return within the Triple Barrier method context. From these strategies, we deduced an average return of 0.81% (sd=0.94%), a value significantly higher than the single candle return.

For clarity, these strategies pertain to the Triple Barrier method's performance metrics, where we assess the impact of various events on the stock price.

Contextualizing the Data with Timeframes and Samples

To provide further context, the analysis revealed that the average trade lasted about 29 days. With this information, we deduced an annual return of (mean=10.185, sd=11.825, n=8530).

Here, the 8530 samples are a culmination of various symbols and trials. To elucidate:

- Symbols refer to the different stock entities in the dataset.
- Trials pertain to 15 individual runs for each stock symbol, each representing a distinct, randomly generated set of stock market events.

The Triple Barrier method then evaluates the performance of these event sets, resulting in what we term a "collective sample." These samples are assessed by using a 7% barrier threshold and a generous time limit of 500 days.

Estimating Annual Returns

By projecting the single candle return (0.035%) over an entire year (365 days), we estimate an annual return of approximately 12.8772%. It is important to recognize that this projection could be on the higher side, as the chosen stocks represent historically high-performing companies.

Comparative Analysis with S&P 500

For a well-rounded understanding, we compared the findings with the S&P 500's historical average return over 30 years. The S&P 500 displayed a return of 9.78% with a standard deviation of 14.96. These figures were deduced by analyzing the SPY index from May 1992 to May 2023. Although some platforms might display slightly varied results, the differences are not statistically significant[70].

To gauge the significance of these numbers, we conducted a two-sample t-test. Before this, we ensured that the returns from the random trades followed a normal distribution, as shown in Figure 7.9. The test comprised two samples: the annual return from the random trades and the S&P 500's historical annual return.

Conclusion: Skewness Originating from S&P 500's Rise

The t-statistic calculated was 0.1871 with a corresponding p-value of 0.8516. This suggests no statistically significant difference between the random trades' performance and the S&P 500's historical returns. Given that the dataset originates from companies once part of the S&P 500 and still operating, this result is consistent with the expectations.

In conclusion, the skewness observed in the random data likely stems from the inherent upward trajectory of the S&P 500. It is vital to consider this factor when assessing the efficacy of different trading patterns.

7.3.2 Statistical Analysis: Significance Testing

The goal is to make sure the patterns we see are genuine and not just random events. In this section, we explain how we used statistics to confirm the reliability of each pattern.

Analyzing Success Rates

To get a clear picture of how well certain patterns like the double bottom, double top, triple bottom, triple top, and head and shoulders work, we checked their success rates. Here's the

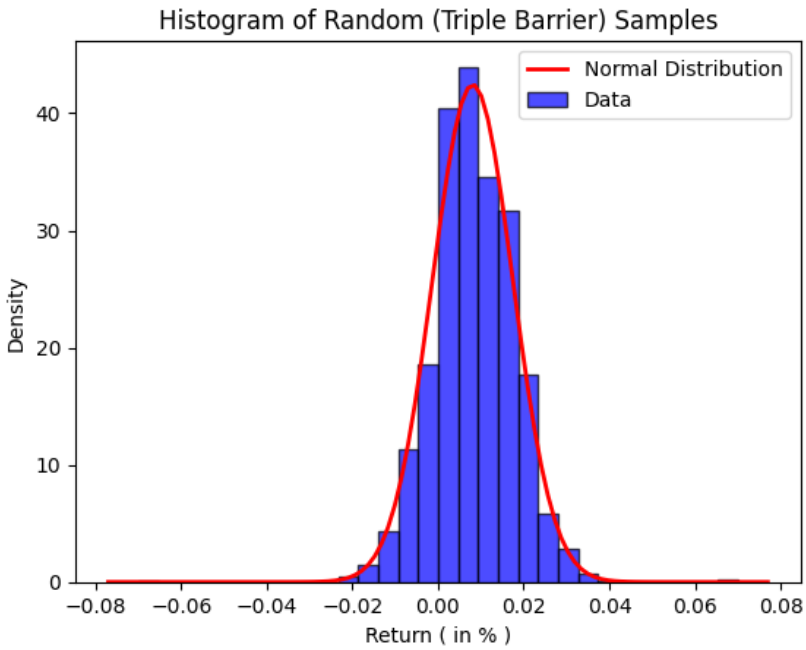


Figure 7.9: Returns of random samples produced by the Triple Barrier Method. (mean=0.81% sd=0.94%)

formula we used:

$$\text{Success Rate} = \frac{\text{Successful occurrences of pattern}}{\text{Total occurrences of pattern}}$$

We used both the main data and a set of random data for this, so we could see how the patterns stacked up against random choices.

Proportions Z-Test: Checking Predictive Power

We used the proportions z-test to see how the success rates of the patterns compare to random events. If we get a high z-score, it means the patterns are doing better than random choices. We also used p-values to show how sure we can be about these findings, based on the idea that there's no real difference between the patterns and random events.

Bonferroni Correction: Making Adjustments for Many Tests

Because we looked at so many patterns and settings, we had to make some adjustments to the p-values. This helps us avoid mistakenly seeing a pattern as important just because we did so many tests. For this, we used the Bonferroni correction, which is a trusted way to make these adjustments when you're doing a lot of comparisons.

Effect Size: Understanding the Real Impact

It is one thing to know there's a difference, but we also need to know how big that difference is. To do this, we calculated the effect size using Cohen's h. This helps us see not just if there's a meaningful difference between the patterns and random events, but also how big that difference really is. By doing this, we get a complete understanding of what the results mean.

7.4 Performance Comparison and Interpretation

In this section, we'll take a closer look at how different evaluation methods measure up against each other and what these results mean for trading patterns.

7.4.1 Performance Metrics and Comparison Tables

In this section, we'll lay out the metrics we use to gauge performance and present tables that show side-by-side comparisons.

7.4.1.1 Comparisons of Triple Barrier Method and Alternative Approach

We perform an in-depth comparative analysis of two distinct approaches for assessing trading pattern outcomes: the Triple Barrier Method and the Hybrid Bucket-based Classifier. Both techniques provide valuable insights into trading patterns, differing fundamentally in how they capture and interpret pattern behaviour.

Initially, we employed the Triple Barrier Method for its straightforwardness and its ability to label results clearly. However, this method has its limitations. A key concern is its propensity to disregard or 'time out' certain outcomes. This tendency could potentially lead to biases in the findings, as it might exclude substantial losses, leaving them unaccounted for in the analysis. Such discrepancies could significantly skew the understanding of a pattern's performance.

To address these limitations and provide a more comprehensive portrayal of trading pattern outcomes, we introduced the Hybrid Bucket-based Classifier. This approach enables us to more precisely capture price movements, including those that might not fully play out within the evaluation timeframe. The bucket-based method offers a more detailed picture of pattern performance and crucially circumvents the data omission problem intrinsic to the Triple Barrier Method.

In the comparison, we observed intriguing subtleties in the representation of win rates for certain patterns. Specifically, when examining the double bottom pattern, it became clear that we tended to underestimate the pattern in nearly all cells (see Figure 7.10). This underestimation by the Triple Barrier Method aligns with the understanding of market biases, in which the market naturally tends to ascend over time. By leaving out incomplete movements, the Triple Barrier Method unintentionally makes the evaluation process more rigorous, tilting the results towards a lower estimation of win rates.

However, this underestimation does not necessarily imply an unreliable evaluator. On the contrary, it suggests a more conservative and stringent evaluation that might bolster the confidence in the outcomes. Notably, a conservative estimate does not always offer an advantage. For instance, in patterns like double bottom, where higher win rates are expected, this tendency might erroneously seem to favor the pattern, particularly at larger thresholds (see Figure 7.11). But, as evident in the figure, this issue only arises when the threshold increases significantly, which is expected as it would require more market volatility. So, while small thresholds are less prone to this concern, we need to maintain caution when interpreting such results, keeping in mind this inherent tendency.

The comparison extends to cases where negative outcomes are expected, with similar conclusions. The differences in outcome representation become more conspicuous as the evaluation

Figure 7.10: Comparison between barriers and bucket-based evaluator where we expect a positive win rate. Green cells indicate an underestimation of the win rate, while red cells indicate an overestimation. Timeout limit = 14.

Figure 7.11: Comparison between barriers and bucket-based evaluator where we expect a negative win rate. Green cells show an underestimation of the win rate, and red cells show an overestimation. Timeout limit = 14.

threshold increases, a logical conclusion since higher thresholds necessitate greater market volatility for trade closure.

In summary, while both methods possess distinct strengths and weaknesses, they contribute collectively to a comprehensive understanding of pattern performance. Acknowledging these subtleties allows for a more accurate interpretation of the results, providing a robust basis for trading decisions rooted in pattern recognition. We will consider these nuances when analyzing results for patterns like double bottom, triple bottom, and head and shoulders.

7.4.1.2 Interpretation of Each Result Table

In this section, we provide a detailed interpretation of the results derived from each of the pattern algorithms: random, double bottom, double top, triple Bottom, triple Top, and head and shoulders. Each algorithm's performance is assessed within the context of the Triple Barrier Method and the Hybrid Bucket-based Classifier, as discussed in the previous sections.

Random

Starting with random, we observe that the win rate increases proportionately with the threshold (see Figure 7.12), which holds across all ranges. This is expected as the range parameter does not influence the generation of random results.

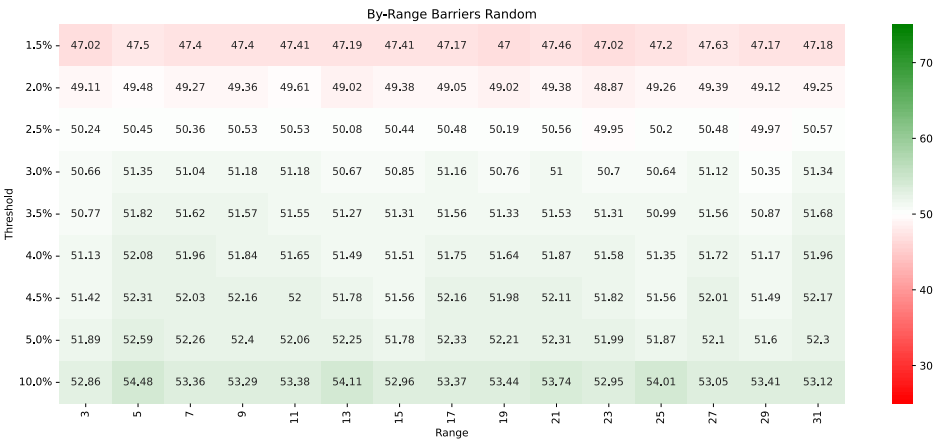


Figure 7.12: Random win rate

Double Bottom

Moving on to the double bottom pattern, the results align with the literature expectations. The pattern displays a high win rate across almost all cases and outperforms the random baseline (see Figure 7.13). However, while a statistical analysis confirms this difference as significant (see Figure A.4), it also reveals a relatively small effect size (see Figure A.5). This suggests that although the double bottom pattern does demonstrate an advantage, the effect is not substantial enough to validate it as a standalone trading strategy.

