UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA



Evolving Financial Trading Agents Using Vectorial Genetic Programming

Rui Miguel Santos Menoita

Mestrado em engenharia informática

Dissertação orientada por:
Doutora Sara Guilherme Oliveira da Silva

2023

Agradecimentos

No percurso desta tese, tive a sorte de estar rodeado por pessoas e instituições que foram fundamentais para o seu sucesso e, por isso, gostaria de expressar o meu sincero agradecimento.

À minha familia, em especial aos meus pais e irmã, que sempre acreditaram em mim e no meu potencial. O vosso apoio constante e palavras de motivação foram a base sólida sobre a qual pude construir este trabalho.

À minha namorada, Inês Raquel, pela sua presença constante, apoio incondicional e motivação. Escrever uma tese é um desafio que ambos enfrentamos juntos e a sua ajuda e compreensão foram essenciais em cada etapa desta jornada.

Ao meu amigo, Francisco Cardoso, pelo seu entusiasmo contagiante e palavras de incentivo. A camaradagem de enfrentarmos a tarefa de escrever uma tese ao mesmo tempo foi uma fonte de motivação mútua.

À minha orientadora, Professora Sara Silva, pelo seu aconselhamento, paciência e direção. A sua expertise e orientação foram cruciais para moldar esta tese e torná-la o que é hoje.

Gostaria também de expressar a minha gratidão às instituições que desempenharam um papel crucial neste processo. À FCUL, pela educação de excelência e recursos providenciados. À Inetum e à Tripadvisor, pelas oportunidades profissionais que me permitiram equilibrar trabalho e estudo, e pela compreensão e apoio demonstrados enquanto escrevia esta tese.

Dedicatória.

Resumo

Num mundo cada vez mais dominado pela tecnologia e pela análise de dados, os mercados financeiros não são exceção. A capacidade de interpretar corretamente as vastas quantidades de dados gerados diariamente nos mercados pode ser a diferença entre um investimento bem-sucedido e uma perda significativa. Neste contexto, surge a motivação desta tese: entender e aplicar técnicas avançadas de análise de dados, especificamente a Programação Genética (GP), para interpretar e prever tendências nos mercados financeiros.

A análise técnica é uma abordagem tradicionalmente usada no mercado financeiro, que se concentra em identificar padrões em dados históricos do preço e volume para prever futuras movimentações do preço. Baseia-se na ideia de que os movimentos do preço passado podem indicar tendências futuras. A análise técnica é amplamente utilizada por traders e investidores para tomar decisões informadas sobre quando comprar ou vender um ativo.

No entanto, à medida que o volume de dados cresce e os mercados se tornam mais complexos, a análise técnica tradicional pode não ser suficiente para capturar todas as nuances e padrões. É aqui que a Programação Genética (GP) entra em cena. A GP é uma técnica de aprendizagem automática que simula a evolução natural para resolver problemas. Em vez de procurar soluções num conjunto pré-definido, a GP permite que as soluções evoluam, adaptando-se e melhorando ao longo do tempo. Quando aplicada ao mercado financeiro, a GP pode ser usada para evoluir e adaptar estratégias de negociação com base na análise de dados históricos.

A combinação da análise técnica com a GP representa uma abordagem moderna para interpretar e responder às complexidades do mercado financeiro. É com essa mentalidade que esta tese procura explorar a aplicabilidade e eficácia da GP no contexto do mercado financeiro.

Um ponto crucial para qualquer pesquisa baseada em dados é a qualidade e relevância dos dados escolhidos. Para este estudo, a seleção de dados foi criteriosa, focando em instrumentos financeiros específicos que representam uma variedade de comportamentos de mercado. Foram escolhidos quatro instrumentos financeiros: COTY, KO, TWTR e PSI20. A escolha desses instrumentos foi feita com o intuito de representar diferentes

fases do mercado, como tendências de alta, baixa e mercados neutros. Os dados selecionados abrangem um período de tempo específico, de 1 de janeiro de 2015 a 14 de abril de 2022, garantindo continuidade e consistência nos dados examinados.

A normalização dos dados é uma etapa importante para assegurar que análises e modelos se fundamentem em dados consistentes e comparáveis. Neste estudo, recorremos a dois métodos de normalização: por linha e por janela. A normalização por linha ajusta o vetor de dados com base nos pontos mínimo e máximo do próprio vetor. Já a normalização por janela toma por base uma janela de tempo específica - como por exemplo os 200 dias anteriores - para determinar os valores máximo e mínimo a serem aplicados.

Além da seleção e normalização dos dados, um passo adicional foi tomado para enriquecer os dados: a hidratação dos dados. Este processo envolveu a adição de indicadores técnicos, como Médias Móveis Exponenciais (EMAs) de diferentes períodos (5, 13, 50 e 200 dias) e o Índice de Força Relativa (RSI). Além disso, foram introduzidos indicadores personalizados, como 'smallEmaDiff' e 'bigEmaDiff', que são derivados das EMAs e ajudam a identificar pontos de inflexão nas tendências do mercado. Estas adições foram feitas com o objetivo de fornecer uma visão mais completa e aprofundada dos movimentos do mercado e auxiliar os modelos GP nas suas previsões e decisões.

Dentro da Programação Genética (GP), existem várias técnicas que podem ser adaptadas e ajustadas para se adequar ao domínio específico em análise. Para esta pesquisa, foram escolhidos quatro paradigmas distintos de GP, cada um com suas peculiaridades e focos.

O primeiro, o método Standard GP, é uma abordagem tradicional que trabalha com expressões matemáticas como soluções. No contexto de negociação financeira, este método visa evoluir expressões que são capazes de prever preços de ações. Diferentemente de alguns outros métodos, o Standard GP utiliza valores escalares como input.

Em seguida, temos o método Vectorial GP (VGP), que é uma extensão do Standard GP. A principal diferença é que o VGP trabalha com vetores em vez de valores escalares. Isto permite que o método opere com funções vetoriais, que aceitam um ou mais vetores como input e produz um vetor de output.

O Strongly Typed Vectorial GP (STVGP) é outro método explorado, que introduz uma combinação de tipos de valores, booleanos e vetores. Ele tem a capacidade de gerar modelos que equilibram precisão e compreensibilidade, tornando-se relevante em domínios onde a lógica subjacente é tão importante quanto a precisão da previsão.

Finalmente, o Complex Vectorial GP é uma variação que traz números complexos para a equação. Isto permite a modelagem de expressões matemáticas que vão além das restrições do domínio real, oferecendo potencialmente uma modelagem mais robusta em certos cenários.

Após a implementação dos métodos de GP escolhidos, seguiu-se uma fase de avaliação e análise dos resultados. Cada técnica foi aplicada ao conjunto de dados preparado, e os

resultados foram registados em termos de retorno sobre o investimento (ROI) para cada instrumento financeiro.

A análise descortinou padrões e tendências notáveis na performance dos métodos de programação genética (GP). Enquanto o método Vectorial GP (VGP) manifestou competência em ambientes específicos, a sua performance não se destacou de forma universalmente superior em todos os cenários. Contrariamente, o Strongly Typed Vectorial GP (STVGP) mostrou-se promissor em diversas situações, apesar de não estar isento de desafios e dificuldades. Importante realçar que os resultados não apontam para a existência de um método categoricamente superior; a eficácia de cada técnica foi substancialmente influenciada pelo contexto no qual foi aplicada.

Foi observado que todos os modelos que empregaram o VGP apresentaram, em média, uma performance superior quando comparados aos que utilizaram GP convencional. Esta superioridade pode estar relacionada ao fato de que, ao utilizar VGP, os modelos recebem vetores como input, proporcionando uma quantidade maior de informações para fundamentar as suas decisões.

Além da análise quantitativa, a interpretabilidade dos modelos gerados também foi uma prioridade. Soluções com alto Retorno sobre o Investimento (ROI) podem não ser práticas se a sua complexidade ou falta de clareza impedirem a compreensão e a aplicabilidade no mundo real. Portanto, buscou-se um equilíbrio entre a eficácia do modelo e a simplicidade, garantindo que os modelos gerados fossem tanto eficazes quanto compreensíveis.

Olhando para o futuro, existem várias áreas em que a pesquisa pode ser expandida e aprofundada. Uma delas é a exploração de outras técnicas de GP não abordadas neste estudo. Além disso, devido à natureza dinâmica dos mercados financeiros, é prudente que os métodos sejam reavaliados regularmente à medida que novos dados tornam-se disponíveis, garantindo que os modelos se mantenham relevantes e eficazes face ás flutuações do mercado. Outro caminho promissor reside na combinação de GP com outras técnicas de aprendizagem de máquina, buscando desenvolver uma abordagem híbrida que possa capitalizar sobre os pontos fortes de múltiplas metodologias.

É importante destacar que a natureza não-estacionária dos dados financeiros apresentou um desafio considerável na comparação do desempenho entre os agentes. Os mercados financeiros são permeados por uma variedade de fatores macroeconômicos e eventos globais, tornando os dados altamente voláteis e menos previsíveis. Portanto, uma investigação futura pode se beneficiar do uso de dados de setores com características mais previsíveis, como as indústrias de energia ou farmacêutica. Explorar a aplicação de técnicas de GP em dados oriundos dessas indústrias poderia proporcionar insights adicionais sobre a eficácia dos métodos em diferentes contextos.

Em resumo, embora esta pesquisa tenha fornecido insights valiosos e aumentado a compreensão das técnicas de GP no contexto da negociação, o campo permanece aberto e

repleto de oportunidades para investigação adicional. A evolução contínua dos mercados, juntamente com os avanços na área de GP, promete um futuro rico em descobertas e inovações.

Palavras-chave: Programação genética, previsão de séries temporais, programação genética vectorial, análise comparativa, mercados financeiros

Abstract

Navigating the intricacies of establishing profitable trading strategies in financial markets is a challenging endeavour. While traditional methods like technical analysis have long served as foundational tools for traders to recognize and act upon market patterns, the evolving landscape has called for more advanced techniques. In this master's research, an approach is introduced that integrates insights from technical analysis with a refined genetic programming (GP) framework, with the aim of improving the creation of trading rules. The methodology merges the principles of strongly typed genetic programming (STGP) with vectorial GP, leading to the development of the strongly typed vectorial GP (STVGP) technique. Within this framework, trading agents work with vector-based inputs, and a set of operations is introduced to adjust these vectors, ensuring compliance with the inherent constraints of STGP.

The core objective of this research is to evaluate various vectorial GP techniques, with a focus on STVGP, to ascertain their effectiveness in formulating trading strategies. The STVGP method demonstrated the ability to construct strategies that navigate various market conditions, secure profit, minimize risks, and prevent overfitting. This research enhances our understanding of the potential applications and efficacy of GP techniques in trading strategy development.

Keywords: Genetic programming, Time series forecast, Vectorial Genetic programming, Comparative analysis, Financial Markets

Contents

Lista de Figuras **xviii**

Lista de Tabelas **xxi**

1 Introduction **1**
- 1.1 Motivation and Objectives . . . 2
- 1.2 Main Contributions . . . 2
- 1.3 Document structure . . . 3

2 Background **5**
- 2.1 Financial markets . . . 5
- 2.2 Fundamental Analysis . . . 6
- 2.3 Technical analysis . . . 6
  - 2.3.1 Technical Indicators . . . 7
- 2.4 Genetic Programming . . . 11
  - 2.4.1 What is Genetic Programming? . . . 11
  - 2.4.2 How does Genetic Programming work? . . . 11
- 2.5 Related work . . . 13
- 2.6 Summary . . . 15

3 Data processing **19**
- 3.1 Data representation . . . 19
- 3.2 Data Normalization . . . 20
  - 3.2.1 Row Normalization . . . 20
  - 3.2.2 Window normalization . . . 21
- 3.3 Data Selection . . . 23
- 3.4 Data Hydration . . . 24
- 3.5 Summary . . . 26

4 Methods **29**
- 4.1 Standard GP . . . 29
- 4.2 Vectorial GP . . . 30

4.3 Complex Vectorial GP . . . 31
4.4 Strongly Typed Vectorial GP . . . 34
4.5 Fitness Function . . . 34
4.6 Parameter Settings . . . 35
4.7 Summary . . . 36

5 Results **41**
5.1 Evolution of Best Agents . . . 41
5.1.1 GP . . . 42
5.1.2 VGP . . . 43
5.1.3 VGP Row normalized (VGP Rn) . . . 43
5.1.4 Complex VGP (CVGP) . . . 44
5.1.5 Complex VGP Row normalized (CVGP Rn) . . . 45
5.1.6 STVGP . . . 45
5.2 Agent Analysis . . . 46
5.2.1 GP . . . 46
5.2.2 VGP . . . 46
5.2.3 VGP Row normalized (VGP Rn) . . . 47
5.2.4 Complex VGP (CVGP) . . . 47
5.2.5 Complex VGP Row normalized (CVGP Rn) . . . 48
5.2.6 STVGP . . . 48
5.2.7 Insights into Strategy Formulation . . . 49
5.3 Comparison . . . 50
5.3.1 Run analysis . . . 50
5.3.2 ROI Table Analysis . . . 56
5.3.3 Boxplot Analysis . . . 57
5.4 Statistical analysis . . . 59
5.4.1 COTY . . . 59
5.4.2 KO . . . 59
5.4.3 PSI20 . . . 61
5.4.4 TWTR . . . 62
5.5 Summary . . . 63

6 Conclusion **65**
6.1 Summary of Work and Key Findings . . . 65
6.2 Implications of Findings . . . 65
6.3 Concluding Remarks . . . 66

Abreviaturas **69**

Bibliografia **75**

Índice **76**

List of Figures

2.1 SMA 50 (green) and SMA 200 (orange) in AAPL instrument 8
2.2 EMA 50 (green) and EMA 200 (orange) in AAPL instrument 9
2.3 RSI in AAPL instrument 10
2.4 Example of an individual built with the function set $\{+, \times\}$ and the terminal set $\{C_0, C_1\}$. 12
2.5 Genetic Programming algorithm. 13
2.6 Example of a standard subtree crossover operation 14

3.1 Illustration of two candlesticks elucidating price trajectories, trends, and cardinal values (high, low, close, open). 20
3.2 Candlestick chart delineating the temporal evolution of prices. 20
3.3 Example of the window normalization algorithm's normalization using a window of length five. 23
3.4 Instrument prices spanning January 1, 2015, to April 14, 2022. 25

4.1 Decision planes represented in a Cartesian plane. The blue plain represents the sell signal and the red plane the buy signal 32

5.1 Evolution of Best Fitness and Test Fitness for GP on COTY dataset. . . . 42
5.2 Evolution of Best Fitness and Test Fitness for VGP on COTY dataset. . . 43
5.3 Evolution of Best Fitness and Test Fitness for VGP(Rn) on COTY dataset. 44
5.4 Evolution of Best Fitness and Test Fitness for Complex VGP on COTY dataset. 44
5.5 Evolution of Best Fitness and Test Fitness for Complex VGP(Rn) on COTY dataset. 45
5.6 Evolution of Best Fitness and Test Fitness for STVGP on COTY dataset. . 45
5.7 Comparison of Best Fitness during Training and Test Fitness across 10 runs for COTY stock. Each color represents one run; the thicker black line represents the median at each generation. The y axis may cut some lines to facilitate the visualization. 51

5.8 Comparison of Best Fitness during Training and Test Fitness across 10 runs for KO stock. Each colour represents one run; the thicker black line represents the median at each generation. The y-axis may cut some lines to facilitate the visualization. 52
5.9 Comparison of Best Fitness during Training and Test Fitness across 10 runs for PSI20 stock. Each colour represents one run; the thicker black line represents the median at each generation. The y-axis may cut some lines to facilitate the visualization. 54
5.10 Comparison of Best Fitness during Training and Test Fitness across 10 runs for TWTR stock. Each color represents one run; the thicker black line represents the median at each generation. The y axis may cut some lines to facilitate the visualization. 55
5.11 Comparative analysis of ROI across methods and runs during the test phase, stratified by instrument 58

List of Tables

4.1 Operations in Standard GP 30
4.2 Operations in Vectorial GP 31
4.3 Complex Vectorial Operations 33
4.4 Operations Strongly Typed Vectorial GP method 39

5.1 ROI Medians by method and stock. 56
5.2 Pairwise p-values from the Kruskal-Wallis H-test for the COTY dataset. . 59
5.3 Pairwise p-values from the Kruskal-Wallis H-test for the KO dataset. . . . 60
5.4 Pairwise p-values from the Kruskal-Wallis H-test for the PSI20 dataset. . 61
5.5 Pairwise comparison between different models on TWTR dataset 62

Chapter 1

Introduction

Financial time series forecasting poses significant challenges, as underscored by some economists through the efficient market hypothesis [15]. It suggests that if market participants could forecast the price, they could generate unlimited profits, and this profit-seeking behaviour would destroy any pattern of predictability that might emerge in the series for some time. This results in market time series being nonstationary, further complicating their predictability. This opens the question of whether the future market price is predictable or not. A considerable number of studies in the stock market [6, 8, 16, 22, 37] and in the forex (foreign exchange) market [44] have shown that some kind of predictability arises and can be exploited to obtain profits.

Technical analysis is a method that attempts to exploit these patterns. This method is used by professional traders to extract useful information from the asset price variance, using information about the historical price movements in the form of charts and formulas to forecast future prices and trends. These investors argue that this method allows them to profit from changes in market psychology, and this view is well expressed in Pring's [47]. This method was initially designed for stock markets but quickly expanded and is now used in various markets such as forex and cryptocurrencies.

This work proposes a system that uses the knowledge of technical analysis indicators and the learning ability of evolutionary algorithms to generate profitable trading rules. In particular, Genetic Programming (GP) is used [33], in one of its newest formulations called Vectorial GP (VGP), introduced by Azzali et al [4]. Vectorial GP extends standard GP by allowing the use of vectors of features as input data, and introduces new operators that work on these vectors. This new approach gives more context to the GP agent to work with, which means that the agent will be aware of past data to make a more informed decision, hopefully resulting in better performance as a forecasting model. Vectorial GP has shown promising results, outperforming standard GP and some standard machine learning techniques predicting physiologic data [2, 3, 4, 18, 19].

Strongly Typed Genetic Programming (STGP) [42] is another GP variant, which reduces the GP problem search space by restricting the data type of variables and the outputs

of functions used to model the solutions. We amalgamate these two frameworks to formulate the Strongly Typed Vectorial GP (STVGP) method, leveraging the data constraints inherent to STGP along with the inputs and functions characteristic of VGP. To implement STVGP we use a framework written in Java called Jenetics [50].

1.1 Motivation and Objectives

The primary motivation behind this research stems from the growing interest and success stories surrounding the application of GP in the field of market forecasting. As detailed in subsequent sections, GP has shown impressive results in financial market forecasting, often surpassing traditional methodologies.

