Dynamic Ensemble of Specialized Models for Multi-Timeframe Stock Market Trend Prediction

João Miguel Pipa Ferreira Caldeira

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisor: Prof. Rui Fuentecilla Maia Ferreira Neves

Examination Committee

Chairperson: Prof. Paolo Romano
Supervisor: Prof. Rui Fuentecilla Maia Ferreira Neves
Member of the Committee: Prof. Luís Manuel Silveira Russo

November 2024

Acknowledgments

Completing this thesis has been a journey filled with challenges and learning experiences, and I am deeply appreciative of everyone who contributed to this endeavor.

First and foremost, I want to express my gratitude to my thesis supervisor, Prof. Rui Neves, for his invaluable guidance, mentorship, and constant feedback throughout the development of this work. Our discussions on the stock market were both productive and engaging, significantly improving the research and my understanding of the whole context. I also appreciate the freedom the Professor provided to explore various aspects within the thesis theme, which allowed me to develop my ideas fully and ultimately develop a work that I am truly proud of.

I would like to extend my sincere gratitude to Instituto Superior Técnico and its teachers for their commitment to excellence in teaching. Their high standards have equipped us to tackle professional challenges and strive for improvement. I am proud and grateful to have studied here.

To all my friends, your patience and support have been invaluable. Your ability to lighten stressful moments and offer a listening ear has made a tremendous difference. The memories we've created together will stay with me forever. I especially want to thank my friends at Instituto Superior Técnico, your companionship made the academic journey enjoyable and filled with laughter while motivating me to strive for excellence.

To my lifelong best friends, thank you for always being there for me. Your support and company has made tough times easier, and your presence brightens every moment. I cherish our memories together, you truly make everything better.

To my girlfriend, I want to express my heartfelt gratitude for your unwavering support throughout this thesis journey. Your encouragement and understanding during the long nights and challenging moments have been my guiding light, making all the difference. I couldn't have done this without you by my side.

Lastly, I am forever grateful to my family for their unconditional love and support. Their faith in my abilities has fueled my determination, and their sacrifices have not gone unnoticed. Thank you for being my backbone and my greatest supporters throughout this journey.

A special thank you to my grandparents, Carlos Alberto and Maria da Piedade, to whom I dedicate this thesis in honor of their final wish to see me graduate.

Abstract

This thesis addresses the challenge of using machine learning to predict stock market trends, by developing a novel dual-model architecture, DTE-2SVM, which separates long-term and short-term trends, utilizing specialized SVMs for each to predict the S&P 500 index. The system aims to handle the stock market's inherent volatility, and complexity by decomposing the market behavior using different scopes of observation, creating distinct strategies enhancing responsiveness and predictive capacity in both bullish and bearish market conditions. To further refine the dual-model approach, a hyperparameter tuning process is employed separately, for each trend model. Additionally, an enhanced data labeling method based on an N-Period Min-Max algorithm is introduced, which captures the markets fluctuations through the identification of critical points, local maxima and minima. The DTE-2SVM system is evaluated through case studies, addressing the key aspects of the proposed solution. Findings reveal that DTE-2SVM significantly outperforms both traditional machine learning models and the benchmark buy-and-hold strategy, achieving a 204.31% return on investment, above the buy-and-hold's 132.11%, on the S&P 500 index between 2015 and 2023. Additionally, the system demonstrates consistent returns and effective risk management, highlighting its robustness in diverse market conditions. Moreover, the research highlights the significance of hyperparameter tuning in enhancing the model's performance and profitability results.

Keywords

Stock Market; Market Trends; Technical Analysis; Support Vector Machines; Genetic Algorithms; Ensemble Learning.

Resumo

Esta tese aborda a previsão de tendências do mercado de ações usando machine learning através do desenvolvimento de uma nova arquitetura de modelo duplo, DTE-2SVM, que separa tendências de longo e curto prazo, utilizando SVMs otimizadas para cada tendência, visando prever o índice S&P 500. O sistema visa lidar com a volatilidade e complexidade inerentes ao mercado de ações, decompondo o comportamento geral do mercado utilizando diferentes perspetivas temporais de observação, criando estratégias distintas que melhoram a capacidade de resposta e de previsão nos variados comportamentos do mercado de ações. De forma a maximizar o efeito da abordagem de modelo duplo são aplicados processos de afinação de hiperparâmetros, separadamente para cada modelo, obtendo hiperparâmetros otimizados para cada estratégia. Adicionalmente, é introduzido uma melhoria de um método de atribuição de classes de dados, baseado num algoritmo de Min-Máx em n-períodos, que captura as flutuações de mercado através da identificação de pontos críticos como o máximo e mínimo local. O sistema proposto, DTE-2SVM, é avaliado através de casos de estudo que abordam aspetos-chave da solução proposta. Os resultados mostram que o DTE-2SVM supera modelos tradicionais de aprendizagem e a estratégia de referência de Buy & Hold, alcançando um retorno sobre investimento de 204,31%, superando os 132,11% da estratégia de Buy & Hold, no índice S&P 500 entre 2015 e 2023. A solução demonstra ainda consistência nos retornos e uma gestão de risco eficaz. Acrescentando ainda, a importância da afinação de hiperparâmetros na melhoria do desempenho e da rentabilidade do modelo.

Palavras Chave

Mercado de Ações; Tendências de Mercado; Análise Técnica; Support Vector Machines; Algoritmos Genéticos; Aprendizagem em Conjunto.