Double Top

The double top pattern, on the other hand, deviates from literature expectations. Contrary to the predicted sub-50% win rates, the results show win rates exceeding 50% in most cases.

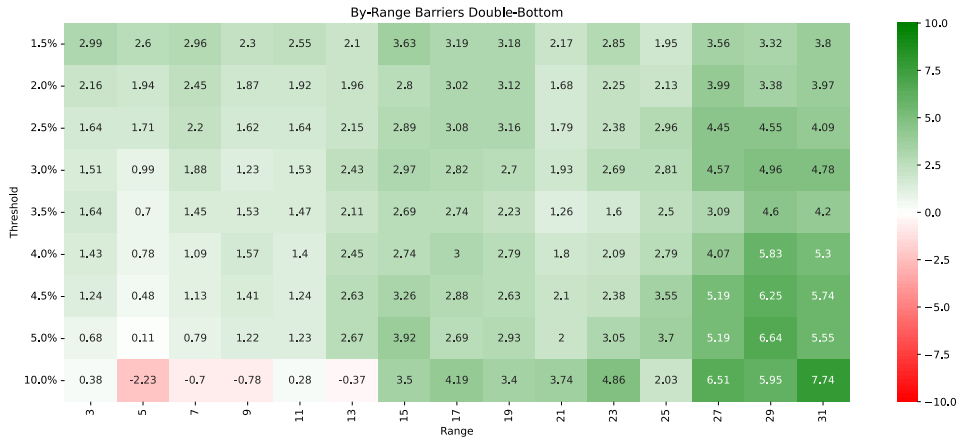


Figure 7.13: Random vs double bottom

We observe statistical significance only up to a 4% threshold, with an effect size similar to the double bottom pattern. Thus, while the double top pattern may be effective in isolated instances, the overall results suggest its performance is inconsistent and should be considered with caution (see Figure 7.14).

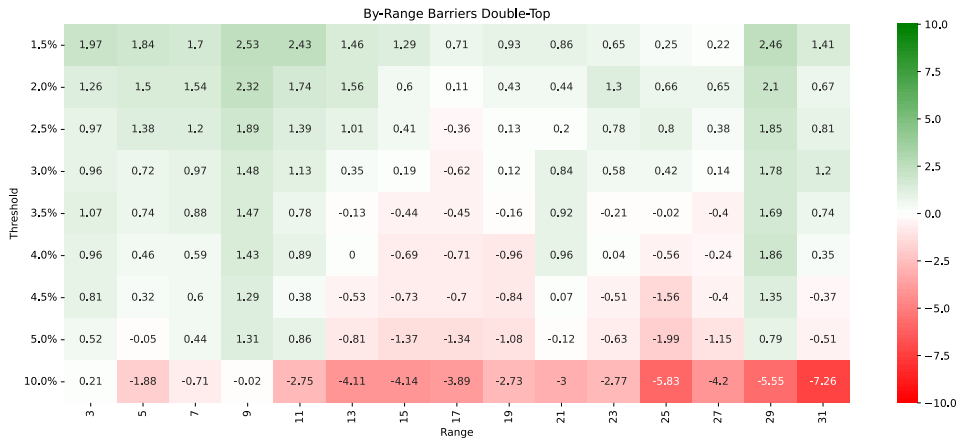


Figure 7.14: Random vs double top

Triple Bottom

With the triple bottom pattern, the results exhibit a marked amplification from the double bottom signal. However, the sample size is considerably smaller (see Figure A.11), and the win rate appears quite chaotic across thresholds and ranges (see Figure A.10). After correction, we find that only up to a barrier threshold of 2.5% the results are significantly different from random, suggesting that the pattern may not be as effective as the literature suggests (see Figure 7.15).

Triple Top

The Triple Top pattern yields significant results only barrier thresholds of 1.5% and 2.5%,

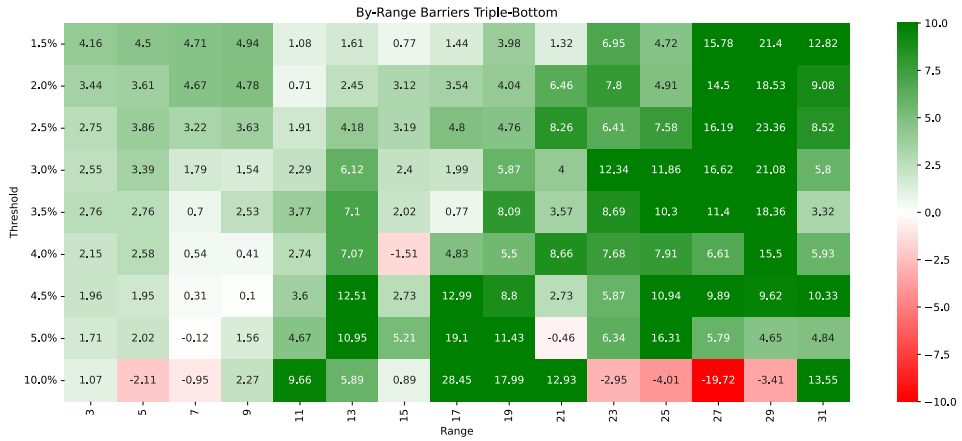


Figure 7.15: Random vs triple bottom

suggesting inconsistent and unreliable performance (see Figure 7.16). This diverges from the expected pattern behavior in literature, further questioning its reliability as a trading strategy.

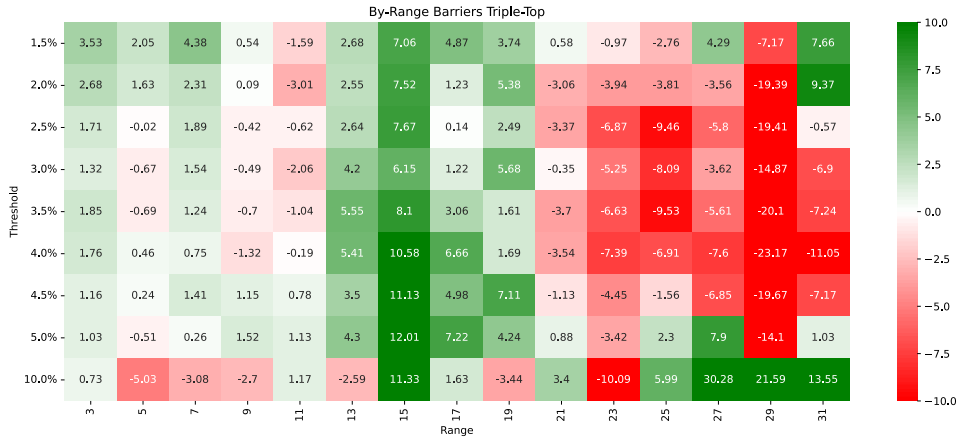


Figure 7.16: Random vs triple top

Head and Shoulders

Despite its prevalence in the literature, the head and shoulders pattern yields fewer samples than anticipated, with performance appearing no different from random (see Figure 7.17). This may result from the focus on daily ranges or a need for wider ranges. Still, these results suggest that the conventional wisdom around this pattern may not hold in the practical trading context.

In summary, the findings underscore the importance of critically evaluating pattern performance in the context of real-world data. While some patterns align with theoretical expectations, others diverge significantly, highlighting the need for a nuanced and data-driven approach in pattern-based trading strategies.

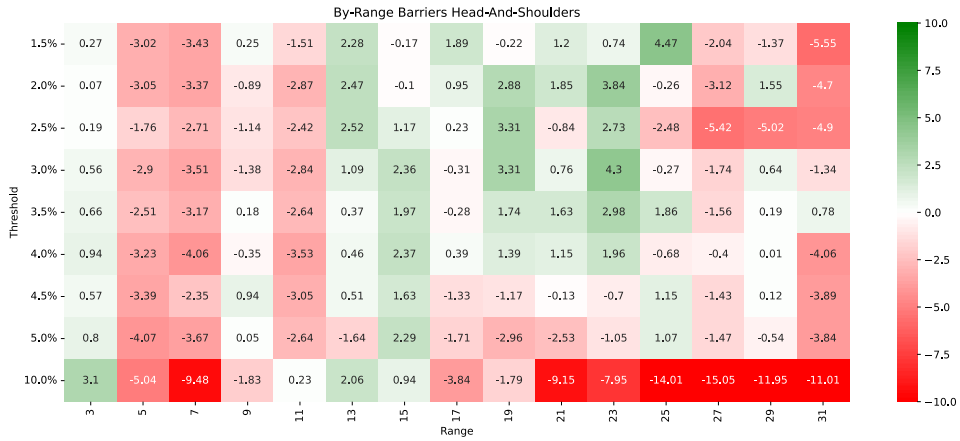


Figure 7.17: Random vs head and shoulders

7.4.2 Observing Trends Over Time

In the concluding segment of the results and discussions, we delve into the temporal aspects of the pattern analysis. Given the span of data collected over numerous years, we've aggregated the results on an annual basis to discern any emergent temporal patterns.

Performance

To investigate temporal trends in pattern detection on a year-over-year basis, we aggregated the outcome metrics for each pattern annually. This methodology counteracts the non-uniform distribution of the dataset by focusing on the win rates for each year rather than raw pattern counts. We conducted a regression analysis on these win rates.

Using the double bottom pattern as an example, there's a noticeable yet slight decline in its win rate (see Figure 7.18). With an annual decrease of 0.08 percent (Slope: -0.0008, P-value: 0.1144, Standard error: 0.0005), this trend is not statistically significant.

A similar trend, or lack thereof, was found for the double top pattern (see Figure 7.19). A near-zero slope (Slope: -0.0000, P-value: 0.9618, Standard error: 0.0006) indicates no noticeable trend over the years. This observation aligns with previous findings where the double bottom pattern showed significance only for smaller thresholds.

This lack of significance was consistent in the analysis of all other patterns, underscoring the recurring theme of non-significant trends.

Occurrences

When examining occurrences, the aim was to present a normalized view of the double bottom pattern. An analysis of the double bottom occurrences shows a consistent increase year-over-year. This is in line with the expectations given the growing sample size as time progresses (see Figure 7.20).

To gain a clearer understanding of pattern growth, we first calculated the relative frequencies of the random pattern. Since the occurrences of the random pattern are determined by a fixed probability, its counts can hint at data density during particular time frames (see Figure 7.21).

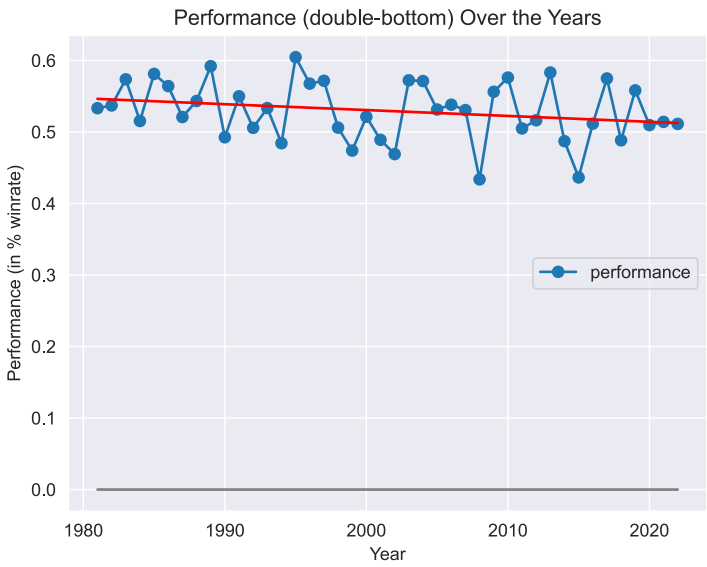


Figure 7.18: Year win rate for double bottom.