In light of the successes achieved by Genetic Programming (GP) in various domains, notably in market forecasting as elaborated in Section 2.5, this study conducts a comparative analysis of different GP methodologies in the financial markets. The research specifically aims to evaluate four GP approaches: standard GP, Vectorial GP (VGP), Complex Vectorial (CVGP) and Strongly Typed Vectorial Genetic Programming (STVGP), with a primary objective of understanding the performance of these methodologies in developing trading strategies and facilitating decision-making across varied financial market scenarios.

A pivotal aspect of our research is the introduction of the STVGP approach—a novel fusion of VGP, which considers multiple inputs simultaneously, with the structured framework and data constraints of STGP. This integration is designed to enhance the capabilities of our trading agents by enabling them to better analyze market data, make context-rich decisions, and construct GP models that resonate with domain experts.

Our ultimate goal is to shed light on the performance disparities and inherent characteristics of standard GP, VGP, and the innovative STVGP approach. Through this research, we aim to contribute to the field of computational finance, providing insights into the potential and intricacies of various GP algorithms in the context of trading. We hope that our findings will pave the way for further academic and practical exploration in this domain.

1.2 Main Contributions

- **Pioneering STVGP:** Our research introduces the groundbreaking STVGP methodology, a fusion of VGP's ability to process multiple inputs simultaneously with STGP's structured framework and data rigor. This innovative approach enhances the capabilities of our trading agents, enabling more nuanced market data analysis, informed decision-making, and the creation of GP models that are inherently interpretable and user-friendly for professionals.

- **Holistic Comparative Analysis:** We conduct an exhaustive comparison of various GP methodologies within the financial market domain. Our focus is on assessing the strengths and weaknesses of standard GP, VGP, and the innovative STVGP approach, particularly in their capacity to develop profitable trading strategies and guide informed decisions across diverse financial market scenarios.
- **In-depth Performance Scrutiny:** Our research includes a meticulous examination of the performance nuances and unique characteristics of standard GP, VGP, and STVGP. This in-depth analysis helps us discern the relative advantages and limitations of each methodology, providing valuable insights into their potential applicability and effectiveness in the field of computational finance.
- **Blueprint for Future Exploration:** Beyond the present study, our findings serve as a guiding light for future research endeavors. Not only do they illuminate the current landscape of GP methodologies in trading, but they also outline potential directions for upcoming research. One prominent theme that emerges is the importance of interpretability in GP models, highlighting the need for innovative approaches that are both powerful and transparent.

In summary, our research offers a comprehensive comparison of diverse GP techniques, making significant contributions to the fields of time series forecasting and computational finance. Additionally, by introducing the STVGP methodology, we push the boundaries of existing methodologies and emphasize the crucial role of model interpretability in future research endeavors.

1.3 Document structure

The remainder of this document is structured as follows: Chapter 2 delves into the background knowledge required, laying the foundation for the subsequent discussions. Chapter 3 details the data used in this research, its sources, and its preprocessing. In Chapter 4, we describe the methods employed, emphasizing the novel approaches and their implementation. Chapter 5 presents the results derived from the experiments, while Chapter **??** offers a comprehensive discussion of the findings, drawing connections to existing literature and suggesting avenues for future work.

Chapter 2

Background

The cornerstone of financial market analysis lies in forecasting price behaviors, which in turn facilitates informed decisions during the purchase or sale of financial assets. Market analysis predominantly bifurcates into two categories: fundamental analysis and technical analysis. Although these two forms of analysis can be used in conjunction to achieve optimal conclusions in the financial market, it is important to acknowledge their distinct areas of focus. For the purpose of this study, we concentrate on harnessing technical analysis to discern market patterns and subsequently formulate investment rules anchored in these patterns.

2.1 Financial markets

Serving as marketplaces, financial markets facilitate the buying and selling of diverse financial instruments including stocks, currencies, and cryptocurrencies. These markets play a crucial role in facilitating lending and borrowing activities among entities including companies, individuals, banks, governments, and others. Through these transactions, financial markets effectively allocate savings and investments, promoting capital accumulation and the production of goods and services.

Well-developed financial markets are essential for meeting the borrowing and lending needs of participants, contributing to a productive economy. Conversely, poorly developed financial markets can hinder economic productivity and stability. Investors engage in trading financial instruments with the goal of generating profits. There are two primary approaches to achieving this goal.

The first approach involves buying an instrument at a lower price and selling it at a higher price. For instance, if an investor purchases €1000 worth of a stock priced at €100 per share and subsequently sells the shares when the stock price reaches €105 per share, a profit of €50 can be realized ($10 \times 105 - 10 \times 100 = 50$).

The second approach is known as a short sell and is employed when the investor anticipates a decline in price. In this transaction, the investor borrows the instrument,

immediately sells it at the prevailing market price, and then repurchases it when the price has decreased. By doing so, the investor can acquire the same quantity of the instrument at a lower cost, resulting in a profit. For example, let's consider a scenario where 100 shares of a company are borrowed and sold when the price is €10 per share. If the shares are repurchased when the price drops to €9 per share, a profit of €100 is achieved ($10 \times 100 - 9 \times 100 = 100$).

Such trading strategies elucidate the tactical manoeuvres investors employ to capitalize on price volatilities, thereby securing profits in the ever-evolving financial market landscape.

2.2 Fundamental Analysis

Fundamental Analysis, a methodological approach, delves into gauging the intrinsic value of a company, juxtaposing it against its stock price, anchored by a plethora of financial and economic indicators. This approach involves a comprehensive evaluation of both internal and external factors that can impact a company's overall worth. By analyzing various aspects, such as financial statements, industry trends, competitive landscape, and macroeconomic conditions, the fundamental analysis aims to provide insights into a company's true value.

With the intrinsic value of a company established via fundamental analysis, the subsequent step involves discerning if the prevailing stock price mirrors an overvaluation or undervaluation of the company. If the stock price surpasses the intrinsic value, indicating that the business is overvalued, selling the stock may be considered a favourable option. On the other hand, if the stock price falls below the intrinsic value, implying that the company is undervalued, it may be a suitable time to consider buying the stock as an investment opportunity.

2.3 Technical analysis

As delineated by Murphy [43], technical analysis epitomizes "the study of market action, predominantly employing charts, with an overarching goal of prognosticating future price trajectories." It is a trading discipline that involves examining statistical trends derived from trading activity, such as price movement and volume, to assess investments and identify potential trading opportunities. Technical analysts focus on price movement patterns, trading signals, and other analytical charting tools to evaluate the strength or weakness of an instrument, in contrast to fundamental analysts who attempt to determine an instrument's underlying value based on financial or economic data [13].

The versatility of technical analysis allows its application across a gamut of instruments, provided they possess historical trading data. This spans stocks, futures, commodi-

ties, currencies, among other assets. However, it is more commonly used in commodities and foreign exchange (FX) markets, where traders place a greater emphasis on short-term price changes.

2.3.1 Technical Indicators

Technical indicators, grounded in heuristics or pattern-based signals, emerge from the analysis of factors like price, volume, and, in some cases, open interest associated with an asset or contract. Unlike fundamental analysis, which focuses on a company's core values like earnings, revenue, or profit margins, technical indicators concentrate on previous trading data such as price, volume, and open interest. These indicators are designed to analyze short-term price changes and are commonly used by active traders, although long-term investors can also use them to identify profitable entry and exit opportunities. Technical analysis bifurcates its indicators into two primary categories: classic and computational [9]. Classic indicators rely on graphical patterns (usually using the candlestick chart, Figure 3.2), while computational indicators use formulas and numerical calculations. Computational technical indicators can be categorized into four groups [13, 43]:

- **Trend followers:** These indicators measure the direction and strength of trends. They are typically represented as lines on a chart and provide sell or buy signals when the price is below or above the indicator, respectively. The slope of the line can also indicate the strength of the trend. Examples of trend follower indicators include Simple Moving Average (SMA), Exponential Moving Average (EMA), and Hull Moving Average.

- **Oscillators:** This type of indicator oscillates between two values and helps measure the strength and momentum of a trend. Oscillators often signal whether the market is overbought or oversold, indicating when the price is unreasonably high or low, which could suggest a trend reversal. Examples of oscillators include the Stochastic Oscillator, Relative Strength Index (RSI), and Money Flow Index (MFI).

- **Band systems:** Band systems assist in determining whether prices are overbought or oversold on a relative basis. They use upper and lower bands in conjunction with a moving average. When there is no defined trend, these indicators give a sell signal when the price is above the upper band and a buy signal when the price falls below the lower band. Examples of band systems include the Keltner Channel and Bollinger Bands.

- **Divergence Identifier:** Divergence occurs when the price of an asset moves in the opposite direction of a technical indicator, such as an oscillator. Divergence warns that the current price trend may be weakening and, in some cases, may lead to a reversal in the price direction. Examples of divergence identifiers include the

Commodity Channel Index, On Balance Volume (OBV), and Moving Average Convergence Divergence (MACD).

Harnessing these technical indicators equips traders with insights into market trajectories, momentum, overbought or oversold conditions, and potential trend reversals.

Simple moving average

The Simple Moving Average (SMA) stands as a pivotal technical indicator in the realm of financial analysis. It is calculated by summing up the prices of an asset over a specified number of time periods and dividing the sum by the number of periods. The SMA provides insights into the average price of an asset over a given time frame, smoothing out short-term fluctuations and highlighting underlying trends.

The formula for calculating SMA is:

$$SMA = \frac{\sum_{n=1}^{N} A_n}{N}$$

where:

- A_n represents the price of the asset at period n.
- N is the total number of periods.

SMA is commonly used to identify current price trends and potential changes in established trends. Short-term SMAs, such as 10-day moving averages, react quickly to price changes, providing timely signals. On the other hand, long-term SMAs, such as 50-day moving averages, are slower to react, offering a broader perspective on price movements.

Two popular trading patterns associated with SMAs are the death cross and the golden cross (Figure 2.1). A death cross occurs when the 50-day SMA crosses below the 200-day



Figure 2.1: SMA 50 (green) and SMA 200 (orange) in AAPL instrument

SMA, indicating a bearish signal and potential further losses. Conversely, a golden cross happens when a short-term SMA crosses above a long-term SMA, suggesting a bullish signal and potential additional gains. Volume analysis is often considered alongside these crosses to validate the strength of the signal.

In essence, the SMA is an indispensable tool for traders and analysts, illuminating price trends and guiding informed investment decisions.

Exponential moving average

The Exponential Moving Average (EMA) is a revered technical indicator, frequently employed in financial analysis. It is a type of weighted moving average (WMA) that assigns more weight to recent price data. The EMA, like the simple moving average (SMA), helps identify price trends over time, but it aims to improve upon the SMA by placing greater emphasis on recent prices, which are considered more relevant than older data. Due to its weighting scheme, the EMA reacts faster to price changes compared to the SMA. Similar to the SMA, the EMA can be used in conjunction with trading patterns such as the death cross and the golden cross.

The EMA is calculated using the following formula:

$$EMA_t = \text{price}_t \times W + EMA_{t-1} \times (1 - W)$$

where:

- price_t represents the price of an asset at period t.
- EMA_{t-1} is the EMA value from the previous period.
- W is the weighting factor calculated as $2/(n + 1)$, where n is the number of total periods.



Figure 2.2: EMA 50 (green) and EMA 200 (orange) in AAPL instrument

- EMA_0 is typically set as the initial value of the simple moving average (SMA) with a window size of n.

Offering a nuanced portrayal of an asset's price oscillations, the EMA is adept at capturing the essence of short-term trends. Traders often rely on the EMA to generate buy and sell signals based on crossovers with other moving averages or price patterns.

Relative strength index

Conceived by J. Welles Wilder Jr. and delineated in his seminal 1978 publication, *New Concepts in Technical Trading Systems* [1], the Relative Strength Index (RSI) is a popular momentum oscillator used to assess overbought or oversold conditions and identify potential trend reversals or corrective price pullbacks.

The RSI quantifies the velocity and magnitude of recent price oscillations in an instrument, juxtaposing the instrument's vigor during bullish days against its resilience during bearish ones. By relating this comparison to price action, traders gain insight into how an instrument may perform. When combined with other technical indicators, the RSI can assist traders in making more informed trading decisions.

The RSI is represented as a line graph oscillator with a scale ranging from 0 to 100 (figure 2.3). A reading of 70 or higher historically indicates an overbought condition, suggesting that the instrument may be due for a price decline. Conversely, a reading of 30 or lower indicates an oversold market, indicating a potential price increase.

The RSI is calculated using the following formula:

$$RSI = 100 - \left(\frac{100}{1 + \left(\frac{\text{Average Gain}}{\text{Average Loss}} \right)} \right)$$



Figure 2.3: RSI in AAPL instrument

In this calculation, the average gain or loss represents the average percentage gain or loss over a given time period. Periods with price losses are counted as zero in the average gain calculation, and periods with price increases are counted as zero in the average loss calculation. The standard period length for calculating the RSI value is 14.

Illustratively, considering a scenario where the market exhibited an upward trajectory on seven out of the past fourteen days, with an average gain of 1%. The remaining seven days closed lower, with an average loss of 0.8%. This would result in an RSI value of 55.55:

$$RSI = 100 - \left(\frac{100}{1 + \left(\frac{1\%}{14}\right)\left(\frac{0.8\%}{14}\right)} \right) = 55.55$$

2.4 Genetic Programming

2.4.1 What is Genetic Programming?

Genetic Programming is a computational algorithm based on evolutionary principles. Proposed by Koza [33], GP shares structural similarities with genetic algorithms (GA) [27]. The process commences with a predetermined number of initial, often randomly selected, individuals constituting a population. Through a series of operations resembling natural genetic processes — evaluation, crossover, mutation, and selection — the population continually evolves. Contrastingly, in GP, individuals are symbolized as program trees, enabling the encoding and optimization of trading rules. As a disadvantage, this structure requires additional control on crossover and mutation operators to deal with invalid solutions. GP thrives in scenarios where the precise configuration of the solution remains undetermined a priori or where an approximate solution is acceptable, making the algorithm suitable for solving a wide range of problems applied in various fields of knowledge such as time series forecast, data modelling, symbolic regression, feature selection, pattern recognition, classification, among others [14, 5, 11, 21, 1, 27, 32].

2.4.2 How does Genetic Programming work?

Genetic Programming begins by generating a fixed number of individuals, typically through a random process. These individuals are represented as programs encoded as trees, with nodes and links. The internal nodes in the tree are referred to as functions, while the leaves are called terminals. Before creating the initial population, it is necessary to define the function set and terminal set that best represent potential solutions to the problem at hand. In Figure 2.4, an example of an individual built with the function set +, × and the terminal set C0, C1 is depicted.

A fitness function delineates the metric to gauge an individual's efficacy in problem-solving. The fitness function returns a value that represents how effectively the individual

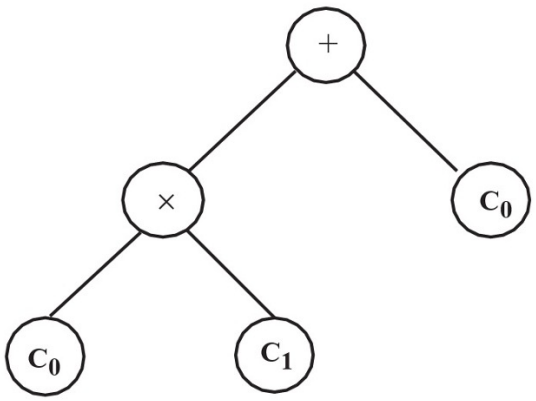


Figure 2.4: Example of an individual built with the function set $\{+, \times\}$ and the terminal set $\{C_0, C_1\}$.

solves the task or the severity of its incorrectness. Upon crafting the initial population, the procedure iteratively invokes operations reminiscent of natural genetic processes (see Figure 2.5). This loop continues until a termination condition is met, such as the best individual's fitness falling below or exceeding a certain threshold, depending on the type of fitness, or when the number of generations reaches a specified limit.

Initiating with 'selection', this operation earmarks individuals slated to parent the succeeding generation. The primary aim of this operation is to bias the selection process towards the best programs by considering each individual's fitness. This is achieved by increasing the probability of selecting an individual in proportion to its fitness value. There are various selection methods available [49, 48], but in this work we will use tournament selection.

The tournament selection modus operandi entails the random drawing of individuals in fixed-length cohorts (with a length greater than or equal to 2) without considering their fitness and then choosing the fittest individual from the group. It's important to note that the worst individual never survives, and the best individual wins every tournament in which it participates. The following operations are employed to create new individuals. Reproduction is a straightforward operation that involves copying the selected individual's program into the new offspring population. The crossover operation generates new individuals by recombining randomly chosen parts of two selected programs. In our case, we employ the standard subtree crossover, also known as single-point crossover, which takes two parents and exchanges their subtrees. Figure 2.6 illustrates this process. The last operation is a mutation, which creates a new offspring for the new population by randomly mutating a randomly chosen node from a selected individual.

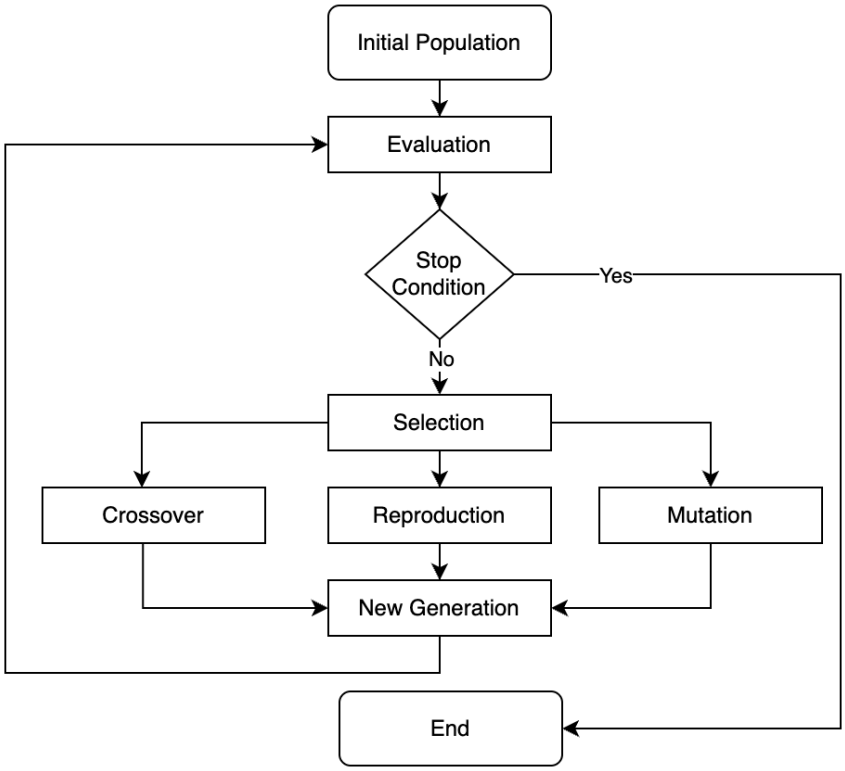


Figure 2.5: Genetic Programming algorithm.