Contents

List of Figures

List of Tables

Acronyms

ADO	Accumulation/Distribution Oscillator
AI	Artificial Intelligence
APO	Absolute Price Oscillator
ATR	Average True Range
BBANDS	Bollinger Bands
BI	Bias Indicator
CVI	Chaikin Volatility Index
CSV	Comma Separated Values
DEAP	Distributed Evolutionary Algorithms in Python
DEM	Dynamic Ensemble Model
DEMA	Double Exponential Moving Average
DGA	Dynamic Genetic Algorithm
DPO	Detrended Price Oscillator
DTE-2SVM	Dynamic Trend Ensemble Dual Support Vector Machine
EA	Evolutionary Algorithm
EMA	Exponential Moving Average
GA	Genetic Algorithm
HMA	Hull Moving Average
KKT	Karush-Kuhn-Tucker
k-NN	k-Nearest Neighbors
MACD	Moving Average Convergence Divergence
MOEA	Multi-Objective Evolutionary Algorithm

MST	Minimum Spanning Tree
NPMM	N-Period Min-Max
OBV	On Balance Volume
PandasTA	Pandas Technical Analysis
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RBF-SVM	Radial Basis Function Support Vector Machine
ROC	Rate of Change
ROI	Return on Investment
RSI	Relative Strength Index
RVI	Relative Volatility Index
SMA	Simple Moving Average
STOCH	Stochastic Oscillator
SVM	Support Vector Machine
WILLR	Larry Williams %R

Glossary

bearish

A financial term indicating a pessimistic outlook or downward trend in a market or asset, where prices are expected to decline. 59, 61, 72

bullish

A financial term indicating an optimistic outlook or upward trend in a market or asset, where prices are expected to rise. 58, 60

1 Introduction

Contents

Predicting stock market trends poses a significant challenge and opportunity in financial analysis [1]. Traditional methods like fundamental and technical analysis focus on company metrics and historical price data. However, these methods have limitations, fundamental analysis misses real-time sentiments, and technical analysis struggles with market randomness and complexity. Furthermore, there are inherent limitations to human capabilities in predicting the stock market, including emotional biases, information overload, and delayed reactions that further hinder predictions. The rise of Artificial Intelligence (AI) has shifted this landscape, addressing many of these challenges.

Machine learning, an application of AI, involves the use of algorithms that can learn and make predictions/decisions based on data. This offers a promising approach in the context of stock market prediction, as machine learning algorithms can process and analyze vast volumes of data while handling the complexity and non-linearity inherent in stock market data to uncover patterns and relationships [2].

1.1 Motivation

The stock market is a highly complex and dynamic system influenced by a wide range of factors, including politics, economic indicators, global events, corporate performance, and market sentiment. This inherent uncertainty and volatility makes the prediction of trends especially challenging. The Efficient Market Hypothesis supports the idea that asset prices reflect all available information, making it difficult to consistently achieve higher returns than the market [1]. Three forms of market efficiency are suggested by this hypothesis: weak-form, where historical prices are reflected in current prices; semi-strong form, where all publicly available information is factored into prices; and strong-form, which asserts that even insider information is accounted for in stock prices. Given these complexities, no technique can fully predict market movements with absolute accuracy.

Machine learning offers the potential to better navigate this uncertainty by uncovering patterns in large datasets. In stock market applications, machine learning algorithms can analyze historical price movements, economic indicators, and financial news to detect patterns and make informed predictions. Algorithms like Support Vector Machine (SVM), Random Forest, XGBoost, and k-Nearest Neighbors (k-NN), have been applied to various stock market prediction tasks, ranging from price prediction to volatility estimation. These models excel in recognizing hidden relationships within the data and can adapt to the nonlinear nature of the market.

Despite these advances, there remains a gap in current approaches that effectively decompose market trends. Most models focus on overall market trends but fail to differentiate between long-term and short-term dynamics. This work introduces a novel approach that addresses this gap by employing specialized models for long and short-term trends, combined with a dynamic ensemble mechanism that leverages market data to generate a final prediction. Furthermore, this study explores labeling strategies within the context of the stock market, proposing an innovative method that identifies critical points in market data (such as local minima and maxima) and labels the surrounding data leading to these points.

1.2 Objectives

This work leverages machine learning for stock market prediction and is guided by the following research objectives:

- Identify and label market trends, separating long trends from short trends, using technical analysis.
- Develop a dual-model framework that effectively captures the dynamics of long and short-term trends, improving on single model approaches.
- Surpass the profitability of the benchmark Buy & Hold trading strategy.

1.3 Contributions

The main contributions of this work to the field of stock market prediction using machine learning are:

- The dual-model architecture, where two models are created and optimized to predict either long or short-term market trends, each using specific features and labels.
- The dynamic ensemble strategy that leverages market information in the form of a trend technical indicator to adaptively combine model predictions based on current market conditions, deciding between a long trend prediction or a short trend prediction.
- The N-Period Min-Max data labeling strategy, which identifies minima and maxima, over a specified period, establishing a relationship between these critical points, generating labels that are then used to label the data ”outside” the critical points.

1.4 Document Outline

This document describes the research and work developed, and it is organized as follows:

- **Chapter** 1 Introduces the developed research, presenting the motivations, objectives and key contributions to the field.
- **Chapter** 2 Discusses the relevant literature in the field, outlining the theoretical background and explaining key concepts essential to the study.
- **Chapter** 3 Describes the architecture of the developed approach, detailing the system’s processes and algorithms to provide a comprehensive overview of the entire system.
- **Chapter** 4 Evaluates the proposed system through case studies, highlighting its most impactful aspects and potential drawbacks, thereby validating the significance of its contributions.
- **Chapter** 5 Summarizes the work developed with the conclusions regarding the presented results, outlining the system’s limitations and future work.

2 Related Work

Contents

This section examines the existing research on stock market prediction through the application of machine learning techniques. It begins by presenting the fundamental concepts of the stock market, with a particular emphasis on technical analysis. We then provide an overview of machine learning techniques, with a particular focus on SVMs, Evolutionary Algorithms (EAs), and Ensemble Learning, exploring how these methods relate to each other and ultimately discussing how their integration can offer a comprehensive solution for stock market prediction. A summary table highlighting the key research studies discussed is provided at the end of the section (see Table 2.2).

2.1 Stock Market

The stock market, alternatively referred to as an equity market or share market, serves as a centralized platform, where buyers and sellers come together to trade financial instruments, such as stocks and indexes. Stocks represent ownership in a company and confer the right to a portion of the company's assets and profits. A stock market index represents and reflects the composite value of a selected group of stocks, giving an indication of the overall health and performance of a specific market sector or of a stock market [3], like the Standard & Poor's 500 Index (S&P 500), which measures the stock performance of the 500 largest companies listed on stock exchanges in the United States (U.S.) and is considered one of the best representations of the U.S. stock market.