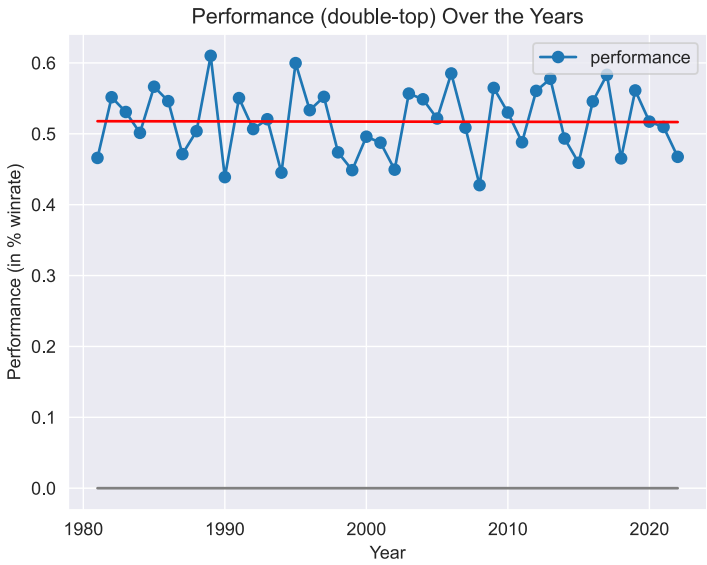


Figure 7.19: Year-over-year win rate for double top.

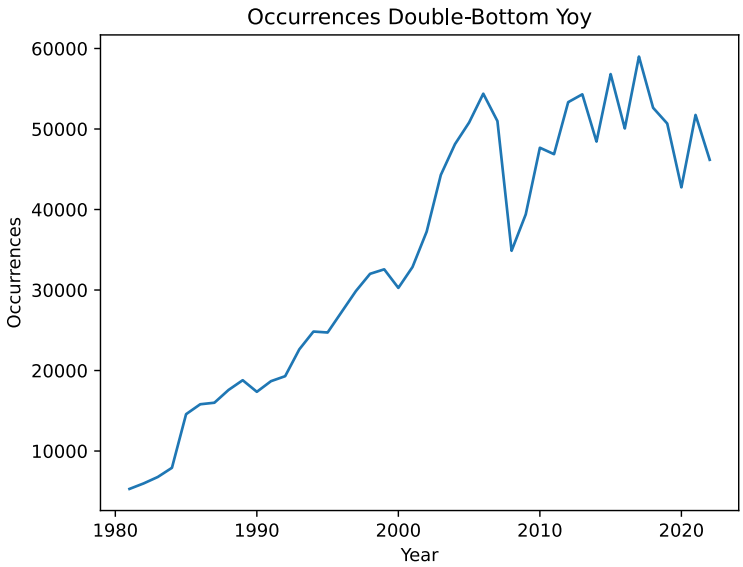


Figure 7.20: Year-over-year occurrences for double bottom.

By comparing these frequencies with the double bottom's occurrences and then normalizing the adjusted samples, we get a clearer picture of how pattern detections have evolved over time.

The resulting chart suggests an upward movement. However, due to its volatility, it is challenging to pinpoint a definite trend. One possibility is that earlier data was heavily influenced by a select group of high-performing stocks. Another potential reason could be an increase in market volatility over time. Drawing firm conclusions, however, would require further investigation.

While patterns such as the double bottom revealed noticeable trends, statistical significance was not uniformly present across all patterns. Furthermore, even when significance was observed, it is likely influenced by factors beyond merely an increase in the number of patterns. Drawing direct conclusions remains challenging, underscoring that understanding the intricacies of financial markets is a nuanced task, necessitating ongoing research and examination.

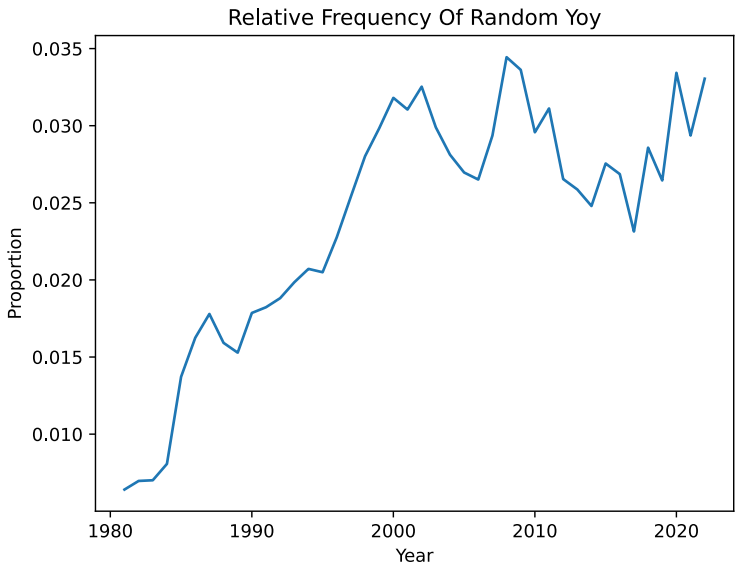


Figure 7.21: Year-over-year frequencies for random.

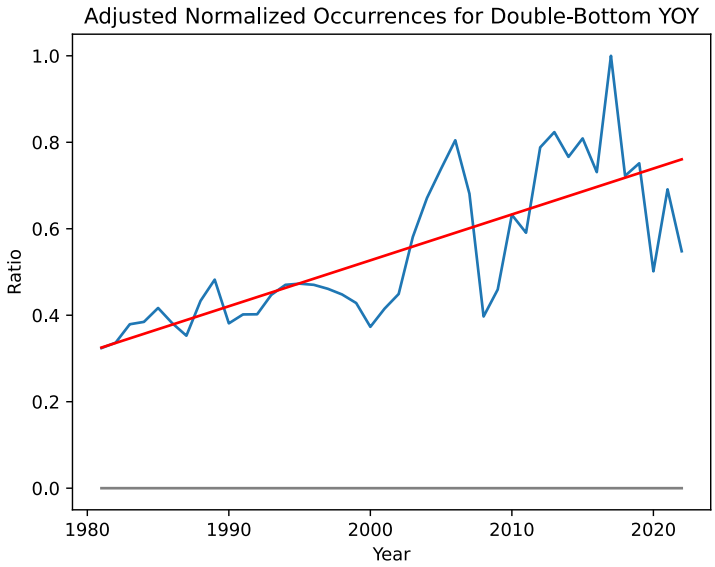


Figure 7.22: Year-over-year normalized for double bottom.

Chapter 8

Conclusions and Future Work

We embarked on this journey with three fundamental research questions about trading patterns in the financial market. The first question revolved around the potential of advanced machine learning techniques in effectively identifying common trading patterns. As we delved deeper, the nuances and complexities inherent in financial markets nudged us towards a more scripted approach, moving slightly away from a pure machine learning focus. This transition into the second research question: Does the discerned pattern truly possess predictive power in stock price movements? Finally, the third line of questioning concentrated on the temporal aspect, contemplating whether the rise in the prevalence of machine learning over the years has influenced this predictive power. Throughout this thesis, the core aim remained to sift the signal from the noise and uncover the real-world relevance of these financial patterns.

Key Observations and Challenges

The initial investigation focused on the potential of machine learning as a means to more effectively identify common trading patterns. However, during the exploration, we encountered challenges in its practical implementation within the confines of this thesis. These experiences prompted us to pivot towards an algorithmic approach, which yielded transparent and reproducible results.

The cornerstone of the work was the development of an extensive software pipeline. This pipeline facilitated efficient analysis of the selected stock datasets. The project made use of a microservices architecture, allowing for a network of specialized modules, each performing a specific function. Such a structure enhanced development time and offered flexibility, as individual modules could be developed without affecting existing pieces.

Interacting with the data became more intuitive due to this pipeline. Previous challenges associated with visualizing findings, such as the limitations of generic tools or the intricacies of plotting, were mitigated. The custom-built visualization tool streamlined this process, granting us the ability to visualize data as per the requirements without being constrained by external tool limitations.

Evaluating the patterns was another hurdle we faced. We recognized that viewing them solely as trading strategies provided a limited perspective. Consequently, we use the triple barrier method, aiming to discern any significant differences in success rates when contrasting these

patterns against random occurrences. To make sure that the triple barrier method does not introduce any bias, we made sure to verify it against the evaluator which also logs timed-out trades.

While we've found success with the algorithmic approach, there is still potential for machine learning in this topic. With the established tools and pipeline, a renewed exploration into the machine learning domain could lead to further insights in future research.

Discussion of Results

The efficacy of classical patterns from the literature was put to the test to determine their predictive power, which is central to the second research question. The following observations were made:

1. The random algorithm's win rate proportionally rises with the triple barrier's threshold.
2. The double bottom pattern, though it outperforms the random baseline, offers only a minimal effect size.
3. Double top patterns, contrary to established literature, had win rates exceeding 50% in most scenarios but again with a minimal effect size.
4. Triple bottom patterns amplify the double bottom signal, yet the effect size remains small.
5. Triple top pattern's performance is inconsistent, with significant results emerging only at specific thresholds.
6. The head and shoulders pattern seemed no different from random events.

The small effect size observed in the different patterns, particularly in the double bottom and double top, may appear insignificant, but it's essential to put this into context. In the realm of trading, where win rates are often the focus, a substantial win rate would be highly unusual and could indicate an error or anomaly. Thus, even a small win rate improvement can be considered a favorable result in certain scenarios. However, the critical question remains whether these minor gains can outperform simpler and more conventional approaches, such as holding indices. The strategies based on trading specific patterns are inherently different from these broad investment strategies, and a direct comparison might not reveal a clear advantage.

This nuanced understanding of the effect size leads us to suggest caution when relying exclusively on these patterns for stock predictions. Exploring the different dynamics further, such as whether trading based on recognized patterns for short periods offers better returns compared to holding an index throughout the year, would add depth to our grasp of the practical application of pattern-based trading.

Patterns Over Time

In the endeavor to understand the evolving nature of trading patterns, we systematically assessed their performance over decades. The findings provide insight into the third research question — evaluating the change in predictive power of these patterns over time:

1. Double Bottom Pattern:
   Over the years, the effectiveness of the double bottom pattern showed a slight decrease. However, this trend was minor and did not reach a level of statistical significance. This suggests that while there might be minor shifts in the pattern's efficacy, it has largely remained stable.

2. Double Top Pattern:
   As a counterpoint to the double bottom, the double top pattern's predictive power remained steadfast throughout the observed period, with no notable variations.