2.5 Related work

The realm of market forecasting has witnessed extensive exploration of GP, with several notable studies showcasing its effectiveness. In one of the pioneering endeavors, Neely C. [44] harnessed GP, aiming to carve out profitability within the forex markets. By analyzing data from six currency pairs between 1981 and 1995, significant improvements were observed in all pairs except the Deutsche mark/yen, when compared to traditional technical analysis rules. Interestingly, the study found that the generated rules did not perform as well on other pairs, even if they involved the same currency.

Iba and Sasaki [29] employed GP to determine optimal Japanese stock companies for investment, as well as the timing and quantities for buying or selling shares. Their comparison of GP with an artificial neural network (ANN) approach revealed that GP outperformed ANN, which tended to suffer from overfitting and yielded inferior results.

Kaboudan [30] introduced a metric to quantify the predictability of time series using GP, highlighting the predictability of stock markets. Based on this finding, the author proposed a model that forecasted the high and low prices of the next trading day for six US stocks (Citigroup, Compaq Computers, General Electric, Pepsi, Sears, and Microsoft) and used these predictions to guide trading decisions. The results showed that trading decisions based on GP forecasts achieved higher returns on investment (ROI) compared to those based on ANN forecasts, with GP outperforming ANN in five out of the six

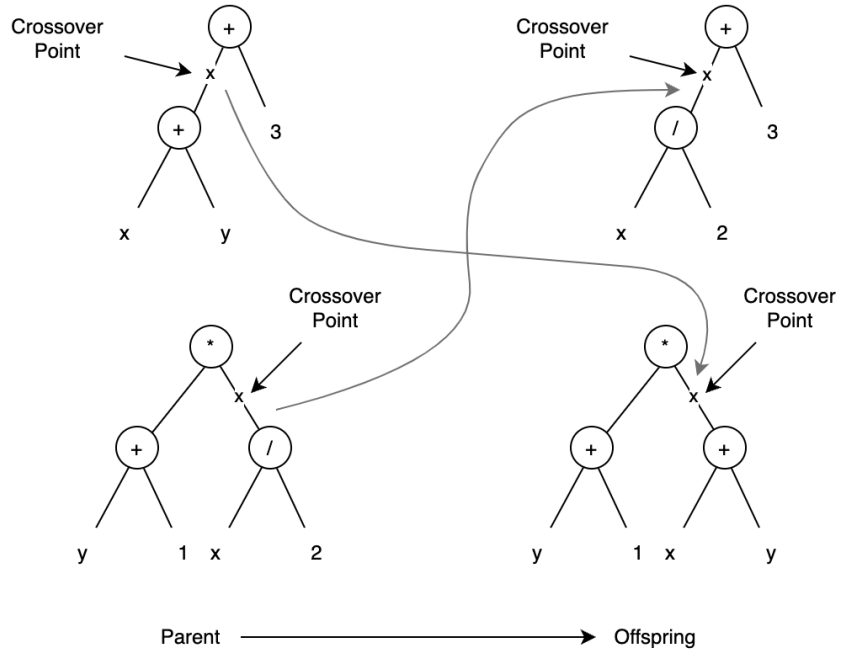


Figure 2.6: Example of a standard subtree crossover operation

stocks.

Potvin et al [46] used GP to generate trading rules in the Canadian stock market, considering data from various industries. Their analysis revealed that GP-generated trading rules proved advantageous during falling or stable market conditions, while the buy and hold approach outperformed GP during rising markets.

Michell and Kristjanpoller [41] adopted Strongly Typed Genetic Programming (STGP) to generate trading rules. They proposed a fitness function that directly evaluated the quality of the generated signals based on their predictive capabilities. The authors used a risk-free metric (rf) to determine buy, sell, and hold signals, considering the stock's return relative to rf. The rules generated by their approach were straightforward and easily interpretable by investors, resulting in higher profits compared to classical optimization approaches and the buy and hold strategy.

Grosan et al [25] and Grosan and Abraham [24] conducted comparative studies between a genetic multi-expression programming algorithm, ANN, and support vector machine models using multiple stock indices for forecasting purposes. The results of both studies strongly supported the use of the genetic model for accurate predictions.

Lee and Tong [36] combined GP with an ARMA statistical model and ANN to forecast the US GDP, yielding promising results.

Karatahansopoulos et al [31] applied GP and gene expression programming (GEP), an evolutionary algorithm, to achieve profits while trading the Greek ASE20 index. Their findings demonstrated that GP and GEP significantly outperformed other machine learn-

ing and trading analysis techniques.

Venturing into automated investment systems, Pimenta et al [45] amalgamated the prowess of STGP with multi-optimization facets. Each individual in the system consisted of two program trees—one for the buy signal and another for the sell signal—which were combined to generate the final trading decision. The authors optimized the algorithm to generate rules that aimed to maximize profits while avoiding overly complex trees. By using the depth of the trees as a complexity metric and removing outliers from the data, their approach achieved returns well above the stock price variation over the same period, surpassing the performance of the other two automated investment strategies tested. Notably, this system generated significant profits even during periods of severe asset depreciation.

Ding et al [12] aimed to generate forecasting models using GP to determine the predictability of stock markets. They explored model accuracy through better specification without introducing new variables. The authors argued that different countries' stock markets should have distinct forecasting models and evaluated stock data from China, India, the US, and Japan. Their findings indicated that GP generated approximately three times more profit in developed countries' stocks compared to developing countries.

Ding et al [12], in their quest to decipher market predictability, employed GP to construct forecasting archetypes. They sought to improve model accuracy through better model specification without introducing additional variables. Furthermore, they argued that stock markets in different countries should have distinct forecasting models. Analyzing stock data from China, India, the US, and Japan, with the US and Japan considered as developed countries and China and India as developing countries, the authors generated models for each group. Interestingly, they discovered that GP generated approximately three times more profit in developed countries' stocks than in developing countries.

These diverse studies collectively highlight the potential of GP and its variants in effectively forecasting markets and generating profitable trading strategies across various financial domains and countries.

2.6 Summary

This chapter explored the application of GP within the financial markets framework. Initially, it underscored the salience of financial data and elucidated how fundamental and technical analyses serve as pivotal techniques for insightful investment decisions. Subsequently, GP emerged in the discourse as a subfield of machine learning, characterized by its reliance on evolutionary algorithms for the induction and optimization of programs or models.

The discourse on GP traversed its applicability in financial markets, emphasizing its prowess in sculpting remunerative trading rules or strategies. The chapter revisited myriad works wherein GP showcased its efficacy, spanning realms like market trend prognosti-

cations, judicious stock selections, and the design of trading paradigms. Additionally, the narrative accentuated the merits of opting for GP vis-à-vis other methodologies, such as artificial neural networks (ANN) and traditional optimization techniques.

In summation, the chapter unfurled a panoramic view of financial data's essence, the quintessential techniques in financial analyses, and GP's potential as a formidable instrument for refining trading models and strategies in financial echelons. Through its contours, the chapter manifested GP's potential in gleaning invaluable insights and bolstering the investment decision-making apparatus.

Chapter 3

Data processing

Financial markets are characterized by detailed and extensive datasets. When examined carefully, these datasets can offer valuable insights, and GP provides a method to explore this data. By using evolutionary algorithms, GP adjusts models to reflect market changes, aiding in more informed investment choices.

The effectiveness of GP is reliant on access to relevant and comprehensive data. This data supports the training and adaptation of the models. A key step in this process is data normalization, which ensures data consistency and reduces potential biases. This chapter discusses the process of data normalization.

3.1 Data representation

Structuring stock market data for GP agents involves breaking the data into evenly sized vectors. Each vector represents a specific data element, such as closing prices or trading volume. Arranging the data in this vector format ensures a consistent chronological order: the most recent data is placed at the end of the current window, followed by earlier data.

Candlestick charts (fig 3.2) serve as a visual method to represent this data arrangement. These charts depict price variations over set periods, like days, weeks, or months. A single candlestick (fig 3.1) represents the price change for a specific period, characterized by a rectangular body and extended lines, known as shadows or wicks. The rectangle indicates the opening and closing prices, while the shadows mark the range of prices for that period.

In these charts, a rise in price from the opening to closing is often shown in green or white, indicating a positive or bullish trend. Conversely, a decline is usually marked in red or black, hinting at a negative or bearish trend. Understanding these chart patterns is essential for effective market data analysis.

Dividing data into consistent segments and representing it as vectors gives GP agents a structured method for analyzing data. By combining recent and historical data, this representation offers an overview, aiding in thoughtful decision-making.

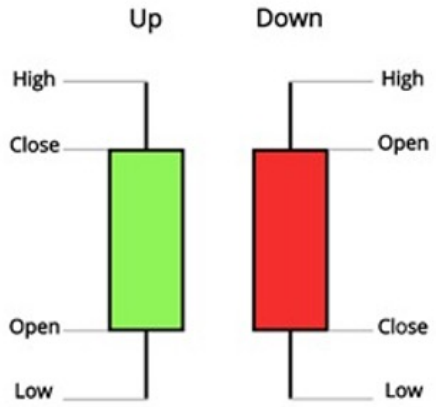


Figure 3.1: Illustration of two candlesticks elucidating price trajectories, trends, and cardinal values (high, low, close, open).



Figure 3.2: Candlestick chart delineating the temporal evolution of prices.

3.2 Data Normalization

Data normalization plays an important role in maintaining the quality and accuracy of data analysis. By adjusting data to a consistent scale, this process addresses differences in magnitude. This ensures fair comparisons across various market aspects, like trends, volumes, and volatility. Essentially, data normalization creates a standardized environment, facilitating precise modelling and analysis.

3.2.1 Row Normalization

Many datasets often have differences in scale, making direct comparisons and analyses problematic. Row normalization addresses this by using the min-max normalization technique. This method aims to standardize values within each data vector, adjusting them to a consistent scale that spans from 0 to 1.

Algorithm 1: Min-Max Normalization

Input: Vector of data v
Output: Normalized vector v_{norm}
Identify the minimum value min $= \min(v)$ and maximum value max $= \max(v)$ within the vector;
for *each value in vector* v **do**
    Compute the normalized value norm_value $= \frac{\text{value}-\text{min}}{\text{max}-\text{min}}$;
    Append norm_value to the normalized vector v_{norm};
end

Row normalization offers several advantages. When applied to each vector individually, it maintains a consistent data scale, reducing biases from varying feature sizes. This becomes crucial, especially when merging multiple data vectors for agent inputs, ensuring a consistent data representation. Additionally, by shifting values to a consistent range, it emphasizes the significance of variations within each vector. This aids agents in identifying and reacting to essential patterns and trends that might influence investment decisions.

In essence, row normalization promotes data consistency, aids in better comparisons, improves pattern detection, and enhances data clarity. This paves the way for agents to make investment choices grounded in a uniform and standardized data context.

3.2.2 Window normalization

In financial domains, large datasets with time dependencies and diverse magnitudes are common. Effectively managing this data is crucial for models to recognize inherent patterns correctly. Window normalization stands out as an effective method to address these issues, enhancing the analysis and representation of financial data. Combining the concepts of min-max normalization with a sliding window approach, this method breaks down the data into uniform, predetermined segments or windows. Each window undergoes normalization, ensuring that modelling agents use only past data, preventing unintended glimpses into future data.

Algorithm 2: Window normalization

```
Inputs: data: a list of n real-valued data points;
period: an integer representing the size of the sliding window;
Outputs: normalized: a list of n real-valued data points, normalized using the
         window normalization technique;
Class: WindowNormalization period
    periodMax ← −∞;
    periodMin ← +∞;
    Function setMaxAndMin(data, start, end):
        for each data point in the subset between start and end do
            if data point > periodMax then
                periodMax ← data point;
            end
            if data point < periodMin then
                periodMin ← data point;
            end
        end
    end
    Function normalize(value, max, min):
        return (value - min) / (max - min);
    end
    Function apply(data):
        normalized ← [];
        for each data point in data do
            if current index ≥ period then
                setMaxAndMin(data, current index - period + 1, current index +
                 1);
            end
            normalized.append(normalize(current data point, periodMax,
             periodMin));
        end
        return normalized;
    end
end
```

The aforementioned algorithm outlines the steps for the window normalization procedure. It initiates by setting a specific window size. Within this boundary, data undergoes normalization using the maximum and minimum values present. This ensures that every data point is adjusted in relation to its recent historical backdrop.

Window normalization offers a slew of advantages:

1. **Prevention of Data Leakage:** Restricting the data to a specific window ensures that the model does not have access to future data, thereby preventing data leakage. This ensures that decision-making is based only on information that would be available at the time in practical applications.

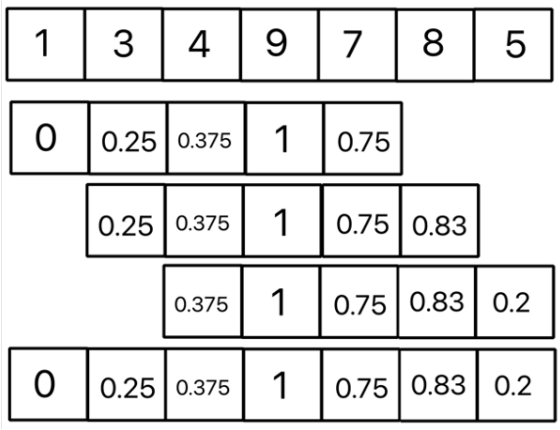


Figure 3.3: Example of the window normalization algorithm's normalization using a window of length five.

2. **Enhanced Pattern Recognition:** The approach amplifies variations within each window, aiding agents in identifying patterns by mitigating magnitude disparities and focusing on relative changes.

3. **Adaptability to Market Dynamics:** Financial markets exhibit continual changes in trends, volumes, and volatilities. Window normalization enables agents to adapt by recalibrating based on localized fluctuations, thereby providing a mechanism for managing changes in market conditions.

Compared to row normalization, window normalization offers a more extended historical context. When the window size is larger than the vector size, the vector captures additional historical data. This gives the agent a broader context for analysis, covering a wider range of market situations.

Consequently, window normalization serves as the default normalization method in this research, applied in instances where row normalization is not utilized.

3.3 Data Selection

The dataset was divided using an 80-20 split for training and testing. Specifically, 80% of the data was used for training the model, and the remaining 20% was used for testing to evaluate the model on new data. The test data was not used during training to ensure an unbiased evaluation of the model. Visual representations of the data split for different datasets are provided in figures 3.4b, 3.4a, 3.4d, and 3.4c, with training and test splits shown in blue and red, respectively."

Given the time series nature of our data, we partitioned it sequentially. To avoid overfitting and ensure agents didn't simply memorize the data, we added a degree of randomness. We used a buffer of 100 data points, and the starting and ending points

within this buffer changed randomly for each training epoch. This maintained consistent data use for all agents but varied the specific data subsets they accessed. For instance, if one dataset started at data point 13, it would end at 1013. Another might begin at point 57 and end at 1057.

In order to mitigate the risk of overfitting, the training data was segmented into three equal parts, an approach inspired by the work of Gonçalves and Silva [23]. Throughout the generational iterations, agents were exposed to only one data segment to circumvent any bias towards a particular subset. However, during substantial 'super generations,' specifically those divisible by 50, agents were provided access to the entirety of the training data, reinforcing their foundational understanding of the problem space.

Regarding data selection, we used four instruments: COTY, KO, TWTR, and PSI20, covering from January 1, 2015, to April 14, 2022. This ensured consistent data availability for all instruments. These instruments were chosen for their varied market behaviors. For instance, COTY shows a declining market trend, KO indicates a rising market with some price drops, while PSI20 and TWTR don't strongly favor any particular trend. This selection aimed to test our methodology across different market situations.

In curating the dataset, our goal was a thorough evaluation of our approach, checking its versatility and performance across diverse market conditions.

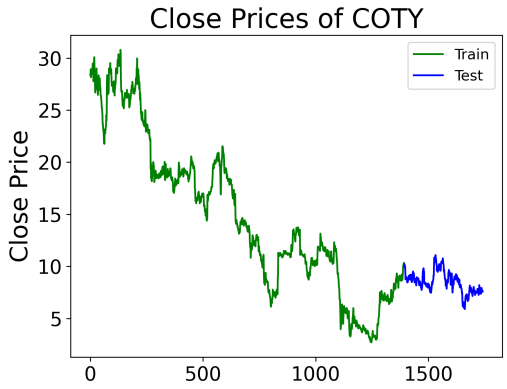
3.4 Data Hydration

Data hydration is a process that involves adding various types of data to boost the analytical prowess of GP agents. In addition to fundamental inputs like instrument prices and volume data, which provide a glimpse into market trends and sentiment, we expanded the dataset with technical indicators and tailored inputs. This was done to give agents a broader perspective for understanding market subtleties and forming effective trading strategies.

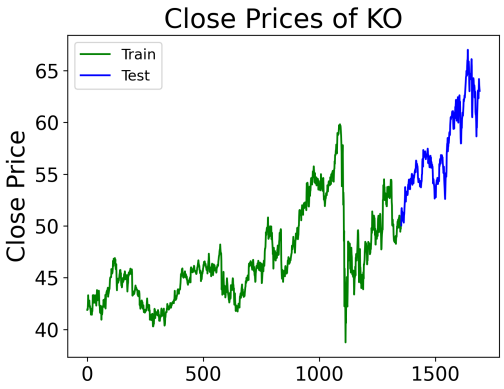
We incorporated Exponential Moving Averages (EMAs) for several periods, such as 5, 13, 50, and 200 days. This range not only offers varying degrees of data smoothing but also helps agents identify trends spanning short to long durations. The selection of these EMA periods was based on insights from technical trading discussions and resources frequently consulted by traders.

To enhance the agents' analytical toolkit, we added the Relative Strength Index (RSI). This metric offers agents an insight into the momentum of instrument price changes. Specifically, the RSI helps identify potential overbought or oversold market scenarios, which are crucial for decision-making in trading. Alongside this, the 'profit percentage' metric acts as a real-time gauge, allowing agents to evaluate the success of their current trades and adapt accordingly.

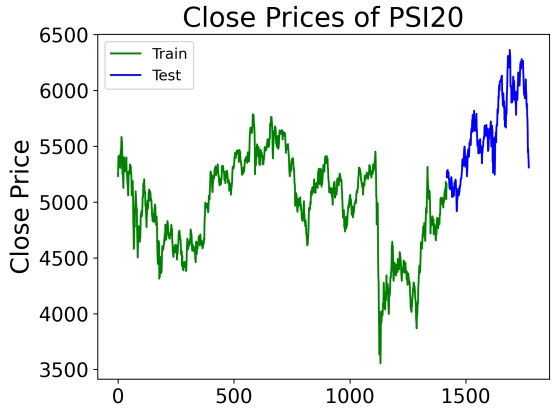
On the custom input front, we introduced 'smallEmaDiff' and 'bigEmaDiff,' both



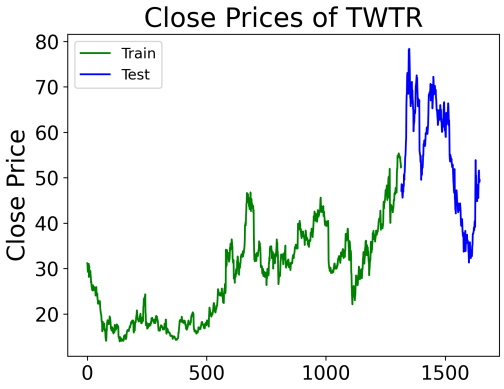
(a) COTY (Beauty Industry) Instrument Prices



(b) KO (Food Industry) Instrument Prices



(c) PSI20 (Portuguese Index) Instrument Prices



(d) TWTR (IT Industry) Instrument Prices

Figure 3.4: Instrument prices spanning January 1, 2015, to April 14, 2022.

derived from EMAs. 'SmallEmaDiff' represents the difference between the 5-day and 13-day EMAs, whereas 'bigEmaDiff' captures the gap between the 50-day and 200-day EMAs. These metrics assist agents in spotting potential trend shifts, which could indicate viable trading opportunities.