The stock market serves as a critical component of the global economy, providing a platform for companies to raise capital and investors to allocate their funds, seeking a return on their investment. Financial markets, particularly the stock market, have long been a focal point for investors, analysts, and researchers, due to their dynamic and complex nature [4]. Understanding the stock market's highly complex and volatile behavior is the main challenge for investors. Although market volatility and unpredictability pose risks, they also offer opportunities, as market fluctuations create the potential for profits through well-timed trades [5]. To exploit these opportunities, investors use information obtained through forms of analysis such as fundamental analysis, sentiment analysis, and technical analysis.

2.1.1 Fundamental Analysis

Fundamental analysis is a comprehensive approach employed by investors to evaluate and analyze the intrinsic value of a stock, taking into account factors such as economic influences, industry trends, and especially company information. Some of the most relevant company information factors are the company's financial statements, dividends, management team, competitive position, and product demand [6]. Fundamental analysis aims to provide a solid foundation for making investment decisions, by assessing the true worth of a stock to determine whether it is overvalued or undervalued, and therefore whether it is a good sell or buy.

One of the key aspects of fundamental analysis is the examination of financial statements, including the income statement, balance sheet, and cash flow statement. These documents offer insights into a company's revenue, expenses, assets, liabilities, and cash flow over a specific period. Given these insights, it is possible to calculate and build fundamental analysis indicators. These indicators offer investors a comprehensive understanding of a company's financial standing and future potential, expressed in quantitative measures through financial ratios. Using financial ratios, we can compare companies inside the same industry and draw conclusions about the best companies to invest in [7].

2.1.2 Sentiment Analysis

Stock market sentiment analysis is a text mining method that involves the process and evaluation of market participants' sentiments expressed in various textual sources, such as financial news, social media, and analyst reports [8]. The analysis of sentiments in financial markets serves as a valuable tool for investors and traders to comprehend the prevailing mood surrounding a particular financial instrument or even the market as a whole when dealing with macroeconomic event sentiment analysis. As the market is driven by the market's participants, by processing textual sources and categorizing sentiments towards a certain financial instrument as positive, negative, and neutral, we can deduce that there is a higher probability that the financial instrument's stock value will rise, fall, or keep its value, respectively.

A real showcase of the power and importance of sentiment analysis was demonstrated in 2019 when a group of individual retail investors, particularly through online forums like Reddit, initiated a coordinated effort to drive up the price of GameStop shares. This effort led to a substantial and rapid increase in GameStop's stock price, causing a short squeeze, forcing some institutional investors with short positions to cover their losses by purchasing shares at elevated prices, contributing further to the stock's upward momentum. This event demonstrated the impact of online communities on stock prices and, consequently, the impact on investor sentiment [9].

2.1.3 Technical analysis

Technical analysis is a method used by traders and investors to predict future price movements in the stock market. It is based on the notion that historical price data, along with other technical indicators, can provide insights into future market trends. Technical analysis involves the use of various quantitative indicators, such as opening and closing prices, as well as volume value, to analyze the stock market. It is particularly suitable for short-term analysis, focusing on predicting short-term stock price changes rather than long-term fluctuations.

To better understand the market through the scope of technical analysis, it is important to understand the Dow theory, as it is the foundational framework for modern technical analysis [10]. The Dow theory emphasizes the importance of trends in the stock market, and it divides the stock market trend into three types of trends. A primary trend that reflects the overall trend of the market over the long term, with years as a time period unit. A secondary trend is an intermediate trend that represents a counter-trend or correction movement and covers weeks until about three months of time period. The last trend represents the short-term trends and lasts for only some days [10]. Therefore, organizing the market in terms of trends and using price to exhaustively explain the market movements, technical analysis strategies are based on the stock price and the mathematical indicators computed on it [4], the technical indicators.

We will now describe the types of technical indicators while highlighting and selecting a set of relevant technical indicators from reviews of domain experts and prior researches [11] [12] [13] [14], presented and summarized in Table 2.1.

- **Trend Indicators**: such as moving averages, help identify the direction of price movements. Moving averages, in particular, smooth out price fluctuations, providing a clearer picture of the underlying trend.
  - **Simple Moving Average (SMA)**: calculates the average price of an asset over a specified period, smoothing out price data to identify trends.
  - **Exponential Moving Average (EMA)**: weighs price observations over a specified period, assigning more significance to the most recent prices, making this indicator more sensitive to new information and better at capturing short-term trends. This indicator has additional variants, such as the **Double Exponential Moving Average (DEMA)** which further increases the sensitivity to newer information.
  - **Hull Moving Average (HMA)**: aims to address the issue of lag associated with traditional moving averages by using a combination of weighted moving averages and differential smoothing. It is often used to identify the current trend direction and potential trend reversals.
  - **Detrended Price Oscillator (DPO)**: seeks to identify an asset's price cycle, designed to remove the influence of the long-term trend from the current price action to highlight the underlying cycles and shorter-term price swings.
  - **On Balance Volume (OBV)**: measures buying and selling pressure by cumulatively adding volume on up days and subtracting volume on down days, based on whether today's closing price is higher or lower than yesterday's. It helps traders confirm price trends and detect potential reversals by analyzing the relationship between volume flow and price movement.

$$\text{OBV}_{\text{today}} = \begin{cases} \text{OBV}_{\text{yesterday}} + \text{Volume}_{\text{today}}, & \text{if Closing Price}_{\text{today}} > \text{Closing Price}_{\text{yesterday}} \\ \text{OBV}_{\text{yesterday}} - \text{Volume}_{\text{today}}, & \text{if Closing Price}_{\text{today}} < \text{Closing Price}_{\text{yesterday}} \\ \text{OBV}_{\text{yesterday}}, & \text{if Closing Price}_{\text{today}} = \text{Closing Price}_{\text{yesterday}} \end{cases} \quad (2.1)$$

- **Momentum Indicators**: gauge the speed and strength of price movements. These indicators assist traders in identifying overbought or oversold conditions, potentially signaling a reversal.
  - **Accumulation/Distribution Oscillator (ADO)**: provides insights into the strength of price movements and potentially helping to identify overbought or oversold conditions.
  - **Absolute Price Oscillator (APO)**: seeks to identify potential trend reversals and trade setups by comparing two moving averages over time.