3. Other Studied Patterns:
   Diversifying this thesis to include other trading patterns, it was observed that the majority maintained a consistent behavior. No significant temporal trends were detected in their predictive prowess.

These findings provide insights into the third research question. Despite the rapid technological evolution, potential changes in trading methods, and the rise of machine learning, the predictive power of these trading patterns has remained largely consistent over the decades. However, this constancy might suggest an inherent limitation in these patterns' foundational efficacy, implying that their baseline effectiveness might have been suboptimal from the outset.

Conclusion and Future Work

As we reflect on the research, it becomes evident that relying solely on specific horizontal trading patterns for stock market predictions does not capture the full complexity of market behavior. This limitation is also highlighted in existing literature, warning against the exclusive use of such patterns. However, our exploration confirms that while there may not be a robust signal in every pattern, there are indeed discernible trends in very few of them. Even so, the effect size is minimal, and it might not be advisable to use them unless they are part of many other layers in a strategy.

The consistent behavior of these patterns over the years, despite rapid technological changes and shifts in trading practices, prompts a closer examination of their underlying effectiveness. This thesis focused mainly on horizontal patterns, but there's a new avenue for exploration in identifying patterns through trend lines and other charting techniques.

Advanced tools and clear visualization play an essential role in this process; their importance in enhancing understanding can pave the way for future exploration. With these tools, one could attempt to detect trend-line based patterns, where patterns can be found on any angle or line instead of exclusively horizontal ones, presenting a promising area that might reveal new insights. Integrating these trend analyses could also prove promising.

Finally, while the journey into the realm of machine learning was brief, its potential as a tool for future stock pattern analysis still remains an enticing prospect, especially for those armed with the right tools and a keen spirit for charting Wall Street.

Appendix A

Appendix

Figure A.1: Random samples

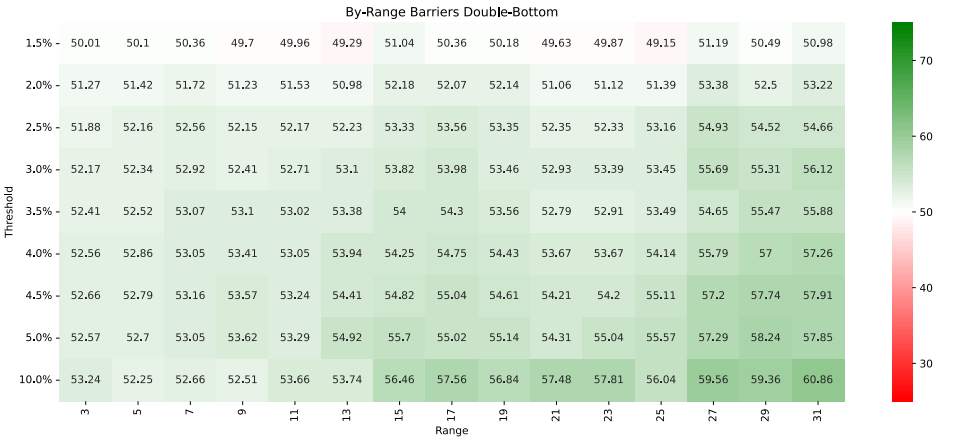


Figure A.2: Double bottom win rate

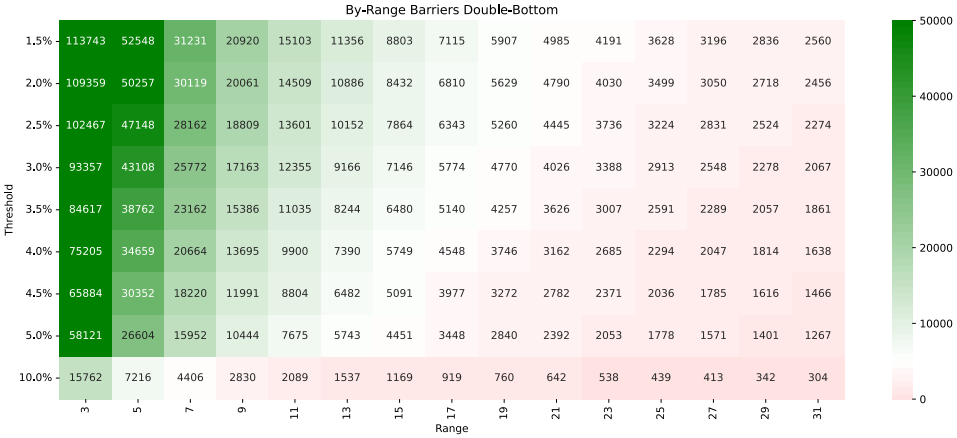


Figure A.3: Double bottom samples

	rng:3.00	rng:5.00	rng:7.00	rng:9.00	rng:11.00	rng:13.00	rng:15.00	rng:17.00	rng:19.00	rng:21.00	rng:23.00	rng:25.00	rng:27.00	rng:29.00	rng:31.00
thld:0.015	0.00000	0.00000	0.00000	0.00000	0.00000	0.00005	0.00000	0.00000	0.00000	0.00341	0.00039	0.02351	0.00009	0.00057	0.00018
thld:0.020	0.00000	0.00000	0.00000	0.00001	0.00005	0.00021	0.00000	0.00000	0.00001	0.02612	0.00600	0.01533	0.00002	0.00061	0.00013
thld:0.025	0.00000	0.00000	0.00000	0.00017	0.00080	0.00009	0.00000	0.00000	0.00001	0.02279	0.00507	0.00115	0.00000	0.00001	0.00014
thld:0.030	0.00000	0.00342	0.00000	0.00623	0.00266	0.00002	0.00000	0.00006	0.00039	0.01883	0.00249	0.00336	0.00001	0.00000	0.00002
thld:0.035	0.00000	0.04877	0.00041	0.00122	0.00604	0.00045	0.00006	0.00021	0.00563	0.14647	0.09047	0.01329	0.00400	0.00005	0.00039
thld:0.040	0.00001	0.03661	0.01153	0.00169	0.01329	0.00012	0.00011	0.00013	0.00111	0.05233	0.03645	0.00942	0.00034	0.00000	0.00003
thld:0.045	0.00018	0.22413	0.01359	0.00785	0.03824	0.00010	0.00001	0.00057	0.00391	0.03354	0.02526	0.00185	0.00002	0.00000	0.00002
thld:0.050	0.05371	0.77999	0.10445	0.03123	0.05294	0.00020	0.00000	0.00269	0.00276	0.05919	0.00734	0.00237	0.00006	0.00000	0.00011
thld:0.100	0.53004	0.00294	0.42343	0.45747	0.81426	0.78609	0.02244	0.01423	0.06815	0.06437	0.02741	0.40419	0.00915	0.02998	0.00761

Figure A.4: Double bottom p-values

	rng:3.00	rng:5.00	rng:7.00	rng:9.00	rng:11.00	rng:13.00	rng:15.00	rng:17.00	rng:19.00	rng:21.00	rng:23.00	rng:25.00	rng:27.00	rng:29.00	rng:31.00
thld:0.015	0.05982	0.05210	0.05938	0.04602	0.05093	0.04193	0.07260	0.06377	0.06359	0.04344	0.05693	0.03891	0.07126	0.06644	0.07593
thld:0.020	0.04312	0.03876	0.04904	0.03746	0.03829	0.03929	0.05603	0.06040	0.06237	0.03367	0.04502	0.04243	0.07980	0.06758	0.07934
thld:0.025	0.03274	0.03416	0.04404	0.03250	0.03272	0.04289	0.05796	0.06154	0.06320	0.03575	0.04766	0.05922	0.08902	0.09097	0.08200
thld:0.030	0.03016	0.01983	0.03757	0.02466	0.03062	0.04858	0.05939	0.05650	0.05409	0.03869	0.05396	0.05617	0.09167	0.09940	0.09586
thld:0.035	0.03301	0.01398	0.02906	0.03065	0.02949	0.04237	0.05389	0.05489	0.04462	0.02517	0.03200	0.05021	0.06194	0.09227	0.08427
thld:0.040	0.02856	0.01558	0.02188	0.03140	0.02800	0.04899	0.05490	0.06008	0.05591	0.03595	0.04182	0.05591	0.08156	0.11713	0.10653
thld:0.045	0.02493	0.00959	0.02263	0.02826	0.02479	0.05266	0.06539	0.05785	0.05286	0.04194	0.04755	0.07113	0.10420	0.12558	0.11545
thld:0.050	0.01356	0.00234	0.01583	0.02441	0.02471	0.05356	0.07859	0.05397	0.05879	0.04006	0.06118	0.07425	0.10430	0.13360	0.11164
thld:0.100	0.00773	-0.04474	-0.01421	-0.01561	0.00560	-0.00738	0.07032	0.08438	0.06851	0.07527	0.09768	0.04066	0.13144	0.11989	0.15646