In essence, our data hydration approach is not just about adding data but integrating meaningful indicators. By combining EMAs, RSI, profit percentage, and our tailored metrics, we provide agents with a rounded view of the market. This well-rounded understanding, based on established trading indicators, aims to guide agents toward sound trading decisions.

3.5 Summary

For our research, we developed a comprehensive data processing and preparation procedure, segmented into main components: data representation, normalization, selection, and hydration.

- **Data Representation:** We began by integrating essential price data elements, such as close, open, high, low, and volume, into fixed-length vectors. This method offered a structured format to our initial data.
- **Data Normalization:** Understanding the need for consistent data for analysis, we performed data normalization. This step adjusted various data values to a common scale, promoting relevant comparisons.
- **Data Selection:** We strategically chose four financial instruments for analysis: COTY, KO, TWTR, and PSI20. These instruments were chosen based on their representation of different market dynamics, from predictable to volatile trends. We set the data timeframe from January 1, 2015, to April 14, 2022, to ensure consistent data for the selected instruments.
- **Data Hydration:** Beyond primary data, we enhanced our dataset with added metrics. This included exponential moving averages (EMAs) over different periods (5, 13, 50, and 200) for trend insights and the relative strength index (RSI) for gauging price momentum. We also added custom metrics like smallEmaDiff and bigEmaDiff to help agents detect trend changes. A real-time profit percentage metric was also added for immediate trade performance evaluation.

In essence, our detailed data processing approach equipped agents with a broad spectrum of data. This combination of data sources aids agents in making informed trading decisions, recognizing market patterns, and accurately assessing market situations. Our selection of metrics and data elements, commonly recognized in trading circles, aimed to offer a grounded and practical perspective for data-informed decision-making.

Chapter 4

Methods

In the realm of algorithmic trading, many methods harness computational capabilities to improve financial outcomes. This chapter explores techniques used in this study and explains GP as a tool for developing trading strategies with a focus on ROI. The discussion encompasses four GP methods: standard GP, Vectorial G, Complex Vectorial GP and Strongly Typed Vectorial GP. All methods derive from the GP paradigm but vary in solution representations and operators.

4.1 Standard GP

Genetic Programming is a technique within evolutionary algorithms. Unlike some other techniques that represent solutions as fixed-length character strings, GP deals with programs or, more broadly, mathematical expressions as solutions. The standard GP methodology exemplifies this approach, employing evolutionary mechanisms to craft mathematical expressions capable of predicting outcomes for a given problem.

In the field of financial trading, we applied standard GP to evolve mathematical expressions tailored for predicting stock prices. These expressions revolve around mathematical operations that encompass basic arithmetic functions like addition and subtraction, as well as more intricate trigonometric functions such as sine and cosine. These operations collaborate with the input variables outlined in Chapter 3, supplemented by other constants, as outlined in Table 4.1. A distinctive characteristic of standard GP, when compared to other methods discussed in this chapter, is its reliance on scalar values as input, rather than vectors.

The resulting output from the agent takes the form of a scalar value, which is interpreted in three distinct ways:

- A value surpassing 1 is construed as a buying signal.
- A value descending below -1 translates to a selling signal.
- Any other value insinuates a state of inactivity or a ”hold” decision from the agent.

Name	Description	Example
ADD	Adds two input variables	$1 + 2 \rightarrow 3$
MUL	Multiplies two input variables	$2 \times 2 \rightarrow 4$
SUB	Subtracts two input variables	$2 - 1 \rightarrow 1$
DIV	Divides two input variables	$10/5 \rightarrow 2$
SIGNUM	Returns the signum of the input variable; zero if it is zero, 1.0 if it is greater than zero, and -1.0 if it is less than zero	$\text{SIGNUM}(10) \rightarrow 1$
NEG	Negates a single input variable	$10 \rightarrow -10$
SIN	Returns the sine of a single input variable	$\text{SIN}(0) \rightarrow 0$
COS	Returns the cosine of a single input variable	$\text{COS}(0) \rightarrow 1$
TAN	Returns the tangent of the argument	$\text{TAN}(0) \rightarrow 0$
GT	Returns 1.0 if its first input variable is greater than its second input variable and returns -1.0 otherwise	$\text{GT}(10, 5) \rightarrow 1$

Table 4.1: Operations in Standard GP

In essence, the standard GP technique serves as a fundamental cornerstone in our methodological arsenal. By leveraging its capability to evolve mathematical expressions, we empower agents to excel in identifying trading opportunities within the intricate dynamics of financial markets.

4.2 Vectorial GP

Vectorial Genetic Programming (VGP), a derivative of standard GP, evolves programs that function with vectors, accepting one or more vectors as inputs and producing a vector as an output. To manage vectors of different sizes, which might lead to computational compatibility challenges, a simplification of VGP was employed in this study.

Unlike the original VGP, which could manage vectors of varying sizes, all vectors in this model are restricted to a size of either 1, for scalar values, or 21, representing sequences over time. The latter approach ensures compatibility during vector operations and minimizes computational inconsistencies. Although this provides a streamlined methodology for our application, it indicates a noteworthy deviation from the original VGP functionality, which might be explored in future research to facilitate dynamic vector resizing.

The output from the VGP agent is a vector, and the mean of the vector is computed to produce insights. This mean value then translates into actions within the trading model: a value above 1 generates a buy signal, a value below -1 generates a sell signal, and any other value indicates a hold position.

In Table 4.2, an overview of the operations foundational to the VGP method is provided, each accompanied by its respective description and example. This iteration of VGP introduces new operations, such as STD_VAR, and omits others, specifically those asso-

ciated with managing vectors of different sizes, which is a divergence from the original VGP methodology. These adjustments reflect the specific requirements and constraints of the current application.

Name	Description	Example
ADD	Adds corresponding elements of two vectors.	$[1, 2, 3] + [4, 5, 6] = [5, 7, 9]$
DOT	Calculates the dot product of two vectors.	$[1, 2, 3] \cdot [4, 5, 6] = [32]$
SUB	subtract corresponding elements of two vectors.	$[4, 5, 6] - [1, 2, 3] = [3, 3, 3]$
DIV	Divides corresponding elements of two vectors.	$[4, 6, 8]/[2, 3, 4] = [2, 2, 2]$
CUM_MEAN	Calculates the cumulative mean of a vector.	$[1, 2, 3, 4] \rightarrow [1, 1.5, 3, 5]$
PROD	Calculates the product of corresponding elements of vectors.	$[1, 2, 3] \times [4, 5, 6] = [4, 10, 18]$
MEAN	Calculates the mean of a vector.	$[1, 2, 3, 4] \rightarrow [5]$
NEG	Negates a vector.	$[-1, 2, -3] \rightarrow [1, -2, 3]$
SIN	Calculates the sine of each element in a vector.	$[0, \pi/2, \pi] \rightarrow [0, 1, 0]$
COS	Calculates the cosine of each element in a vector.	$[0, \pi/2, \pi] \rightarrow [1, 0, -1]$
TAN	Calculates the tangent of each element in a vector.	$[0, \pi/4] \rightarrow [0, 1]$
GT_THEN	Returns 1 if the mean of the first element is greater than the mean of the second, -1 otherwise, 0 otherwise.	$[5, 3], [1, 3] \rightarrow [1]$
STD_VAR	Calculates the standard deviation of a vector.	$[1, 2, 3, 4] \rightarrow [1.29]$

Table 4.2: Operations in Vectorial GP

4.3 Complex Vectorial GP

Complex Vectorial GP extends the foundational VGP method by enabling the use of complex numbers in computational processes. Complex numbers contain both a real and an imaginary component, providing additional avenues for computational exploration and learning.

The incorporation of complex numbers unlocks several computational advantages. It equips the agent with the ability to perform operations that were previously infeasible

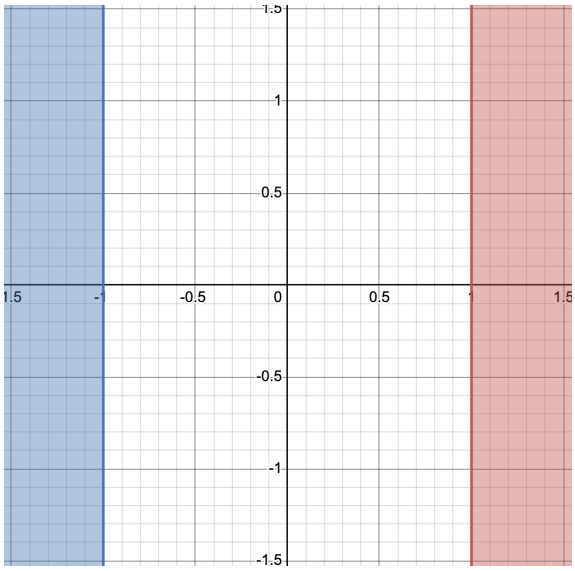


Figure 4.1: Decision planes represented in a Cartesian plane. The blue plain represents the sell signal and the red plane the buy signal

when relying solely on real numbers. For instance, the logarithmic transformation of a negative real number remains undefined in the realm of real numbers. However, within the domain of complex numbers, the logarithm of a negative complex number becomes valid, incorporating an imaginary dimension. This newfound capability empowers the agent to discern patterns within the realm of imaginary operations, potentially offering a competitive advantage over traditional methods confined to real number operations.

However, the integration of the Complex Vectorial GP method comes with its own set of challenges. The inherent complexity of handling complex numbers amplifies the intricacy of the training process. The agent is now tasked with simultaneously optimizing both the real and imaginary components of each complex number. This increased dimensionality of the search space intensifies the computational challenge of identifying successful solutions.

Similar to the VGP method, Complex Vectorial GP accepts vectorial inputs. Each vector can be either unidimensional or possess a dimensionality of 21, accommodating singular values or time-sequenced data sets, respectively. The agent's output materializes as a complex vector. To derive actionable insights, the mean of the complex vector is computed, resulting in a single complex number. The real component of this resultant complex mean is then subject to the following interpretations: values exceeding 1 indicate a buy signal, values falling below -1 signify a sell signal, and all other values suggest a stance of inaction.

In a Cartesian coordinate system, with the x-axis representing the real component and the y-axis representing the imaginary component, three action regions are defined: buy, sell, or hold (Figure 4.1).

Table 4.3 details operations fundamental to the Complex Vectorial GP method, each with a description and example. Differences between these operations and those in VGP, outlined in Table 4.2, are evident. Complex Vectorial GP introduces operations like LOG, SQRT, and TAN that apply to all complex numbers, as well as operations that compare the real and imaginary parts of complex numbers. In contrast, VGP limits its operations to real number vectors, excluding those unique to complex number manipulation.

Operation	**Description**	**Example**
ADD	Element-wise addition of two complex vectors	ADD([1+2i, 3-4i], [5-1i, 2+3i]) = [6+1i, 5-i]
DOT	Dot product of two complex vectors	DOT([1+2i, 3-4i], [5-1i, 2+3i]) = (-7-7i)
SUB	Element-wise subtraction of two complex vectors	SUB([1+2i, 3-4i], [5-1i, 2+3i]) = [-4+3i, 1-7i]
DIV	Element-wise division of two complex vectors	DIV([1+2i, 3-4i], [5-1i, 2+3i]) = [(0.115+0.423i),(-0.462-1.308i)]
LOG	Natural logarithm of a complex vector	LOG([1+2i, 3-4i]) = [(0.805+1.107i),(1.609-0.927i)]
SQRT	Square root of a complex vector	SQRT([1+2i, 3-4i]) = [(1.272+0.786i), (2+1i)]
SIN	Sine of a complex vector	SIN([1+2i, 3-4i]) = [(3.166+1.960i),(3.854+27.017i)]
COS	Cosine of a complex vector	COS([1+2i, 3-4i]) = [(-2.033-3.052i),(-27.035+3.851i)]
TAN	Tangent of a complex vector	TAN([1+2i, 3-4i]) = [(0.034+1.015i),(-0.000122-1i)]
CUM_MEAN	Cumulative mean of a complex vector	CUM_MEAN([1+2i, 3-4i, 5-6i]) = [(1+2i),(2-1i),(3-2.667i)]
MEAN	Mean of a complex vector	MEAN([1+2i, 3-4i, 5-6i]) = (3-2.667i)
NEG	Negation of a complex vector	NEG([1+2i, 3-4i]) = [-1-2i,-3+4i]
GT_THEN_REAL	Returns 1 if the real part mean of the first argument is greater than the real part mean of the second argument, otherwise returns -1	GT_THEN_REAL([1+2i, 3-4i], [5-1i, 2+3i]) = [-1, 0i]
GT_THEN_COMPLEX	Returns i if the imaginary part mean of the first argument is greater than the imaginary part mean of the second argument, otherwise returns -i	GT_THEN_COMPLEX([1+2i, 3-4i], [5-1i, 2+3i]) = [0,-i]

Table 4.3: Complex Vectorial Operations

4.4 Strongly Typed Vectorial GP

Understanding Strongly Typed Vectorial GP (STVGP) necessitates a brief overview of Strongly Typed Genetic Programming (STGP). STGP introduces a type system to standard GP, dictating the input and output types for functions and terminals and ensuring syntactic correctness in the resultant solutions. This type system influences genetic operations to yield syntactically valid offspring and can influence the trajectory of the evolutionary process toward feasible solutions. Tests with standard STGP were not conducted in this study due to time restrictions.

STVGP extends GP by integrating boolean and vector value types, aiming to improve model interpretability and validation in domains such as financial trading. The method intends to formulate agents that provide both interpretability and accuracy in scenarios where understanding the reasoning behind predictions is necessary.

In the context of finance, understanding the logic of investment strategies is crucial for informed decision-making. STVGP enables domain experts to incorporate their knowledge, potentially improving model performance.

STVGP utilizes three main function categories:

- **Vectorial Functions:** Functions accept and produce vectors, enabling data structure manipulation.
- **Relational Functions:** Functions utilize vectors and/or booleans, outputting boolean values, and aid in modeling relationships and conditions.
- **Boolean Functions:** Functions operate with boolean inputs and outputs, providing logical operations and decision-making capabilities.

In STVGP, an agent outputs a boolean value, and the initial node of the agent's decision tree is a relational or a boolean function. This structure translates 'true' to a "buy" signal and 'false' to "sell."

STVGP can create models that include detailed decision-making logic and rules. Its transparent, "white-box" design makes it a viable choice for specialists from varied domains, presenting GP as a tool to navigate and solve complex problems. Such transparency also broadens the accessibility of GP techniques, inviting further exploration by experts.

Table 4.4 presents the operations used in the STVGP method, along with descriptions and examples.

4.5 Fitness Function

In evolutionary algorithms, the fitness function is pivotal, acting as the yardstick to evaluate an agent's problem-solving prowess. It measures an agent's accuracy or errors in

solving a specified problem. For our research, we selected the return on investment (ROI) to gauge the fitness function. This choice offers a direct indication of the returns an agent derives from its decisions.

However, applying this metric posed certain challenges. A notable concern was the inclination of some agents to abstain from trading. While these agents might seem to showcase high fitness due to the absence of losses, their passive stance stymied the learning trajectory by prematurely dominating the gene pool. This inactivity contradicts the primary objective of a trading algorithm, which is to actively participate in trading. To counter this, we imposed a penalty: agents that remained inactive were attributed a fitness score symbolizing the lowest possible value. This ensured such agents wouldn't progress to the next generations.

But penalizing inactivity alone wasn't the complete solution. It was vital to incentivize agents that actively traded and exhibited discernment in their decisions. Thus, we fine-tuned our fitness assessment by incorporating the product of the win rate and ROI. This combined score accentuates the rewards for agents boasting a high win rate. However, it's worth noting that this adjustment is only considered when the ROI is positive. Implementing it on negative ROI scores would unintentionally diminish loss magnitudes, deviating from our intent of nurturing agents that trade actively and wisely.

4.6 Parameter Settings

Parameter configuration in GP can influence the algorithm's evolutionary progression, affecting duration, population size, and the probability of performing genetic operations. The following list describes the parameters used in our GP methodology, while other parameters used default settings as per [50]:

- **MaxGenerations:** The algorithm was set to run at most 100 generations. The selection of 100 was determined through preliminary tests, which suggested it provided a balance between computational efficiency and solution optimization.
- **MaxPhenotypeAge:** The maximum age of a phenotype was limited to 50 generations, after which it is replaced. This parameter, specific to our use of the Jenetics library, helps manage population diversity and prevent stagnation in evolutionary progress.
- **PopulationSize:** The population at each generation was held constant at 3000 individuals. This size was chosen to maintain a broad search space without demanding excessive computational resources.
- **Number of Runs for Each Method:** Each method was run 10 times. While larger numbers of runs (e.g., 30) are common, resource and time constraints led to this choice, which still allows for a preliminary assessment of consistency.

- **Tournament Selector Size:** The tournament selector size was set to 30, approximately 1% of the population size, aiming to strike a balance between selective pressure and maintaining genetic diversity.

- **MutationProb:** The mutation probability was set to 0.001. This relatively low probability ensures genetic diversity without introducing excessive disruption to evolving solutions.

To manage the risk of overfitting—where models might adapt too closely to training data and perform poorly on new data—depth and size limits were implemented. The models were confined to a maximum depth of 13 and a maximum size of 90. These values were derived from observational trials and aimed to ensure that the agent representations were succinct yet effective.

These constraints are necessary to guide the agents towards a balance between simplicity and capability, mitigating the risk of developing overly complex solutions that perform inadequately on new data.

It is crucial to note that while the parameter choices in this study were informed by understanding and observational data, they remain adjustable. These parameters may require adaptation to optimize outcomes depending on the specific characteristics of different problems and datasets. Nonetheless, within the scope of this work, the parameters were consistently applied across all experiments and datasets to maintain experimental rigour and comparability.

4.7 Summary

The research presented encompasses an exploration of several GP methods, each tailored to address specific challenges and use cases. These methods include:

- **Standard GP:** Rooted in classic GP concepts, this technique stays clear of the intricacies associated with specialized vectors or complex operations. It primarily focuses on market data, translating its results into practical trading directives.