  - **Bias Indicator (BI)**: reflects the degree of deviation between the stock price and its moving average in a certain period.
  - **Moving Average Convergence Divergence (MACD)**: is a trend-following momentum indicator that shows the relationship between two moving averages of an asset's price. Traders often look for crossovers between the MACD line and its signal line to identify potential trend changes.
  - **Rate of Change (ROC)**: measures the percentage change in price between the current price and a price from a specified number of periods in the past, helps traders identify the strength and direction of price movements.
  - **Relative Strength Index (RSI)**: measures the speed and change of price movements. It is expressed on a scale of 0 to 100 and is commonly used to identify overbought or oversold conditions.
  - **Stochastic Oscillator (STOCH)**: measures the closing price relative to the high-low range over a set period. It indicates overbought or oversold conditions and is valuable for identifying potential reversal points. Two lines are used in the stochastic process: the %K line and the %D line.
  - **Larry Williams %R (WILLR)**: measures overbought and oversold levels by comparing a stock's closing price to the high-low range over a specified period. It helps traders identify potential entry and exit points by indicating whether an asset is trading near its high or low.

- **Volatility indicators**: measure the rate and magnitude of price fluctuation. High volatility indicates large fluctuations of price, and low volatility indicates a stabilization of the price.
  - **Average True Range (ATR)**: measures market volatility by calculating the average range between the high and low prices over a specified period. A higher ATR indicates higher volatility, influencing risk management decisions.
  - **Bollinger Bands (BBANDS)**: consist of a middle band being an N-period simple moving average and upper and lower bands representing K standard deviations from the moving average. These bands expand and contract based on market volatility, aiding traders in identifying potential trend reversals.
  - **Chaikin Volatility Index (CVI)**: compares the spread between an asset's high and low prices, quantifying volatility as a widening of the range between the high and the low price.
  - **Relative Volatility Index (RVI)**: measures the direction of volatility, providing insight into the strength and potential direction of market trends. It compares the standard deviation of recent price changes to determine whether volatility is increasing or decreasing.

Machine learning algorithms can be used to complement these traditional analysis methods by being able to process and correlate vast amounts of data to capture complex patterns that may not be easily identified using manual techniques, creating a system capable of leveraging complex algorithms to automatically execute trades based on predefined criteria. Machine learning represents a paradigm shift in computational approaches, enabling systems to automatically learn and improve from experience through data. Its ability to identify patterns, make predictions, and uncover hidden insights has found applications across various domains, with finance standing out as a prominent arena for its implementation.

Given the described capabilities of machine learning, it can be used to frame stock market analysis as a classification problem and use classification algorithms to predict the market state, which provides an advantage to performing a more informed decision by tailoring specific strategies, adjusting risk tolerance, and adjusting portfolio allocation accordingly [15]. Due to the stock market's high volatility and influential factors, the resulting data is very non-linear.

Technical Indicators	**Formula**
DEMA	$2 \times EMA_n - EMA(EMA_n)_n$
HMA	$WMA\left(2 \times WMA_{\frac{n}{2}} - WMA_n\right)_{\sqrt{n}}$
DPO	$P_{\left(\frac{n}{2}+1\right)} - SMA_n$
ADO	$\frac{H_t - C_{t-1}}{H_t - L_t}$
APO	$EMA_n - EMA_m, m > n$
BI	$\frac{P_t - MA_n}{MA_n}$
MACD	$EMA(EMA_n - EMA_m)_k, m > n$
ROC	$\frac{P_t - P_{t-n}}{P_{t-n}} \times 100$
RSI	$100 - \frac{100}{1+\frac{AG_n}{AL_n}}$
STOCH %K	$\frac{(C_t - LL_n)}{(HH_n - LL_n)} \times 100$
STOCH %D	$SMA_3(\%K)$
WILLR	$\frac{HH_n - C_t}{HH_n - LLn} \times (-100)$
ATR	$\frac{ATR_{t-1} \times (n-1) + TR}{n}$
CVI	$\frac{HL\_AVG_t - HL\_AVG_n}{HL\_AVG_n} \times 100$
RVI	$100 \times \frac{U_n}{U_n + D_n}$

Table 2.1: Key Literature Review Technical Indicators

Symbol	Description
AG_n	Average Gain over a period n
AL_n	Average Loss over a period n
C_t	Close Price at time t
D_n	Average standard deviation of down days over a specified period n
EMA_n	Exponential Moving Average in n periods
H_t	High at time t
HH_n	Highest High over a period n
HL_AVG_t	High-Low Average: $EMA(H_t - L_t)_n$
L_t	Low at time t
LL_n	Lowest Low over a period n
P_t	Price at time t
P_{t-n}	Price n periods before t
SMA_n	Simple Moving Average in n periods
TR	True Range: $TR = \max\left[(H_t - L_t), \lvert H_t - C_{t-1}\rvert, \lvert L_t - C_{t-1}\rvert\right]$
U_n	Average standard deviation of up days over a specified period n

2.2 Support Vector Machine

Introduced by Vapnik and Cortes in 1995 [16] SVMs are a powerful classification algorithm that excels at handling non-linearity and high-dimensional data, solving not only classification problems but regression tasks as well. The idea of an SVM is to map the input vectors into an N-dimensional space (where N is the number of features), in which a linear decision surface is constructed [16], called a hyperplane, that distinctly classifies data into different classes. This hyperplane is selected by maximizing the margin between the two classes, defined as the distance between the hyperplane and the nearest data point from each class, commonly referred to as support vectors, obtaining the *optimal hyperplane*.