Figure A.5: Double bottom Cohen's h

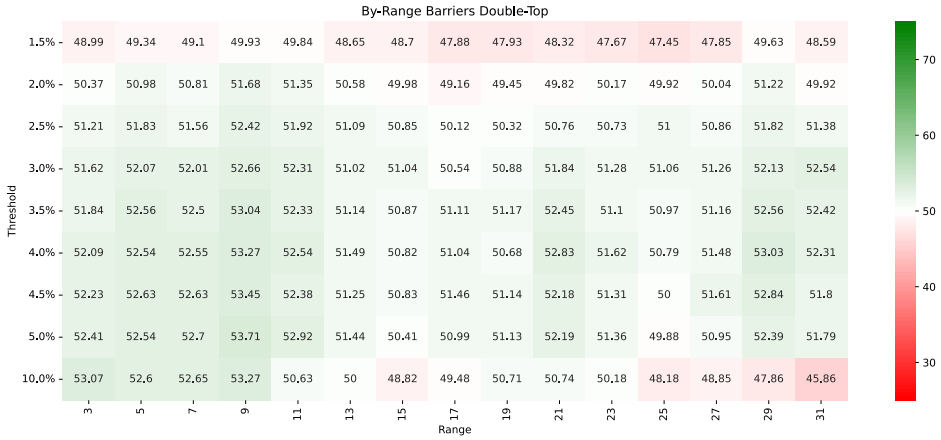


Figure A.6: Double top win rate

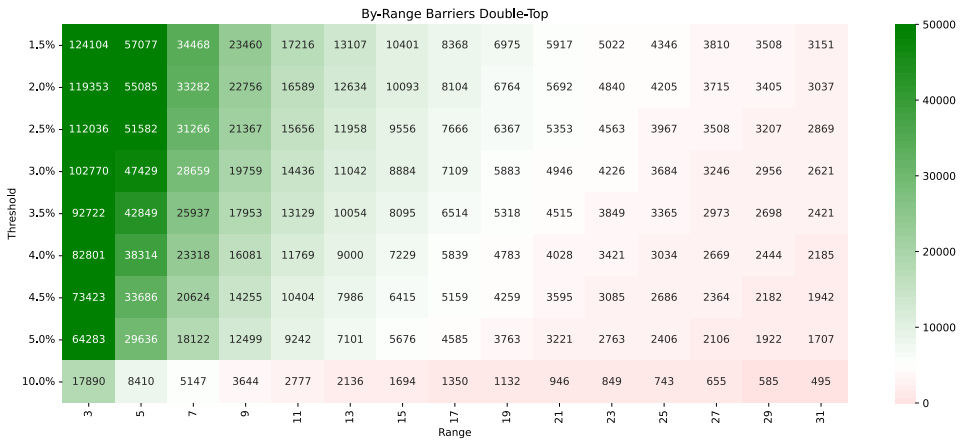


Figure A.7: Double top samples

	rng:3,00	rng:5,00	rng:7,00	rng:9,00	rng:11,00	rng:13,00	rng:15,00	rng:17,00	rng:19,00	rng:21,00	rng:23,00	rng:25,00	rng:27,00	rng:29,00	rng:31,00
thld:0,015	0,00000	0,00000	0,00000	0,00000	0,00000	0,00251	0,01300	0,27230	0,11866	0,16727	0,31699	0,74740	0,76328	0,00567	0,15707
thld:0,020	0,00000	0,00000	0,00002	0,00000	0,00011	0,00132	0,26188	0,87901	0,45136	0,53589	0,07498	0,41355	0,50354	0,02064	0,50779
thld:0,025	0,00046	0,00001	0,00087	0,00000	0,00249	0,03890	0,48023	0,53306	0,84112	0,73737	0,28999	0,34531	0,70082	0,05201	0,38495
thld:0,030	0,00109	0,02961	0,01083	0,00107	0,01810	0,41722	0,75111	0,29500	0,77850	0,23740	0,37762	0,56175	0,97837	0,07966	0,20864
thld:0,035	0,00047	0,02850	0,01889	0,00154	0,10330	0,98771	0,46030	0,46142	0,90044	0,21933	0,87888	0,99499	0,73969	0,08631	0,33769
thld:0,040	0,00153	0,22354	0,14296	0,00244	0,08334	0,97731	0,25415	0,31675	0,28399	0,17865	0,92226	0,58455	0,85379	0,07722	0,57287
thld:0,045	0,01788	0,36341	0,14087	0,01189	0,37564	0,46597	0,25073	0,30785	0,35131	0,77339	0,62412	0,21199	0,75203	0,21857	0,90161
thld:0,050	0,17436	0,87342	0,34727	0,00994	0,14192	0,21112	0,02881	0,09967	0,36198	0,85036	0,60085	0,05928	0,35114	0,31890	0,63735
thld:0,100	0,75784	0,00347	0,41375	0,92156	0,02230	0,00216	0,00091	0,00433	0,08303	0,10630	0,14267	0,00093	0,01898	0,02379	0,00106

Figure A.8: Double top p-values

	rng:3.00	rng:5.00	rng:7.00	rng:9.00	rng:11.00	rng:13.00	rng:15.00	rng:17.00	rng:19.00	rng:21.00	rng:23.00	rng:25.00	rng:27.00	rng:29.00	rng:31.00
thld:0,015	0,03917	0,03659	0,03421	0,05038	0,04855	0,02968	0,02677	0,01297	0,01994	0,01899	0,01482	0,00509	0,00506	0,04830	0,02605
thld:0,020	0,02505	0,03009	0,03046	0,04566	0,03479	0,03205	0,01227	0,00183	0,00977	0,00868	0,02682	0,01315	0,01139	0,04107	0,01239
thld:0,025	0,01927	0,02797	0,02435	0,03799	0,02796	0,02116	0,00793	-0,00769	0,00268	0,00484	0,01641	0,01560	0,00673	0,03554	0,01676
thld:0,030	0,01862	0,01440	0,01933	0,02809	0,02268	0,00864	0,00370	-0,01339	0,00390	0,01774	0,01422	0,00998	0,00049	0,03344	0,02525
thld:0,035	0,02076	0,01516	0,01864	0,02841	0,01636	-0,00017	-0,00900	-0,00983	-0,00182	0,01929	-0,00257	0,00011	-0,00632	0,03416	0,02008
thld:0,040	0,01973	0,00885	0,01221	0,02865	0,01831	0,00033	-0,01465	-0,01409	-0,01644	0,02230	0,00174	-0,01032	-0,00370	0,03700	0,01243
thld:0,045	0,01551	0,00697	0,01296	0,02517	0,00988	-0,00902	-0,01561	-0,01520	-0,01513	0,00505	-0,00921	-0,02495	-0,00673	0,02737	-0,00288
thld:0,050	0,00939	-0,00129	0,00877	0,02732	0,01740	-0,01640	-0,03155	-0,02610	-0,01571	0,00351	-0,01043	-0,04007	-0,02105	0,02350	-0,01173
thld:0,100	0,00370	-0,04208	-0,01367	0,00187	-0,04845	-0,07264	-0,08617	-0,08229	-0,05380	-0,05427	-0,05244	-0,12438	-0,09431	-0,09666	-0,14937

Figure A.9: Double top Cohen's h

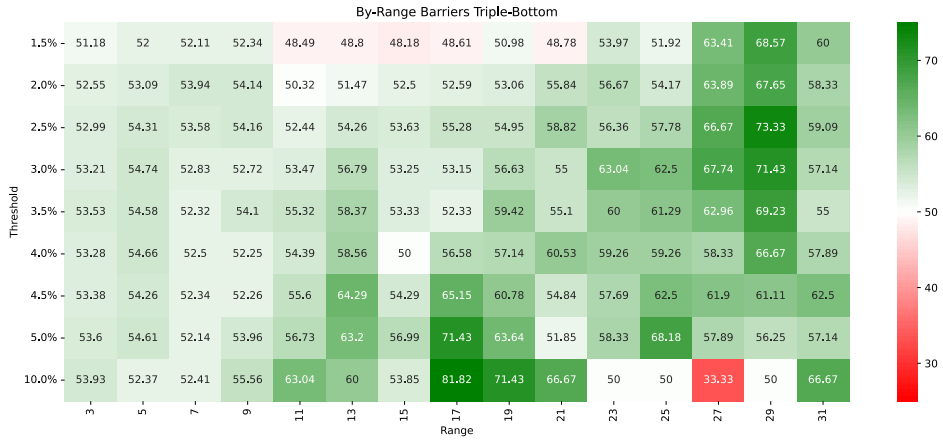


Figure A.10: Triple bottom win rate

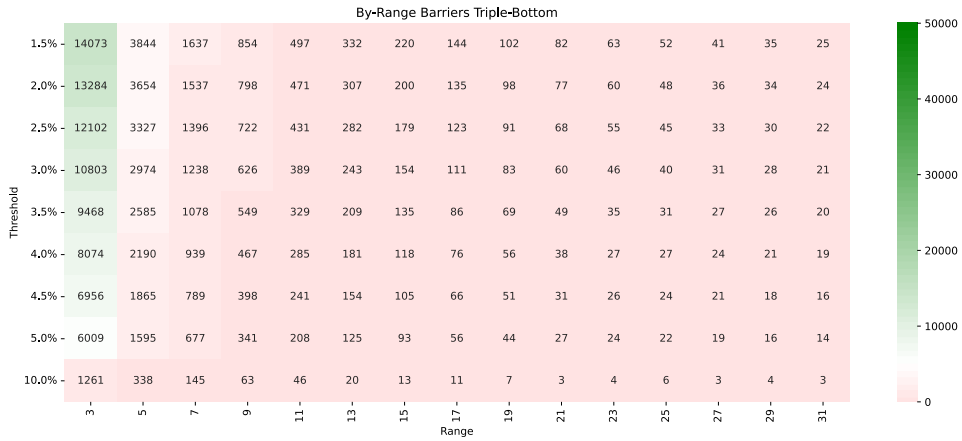


Figure A.11: Triple bottom samples

	rng:3,00	rng:5,00	rng:7,00	rng:9,00	rng:11,00	rng:13,00	rng:15,00	rng:17,00	rng:19,00	rng:21,00	rng:23,00	rng:25,00	rng:27,00	rng:29,00	rng:31,00
thld:0,015	0,00000	0,00000	0,00021	0,00298	0,59466	0,56139	0,87440	0,61115	0,41874	0,98509	0,32931	0,58547	0,04305	0,01124	0,19929
thld:0,020	0,00000	0,00004	0,00024	0,00305	0,69174	0,36202	0,55062	0,41111	0,48618	0,26198	0,28324	0,50166	0,08197	0,03079	0,37347
thld:0,025	0,00000	0,00001	0,01293	0,07030	0,35318	0,19938	0,47806	0,21246	0,36463	0,21803	0,34149	0,31209	0,06293	0,01048	0,54659
thld:0,030	0,00000	0,00065	0,18670	0,40868	0,34520	0,08669	0,66703	0,74638	0,28522	0,53662	0,09417	0,13352	0,06418	0,02568	0,59493
thld:0,035	0,00000	0,00735	0,68693	0,39466	0,18011	0,06249	0,69795	0,97562	0,17997	0,61647	0,30395	0,34396	0,23544	0,06127	0,76826
thld:0,040	0,00087	0,01499	0,61634	0,90033	0,21514	0,04727	0,88611	0,39985	0,41083	0,28545	0,42474	0,41153	0,64544	0,15560	0,60398
thld:0,045	0,00550	0,06733	0,78182	0,93886	0,29171	0,00193	0,58108	0,03491	0,20777	0,61754	0,54936	0,28379	0,36540	0,41430	0,40967
thld:0,050	0,02357	0,22245	0,97991	0,55192	0,17533	0,01445	0,30423	0,00424	0,13091	0,96210	0,53237	0,12648	0,61340	0,70823	0,71614
thld:0,100	0,49351	0,37609	0,74601	0,78630	0,30833	0,59537	0,95538	0,05802	0,33999	0,65252	0,90500	0,85089	0,49262	0,88773	0,63948

Figure A.12: Triple bottom p-values

	rng:3,00	rng:5,00	rng:7,00	rng:9,00	rng:11,00	rng:13,00	rng:15,00	rng:17,00	rng:19,00	rng:21,00	rng:23,00	rng:25,00	rng:27,00	rng:29,00	rng:31,00
thld:0,015	0,08284	0,08711	0,09286	0,10212	0,02386	0,03196	0,01070	0,04239	0,07927	0,00206	0,12188	0,07487	0,31910	0,43712	0,25776
thld:0,020	0,06834	0,07072	0,09490	0,10598	0,01828	0,05210	0,04208	0,07085	0,07007	0,12655	0,13770	0,09514	0,29368	0,37823	0,18248
thld:0,025	0,05444	0,07842	0,06749	0,06755	0,04480	0,07700	0,05346	0,11287	0,09529	0,14907	0,12869	0,15139	0,33033	0,48631	0,13207
thld:0,030	0,05084	0,06460	0,03794	0,03312	0,04793	0,11074	0,03475	0,03089	0,11779	0,07998	0,24994	0,24000	0,34037	0,43610	0,11650
thld:0,035	0,05414	0,05443	0,01242	0,03631	0,07369	0,13070	0,03360	0,00328	0,16262	0,07173	0,17513	0,16844	0,23129	0,37727	0,06602
thld:0,040	0,04081	0,05341	0,01659	0,00580	0,07285	0,14910	-0,01320	0,09698	0,11041	0,17482	0,15467	0,15911	0,09628	0,31638	0,11959
thld:0,045	0,03661	0,04351	0,00992	0,00384	0,06816	0,25435	0,05400	0,26448	0,17809	0,09151	0,11799	0,22143	0,19971	0,19425	0,20863
thld:0,050	0,03193	0,03143	0,00097	0,03233	0,09461	0,22212	0,10827	0,39642	0,23113	-0,00915	0,12818	0,33420	0,11649	0,09386	0,09758
thld:0,100	0,02036	-0,04892	-0,02714	0,03514	0,15186	0,11973	0,01553	0,62390	0,37419	0,26571	-0,05962	-0,07661	-0,40198	-0,07048	0,27640