- **Vectorial GP:** Differing from the scalar-based approaches, this method embraces vector operations. It empowers models to conduct element-wise computations on vector inputs, offering a deeper context-specific analysis.

- **Complex Vectorial GP:** An extension of the VGP approach, this method integrates complex numbers into its operational fabric. This addition provides a greater mathematical depth, facilitating sophisticated mathematical expressions beyond the scope of real numbers.

- **Strongly Typed Vectorial GP:** This technique integrates both boolean and vector values, marking a shift from standard GP practices. Its main strength lies in its improved model clarity and validation capabilities. Its design lends itself to effectively managing intricate decision-making logic and rules, making it particularly useful in domains with complex decision frameworks.

Operation	Description	Example
MEAN_GT	Returns true if the mean of the input vector is greater than the given threshold, false otherwise	MEAN([1, 2, 3, 4]) $>$ 2.5 = true
CUM_MEAN_GT	Returns true if the cumulative mean of the input vector is greater than the given threshold, false otherwise	CUM_MEAN([1, 2, 3, 4]) $>$ 2.5 = true
SUM_GT	Returns true if the sum of the input vector is greater than the given threshold, false otherwise	SUM([1, 2, 3, 4]) $>$ 8 = false
AND	Returns true if all input boolean values are true, false otherwise	AND(true, false) = false
OR	Returns true if at least one input boolean value is true, false otherwise	OR(false, true) = true
XOR	Returns true if exactly one input boolean value is true, false otherwise	XOR(true, true) = false
NOT	Returns the inverse of the input boolean value	NOT(true) = false
ADD	Returns the element-wise sum of two input vectors	ADD([1, 2, 3], [4, 5, 6]) = [5, 7, 9]
SUB	Returns the element-wise difference of two input vectors	SUB([4, 5, 6], [1, 2, 3]) = [3, 3, 3]
DOT	Returns the dot product of two input vectors	DOT([1, 2, 3], [4, 5, 6]) = 32
DIV	Returns the element-wise division of two input vectors	DIV([6, 8, 10], [2, 4, 5]) = [3, 2, 2]
COS	Returns the element-wise cosine of the input vector	COS($[0, \pi/2, \pi]$) = $[1, 0, -1]$
SIN	Returns the element-wise sine of the input vector	SIN($[0, \pi/2, \pi]$) = $[0, 1, 0]$
TAN	Returns the element-wise tangent of the input vector	TAN($[0, \pi/4]$) = $[0, 1]$
NEG	Returns the element-wise negative of the input vector	NEG([1, -2, 3]) = [-1, 2, -3]
VECT_IF_ELSE	Returns the element-wise result of a ternary operator	VECT_IF_ELSE([true, [1, 2, 3], [4, 5, 6]) = [1, 2, 3]
STD_VAR	Returns the standard deviation of the input vector	STD_VAR([1, 2, 3, 4, 5]) = 1.58

Table 4.4: Operations Strongly Typed Vectorial GP method

Chapter 5

Results

In this chapter, we explore the results of applying different Genetic Programming (GP) techniques to create trading strategies. The focus is on how various agents evolve and make decisions within the trading environment.

We begin with an examination of the best agents' evolution over time, presenting the changes and adaptations in their strategies and decision-making processes for each proposed GP variant. The sections dedicated to these methods include an analysis of evolution followed by a discussion.

The subsequent part of the chapter investigates the agents' structure, exploring the use of different operations and leaf nodes in decision-making. For each GP method, an agent structure analysis is conducted to dissect the components of their trading strategies.

A comparative analysis of the GP approaches is then presented, assessing their performance in varying market contexts through run, table, and boxplot analyses.

The chapter proceeds with a statistical analysis of the GP methods across different datasets: COTY, KO, PSI20, and TWTR. Each section offers an interpretation of the results.

5.1 Evolution of Best Agents

The analysis examines the evolutionary path of the agent with the best fitness during test on the COTY stock dataset to uncover emerging patterns or strategies and clarify the operational mechanics and strategic adjustments of the agents. COTY was selected due to the consistent positive results across several agents, providing a consistent basis for comparison.

Two aspects were analyzed:

1. The progression of the agent's fitness through successive generations during the training phase, to understand how the agent adapts and improves over the course of the evolutionary process.

2. The fitness of the best agent from each generation when applied to the test dataset, to observe how the evolving strategies generalize to unseen data and to highlight instances of overfitting to the training data.

Two charts are provided, the left chart shows the fitness evolution of the best agent across generations. The right chart presents the fitness of the best agent from each generation as applied to the test dataset, offering a view of the agent's generalization capabilities across its evolutionary timeline.

Note the difference in the magnitude of the fitness between the training and test datasets. The reduced fitness in the test dataset is attributed to the smaller temporal span available for the agent to accrue financial gains. Consequently, the fitness function, which is dependent upon the agent's monetary acquisition capabilities, exhibits a lower magnitude in the test context. This difference in magnitude should be considered when interpreting the results.

5.1.1 GP

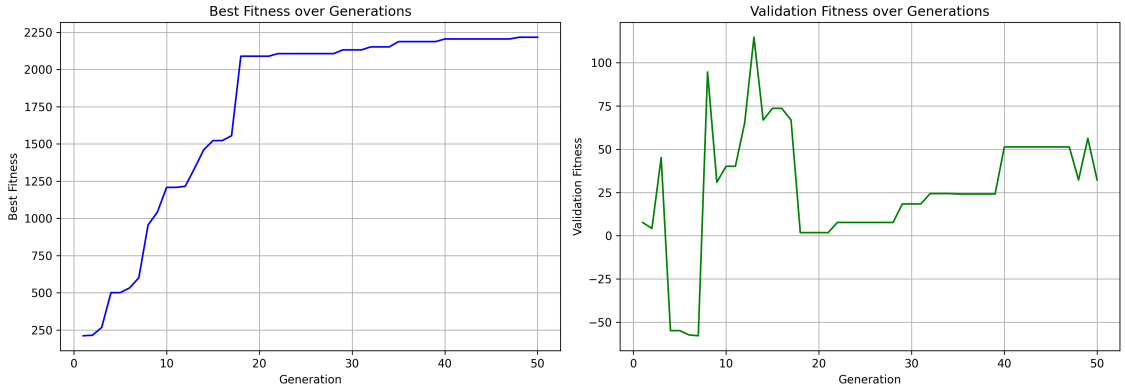


Figure 5.1: Evolution of Best Fitness and Test Fitness for GP on COTY dataset.

Analysis of Evolution Figure 5.1 shows the changes in 'Best Fitness' and 'Test Fitness' over generations. 'Best Fitness' goes up and then stops changing much. 'Test Fitness' moves up and down, drops when 'Best Fitness' goes up, and then shows an upward tendency until the end.

Discussion 'Best Fitness' stops changing much, which might mean the algorithm has found a solution that works well for the training data. 'Test Fitness' suffers wild variations in the beginning, which is indicative of an exploration phase where the best individuals being created are very diverse from each other. Then it stabilizes with an upward tendency, suggesting an exploitation phase where the best individuals are similar to each other.

5.1.2 VGP

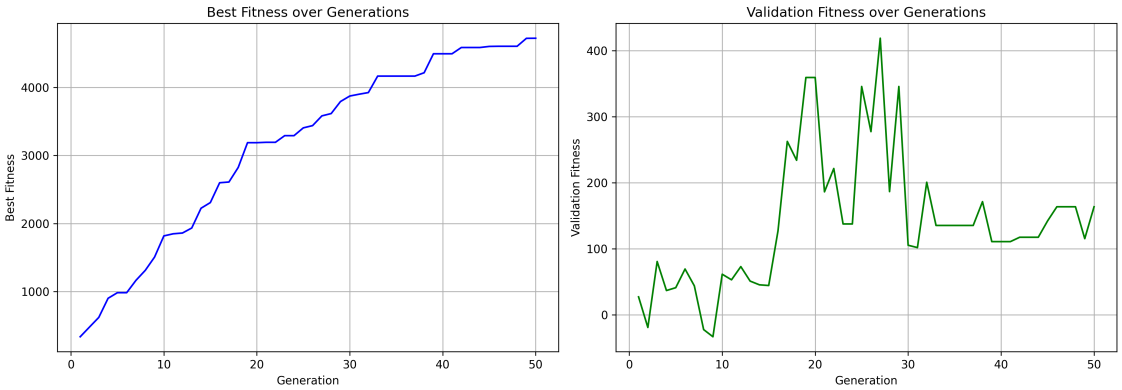


Figure 5.2: Evolution of Best Fitness and Test Fitness for VGP on COTY dataset.

Analysis of Evolution Figure 5.2 displays the progression of 'Best Fitness' and 'Test Fitness'. 'Best Fitness' increases and exhibits a tendency to plateau toward the end of the generation. 'Test Fitness' displays oscillation, reaches the highest peaks mid-run and demonstrates a stabilization toward lower values at the conclusion of the generations.

Discussion The apparent plateau in 'Best Fitness' might suggest potential convergence or a slow solution improvement as the generations progress. 'Test Fitness' seems to go through three different phases. In the first phase, the best individuals do not seem to be useful on unseen data, as the test fitness remains low. In the second phase, the test fitness reaches high values with wild variations, suggesting that evolution is still performing a wide exploration of the search space. Finally, in the third phase, evolution seems to enter the exploitation phase and test fitness remains more or less stable.

5.1.3 VGP Row normalized (VGP Rn)

Analysis of Evolution Figure 5.3 displays how 'Best Fitness' and 'Test Fitness' change over generations. 'Best Fitness' keeps going up, while 'Test Fitness' remains steadily low for almost 20 generations before going up and oscillating.

Discussion 'Best Fitness' keeps going up, meaning the algorithm does not find a stable solution during the generations we see. The 'Test Fitness' suggests that nothing useful is found during the first phase of the evolution, then there is a useful exploration of the search space, and finally some exploitation. However, at the end of the evolution, the test fitness is accompanying the 'Best Fitness' upwards, which suggests there could have been an advantage in letting the evolution run for longer.

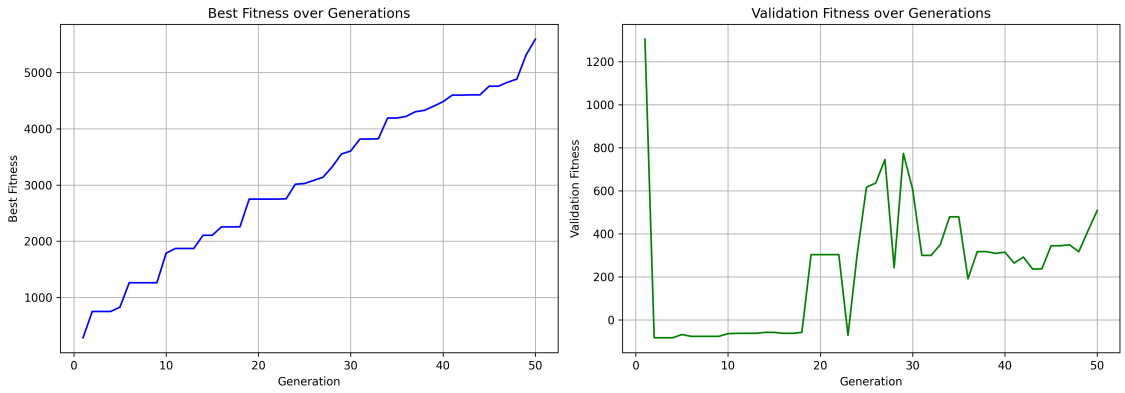


Figure 5.3: Evolution of Best Fitness and Test Fitness for VGP(Rn) on COTY dataset.

5.1.4 Complex VGP (CVGP)

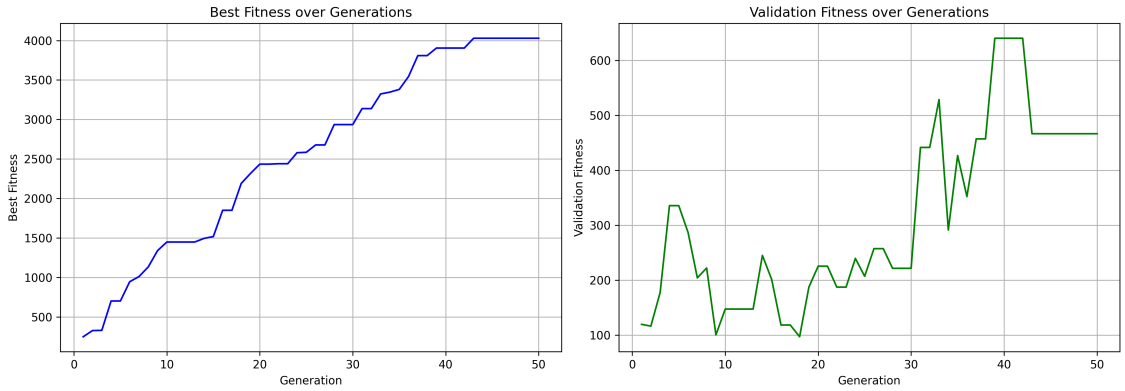


Figure 5.4: Evolution of Best Fitness and Test Fitness for Complex VGP on COTY dataset.

Analysis of Evolution Figure 5.4 shows how 'Best Fitness' and 'Test Fitness' change over generations for Complex VGP on the COTY dataset. 'Best Fitness' increases and then levels off. 'Test Fitness' goes up and down but also levels off towards the later generations.

Discussion The levelling off of 'Best Fitness' suggests the algorithm found a good solution for the training data. The 'Test Fitness' does not show a clear upward tendency until more than half the run, and then finally the exploration of solutions seems to pay off. Although, at the end of the evolution, the test fitness is accompanying 'Best Fitness' in its stabilization phase, it is unclear whether the evolution had already actually entered the exploitation phase.

5.1.5 Complex VGP Row normalized (CVGP Rn)

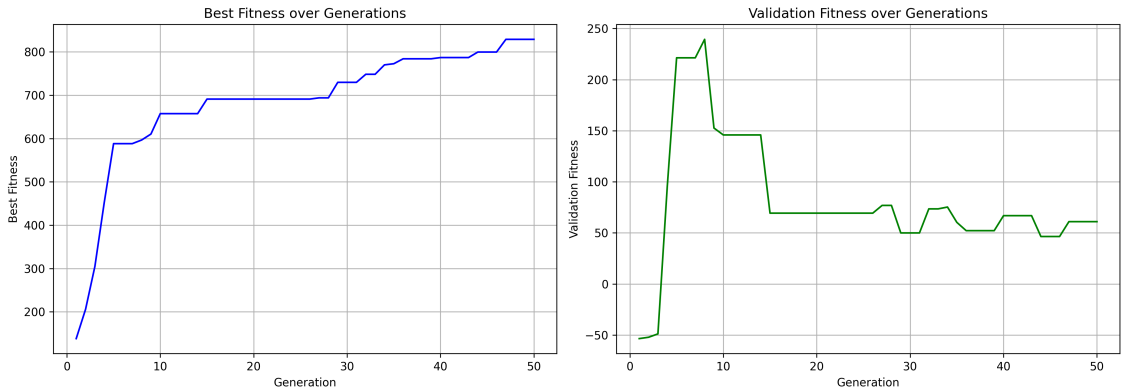


Figure 5.5: Evolution of Best Fitness and Test Fitness for Complex VGP(Rn) on COTY dataset.

Analysis of Evolution Figure 5.5 shows the change in 'Best Fitness' and 'Test Fitness' over generations for Complex VGP(Rn) on the COTY dataset. Best Fitness' goes up quickly and then keeps having small improvements. 'Test Fitness' goes up at first and then comes down.

Discussion The rapid increase in 'Best Fitness' suggests the algorithm finds a stable solution for the training data early in the evolution. The initial rise and following drop in 'Test Fitness' suggests the evolution found an agent capable of generalizing to unseen data early on, but then lost it upon further optimization. Nevertheless, test fitness quickly stabilizes, even as the fitness in training data keeps rising.

5.1.6 STVGP

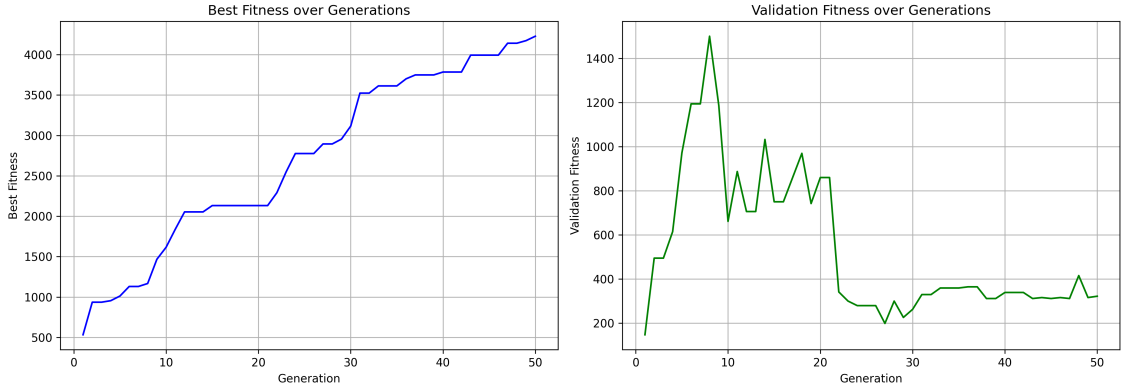


Figure 5.6: Evolution of Best Fitness and Test Fitness for STVGP on COTY dataset.

Analysis of Evolution Figure 5.6 shows the changes in 'Best Fitness' and 'Test Fitness' over generations. 'Best Fitness' shows a steady increase, while 'Test Fitness' shows a peak, then a drop, and a further drop with stabilization.

Discussion The steady increase in 'Best Fitness' suggests that it might be worth prolonging the evolution to find yet better solutions. The peak and first drop in 'Test Fitness' suggests an active exploration phase. The second drop, followed by stabilization, strongly suggests that the evolution has entered an exploitation phase.

5.2 Agent Analysis

This section provides an examination of the optimal agent for each method, using the COTY dataset as a common point of reference. Each agent is analyzed in terms of its utilized operations and variables, followed by a discussion on its tree structure. Essentially, an agent is formulated as a tree of operations, which can be interpreted as a program or mathematical formula, guiding its actions and decisions within the trading environment.

5.2.1 GP

Agent Structure Analysis The agent employs a range of operations and leaf nodes. Specifically, subtraction (-) is utilized 26 times, multiplication (*) six times, division (/) five times, addition (+) once, `sin` is applied ten times,`tan` two times, and `signum` eleven times. Regarding leaf nodes, `ema50` appears eight times, `smallEmaDiff` six times, `ema13` five times, and `rsi` once.

Discussion The agent reveals a notable focus on determining disparities or changes in various market metrics, as evidenced by the frequent use of subtraction. The prominent application of `signum` and `sin` operations could indicate an inclination toward identifying and potentially capitalizing on cyclical or directional market patterns in its decision-making strategy.