Consider $\{(x_i, y_i)\}, \quad i = 1, \ldots, l, \quad y_i \in \{-1, 1\}, \quad x_i \in \mathbb{R}^d$ to be a set of training data-labeled points in a linear problem. The goal of SVM is to find some hyperplane that separates the positive examples from the negative examples, as shown in Figure 2.1. SVM classifiers are based on the class of hyperplanes (2.2) corresponding to decision functions (2.3).

$$(\mathbf{w} \cdot \mathbf{x}) + b = 0 \quad \text{where} \quad w \in \mathbb{R}^N,\, b \in \mathbb{R} \tag{2.2}$$

$$f(x) = \text{sign}((\mathbf{w} \cdot \mathbf{x}) + b) \tag{2.3}$$

The points $\mathbf{x}$ that are part of the hyperplane satisfy (2.2) where $\mathbf{w}$ is normal to the hyperplane and $\frac{|b|}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin [17] [18]. With the definition of the hyperplane, we can define the margin, which is the sum of the shortest distance from the *separating hyperplane* to the closest positive example and the shortest distance from the *separating hyperplane* to the closest negative example. Therefore, by maximizing the margin, we will find the best *separating hyperplane* solution, the *optimal hyperplane*. Applying the equation (2.2) to the considered set of training data points, we can infer $margin = \frac{2}{\|w\|}$ and a set of inequalities:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i \tag{2.4}$$

By defining a pair of hyperplanes, one that holds positive examples H_1 and one that holds negative examples H_2, we can find the pair of hyperplanes $\{(H_1, H_2)\}$ that give the maximum margin by minimizing $\|w\|^2$, subject to constraints (2.4). The training points that are in $\{(H_1, H_2)\}$ and that, once removed, would change the solution are called support vectors, which carry all relevant information about the classification problem [17] [18].

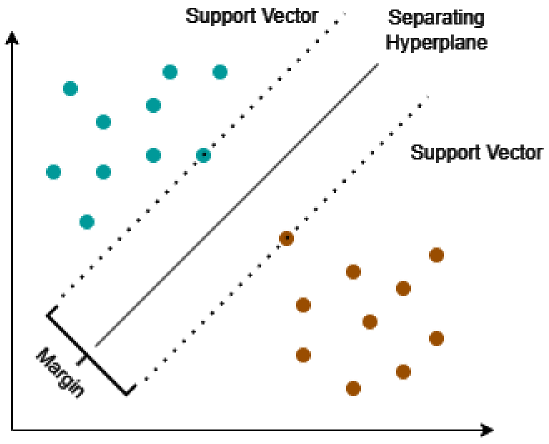


Figure 2.1: Optimal Separating Hyperplane Example

However, note that we have been considering a linear case to formulate and solve the problem, but not every case is a linear case. For that reason, we use a Lagrangian formulation of the problem that allows us to generalize the procedure to the nonlinear case while also providing an easier framework to handle the constraints, as they will be replaced by constraints on the Lagrange multipliers themselves. For each inequality constraints (2.4) we will have a positive Lagrange Multiplier α_i, $i = 1, \ldots, l$, giving the Lagrangian:

$$L_P \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i(\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{2.5}$$

The solution is now found by solving this primal problem, which is minimizing L_P which is a convex quadratic programming problem since the objective function is convex and the points which satisfy the constraints also form a convex set [17]. We can now formulate the *dual* problem, which is to maximize L_P subject to the constraints that the gradient of L_P with respect to w and b vanish, and subject also to the constraint $\alpha_i \geq 0$. Therefore, giving the conditions:

$$w = \sum_i \alpha_i y_i \mathbf{x}_i \tag{2.6}$$

$$\sum_i \alpha_i y_i = 0 \tag{2.7}$$

We obtain the following Lagrangian formulation:

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{2.8}$$

Therefore, training a SVM model comes down to maximizing L_D with respect to α_i and regarding that $\alpha_i \geq 0$, subject to constraint (2.7) with the solution given by (2.6). In this formulation, the support vectors are the training data points in which $\alpha_i > 0$ that lie in $\{(H_1, H_2)\}$.

It is also worth noting that this is not the only approach for training an SVM and finding the *optimal hyperplane*. We can use the first Lagrangian formulation L_P, the primal problem where the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient for the solution to be optimal. The KKT conditions involve constraints on the Lagrange multipliers, and they ensure that the solution satisfies the constraints and optimizes the objective function [17] [19].

Using the Lagrangian formulation, both the training data and the final decision function of the SVM (2.9) appear in the form of dot products between vectors, and that is precisely what lets us generalize to the nonlinear case [18].

$$f(x) = \text{sign}\left(\sum_i \alpha_i y_i (x \cdot x_i) + b\right) \tag{2.9}$$

However, if we apply the above formulations and the described algorithm to non-separable data, we will find no solutions. Therefore, there is a need to extend the formulations to non-separable data. This can be achieved by introducing positive slack variables $\xi_i,\ i = 1, \ldots, l)$ in the constraints, given $\xi_i \geq 0, \forall i$ where an upper bound for the training errors is formulated as $\sum_i \xi_i$. The objective function is accordingly changed to adapt an extra cost for errors,

$$min \quad \frac{\|w\|^2}{2} + C\sum_i \xi_i \tag{2.10}$$

where we introduce a parameter C that is a regularization/penalty parameter that assigns a penalty to errors (2.10).

Support Vector Machines are powerful and widely used in machine learning, however, like any algorithm, there are drawbacks that need to be acknowledged. Solving a SVM requires solving a constrained quadratic programming optimization problem, which has a computational complexity of $\mathcal{O}(|\tau|^3)$, where $(|\tau|)$ is the number of instances [20] in the dataset to train the SVM. This is a very expensive computational algorithm that makes it prohibitive for large-scale datasets, as the algorithm would take unpractical amounts of time to train and deliver a model. Moreover, this algorithm is not only computationally expensive, time-wise, but also memory-wise, as it needs to store the input vectors and associated class labels that grow proportionally to $(|\tau|)$ (the number of instances), becoming also of complexity $\mathcal{O}(|\tau|^3)$. This high computational cost is also a consequence of solving non-linear problems. When solving a non-linear problem, a further extension of the algorithm is needed. The extension for the algorithm that allows SVMs to handle the case where the decision function is not a linear function of the data, is the *kernel trick* [17]. The first step is to map the training data to an Euclidean space H using a mapping Φ:

$$\Phi : \mathbb{R}^d \to H \tag{2.11}$$

Consequently, the training algorithm would only depend on functions of the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. We can then use a kernel function K such that:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{2.12}$$

The following kernel functions are commonly used in the *kernel trick* [21] [22]:

- **:**

  $$K(\mathbf{x}_i, \mathbf{x}_j) = x \cdot x^T \tag{2.13}$$

  This kernel represents the simplest case where no mapping to a higher-dimensional space is performed. It's essentially just the dot product between two vectors, thus it is suitable for linearly separable data.