Figure A.13: Triple bottom Cohen's h

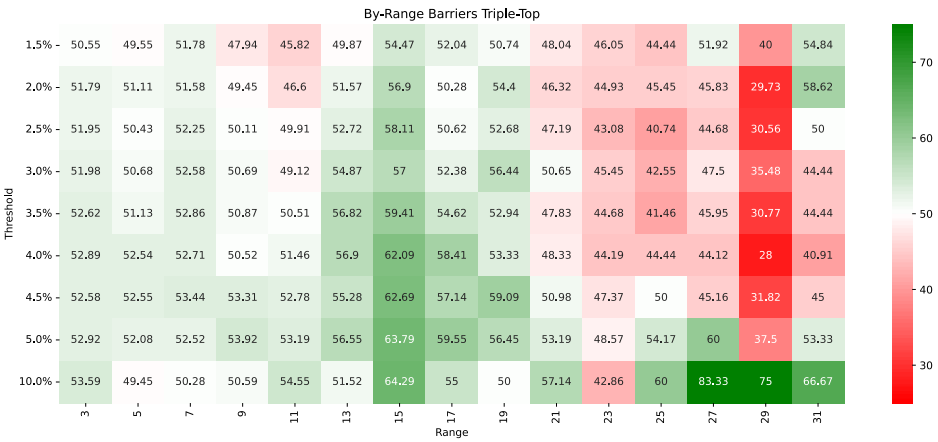


Figure A.14: Triple top win rate

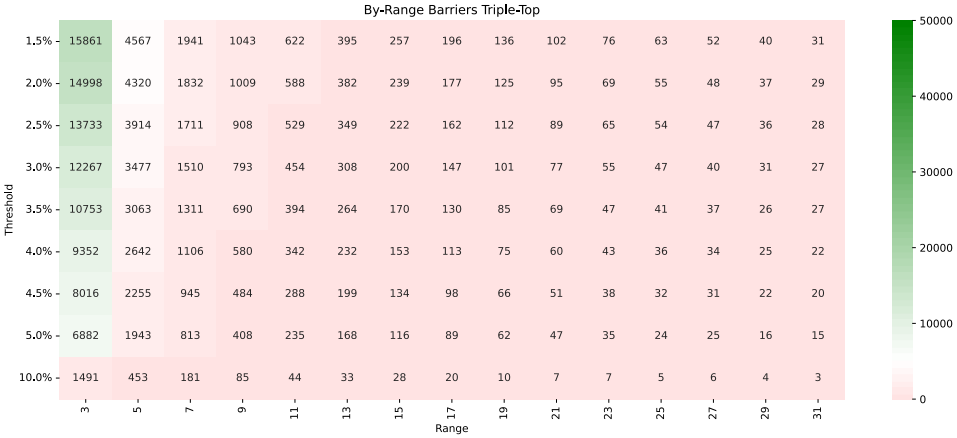


Figure A.15: Triple top samples

	rng:3.00	rng:5.00	rng:7.00	rng:9.00	rng:11.00	rng:13.00	rng:15.00	rng:17.00	rng:19.00	rng:21.00	rng:23.00	rng:25.00	rng:27.00	rng:29.00	rng:31.00
thld:0.015	0.00000	0.00525	0.00016	0.66686	0.43201	0.22436	0.02369	0.17433	0.48073	0.89950	0.87620	0.50563	0.44843	0.46414	0.39693
thld:0.020	0.00000	0.02878	0.05542	0.92526	0.18405	0.27356	0.01944	0.80156	0.23208	0.55630	0.45329	0.50104	0.73160	0.03057	0.23899
thld:0.025	0.00060	0.76213	0.08661	0.82957	0.80850	0.25537	0.01574	0.97000	0.48313	0.52947	0.22897	0.17026	0.52027	0.03305	0.90057
thld:0.030	0.01342	0.53899	0.18041	0.86401	0.46343	0.14421	0.08316	0.77192	0.35891	0.94668	0.44286	0.27649	0.76466	0.14643	0.60305
thld:0.035	0.00110	0.58663	0.37044	0.70798	0.83093	0.07345	0.04676	0.48648	0.85587	0.53915	0.36939	0.23038	0.72430	0.04746	0.57635
thld:0.040	0.00144	0.58430	0.60467	0.58733	0.84515	0.08815	0.00680	0.15725	0.86372	0.58442	0.20247	0.41444	0.47682	0.03747	0.54224
thld:0.045	0.07457	0.75629	0.36950	0.65190	0.83322	0.33339	0.01672	0.31566	0.37431	0.87176	0.58446	0.85545	0.44391	0.10896	0.51921
thld:0.050	0.12517	0.85375	0.87343	0.55510	0.67481	0.30382	0.00762	0.17284	0.50859	0.90364	0.81373	0.84247	0.33570	0.39116	0.75107
thld:0.100	0.48913	0.04663	0.53460	0.78466	0.98672	0.95701	0.23497	0.68779	0.82777	0.85521	0.59169	0.78192	0.13798	0.38916	0.38167

Figure A.16: Triple top p-values

	rng:3.00	rng:5.00	rng:7.00	rng:9.00	rng:11.00	rng:13.00	rng:15.00	rng:17.00	rng:19.00	rng:21.00	rng:23.00	rng:25.00	rng:27.00	rng:29.00	rng:31.00
thld:0.015	0.06978	0.04303	0.08703	0.01350	-0.03168	0.06108	0.14142	0.09711	0.06004	0.01239	-0.01766	-0.08291	0.10411	-0.11500	0.14744
thld:0.020	0.05251	0.03461	0.04524	0.00298	-0.05511	0.05609	0.15326	0.01898	0.10627	-0.05985	-0.08865	-0.08858	-0.04906	-0.35979	0.21639
thld:0.025	0.03322	0.00502	0.04216	-0.00721	-0.01053	0.06089	0.16329	0.00296	0.06525	-0.06600	-0.14653	-0.18433	-0.09294	-0.35822	0.02321
thld:0.030	0.02524	-0.01078	0.03510	-0.00615	-0.03454	0.08337	0.12324	0.02396	0.08982	-0.00753	-0.10181	-0.15603	-0.04677	-0.25949	-0.09834
thld:0.035	0.03529	-0.01017	0.02510	-0.01440	-0.01078	0.11049	0.15559	0.06123	0.01961	-0.07398	-0.12845	-0.18369	-0.05799	-0.38273	-0.10563
thld:0.040	0.03684	0.01101	0.01574	-0.02274	0.01060	0.11248	0.22253	0.13397	0.01972	-0.07063	-0.19511	-0.13261	-0.12035	-0.41825	-0.13001
thld:0.045	0.02207	0.00674	0.02960	0.02064	0.01244	0.06823	0.20932	0.10297	0.10829	-0.02261	-0.08649	-0.03125	-0.13762	-0.33910	-0.14421
thld:0.050	0.02041	-0.00430	0.00566	0.02940	0.02743	0.07954	0.25086	0.14570	0.08429	0.01768	-0.03927	0.03903	0.19080	-0.20887	0.07956
thld:0.100	0.01912	-0.09500	-0.04660	-0.02973	0.00254	0.00940	0.22829	0.09268	-0.06872	0.06922	-0.20296	0.12474	0.66758	0.45312	0.46016