The recurrent inclusion of `ema50` and `smallEmaDiff` as leaf nodes suggests a strategy that gives weight to both medium-term price trends and short-term price shifts when making trading decisions. Meanwhile, the utilization of `tan` and `rsi` may signify an attempt to navigate through oscillatory market patterns and assess instrument momentum, respectively.

5.2.2 VGP

Agent Structure Analysis `smallEmaDiff` Within the agent structure `V_TAN`, `V_ADD`, `V_NEG`, and `GT_THEN` emerge as recurrent operations, each with a distinct function.

V_TAN is used 16 times, suggesting a strategy that potentially grapples with cyclical data. V_ADD, is used 14 times, enabling the combination of various metrics or values. V_NEG, utilized 11 times, alters the sign of metrics, and GT_THEN, also used 11 times, imparts conditional logic to the agent's decision-making process.

Leaf Nodes Analysis Close appears 16 times, while Ema200 and ema50 are used 8 times each, highlighting a focus on these specific moving averages in the agent's calculations.

Discussion The substantial use of V_TAN may be a strategy to identify and navigate periodic patterns in the market data. The frequent employment of Close e may suggest a prioritization of recent closing prices in decision-making. The incorporation of ema200 and ema50 indicates that the agent values both short-term and long-term price trends in its calculations, reflecting a strategy that melds both immediate and more protracted market movements in its decision-making framework.

5.2.3 VGP Row normalized (VGP Rn)

Agent Structure Analysis The agent employs several operations and leaf nodes to formulate its decision-making strategy. The operations and their respective occurrences are as follows: V_MIN one time, GT_THEN eleven times, V_DIV seven times, V_DOT seven times, V_SUB three times, and V_TAN eight times. Regarding the leaf nodes: ema13 is used two times, ema50 four times, vol three times, bigEmaDiff two times, and close nine times.

Discussion The agent shows a predilection for GT_THEN, V_TAN, and V_DIV operations, alongside the close leaf node. The frequent use of GT_THEN signals a strategy that substantially employs conditional logic, possibly to adapt its actions to diverse market scenarios. The use of V_TAN might indicate a strategy that attempts to navigate through cyclical market patterns, while the common use of V_DIV suggests the agent often performs division calculations, potentially for normalization or determining the rate of change. The recurrent inclusion of the close leaf node indicates a strategy that deems the closing price a significant indicator for trading decisions, potentially using it as a predictive metric for future prices.

5.2.4 Complex VGP (CVGP)

Agent Structure Analysis The agent structure employs various operations and leaf nodes to navigate its decision-making logic. The operations include GT_THEN_REAL, which is utilized 20 times, V_COS, used seven times, V_SUB, applied nine times,V_TAN,

implemented fourteen times, and V_DOT, which is used three times. In terms of leaf nodes, Close is utilized eight times, vol four times,rsi thirteen times, ema50 once, and ema5 seven times.

Discussion The usage of GT_THEN_REAL signals a strategy that operationalizes condition-based decision-making, involving the triggering of trading actions under specified market circumstances. The incorporation of V_COS and V_TAN suggests an approach that factors in cyclical or periodic market patterns, attempting to leverage consistent market oscillations. The presence of V_SUB indicates a strategy that might identify and act upon divergences or crossovers in trading indicators by evaluating differences between various metrics. In leaf nodes, the recurrence of rsi signifies a strategy that potentially acts upon overbought or oversold market conditions. The inclusion of Close might indicate a method that considers closing prices to focus on relative price movements while mitigating the impact of overall price level changes. The use of ema5 suggests a tendency to respond to recent market trends by incorporating a short-term exponential moving average into decision-making.

5.2.5 Complex VGP Row normalized (CVGP Rn)

Agent Structure Analysis The agent structure consists of both operations and leaf nodes, with each contributing to its decision-making in the trading environment. The operations and their occurrences in the agent's structure are V_ADD with 23 occurrences, V_DIV with one occurrence, V_SIN with four occurrences, V_COS with one occurrence, GT_THEN_COMPLEX with three occurrences, and CUM_MEAN with 10 occurrences. On the leaf nodes side, ema50 appears 10 times, Close nine times, smallEmaDiff three times, and bigEmaDiff two times.

Discussion The usage of V_ADD may indicate a strategy that synthesizes various metrics to derive new indicators, potentially amalgamating different market information into a composite metric to guide trading decisions. The deployment of CUM_MEAN might imply an approach that seeks to stabilize decision-making by minimizing the impact of short-term market volatilities through the aggregation of data over extended periods. The inclusion of the ema50 and close leaf nodes in the agent's structure suggests that medium-term price trends and the most recent closing price are considered during the decision-making process.

5.2.6 STVGP

Agent Structure Analysis The agent integrates various operations in its structure. These comprise CUM_MEAN_GT, which is employed once, DOT, used five times, VECT_IF_ELSE,

applied twice, STD_VAR, used once, TAN, incorporated 13 times, SUM_GT, applied once, SUB, used eight times, MINUS_ONE, employed once, COS, utilized four times, NEG, used once, and GT_THEN_REAL, which is incorporated 13 times. In terms of leaf nodes, the agent uses smallEmaDiff and ema50 each once, Close twice, rsi ten times, ema5 eight times, and vol three times.

Discussion The recurrent application of the TAN and GT_THEN_REAL operations, 13 times each, signals a balance in the agent's strategy between adapting to periodic market patterns and employing conditional logic to navigate through various market scenarios. The leaf node rsi is utilized ten times, possibly indicating a strategy that factors the Relative Strength Index significantly in decision-making. Likewise, the ema5 node is used eight times, suggesting a strategy that might be capitalizing on short-term market trends.

To illustrate, a segment of the agent's structure is:

VECT_IF_ELSE(GT_THEN_REAL(TAN(Close), TAN(rsi)) COS(rsi), SUB(ema5, vol)),

This segment demonstrates a conditional operation: if the real part of the mean of tangent Close is bigger than the real part of the mean of tangent rsi, it checks the cosine of rsi; otherwise, it computes the difference between ema5 and vol, and the output determines the tangent of rsi. This extract underscores the agent's ability to combine conditional and trigonometric operations, allowing for different responses to market patterns and shifts.

5.2.7 Insights into Strategy Formulation

The frequent appearance of the trigonometric operations suggests a focus on leveraging periodic market patterns, proposing that models may exploit consistent oscillations in prices.

The prevalent use of ema50 indicates a strategy that incorporates past price trends into predictive frameworks, suggesting models may assume historical price trends provide a reliable guide for future market movements.

The incorporation of cumulative functions, such as CUM_MEAN, implies an approach that considers long-term market movements. This raises the hypothesis that models might aim to stabilize decision-making by minimizing the impact of short-term market volatilities through the aggregation of data over extended periods.

The recurrent use of the addition operation (V_ADD) points towards a strategy that synthesizes various metrics to derive new indicators for decision-making. This hints that models might attempt to amalgamate diverse market information into a singular, composite metric to guide trading decisions.

5.3 Comparison

In this section, we dive into a detailed examination of the performance of the various GP methods applied to each stock.

Three primary tools assist in our analysis:

- Charts of all 10 runs, which provide a visual representation of how each run evolves over time.
- Tables that display median values, offering a snapshot of central tendencies in the results.
- Box plots, which give insights into the data distribution, highlighting the central values, spread, and potential outliers.

By utilizing these tools, we aim to understand not just the overall performance of each method, but also the nuances in how they react under different conditions or stages of the analysis.

5.3.1 Run analysis

COTY Analyzing the COTY dataset (Figure 5.7) offers an understanding of the performance of the various methods: GP, shown in the first panel, tends to plateau in training around the 30th generation, while its test remains near zero.

In the second panel, VGP consistently improves in training but mirrors GP's test pattern, suggesting limited data might influence this behaviour.

VGP(Rn), in the third panel, steadily progresses in training with notable dispersion between runs. Test shows an upward median trend despite the noise.

Complex VGP, in the fourth panel, has a clear upward training trajectory with minimal variation between runs. Its test results trend upward.

The fifth panel, Complex VGP(Rn), displays consistent training improvement with run-to-run dispersion, yet an almost flat line in test.

STVGP, in the sixth panel, indicates an upward training trend with varied runs. Its test, however, trends downward amidst noise suggesting overfitting on the train data.

KO In the first panel (Figure 5.8), representing GP, the training displays a concise upward trend, albeit with one outlier. The test, however, is characterized by noisy results trending downward.

The second panel, VGP, reveals a concise upward trajectory during training. Its test results, while noisy, don't display a clear trend but remain in the positive spectrum.

VGP(Rn), depicted in the third panel, shows a slightly dispersed upward trend in training with an outlier. In test, despite the noise, there's an observable positive upward trend.

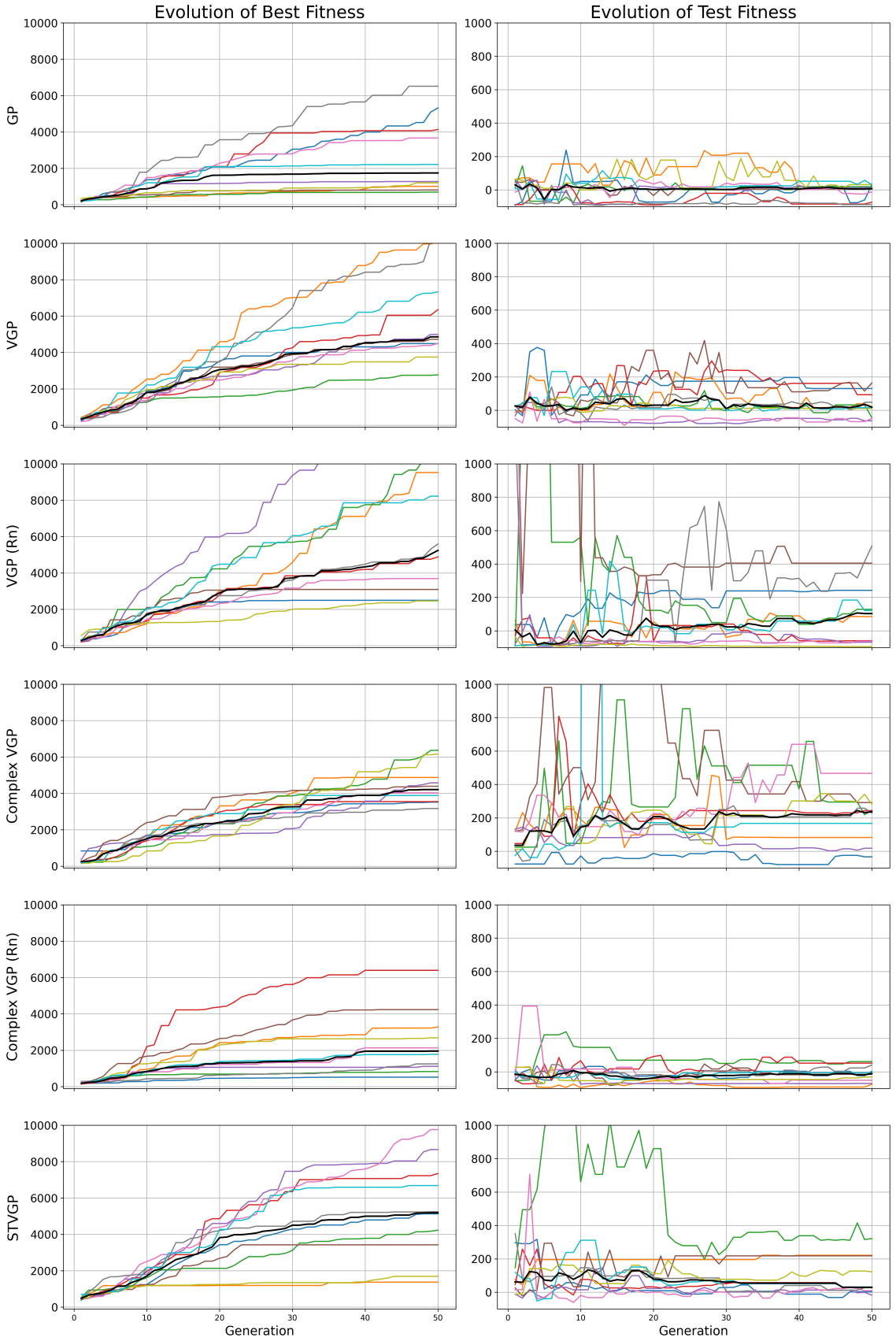


Figure 5.7: Comparison of Best Fitness during Training and Test Fitness across 10 runs for COTY stock. Each color represents one run; the thicker black line represents the median at each generation. The y axis may cut some lines to facilitate the visualization.

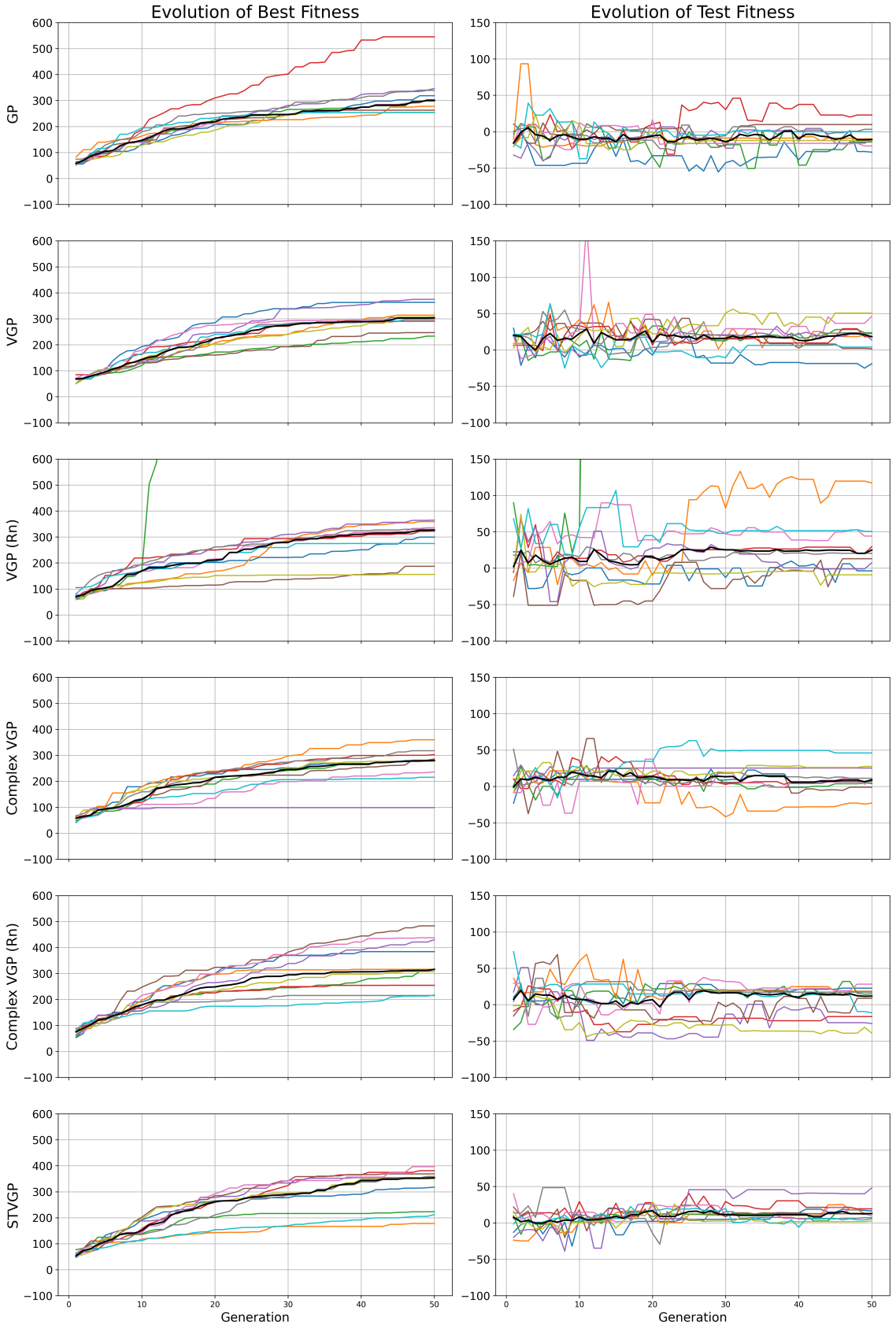


Figure 5.8: Comparison of Best Fitness during Training and Test Fitness across 10 runs for KO stock. Each colour represents one run; the thicker black line represents the median at each generation. The y-axis may cut some lines to facilitate the visualization.

Complex VGP, shown in the fourth panel, offers a concise upward training trajectory, with a flat outlier. Its test results, though noisy, don't show a distinct trend but stay positive.

The fifth panel, representing Complex VGP(Rn), indicates a bit of dispersion in its upward training trend. In test, while the results are noisy, there's no clear trend, yet the outcomes remain positive.

Lastly, STVGP, in the sixth panel, suggests a somewhat concise upward trend in training. The test, despite being noisy, presents a slow-sloping positive upward trend.

PSI20 In the first panel (Figure 5.9), representing GP, training data displays a somewhat concise upward trend. However, test data doesn't show a clear trend, remaining fairly neutral.

The second panel, VGP, showcases a concise upward trajectory in training. In contrast, its test data is noisy and doesn't display a discernible trend.

VGP(Rn), depicted in the third panel, presents a slightly dispersed upward trend in training, with an outlier. Its test data, while noisy, shows a slow upward slope.

In the fourth panel, Complex VGP offers a somewhat dispersed upward training trajectory. Test results, though noisy, stay positive without a clear trend.

The fifth panel, representing Complex VGP(Rn), indicates a dispersed upward trend during training. However, its test outcomes are noisy and trend negatively.

Lastly, STVGP, shown in the sixth panel, suggests a somewhat concise upward trend in training. The test data, though noisy, exhibits a strong upward trend.

TWTR In Figure 5.10, GP, depicted in the first panel, shows a dispersed upward trend in training. However, its test remains negative without a clear trend.

In the second panel, VGP presents a dispersed upward trajectory in training. The test data, while noisy, displays a slow upward slope.

VGP(Rn), in the third panel, reveals a very dispersed upward trend in training with four noticeable outliers. Its test is notably noisy but follows an upward trend.

Complex VGP, shown in the fourth panel, stands out with a concise upward trajectory in training. In contrast, its test data trends negatively without a specific pattern.

The fifth panel, Complex VGP(Rn), offers a dispersed upward movement during training. Test, however, is mainly negative with noise, except for a notable positive outlier.

Lastly, STVGP, represented in the sixth panel, suggests a somewhat concise upward trend in training. Its test data remains negative without a discernible trend.