- **Polynomial Kernel Function**:

  $$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \cdot (\mathbf{x}_i^T \cdot \mathbf{x}_j) + r)^d \tag{2.14}$$

  where γ is the scale factor, r is a constant term and d is the polynomial degree. The polynomial kernel allows for curved boundaries in the input space. The parameters control the shape and complexity of the decision boundary.

- **Radial Basis Function**:

  $$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \cdot \|x - y\|^2) \tag{2.15}$$

  where γ controls the width of the Gaussian kernel. The Radial Basis Function (RBF) kernel is a popular choice for many problems. It can handle non-linear relationships well and can create complex regions within the feature space.

- **Sigmoid Kernel Function**:

  $$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \cdot (x \cdot y) + r) \tag{2.16}$$

  where γ is the scale factor and r is a constant term. This kernel function is similar to the activation function used in neural networks. It can create complex boundaries, but it's less commonly used compared to the RBF kernel.

Using kernel functions is crucial to solving non-linear problems, however, choosing the correct kernel function while tuning its hyperparameters, i.e. model selection, can be a very complex and expensive operation in itself.

2.2.1 Optimization Strategies

Expanding on the complexities of model selection in SVMs, where the kernel trick plays a pivotal role, it is evident that for optimizing SVM performance it is essential to explore various techniques that enhance the model's performance. This process involves a broader set of strategies designed to enhance the model's effectiveness and efficiency. In this section, we will delve into three critical techniques: hyperparameter tuning, instance selection and feature engineering. These methods are essential for refining SVM models, ensuring they not only perform well but also generalize effectively across various datasets. By integrating insights from studies and research, we will explore how each technique contributes to building a more robust and high-performing SVM model.

2.2.1.A Hyperparameter Tuning

The task of tuning and selecting the hyperparameters is seen as an optimization problem whose objective function captures the predictive performance of the model induced by the algorithm. For a given dataset D, the optimal hyperparameter configuration maximizes the performance of this algorithm in D [23]. One of the most common approaches to tuning and selecting the hyperparameters is the Grid-Search algorithm, due to its simplicity and good results. However, this algorithm is an exhaustive search method, and for large datasets, this algorithm becomes computationally inconceivable. To address this issue, a study [23] explored Random Search and found it effective in producing SVM models with predictive accuracy comparable to more complex methods. However, the tests were conducted on small datasets, where larger datasets might diminish its performance. To improve hyperparameter tuning for large datasets, parallel computing strategies have been explored [24] [25]. These techniques significantly reduce computation time while maintaining high classification accuracy, though they require more complex and resource-intensive systems. Another common approach involves using EAs for optimization. A Multi-Objective Evolutionary Algorithm (MOEA) was employed [20] to discover Pareto optimal solutions, which were then combined into a single ensemble. This method integrated instance selection with hyperparameter optimization, resulting in improved accuracy and dataset reduction for SVM classification tasks, showing greater flexibility on larger datasets than Random Search. Therefore, it is clear that hyperparameter tuning is a very necessary step to extract the full potential of the SVM algorithm and enhance its performance, avoiding suboptimal models and overfitting.

However, hyperparameter tuning alone does not resolve all challenges. Machine Learning algorithms and SVMs in particular, struggle with overlapping classes, where data points are not clearly separable by a decision boundary, leading to difficulties in accurate classification and consequently inferior results. To overcome these challenges, feature engineering and data preprocessing are critical in extracting the most relevant information, enabling the SVM model to make more accurate predictions.

2.2.1.B Instance Selection

Instance selection is a data preprocessing technique aims to select a subset of instances from the original dataset that can represent the entire dataset effectively, improving the efficiency and effectiveness of machine learning models. To select a diverse subset of training instances without compromising accuracy, the Training Set Selection algorithm was proposed [26]. By integrating a sub-region division strategy, this approach enhances the quality of the selected subset and guides the evolutionary process toward better solutions. When compared to previously presented work [20], it showed an improvement in terms of accuracy, reduction rate, and running time.

Focusing on direct control of the instance selection rate to enhance accuracy, the authors proposed a novel Cluster Instance Selection algorithm [27]. The idea is to apply a K-means Clustering algorithm to the training instances to select instances from the centers and borders of the clusters. This algorithm combines a focus on decision boundaries by selecting border instances with a focus on ”typical” instances by selecting center instances in a more explicit way than the previous presented work.

Instance selection techniques leverage a trade-off between reducing the dataset and maximizing accuracy, which is not optimal when the main objective is to obtain the best classification accuracy. Moreover, designing generic instance selection techniques does not produce the best results, as these techniques are best used when aligned with the specificity of the problem and with the classification algorithm to be used.

2.2.1.C Feature Engineering

Feature engineering operates on the idea that not all features are relevant and beneficial for a machine learning model. The objective is to select the most relevant features from the original dataset that better capture the underlying patterns. Feature engineering often leverages feature correlation, favoring less correlated features for their additional information value. A local search strategy was integrated into a Particle Swarm Optimization (PSO) algorithm [28], utilizing Pearson correlation to enhance accuracy and reduce feature subset size. Similarly, Hamming distance was incorporated into a PSO algorithm [29] to evaluate diversity among feature subsets, guiding the selection process toward optimal subsets with minimal redundancy. Additionally, a quartile-based normalization technique was employed to ensure balanced gene priority in the feature selection process. Some approaches tackle feature engineering and hyperparameter tuning in a single solution, as both techniques are correlated. A Genetic Algorithm (GA) was used to optimize both classification accuracy and the number of features and their cost [30]. This approach enhances SVM performance with an optimal feature subset and hyperparameters. However, integrating concepts like Hamming distance [29], could further optimize the GA by improving computational efficiency and avoiding local maxima. Further exploration of EA processes is discussed in the next section.