Figure A.17: Triple top Cohen's h

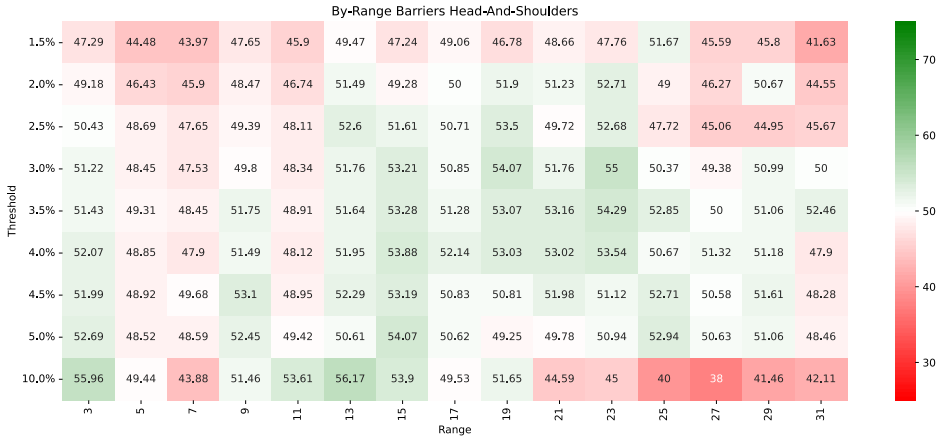


Figure A.18: Head and shoulders win rate

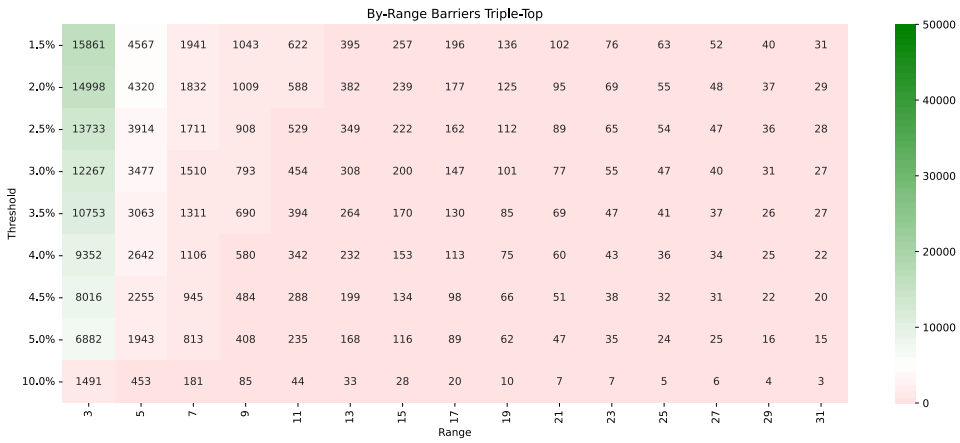


Figure A.19: Head and shoulders samples

	rng:3,00	rng:5,00	rng:7,00	rng:9,00	rng:11,00	rng:13,00	rng:15,00	rng:17,00	rng:19,00	rng:21,00	rng:23,00	rng:25,00	rng:27,00	rng:29,00	rng:31,00
thld:0,015	0,95450	0,01205	0,01424	0,78941	0,48947	0,30782	0,80479	0,29869	0,94071	0,56738	0,62510	0,22782	0,40102	0,66504	0,10207
thld:0,020	0,78324	0,00496	0,01037	0,64465	0,18988	0,23538	0,92775	0,51593	0,11406	0,51472	0,10667	0,75438	0,25625	0,74168	0,14563
thld:0,025	0,96273	0,12824	0,06692	0,54048	0,22242	0,23471	0,57372	0,70876	0,17759	0,75408	0,23516	0,23330	0,04321	0,14026	0,14046
thld:0,030	0,61069	0,01878	0,02796	0,48612	0,15265	0,68266	0,33064	0,88729	0,17328	0,74140	0,13753	0,60914	0,35560	0,97796	0,55615
thld:0,035	0,63919	0,07071	0,04196	0,97767	0,21288	0,82088	0,46903	0,86765	0,61761	0,56457	0,29438	0,79881	0,42706	0,75502	0,92309
thld:0,040	0,42347	0,01504	0,01068	0,89102	0,08350	0,98732	0,44936	0,91428	0,68558	0,79617	0,52827	0,68399	0,79802	0,82172	0,23086
thld:0,045	0,74412	0,01604	0,20050	0,83251	0,14122	0,93709	0,62949	0,50881	0,72399	0,85764	0,83129	0,96921	0,49067	0,76911	0,26705
thld:0,050	0,49605	0,00267	0,03397	0,91264	0,26764	0,37154	0,46799	0,54073	0,45704	0,44231	0,71672	0,95455	0,54095	0,71017	0,29518
thld:0,100	0,05856	0,02396	0,00015	0,51635	0,81603	0,74987	0,72556	0,49501	0,73385	0,05609	0,17371	0,02948	0,03477	0,12232	0,09121

Figure A.20: Head and shoulders p-values

	rng:3,00	rng:5,00	rng:7,00	rng:9,00	rng:11,00	rng:13,00	rng:15,00	rng:17,00	rng:19,00	rng:21,00	rng:23,00	rng:25,00	rng:27,00	rng:29,00	rng:31,00
thld:0,015	-0,00103	-0,05812	-0,06520	0,00801	-0,02335	0,03744	0,00981	0,04509	0,00348	0,02959	0,02669	0,06907	-0,05086	-0,02828	-0,11029
thld:0,020	-0,00498	-0,06527	-0,06848	-0,01395	-0,04466	0,04389	-0,00365	0,02850	0,07428	0,03400	0,08861	-0,01806	-0,06937	0,02193	-0,09834
thld:0,025	0,00086	-0,03573	-0,04968	-0,01887	-0,04211	0,04504	0,02321	0,01688	0,06477	-0,01662	0,06653	-0,07013	-0,12697	-0,10029	-0,10231
thld:0,030	0,00950	-0,05654	-0,06094	-0,02193	-0,05059	0,01578	0,04122	-0,00659	0,06749	0,01794	0,08619	-0,03127	-0,05942	-0,00193	-0,04223
thld:0,035	0,00894	-0,04520	-0,05813	-0,00091	-0,04521	0,00901	0,03141	-0,00806	0,02574	0,03245	0,06286	0,01627	-0,05380	-0,02287	-0,00711
thld:0,040	0,01571	-0,06266	-0,07548	-0,00460	-0,06523	-0,00066	0,03415	0,00544	0,02184	0,01536	0,03990	-0,02715	-0,01866	-0,01747	-0,09289
thld:0,045	0,00662	-0,06467	-0,03931	0,00735	-0,05745	0,00342	0,02268	-0,03503	-0,02008	-0,01121	-0,01425	-0,00267	-0,05206	-0,02348	-0,09168
thld:0,050	0,01431	-0,08411	-0,06775	-0,00399	-0,04534	-0,04091	0,03624	-0,03414	-0,04510	-0,05048	-0,02505	0,00415	-0,04828	-0,03147	-0,09194
thld:0,100	0,06269	-0,10001	-0,19779	-0,03957	0,01703	0,02549	0,02957	-0,06588	-0,03575	-0,22132	-0,17462	-0,29257	-0,29502	-0,24206	-0,27555

Figure A.21: Head and shoulders Cohen's h

A.1 Dataset Information

Last retrieval data

April 23, 2023

List of Symbols

A, AAPL, ABB, ABC, ABEV, ABMD, ABT, ACGL, ACIW, ACN, ADI, ADP, AEE, AEG, AEM, AEP, AES, AFG, AFL, AGCO, AGR, AIG, AIN, AIT, AJG, AJRD, ALGN, ALK, ALKS, ALL, ALV, AMAT, AMD, AME, AMG, AMGN, AMKR, AMOV, AMT, AN, ANSS, AON, AOS, APA, APD, APH, ARE, ARW, ASH, ASML, ATI, ATVI, AU, AVA, AVT, AVY, AZN, AZO, BA, BAC, BALL, BAM, BAP, BB, BBD, BBVA, BBWI, BBY, BC, BCE, BCS, BEN, BG, BHC, BIIB, BIO, BK, BKNG, BLK, BMO, BMRN, BMY, BOH, BP, BPOP, BRC, BRFS, BRKR, BRO, BSAC, BWA, BXP, BYD, CAG, CAH, CAJ, CAR, CASY, CB, CBRL, CBSH, CBT, CCEP, CCI, CCJ, CCL, CDNS, CEA, CHD, CHE, CHKP, CHRW, CI, CIB, CIEN, CINF, CL, CLF, CMA, CMC, CMI, CMS, CNA, CNC, CNI, CNP, CNQ, COF, COKE, COO, COP, COST, CPB, CPRT, CR, CRH, CRUS, CS, CSCO, CSGP, CSGS, CSL, CTAS, CTRA, CTSH, CUK, CVS, CZR, D, DB, DCI, DD, DDS, DHI, DHR, DIS, DISH, DLTR, DOV, DRI, DTE, DUK, DVA, DVN, DXC, E, EBAY, ECL, ED, EHC, EL, ELP, ELV, EMN, ENB, EOG, EPD, EQNR, EQR, ERIC, ERIE, ES, ESS, ETR, EVRG, EW, EXC, EXPO, F, FAST, FE, FFIV, FHN, FIS, FISV, FITB, FL, FLR, FLS, FMC, FMS, FRHC, FSS, FUL, GD, GE, GEF, GFI, GGB, GGG, GHC, GIB, GILD, GIS, GL, GLW, GOLD, GPC, GPN, GPS, GRMN, GS, GSK, GT, GWW, HAL, HAS, HBAN, HD, HDB, HE, HEI, HES, HIG, HL, HMC, HOG, HON, HP, HPQ, HRB, HRL, HSBC, HSIC, HST, HSY, HUBB, HUM, HXL, IBM, IBN, IDA, IFF, ILMN, INCY, INFY, ING, INTC, INTU, IP, IPG, ISRG, IT, ITUB, ITW, IVZ, J, JBHT, JBL, JBLU, JCI, JEF, JOE, JPM, JWN, K, KAI, KBH, KEP, KEY, KGC, KLAC, KLIC, KMB, KMPR, KMT, KO, KOF, KR, KSS, KT, L, LAMR, LANC, LEG, LEN, LH, LIN, LLY, LMT, LNC, LNT, LUMN, LUV, M, MAR, MAS, MAT, MCD, MCHP, MCK, MCO, MDU, MEOH, MET, MFC, MGA, MGM, MKC, MKL, MLM, MMC, MMM, MNST, MO, MOS, MRO, MS, MSA, MSI, MSTR, MT, MTB, MTD, MTG, MTZ, MU, MUFG, MUR, NCR, NEE, NEM, NEU, NFG, NI, NKE, NLOK, NLY, NOK, NOV, NSC, NTAP, NTES, NUE, NUS, NVDA, NVO, NVR, NVS, NWL, NYT, O, ODFL, OGE, OI, OKE, OLN, OMC, ON, ORCL, ORI, ORLY, OSK, OVV, OXY, PAA, PARA, PBR, PCAR, PCG, PCH, PDCO, PEAK, PEG, PEP, PFE, PG, PGR, PH, PHG, PHI, PHM, PKI, PLD, PLUG, PLXS, PNR, PNW, PPC, PPG, PPL, PRGO, PRU, PSA, PTR, PVH, QCOM, QGEN, R, RCI, RCL, RDN, REGN, RF, RGEN, RHI, RIG, RL, RMBS, RMD, ROG, ROL, ROP, ROST, RPM, RRC, RY, SAN, SANM, SAP, SBAC, SCCO, SCI, SEB, SEE, SEIC, SGEN, SHI, SHW, SID, SIG, SIGI, SIRI, SIVB, SJM, SJR, SKM, SKY, SLAB, SLB, SLF, SLM, SMG, SNA, SNN, SNPS, SNV, SO, SON, SONY, SPG, SPGI, SQM, SRE, SSL, STE, STM, STT, STZ, SWK, SWKS, SWN, SXT, SYK, SYY, T, TD, TEF, TEO, TEVA, TFC, TGNA, TGT, THC, TIMB, TKC, TKR, TLK, TM, TMO, TOL, TPL, TPR, TR, TRMB, TRN, TROW, TRP, TRQ, TRV, TSM, TT, TTC, TTE, TTEC, TTWO, TV, TXN, TXT, TYL, UGP, UHS, UL, UMBF, UMC, UNF, UNH, UNM, UNP, UPS, URI, USB, VALE, VFC, VIAV, VICR, VIV, VLO, VLY, VMC, VMI, VNO, VOD, VRSN, VTR, VTRS, VZ, WAFD, WAT, WBA, WEC, WELL, WERN, WHR, WIT, WM, WMB, WMK, WMT, WOLF, WOR, WPP, WRB, WSM, WST, WTM, WTS, WTW, WY, XEL, XOM, XRAY, Y, YPF, YUM, ZBH, ZBRA, ZION, ZNH

References

1. Nison S. Japanese candlestick charting techniques: A contemporary guide to a client investment technique far east. Philadelphia, PA: Prentice Hall; 1991.

2. Murphy JJ. Technical analysis of the financial markets: Study guide: A comprehensive guide to trading methods and applications. 2nd ed. Welwyn Garden City, England: NYIF; 1998.