Summary The displayed figures highlight a consistent improvement in training across all GP methods. However, when it comes to test, the results become more diverse, hinting at challenges in handling unseen data. For instance, while the GP method exhibits robust

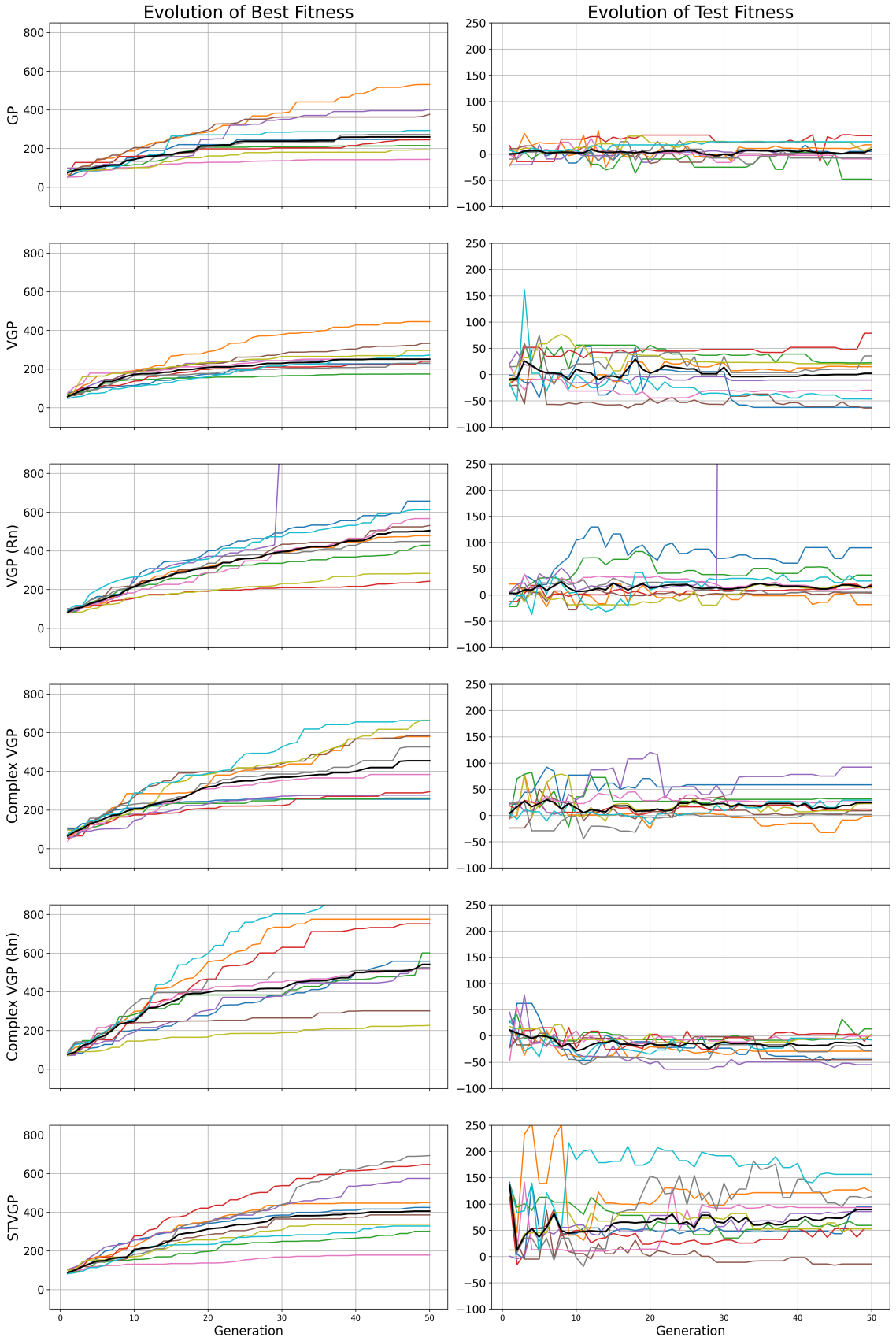


Figure 5.9: Comparison of Best Fitness during Training and Test Fitness across 10 runs for PSI20 stock. Each colour represents one run; the thicker black line represents the median at each generation. The y-axis may cut some lines to facilitate the visualization.

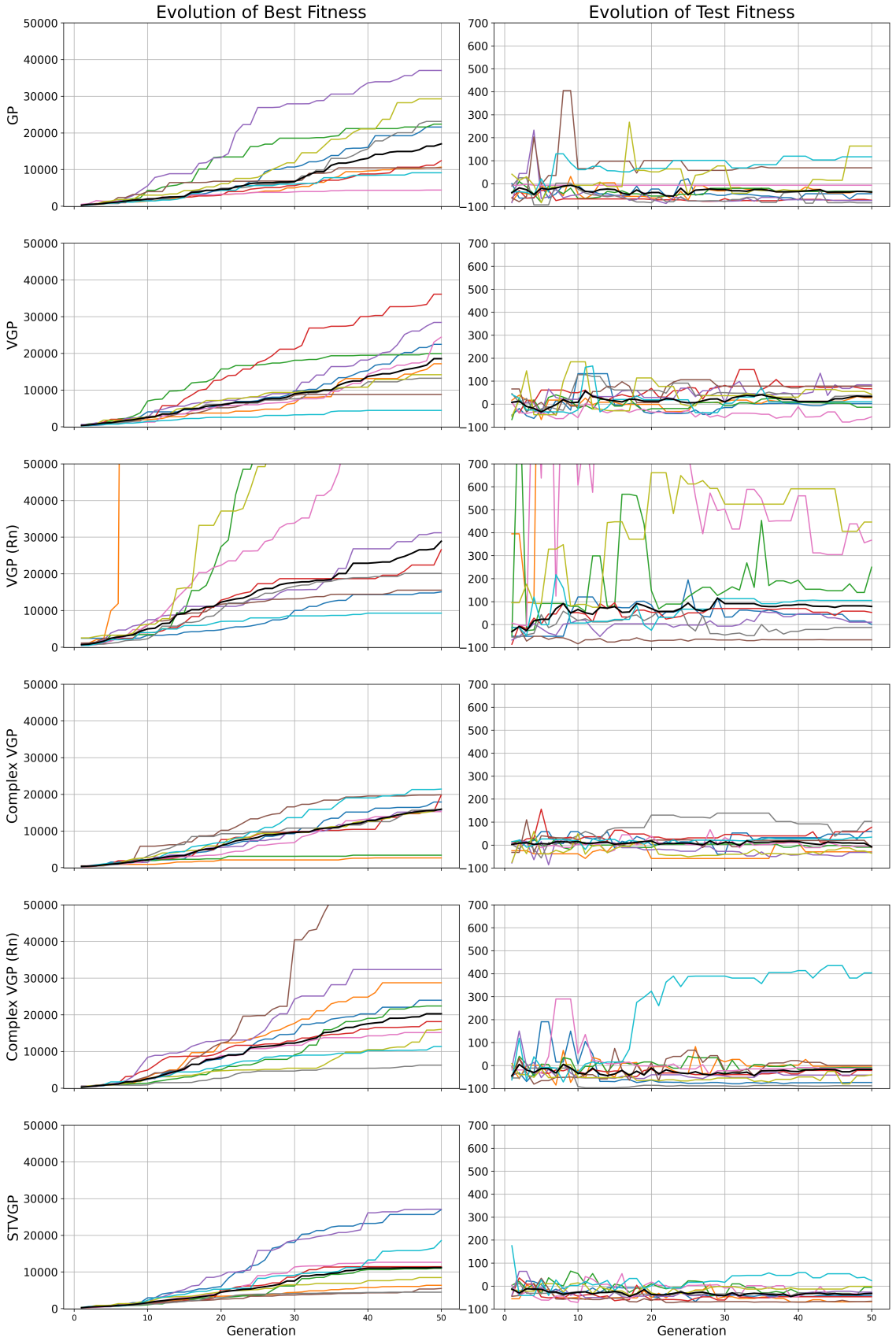


Figure 5.10: Comparison of Best Fitness during Training and Test Fitness across 10 runs for TWTR stock. Each color represents one run; the thicker black line represents the median at each generation. The y axis may cut some lines to facilitate the visualization.

Instrument	GP	VGP	VGP (Rn)	Complex VGP	Complex VGP (Rn)	STVGP
COTY	17.72	38.54	201.75	479.21	-10.61	82.69
KO	-10.76	30.13	42.31	16.16	20.87	21.48
PSI20	16.92	9.27	33.85	45.58	-17.87	132.78
TWTR	-36.73	68.22	164.17	-7.35	-17.23	-32.48
Medians Average	-3.21	133.40	36.54	110.52	-6.21	51.12
Variance	674.99	53618.45	597.42	7244.50	336.77	5177.82

Table 5.1: ROI Medians by method and stock.

optimization during training, its test performance is less convincing. The VGP method progresses steadily in training, yet its test results are more erratic, hinting at occasional overfitting. The Complex VGP method improves during training, but its test suggests difficulties in adapting to new patterns. Finally, the STVGP method demonstrates reliable training optimization but faces inconsistent test outcomes.

It's important to note that test fitness, representing performance on unseen data, has a shorter time frame in the market compared to training. This naturally results in a smaller magnitude for test outcomes.

Potential reasons for these observations include the possibility of overfitting, which might explain the disparity between training and test results. The significant changes in test fitness could be due to obstacles inherent in the optimization procedure or the nature of the problem itself.

5.3.2 ROI Table Analysis

The data presented in Table 5.1 shows the median ROI for different GP methods across selected financial instruments. From the table, we can see:

- Complex VGP stands out with a high median ROI for COTY but shows a significant drop in performance with TWTR, suggesting it may excel in certain market conditions but not others.
- VGP and STVGP seem to deliver more consistent median ROIs across the board. VGP, with its positive medians and lower variance, suggests a stable pattern recognition across different markets.
- GP shows a mixed outcome, with negative medians for KO and TWTR, indicating challenges in those markets.
- The negative medians for Complex VGP (Rn) across several instruments could point to a struggle in pattern identification or a lack of adaptability to varying market conditions.

- STVGP's strong performance in PSI20 contrasts with its negative median for TWTR, indicating possible specialization in the method's approach to different market dynamics.

The average of medians provides an overview of each method's typical performance, with VGP showing a positive average suggesting overall stability. In contrast, the high variance for Complex VGP indicates a less predictable performance.

This performance snapshot across different models and stocks illustrates the varied success of each method, with some showing potential for robustness and others indicating areas where improvement might be necessary.

5.3.3 Boxplot Analysis

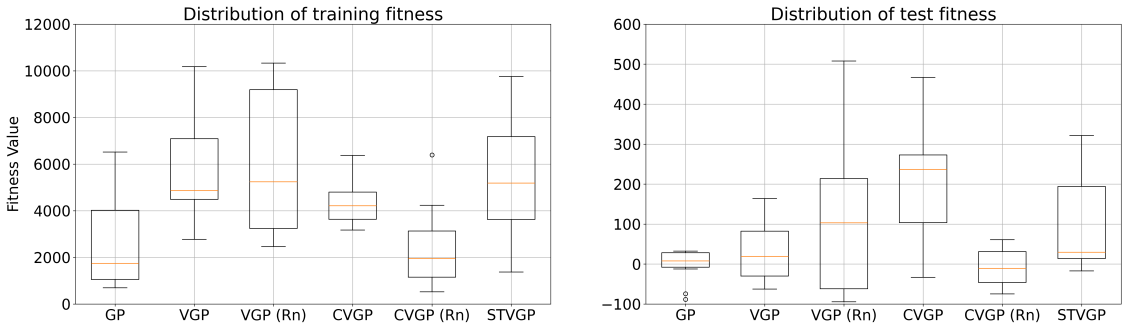
The boxplots 5.11 summarize the performance of each GP method, and a closer look at the numbers provides a clearer picture of their behavior across multiple runs.

COTY CVGP stands out in the COTY dataset with a median test fitness of 236.68, showing potential for high returns despite a wide IQR from 103.60 to 273.06, indicating substantial variability. Conversely, CVGP (Rn) presents a notably lower median of -10.61 and a tighter IQR, suggesting less volatility but also reduced performance. Compared to these, STVGP and VGP exhibit more moderate and consistent medians, with STVGP offering a better balance between risk and return.

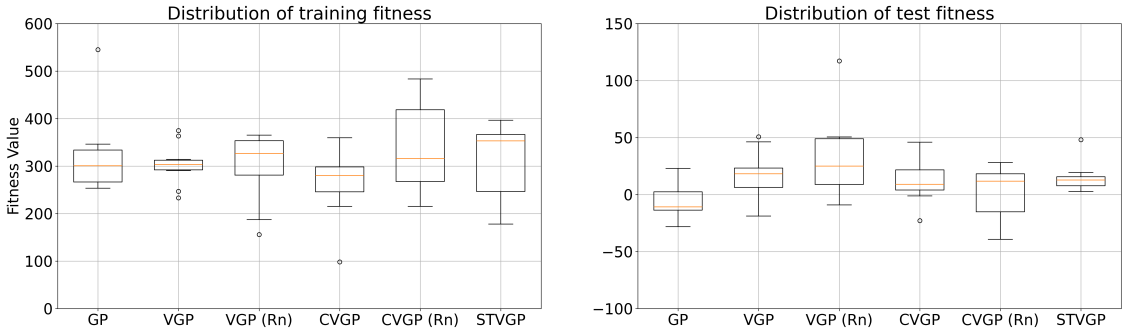
KO In the KO dataset, VGP (Rn) has high outliers up to 453.77, hinting at the potential for exceptional gains, although its median of 24.98 and wide IQR point to inconsistent performance. STVGP, with its narrow IQR and median of 12.79, indicates steadier performance. The other methods, GP and CVGP, show lower medians and smaller IQRs, reflecting a tendency towards consistent but lower returns, suggesting a more conservative approach in this market.

TWTR VGP (Rn) presents a large IQR and median of 78.70 in the TWTR dataset, suggesting a high-risk, high-reward profile. VGP's narrower IQR and median of 32.47 show a more stable strategy with less extreme results. STVGP, although having a lower median, shows a relatively small spread, indicating consistency. In comparison, CVGP and CVGP (Rn) fall behind, with their lower median test fitness and wider spreads, suggesting less reliability in this context.

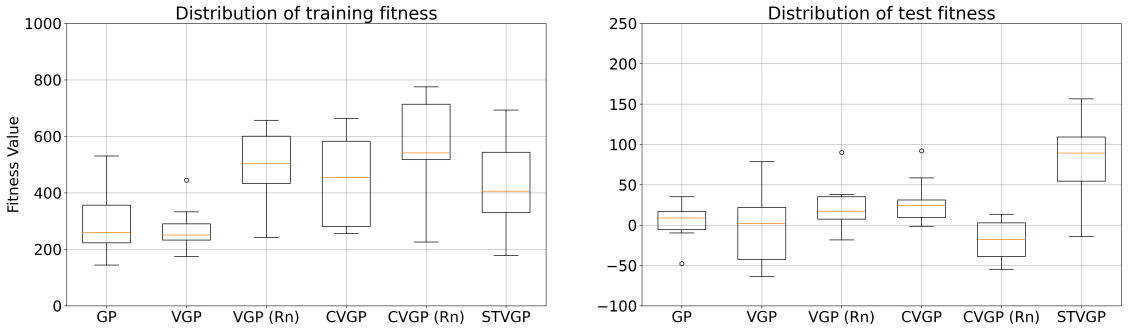
PSI20 For PSI20, STVGP's high median test fitness of 89.26 with a compact IQR signals reliable performance. The other methods, such as GP, VGP, and CVGP, display wider ranges and lower medians, indicating a mix of conservative strategies and less predictable outcomes.



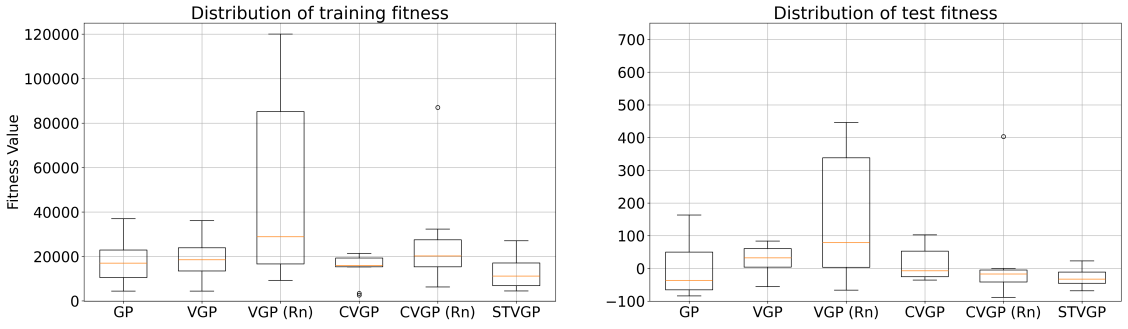
(a) COTY dataset



(b) KO dataset



(c) PSI20 dataset



(d) TWTR dataset

Figure 5.11: Comparative analysis of ROI across methods and runs during the test phase, stratified by instrument

Cross-Dataset Performance Patterns Across all datasets, STVGP generally delivers consistent medians and narrower IQRs, suggesting stability and reliability. Methods like CVGP and VGP (Rn) offer the possibility of higher returns but also exhibit greater variability in their results, indicating a higher risk profile.

5.4 Statistical analysis

In the following subsections, the comparative analysis of various methods across different datasets is presented. The analytical foundation is built upon p-values, derived from the Kruskal-Wallis H-test, which serve to illustrate statistically significant differences in the median results of the methods under examination. The relationships between methods are visually symbolized in tabular form: green signifies the method in the row has a statistically higher median than the method in the column, red signals the contrary, and the absence of colour indicates a non-significant statistical difference.

5.4.1 COTY

The table 5.2 lists the pairwise p-values from the Kruskal-Wallis H-test for the COTY dataset.

COTY	GP	VGP	VGP(Rn)	CVGP	CVGP(Rn)	STVGP
GP	—	0.2697	0.1223	0.0092	0.5078	0.0576
VGP	0.2697	—	0.4529	0.0193	0.1711	0.2004
VGP(Rn)	0.1223	0.4529	—	0.4015	0.1223	0.8946
CVGP	0.0092	0.0193	0.4015	—	0.0041	0.2332
CVGP(Rn)	0.5078	0.1711	0.1223	0.0041	—	0.0152
STVGP	0.0576	0.2004	0.8946	0.2332	0.0152	—

Table 5.2: Pairwise p-values from the Kruskal-Wallis H-test for the COTY dataset.

Interpretation The analysis indicates that CVGP often performs well, while VGP(Rn) maintains consistent results with other methods. GP and CVGP(Rn) show lower performance in specific instances.

5.4.2 KO

Table 5.3 we compare the performance of different methods against the KO dataset, using the Kruskal-Wallis H-test to assess the statistical significance of differences between their median results.

- **GP**: Shows similar median results to most methods but is outperformed by CVGP.
- **VGP**: This is generally on par with other methods except for CVGP, which shows a higher median.
- **VGP(Rn)**: Median results do not significantly differ from any methods.
- **CVGP**: Tends to have higher median results than GP, VGP, and CVGP(Rn), suggesting a strong performance in this dataset.
- **CVGP(Rn)**: Performs less well compared to STVGP but is comparable to the rest.
- **STVGP**: Shows superiority over CVGP(Rn) but has comparable results to other methods.