2.3 Evolutionary Algorithms

As discussed in the previous section, several studies have demonstrated the value of EAs in addressing the challenges of feature engineering and hyperparameter tuning. Evolutionary Algorithms are highly effective in these domains due to their ability to explore vast search spaces and optimize complex multi-objective problems. These algorithms have the ability to iteratively refine feature subsets and hyperparameter configurations, leading to models that are both robust and efficient. Their adaptability allows them to handle the intricacies of high-dimensional data, reducing the dimensionality while maintaining or even enhancing predictive accuracy [31]. Consequently, EAs have become a powerful tool in the optimization of machine learning models, offering a flexible and scalable approach to fine-tuning models, producing better results.

Evolutionary Algorithms are population-based, optimization and search algorithms that, in some sense, are inspired by the processes of natural selection and evolution [32]. These algorithms start with an initial population (most often randomized) of individuals that represent a candidate solution. For each individual, it is assigned a fitness value obtained from a fitness function that reflects the quality of the individual as a solution. The fitness function is a critical component that evaluates the quality of each individual based on how well the solution satisfies the optimization criteria. Upon obtaining the fitness values, genetic operations are used in the population.

2.3.1 Genetic Operators

1. **Selection**: Determines which individuals from the current population will be chosen as parents for the next generation based on their fitness. There are several types of selection operators, each with its own strategy for choosing individuals:

   - **Roulette Wheel Selection**: Individuals are selected based on their fitness value. The higher the fitness, the higher the probability of being selected.

- **Tournament Selection**: A set number of individuals are randomly chosen from the population, and the one with the highest fitness in this group is selected.
- **Rank Selection**: Individuals are selected based on their rank in the population, not their absolute fitness value. The population is sorted by fitness, and selection probability is assigned based on this ranking.

There is also an important concept within the selection operator, which is Elitism, that is a rule to guarantee that the best performing individuals (the elites) are guaranteed to survive to the next generation.

2. **Crossover**: Combines genetic material (features) from two parent individuals to create one or more offspring. There are several types of crossover operators, such as:

   - **Single-Point Crossover**: The chromosomes from two parents are split at a randomly selected point, and the subparts are swapped to create offspring
   - **Multi-Point Crossover**: Similar to single-point crossover, but with multiple points of split and swap
   - **Uniform Crossover**: Each gene in the offspring is independently chosen from one of the corresponding genes of the parent chromosomes.

3. **Mutation**: Introduces small random changes to the genetic information of individuals, helping explore new regions of the search space. There are several types of mutation operators, such as:

   - **Bit Flip Mutation**: In binary representation, bits in a chromosome are flipped (0 to 1 or 1 to 0) with a given mutation probability.
   - **Random Resetting**: Used in integer or real-valued representations, where a gene's value is replaced with a random value within its predefined range.
   - **Swap Mutation**: Two genes are randomly selected, and their values are swapped.
   - **Scramble Mutation**: A subset of genes is chosen, and their values are scrambled or shuffled randomly.

Upon completion of the genetic operations, we again calculate the fitness values of the individuals in the population, describing a control flow (see Figure 2.2) and marking a cycle termed generation [32]. Applying genetic operations to the population leads to the evolution of solutions over successive generations.

Before every new generation, it is checked if the algorithm has reached a termination criterion to decide if it will generate a new generation or present the current generation as a solution. The choice

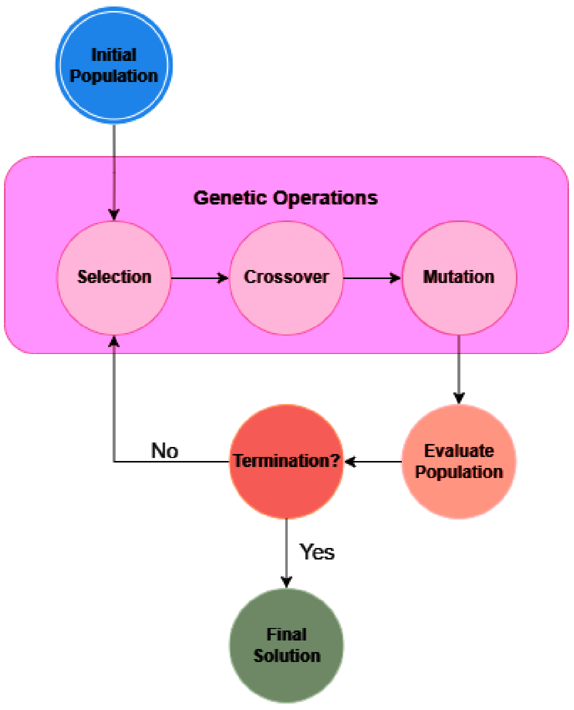


Figure 2.2: Control Flow of an Evolutionary Algorithm

of termination criteria depends on the nature of the optimization problem, available computational resources, and the desired characteristics of the algorithm's behavior. Implementing appropriate termination criteria is crucial for balancing the trade-off between computational efficiency and the likelihood of finding high-quality solutions.

2.3.2 Family of Evolutionary Algorithms

The Evolutionary Algorithms subdivide into other categories of Evolutionary Algorithms with specific specializations [32] [33]:

- **Evolutionary Strategies**: Focuses on mutation operations with self-adaptive mutation rates to seek the solution encoded as a vector of real numbers.
- **Evolutionary Programming**: Has a fixed representation and encoding of the solution, allowing its numerical parameters to evolve.
- **Genetic Programming**: Generates the solution encoded in the form of a computer program. It employs a tree-based representation of computer programs and applies genetic operations such as mutation and crossover to evolve the programs to perform specific tasks.

- **Genetic Algorithm**: Focuses on mutation and recombination operations to seek the solution encoded in the form of strings and numbers.