3. Gregory-Allen RB, Shawky HA, Stangl J. Quantitative vs. Fundamental analysis in institutional money management: Where's the beef? The journal of investing [Internet]. 2009;18(4):42–52. Available from: http://dx.doi.org/10.3905/joi.2009.18.4.042

4. Barber BM, Odean T. Online investors: Do the slow die first? The review of financial studies [Internet]. 2002;15(2):455–88. Available from: http://dx.doi.org/10.1093/rfs/15.2.455

5. Ankenbrand T, Bieri D. Assessment of cryptocurrencies as an asset class by their characteristics. Investment management and financial innovations [Internet]. 2018;15(3):169–81. Available from: http://dx.doi.org/10.21511/imfi.15(3).2018.14

6. Jiang W. Applications of deep learning in stock market prediction: Recent progress. Expert systems with applications [Internet]. 2021;184(115537):115537. Available from: http://dx.doi.org/10.1016/j.eswa.2021.115537

7. Menkhoff L, Taylor MP. The obstinate passion of foreign exchange professionals: Technical analysis. Journal of Economic Literature [Internet]. 2007;45(4):936–72. Available from: http://dx.doi.org/10.1257/jel.45.4.936

8. Park CH, Irwin SH. What do we know about the profitability of technical analysis? Journal of Economic Surveys [Internet]. 2007;21(4):786–826. Available from: http://dx.doi.org/10.1111/j.1467-6419.2007.00519.x

9. Teall JL. Financial trading and investing. San Diego, CA: Academic Press; 2023.

10. Mitchell WC. Quantitative analysis in economic theory. American Economic Review [Internet]. 1925;15(1):1–12. Available from: http://www.jstor.org/stable/1808475

11. Osler CL. Support for resistance: Technical analysis and intraday exchange rates [Internet]. 2006. Available from: https://papers.ssrn.com/abstract=888805

12. Zhu Y, Zhou G. Technical analysis: An asset allocation perspective on the use of moving averages. Journal of financial economics [Internet]. 2009;92(3):519–44. Available from: http://dx.doi.org/10.1016/j.jfineco.2008.07.002

13. Welles Wilder J. New concepts in technical trading systems. Trend Research; 1978.

14. Bollinger J. Bollinger on bollinger bands. New York, NY: McGraw-Hill Professional; 2002.

15. Smith BM. A history of the global stock market: From ancient rome to silicon valley. Chicago, IL: University of Chicago Press; 2004.

16. Quinn W, Turner JD. Boom and bust: A global history of financial bubbles. Cambridge, England: Cambridge University Press; 2021.

17. Angel JJ, McCabe D. Fairness in financial markets: The case of high frequency trading. Journal of business ethics [Internet]. 2013;112(4):585–95. Available from: http://dx.doi.org/10.1007/s10551-012-1559-0

18. Kirilenko AA, Lo AW. Moore's law versus murphy's law: Algorithmic trading and its discontents. The journal of economic perspectives: a journal of the American Economic Association [Internet]. 2013;27(2):51–72. Available from: http://dx.doi.org/10.1257/jep.27.2.51

19. Fama EF. Efficient capital markets: A review of theory and empirical work. The journal of finance [Internet]. 1970;25(2):383. Available from: http://dx.doi.org/10.2307/2325486

20. Malkiel BG. The efficient market hypothesis and its critics. The journal of economic perspectives: a journal of the American Economic Association [Internet]. 2003;17(1):59–82. Available from: http://dx.doi.org/10.1257/089533003321164958

21. Kahneman D, Tversky A. Prospect theory: An analysis of decision under risk. In: Handbook of the fundamentals of financial decision making. WORLD SCIENTIFIC; 2013. p. 99–127.

22. Barberis N, Shleifer A, Vishny R. A model of investor sentiment. Journal of financial economics [Internet]. 1998;49(3):307–43. Available from: https://www.sciencedirect.com/science/article/pii/S0304405X98000270

23. Dang HV, Lin M. Herd mentality in the stock market: On the role of idiosyncratic participants with heterogeneous information. International review of financial analysis [Internet]. 2016;48:247–60. Available from: https://www.sciencedirect.com/science/article/pii/S1057521916301570

24. Odean T. Volume, volatility, price, and profit when all traders are above average. The journal of finance [Internet]. 1998;53(6):1887–934. Available from: http://dx.doi.org/10.1111/0022-1082.00078

25. Huberman G, Regev T. Contagious speculation and a cure for cancer: A nonevent that made stock prices soar. The journal of finance [Internet]. 2001;56(1):387–96. Available from: http://dx.doi.org/10.1111/0022-1082.00330

26. Bulkowski TN. Trading classic chart patterns. Nashville, TN: John Wiley & Sons; 2002.

27. Schwager JD, Etzkorn M. Getting started in technical analysis. Nashville, TN: John Wiley & Sons; 1999.

28. Bulkowski Thomas N, Bulkowski TN. Encyclopedia of chart patterns. 2nd ed. John Wiley & Sons; 2005.

29. O'Neil WJ. How to make money in stocks: A winning system in good times or bad. 3rd ed. New York, NY: McGraw-Hill Professional; 2002.

30. Magee J, Edwards RD. Technical analysis of stock trends. 5th ed. John Magee; 1987.

31. Grimes A. The art and science of technical analysis: Market structure, price action, and trading strategies. Grimes A, editor. Nashville, TN: John Wiley & Sons; 2012.

32. Murphy JJ. The visual investor: How to spot market trends. 2nd ed. Murphy JJ, editor. Chichester, England: John Wiley & Sons; 2009.

33. Farias Nazário RT, Silva JL e, Sobreiro VA, Kimura H. A literature review of technical analysis on stock markets. The Quarterly review of economics and finance: journal of the Midwest Economics Association [Internet]. 2017;66:115–26. Available from: http://dx.doi.org/10.1016/j.qref.2017.01.014

34. Vu JDV. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. The CFA digest [Internet]. 2001;31(1):26–8. Available from: http://dx.doi.org/10.2469/dig.v31.n1.811

35. Tsinaslanidis P, Guijarro F. What makes trading strategies based on chart pattern recognition profitable? Expert systems [Internet]. 2021;38(5). Available from: http://dx.doi.org/10.1111/exsy.12596

36. Li X, Wu P. Stock price prediction incorporating market style clustering. Cognitive computation [Internet]. 2022;14(1):149–66. Available from: http://dx.doi.org/10.1007/s12559-021-09820-1

37. Warren Liao T. Clustering of time series data—a survey. Pattern recognition [Internet]. 2005;38(11):1857–74. Available from: http://dx.doi.org/10.1016/j.patcog.2005.01.025

38. Lento C, Gradojevic N. The profitability of technical trading rules: A combined signal approach. Journal of Applied Business Research (JABR) [Internet]. 2011;23(1). Available from: http://dx.doi.org/10.19030/jabr.v23i1.1405

39. Lopez de Prado M. Advances in financial machine learning. Nashville, TN: John Wiley & Sons; 2018.

40. Black F. Noise. The journal of finance [Internet]. 1986;41(3):528–43. Available from: http://dx.doi.org/10.1111/j.1540-6261.1986.tb04513.x

41. go.dev. Available from: https://go.dev

42. Lenarduzzi V, Lomio F, Saarimäki N, Taibi D. Does migrating a monolithic system to microservices decrease the technical debt? The Journal of systems and software [Internet]. 2020;169(110710):110710. Available from: https://www.sciencedirect.com/science/article/pii/S0164121220301539

43. Prices EH, Api FD. Fundamental, EOD historical prices and financial data API [Internet]. Eodhd.com. Available from: https://eodhd.com

44. Binance - cryptocurrency exchange for bitcoin, ethereum & altcoins [Internet]. Binance. Available from: https://www.binance.com

45. Liu J, Serletis A. Volatility in the cryptocurrency market. Open economies review [Internet]. 2019;30(4):779–811. Available from: http://dx.doi.org/10.1007/s11079-019-09547-5

46. Pernici B, Bernardini CSCC. Big data quality: Stock market data sources assessment [Internet]. Polimi.it. Available from: https://www.politesi.polimi.it/bitstream/10589/154164/3/Masters%20Thesis%20of%20Mohamed%20Elfakhfakh___A.pdf

47. Rolemberg H, Rolemberg H, Koike G. Backtesting is about the past [Internet]. Daemoninvestments.com. Available from: https://daemoninvestments.com/wp-content/uploads/2022/06/WP_backtesting-1.pdf

48. Kan R, Kirikos G, Halpern P, Andrew G. Biases in evaluating trading strategies [Internet]. Utoronto.ca. Available from: https://www-2.rotman.utoronto.ca/~kan/papers/mom1.pdf

49. Apache SparkTM - unified engine for large-scale data analytics [Internet]. Apache.org. Available from: https://spark.apache.org

50. Apache kafka [Internet]. Apache Kafka. Available from: https://kafka.apache.org

51. Salah T, Jamal Zemerly M, Yeun CY, Al-Qutayri M, Al-Hammadi Y. The evolution of distributed systems towards microservices architecture. In: 2016 11th international conference for internet technology and secured transactions (ICITST). IEEE; 2016. p. 318–25.

52. Laigner R, Zhou Y, Salles MAV, Liu Y, Kalinowski M. Data management in microservices: State of the practice, challenges, and research directions [Internet]. arXiv [cs.DB]. 2021. Available from: http://arxiv.org/abs/2103.00170

53. Balamash A, Krunz M. An overview of web caching replacement algorithms. IEEE Communications Surveys & Tutorials [Internet]. Second 2004;6(2):44–56. Available from: https://ieeexplore.ieee.org/abstract/document/5342239

54. Mux: Package gorilla/mux [Internet]. Available from: https://github.com/gorilla/mux

55. Negoni: Package urfave/negroni [Internet]. Available from: https://github.com/urfave/negroni

56. Miretskiy Y. High-performance JSON parsing in go [Internet]. Cockroach Labs. 2023. Available from: https://www.cockroachlabs.com/blog/high-performance-json-parsing/

57. Gob package documentation [Internet]. Pkg.go.dev. Available from: https://pkg.go.dev/encoding/gob

58. Http package documentation [Internet]. Pkg.go.dev. Available from: https://pkg.go.dev/net/http

59. HTML5 financial charting library for stocks, forex and crypto_2021 [Internet]. Devexperts. Devexperts GMBH; 2021. Available from: https://devexperts.com/dxcharts/

60. Free charting library by TradingView [Internet]. TradingView. Available from: https://www.tradingview.com/HTML5-stock-forex-bitcoin-charting-library/

61. React [Internet]. React.dev. Available from: https://react.dev

62. Clark A. React-fiber-architecture: A description of react's new core algorithm, react fiber [Internet]. Available from: https://github.com/acdlite/react-fiber-architecture

63. Built-in react hooks [Internet]. React.dev. Available from: https://react.dev/reference/react

64. Keeping components pure [Internet]. React.dev. Available from: https://react.dev/learn/keeping-components-pure

65. Effective go [Internet].go.dev. Available from: https://go.dev/doc/effective_go

66. Binance API documentation [Internet]. Binance. Available from: https://www.binance.com/en/binance-api

67. Asquith P, Healy P, Palepu K. Earnings and stock splits. The accounting review [Internet]. 1989;64(3):387–403. Available from: http://www.jstor.org/stable/247596

68. Lakonishok J, Lev B. Stock splits and stock dividends: Why, who, and when. The journal of finance [Internet]. 1987;42(4):913–32. Available from: http://dx.doi.org/10.1111/j.1540-6261.1987.tb03919.x

69. Ristretto: Package dgraph-io/ristretto [Internet]. Available from: https://github.com /dgraph-io/ristretto

70. SPY backtesting for the european index investor [Internet]. Curvo.eu. Available from: https://curvo.eu/backtest/portfolio