KO	GP	VGP	VGP(Rn)	CVGP	CVGP(Rn)	STVGP
GP	—	0.0009	0.0031	0.0023	0.3099	0.0013
VGP	0.0009	—	0.9648	0.4015	0.1223	0.2332
VGP(Rn)	0.0031	0.9648	—	0.4015	0.0851	0.4529
CVGP	0.0023	0.4015	0.4015	—	0.3538	0.5660
CVGP(Rn)	0.3099	0.1223	0.0851	0.3538	—	0.4015
STVGP	0.0013	0.2332	0.4529	0.5660	0.4015	—

Table 5.3: Pairwise p-values from the Kruskal-Wallis H-test for the KO dataset.

Interpretation

- **GP**: This method shows lower median results than several others, including CVGP, STVGP, VGP, and VGP(Rn) with significant p-values.
- **VGP**: Demonstrates better performance than GP and is statistically similar to the rest, implying it's a competitive option.
- **VGP(Rn)**: Also outperforms GP and is comparable to the rest, indicating consistency.
- **CVGP**: Beats GP in performance and is statistically comparable to other methods, suggesting it's neither at the top nor bottom in this group.
- **CVGP(Rn)**: Shows no significant difference in median results when compared with other methods, indicating it performs on par.
- **STVGP**: Outperforms GP significantly and matches other methods, positioning it as a consistently strong performer.

The results indicate GP is generally outperformed, suggesting other methods may be more effective for the KO dataset. Both VGP(Rn) and CVGP(Rn) show a level of performance consistent with most methods.

5.4.3 PSI20

The table 5.4 shows the p-values from the Kruskal-Wallis H-test for the PSI20 dataset, which help compare the median results of various methods.

PSI20	GP	VGP	VGP(Rn)	CVGP	CVGP(Rn)	STVGP
GP	—	0.7573	0.1223	0.0305	0.1223	0.0041
VGP	0.7573	—	0.1711	0.0851	0.7573	0.0031
VGP(Rn)	0.1223	0.1711	—	0.7573	0.0031	0.0576
CVGP	0.0305	0.0851	0.7573	—	0.0041	0.0152
CVGP(Rn)	0.1223	0.7573	0.0031	0.0041	—	0.0017
STVGP	0.0041	0.0031	0.0576	0.0152	0.0017	—

Table 5.4: Pairwise p-values from the Kruskal-Wallis H-test for the PSI20 dataset.

Interpretation

- **GP** has lower median results compared to CVGP and STVGP, as indicated by p-values of 0.0305 and 0.0041.
- **VGP** falls short in performance compared to STVGP with a p-value of 0.0031, indicating STVGP's stronger position.
- **VGP(Rn)** shows a better performance over CVGP(Rn) as reflected by a p-value of 0.0031.
- **CVGP** has a higher median result than GP and CVGP(Rn) with p-values of 0.0305 and 0.0041, but it doesn't hold up as well against STVGP, which has a p-value of 0.0152.
- **CVGP(Rn)** is outperformed by CVGP and STVGP with p-values of 0.0041 and 0.0017 but shows better results compared to VGP(Rn).
- **STVGP** consistently shows higher median results compared to GP, VGP, CVGP, and CVGP(Rn) with statistically significant p-values, suggesting its robustness in this dataset.

For the PSI20 dataset, STVGP tends to yield the most robust median values, while GP often has lower outcomes relative to others. CVGP and CVGP(Rn) show a mix of performances, with CVGP generally outperforming CVGP(Rn). These findings suggest that for PSI20, STVGP might be the method to lean towards due to its consistent performance in this analysis.

5.4.4 TWTR

The table below outlines the p-values from the Kruskal-Wallis H-test for the TWTR dataset, comparing the median results of the methods used.

TWTR	GP	VGP	VGP(Rn)	CVGP	CVGP(Rn)	STVGP
GP	—	0.0851	0.0193	0.1023	0.8253	0.6911
VGP	0.0851	—	0.1711	0.3538	0.0469	0.0092
VGP(Rn)	0.0193	0.1711	—	0.0703	0.0243	0.0092
CVGP	0.1023	0.3538	0.0703	—	0.2004	0.0380
CVGP(Rn)	0.8253	0.0469	0.0243	0.2004	—	0.6911
STVGP	0.6911	0.0092	0.0092	0.0380	0.6911	—

Table 5.5: Pairwise comparison between different models on TWTR dataset

Interpretation

- **GP** is shown to have a lower median result compared to VGP(Rn) with a p-value of 0.0193.
- **VGP** scores lower than STVGP, indicated by a p-value of 0.0092, and has a lower median result than CVGP(Rn) as well, with a p-value of 0.0469.
- **VGP(Rn)** shows a stronger performance compared to GP and CVGP(Rn), with p-values of 0.0193 and 0.0243, respectively, and stands on equal footing with STVGP.
- **CVGP** has a median result that is not statistically different from most methods except for a lower performance compared to STVGP, with a p-value of 0.0380.
- **CVGP(Rn)** is outperformed by VGP and VGP(Rn), with p-values of 0.0469 and 0.0243, respectively.
- **STVGP** shows a stronger median result compared to VGP and VGP(Rn) with p-values of 0.0092 for both comparisons.

The data indicates that VGP(Rn) tends to perform well in this dataset, showing higher median results against several methods. STVGP's performance is also notable, particularly in comparison to VGP and VGP(Rn). CVGP exhibits a mid-range performance, whereas CVGP(Rn) and GP are often outperformed in this dataset. For the TWTR dataset, VGP(Rn) stands out for its effectiveness in the pairwise comparisons shown.

5.5 Summary

In this chapter, we examined the performance of GP methods in trading, gleaning insights from both visual and numerical analyses.

Training Consistency vs. Test Variability: The figures, particularly figure 5.9, illustrate a stable increase during training but a spread in results during testing, indicating a struggle to extend learned strategies to new data.

Variation Within Methods: Variability within a single method's results, as seen in 5.11d, raises points about the unpredictability of financial markets and the stochastic nature of genetic programming.

Potential Overfitting: The sharp disparity between training and test results, as shown in figure 5.10, suggests that models might be overfitting the training data and failing to predict new market conditions accurately.

Performance Across Markets: The performance metrics in Table 5.1 highlight how models like Complex VGP can excel with certain stocks but falter with others, underscoring the challenges in achieving broad market reliability.

Return on Investment (ROI) Trends: The ROI boxplots (figure 5.11) provide a snapshot of consistency and risk. STVGP's narrow ROI range for PSI20 suggests a reliable method for capturing market trends that could deliver consistently high returns.

Exploration vs. Exploitation Dynamics: The exploration-exploitation balance is evident, where methods like VGP(Rn) show a wide distribution of ROI (figures 5.10), reflecting a risk-taking approach, and STVGP shows a narrower range, reflecting a cautious strategy.

Observations on Specific Methods:

STVGP: Exhibits tight ROI ranges, indicating its strategies are not only stable but also potentially easier to understand, making it a strong candidate for reliable trading strategies (figure 5.9).

Complex VGP: With its broad ROI distributions, it suggests a method that, while possibly capturing a wider array of market behaviours, also introduces greater risk and less predictability (figure 5.7).

Chapter 6

Conclusion

6.1 Summary of Work and Key Findings

Our study has delved into the application of Genetic Programming (GP) techniques to develop trading strategies. The research has provided a comparison of GP methods, highlighting distinctive performance traits and behaviours in various market environments, with an emphasis on the technical mechanisms and innovations introduced by these approaches.

The STVGP method has contributed to model interpretability by generating rule-like structures, which could improve the clarity of trading strategies for expert analysis.

CVGP has been noted to broaden the solution space, suggesting that its utility could extend to diverse scenarios. This preliminary observation points to the necessity for additional research to determine its full range of applicability.

VGP has shown favourable results, indicating a solid performance potential in trading strategy development. Its efficacy across different contexts warrants further exploration to confirm its promise.

Analysis of strategy formulation within the models revealed specific patterns. The incorporation of trigonometric functions has pointed to a strategy leaning on periodic market trends, while the integration of ema50 has been indicative of utilizing long-term price information. Cumulative functions have hinted at a focus on long-term market dynamics, and the use of addition operations (V_ADD) has been associated with strategies that merge various market metrics into composite decision-making indicators.

6.2 Implications of Findings

STVGP: Clearer Models STVGP's consistent performance in both the training and testing phases underlines its potential for generating understandable strategies. Its ability to form rule-like structures not only aids in transparency but also may contribute to ease of modification and refinement of trading strategies, making it a practical choice for real-

world application.

VGP: Strong Results VGP demonstrated solid performance, adapting effectively to different market conditions. Its capacity to consistently leverage market patterns across datasets underscores its potential as a versatile tool in financial strategy development. VGP's strength lies in its balance between adaptability and performance, making it a reliable choice for traders.

CVGP: Expanded Possibilities While CVGP showed varying outcomes, its approach to exploring a larger solution space is notable. Its diverse performance hints at an inherent flexibility, which could be advantageous when dealing with complex and unpredictable market scenarios. Further analysis could uncover specific contexts where CVGP excels, guiding its application in other domains.

Stable and Exploratory Methods Both VGP and STVGP presented stable patterns, suggesting they can reliably extract and utilize market trends. The range of outcomes seen with VGP(Rn) introduces the idea that this method could be beneficial for scenarios that reward higher risk-taking for potentially greater returns.

Choosing the Right Method The study revealed that no single method uniformly outperforms others; the choice of method must be informed by the specific characteristics of the dataset and the intended application. This decision-making process is crucial for optimizing the effectiveness of GP methods in practice.

Further Investigation and Development These findings suggest several directions for future research, including the creation of hybrid models that combine the capabilities of multiple GP techniques. The goal would be to develop models that maintain performance stability while also exploring new patterns, thus meeting the dual needs of financial trading reliability and discovery.

6.3 Concluding Remarks

This study has examined a range of Genetic Programming (GP) methods and their roles in solving problems. The findings suggest that these methods could be useful not just in trading but in other complex areas as well. As technology and machine learning continue to advance, it's clear that the methods we've explored have the potential to be applied to a variety of challenges. Future research should aim to further understand these techniques and explore how they can be adapted to meet different needs in a rapidly changing world.

Bibliography

[1] Eva Alfaro-Cid, Ken Sharman, and Anna I. Esparcia-Alcázar. Genetic programming and serial processing for time series classification. *Evol. Comput.*, 22(2):265–285, jun 2014.

[2] I. Azzali, L. Vanneschi, S. Silva, I. Bakurov, and M. Giacobini. A vectorial approach to genetic programming. In L. Sekanina, T. Hu, N. Lourenço, H. Richter, and P. García-Sánchez, editors, *Genetic Programming. EuroGP 2019. Lecture Notes in Computer Science*, volume 11451. Springer, Cham, 2019.

[3] Irene Azzali, Leonardo Vanneschi, Illya Bakurov, Sara Silva, Marco Ivaldi, and Mario Giacobini. Towards the use of vector based gp to predict physiological time series. *Applied Soft Computing Journal*, 89, 4 2020.

[4] Irene Azzali, Leonardo Vanneschi, Sara Silva, Illya Bakurov, and Mario Giacobini. A vectorial approach to genetic programming. In *Lecture Notes in Computer Science*, pages 213–227. Springer International Publishing, 2019.

[5] Rodrigo Coelho Barros, Márcio Porto Basgalupp, André C. P. L. F. de Carvalho, and Alex A. Freitas. A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(3):291–312, 2012.

[6] Tim Bollerslev, Viktor Todorov, and Lai Xu. Tail risk premia and return predictability. *Journal of Financial Economics*, 118(1):113–134, October 2015.

[7] O. Bustos and A. Pomares-Quimbaya. Stock market movement forecast: A systematic review, 10 2020.

[8] John Y. Campbell and Motohiro Yogo. Efficient tests of stock return predictability. *Journal of Financial Economics*, 81(1):27–60, 2006.

[9] Shu-Heng Chen and Chia-Hsuan Yeh. Toward a computable approach to the efficient market hypothesis: An application of genetic programming, 1997.

[10] Shu-Heng Chen and Chia-Hsuan Yeh. Evolving traders and the business school with genetic programming: A new architecture of the agent-based arti"cial stock market, 2001.

[11] Paulo Cortez. Modelos inspirados na natureza para a previsão de séries temporais. 2002.

[12] Shusheng Ding, Tianxiang Cui, Xihan Xiong, and Ruibin Bai. Forecasting stock market return with nonlinearity: a genetic programming approach. *Journal of Ambient Intelligence and Humanized Computing*, 11:4927–4939, 11 2020.

[13] Alexandre Elder. *Trading for a living: Psychology, trading tactics, money management(Vol. 31)*. Wiley, New York, 1993.

[14] Pedro Espejo, Sebastian Ventura, and Francisco Herrera. A survey on the application of genetic programming to classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40:121 – 144, 04 2010.

[15] Eugene F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, 1970.

[16] Eugene F. Fama and Kenneth R. French. Dividend yields and expected stock returns. *Journal of Financial Economics*, 22(1):3–25, 1988.

[17] Rodolfo Toríbio Farias Nazário, Jéssica Lima e Silva, Vinicius Amorim Sobreiro, and Herbert Kimura. A literature review of technical analysis on stock markets. *The Quarterly Review of Economics and Finance*, 66:115–126, 2017.

[18] P. Fleck, S. Winkler, M. Kommenda, and M. Affenzeller. Grammar-based vectorial genetic programming for symbolic regression. In W. Banzhaf, L. Trujillo, S. Winkler, and B. Worzel, editors, *Genetic Programming Theory and Practice XVIII. Genetic and Evolutionary Computation*. Springer, Singapore, 2022.

[19] Philipp Fleck, Stephan Winkler, Michael Kommenda, and Michael Affenzeller. Vectorial genetic programming – optimizing segments for feature extraction. 2023. Preprint. Submitted to Eurocast 2022, but was not published in the 2022 proceedings due to an error in the submission information system. Will be published in the Eurocast 2024 proceedings.

[20] A. J. Fox. Outliers in time series. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34:350–363, 7 1972.

[21] Amir H. Gandomi, Amir H. Alavi, and Conor Ryan, editors. *Handbook of Genetic Programming Applications*. Springer International Publishing, Cham, 2015.

[22] Benjamin Golez and Peter Koudijs. Four centuries of return predictability. *Journal of Financial Economics*, 127(2):248–263, 2018.

[23] I. Gonçalves and S. Silva. Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In K. Krawiec, A. Moraglio, T. Hu, A.Ş. Etaner-Uyar, and B. Hu, editors, *Genetic Programming. EuroGP 2013*, volume 7831 of *Lecture Notes in Computer Science*, Berlin, Heidelberg, 2013. Springer.

[24] Crina Grosan and Ajith Abraham. *Stock Market Modeling Using Genetic Programming Ensembles*, pages 131–146. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[25] Crina Grosan, Ajith Abraham, Vitorino Ramos, and Sang Yong Han. Stock market prediction using multi expression programming. In *EPIA*, pages 73–78. IEEE, 2005.

[26] D. M. Hawkins. *Identification of Outliers*. Springer Netherlands, 1980.

[27] John H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press, 2010.

[28] H. Iba and T. Sasaki. Using genetic programming to predict financial data. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 1, pages 244–251 Vol. 1, 1999.

[29] Hitoshi Iba and Takashi Sasaki. Using genetic programming to predict financial data, 1999.

[30] M. A. Kaboudan. Genetic programming prediction of stock prices. *Computational Economics*, 16(3):207–236, Dec 2000.

[31] Andreas Karatahansopoulos, Georgios Sermpinis, Jason Laws, and Christian Dunis. Modelling and trading the greek stock market with gene expression and genetic programing algorithms. *Journal of Forecasting*, 33:596–610, 12 2014.

[32] Ahmed Kattan, S. Fatima, and Muhammad Arif. Time-series event-based prediction: An unsupervised learning framework based on genetic programming. *Information Sciences*, 301, 04 2015.

[33] J. KOZA. On the programming of computers by means of natural selection. *Genetic Programming*, 1992.

[34] John Koza and Riccardo Poli. *Genetic Programming*, pages 127–164. 01 2005.

[35] Leslie Lamport. *LaTeX - A Document Preparation System: User's Guide and Reference Manual, Second Edition*. Pearson / Prentice Hall, New York, 1994.

[36] Yi Shian Lee and Lee Ing Tong. Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming. *Knowledge-Based Systems*, 24:66–72, 2 2011.

[37] Ruobing Liu, Jianhui Yang, and Chuanyang Ruan. Expected stock return and mixed frequency variance risk premium data. *J. Ambient Intell. Humaniz. Comput.*, 11(9):3585–3596, September 2020.

[38] Ioannis E Livieris, Niki Kiriakidou, Stavros Stavroyiannis, and Panagiotis Pintelas. An advanced cnn-lstm model for cryptocurrency forecasting. 2021.

[39] Alexander Loginov, Malcolm Heywood, and Garnett Wilson. Stock selection heuristics for performing frequent intraday trading with genetic programming. *Genetic Programming and Evolvable Machines*, 22:35–72, 3 2021.

[40] Viktor Manahov and Andrew Urquhart. The efficiency of bitcoin: A strongly typed genetic programming approach to smart electronic bitcoin markets. *International Review of Financial Analysis*, 73, 1 2021.

[41] Kevin Michell and Werner Kristjanpoller. Generating trading rules on us stock market using strongly typed genetic programming. *Soft Computing*, 24:3257–3274, 3 2020.

[42] David J. Montana. Strongly Typed Genetic Programming. *Evolutionary Computation*, 3(2):199–230, 06 1995.

[43] John J. Murphy. *Study Guide to Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. 1999.

[44] Christopher Neely, Paul Weller, and Rob Dittmar. Is technical analysis in the foreign exchange market profitable? a genetic programming aproach, 1997.

[45] Alexandre Pimenta, Ciniro A.L. Nametala, Frederico G. Guimarães, and Eduardo G. Carrano. An automated investing method for stock market based on multiobjective genetic programming. *Computational Economics*, 52:125–144, 6 2018.

[46] Jean Yves Potvin, Patrick Soriano, and Vallée Maxime. Generating trading rules on the stock markets with genetic programming. *Computers and Operations Research*, 31:1033–1047, 2004.

[47] Martin J. Pring. *Technical analysis explained*. McGraw-Hill Education, 2014.

[48] Anupriya Shukla, Hari Mohan Pandey, and Deepti Mehrotra. Comparative review of selection techniques in genetic algorithm. In Bhawna Kumar, Gurinder Singh, and J.S. Jassi, editors, *2015 1st International Conference on Futuristic Trends in Computational Analysis and Knowledge Management, ABLAZE 2015*, 2015 1st International Conference on Futuristic Trends in Computational Analysis and Knowledge Management, ABLAZE 2015, pages 515–519, United States, July 2015. Institute of Electrical and Electronics Engineers Inc.

[49] Dirk Thierens and David Goldberg. Convergence models of genetic algorithm selection schemes. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature — PPSN III*, pages 119–129, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[50] Franz Wilhelmstötter. Jenetics. *URL: http://jenetics. io*.