Delving more in-depth into the Genetic Algorithms, being a class of Evolutionary Algorithms, GAs follow the EAs control flow (see Figure 2.2) and are used as a heuristic search method based on ”survival of the fittest” [34].

2.3.3 Genetic Algorithm Components

GAs are composed of specific key elements (see Figure 2.3) that integrate the algorithm to guide the search for optimal solutions in the search space:

- **Population**: Consists of a set of individuals, each representing a potential solution to the problem. These individuals are often encoded as strings known as chromosomes.

- **Chromosomes**: Comprises a string of data that encodes a candidate solution. Its encoding format can vary depending on the problem, e.g. Binary Encoding, Real-valued Encoding, Permutation Encoding. Chromosomes are composed of genes, and their length can be fixed or variable.

- **Genes**: It is the basic unit of information that represents a specific trait or parameter of the solution, contributing to the overall fitness of the chromosome.

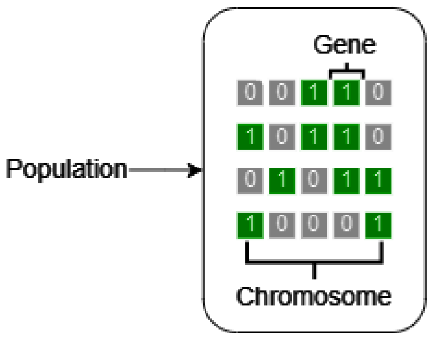


Figure 2.3: Genetic Algorithm Components

Employing the control flow of EAs (see Figure 2.2), after creating the initial population through some generalization method or randomized, the genetic operators are used, where the selection operator selects the chromosomes based on their fitness value, the crossover operator executes at the gene level of the selected chromosomes to combine genes, creating offspring chromosomes, and the mutation operator also executes at the gene level, introducing diversity by changing the value of genes. The fitness values are again calculated and it is checked if the algorithm has reached the stopping criterion.

2.3.4 Evolutionary Algorithm Refinements

There are also very valuable extensions to Evolutionary Algorithms that increase their performance and present an improvement to the initial concept of the algorithm [33]. There are two key extensions:

- **Memetic Algorithms**: This extension combines local search heuristics with EA. It allows individuals within the EA to adapt and change their properties, similar to the process of adaption in nature. The weight between local search and the EA method can be shifted randomly, enabling effective multi-start search of the local search heuristic.
- **Distributed Evolutionary Algorithms**: This extension involves the distribution of the EA process across multiple computing nodes or systems. By distributing the EA, it becomes possible to handle larger problem spaces and computational loads, leading to improved scalability and efficiency.

Due to the EAs extension and refinement possibilities, there are already very popular refined and hybridized techniques, such as Genetic Swarm Optimization, Particle-Swarm-Optimization, Self-Adaptive Differential Evolution, Immune Self-Adaptive Differential Evolution and Multi-Objective Evolutionary Algorithms, like the Nondominated Sorting Genetic Algorithm II, that aim to address specific challenges and improve the effectiveness of evolutionary algorithms in solving more complex and specific real-life applications.

2.3.5 Application of Evolutionary Algorithms

As shown in the previous section, Evolutionary Algorithms are a popular choice of algorithm to optimize and enhance SVM's performance. As Evolutionary Algorithms are a large family of algorithms with several particularities, arises the question on what should be the EA employed for each problem. Building on the aforementioned studies [20] [30], Evolutionary Algorithms were used to optimize Support Vector Machine's hyperparameters, with the former employing a multi-objective algorithm and the latter a single objective algorithm. This raises the question of whether single-objective or multi-objective algorithms are more effective for optimization. Given the conflict between maximizing accuracy and minimizing training set size, a multi-objective algorithm was preferred for handling these competing objectives [26].

The objective function is also a crucial component of EAs, and various approaches to its design exist. Researches often propose hybrid GA-SVM models for feature engineering and hyperparameter optimization [35] [30]. The first research proposes an objective function based on the SVM classifier's accuracy, whereas the second uses a weighted formula to simultaneously maximize accuracy and minimize the number of features and their cost. Both researches evaluated their solutions on real-world datasets from the UCI database, and by direct comparison, we can infer that using the weighted formula as the objective function was an improvement to using the SVM classifier's accuracy regarding the final model's accuracy.

Concerning the application of multi-objective approaches, a MOEA is utilized [36] to separately optimize accuracy, sensitivity, and specificity, resulting in three distinct solutions tailored for each objective. However, this approach lacks a final convergence step for unifying the solutions. Conversely, a MOEA for feature selection and weighting is proposed, incorporating a final convergence step to determine the optimal compromise solution [37]. This is achieved by assigning fuzzy membership values to each solution, reflecting the degree of satisfaction of each objective, and selecting the solution with the highest total normalized membership value. Both were evaluated on the breast cancer dataset, and therefore we can use the results on this dataset to compare them. By doing this, it is observable that the second mentioned study achieved greater accuracy results, and so we can infer that the final convergence step is a very probable improvement.

Genetic algorithms have several parameters, such as population size, mutation rates, and crossover rates. Researches have been done on how these parameters affect the solutions provided by GAs and how to optimize these parameters. Regarding the mutation and crossover rates, a dynamic fitness function for a GA was developed [38] to address the interplay between crossover and mutation rates using Hamming distance, similar to aforementioned research [29]. The Hamming distance measures the rate of change between generations, enabling a dynamic fitness function that outperforms static rates in dynamic environments. Similarly, a Dynamic Genetic Algorithm (DGA) was introduced [39] with adaptive crossover and mutation rates, showing better performance than standard GAs with fixed rates. Both studies highlight that dynamic approaches improve performance, adaptability, and generalization in Genetic Algorithms.

Research on population size and initial population formulation in EAs explores how these factors influence optimization performance, with studies examining the balance between diversity and convergence speed to improve algorithmic efficiency and outcomes. The significance of population size is emphasized [40], demonstrating that a larger population does not always result in improved performance. A self-regulating population size is proposed [41] through the Self-Regulated Population size Evolutionary Algorithm, which adjusts based on the genetic diversity within the population. In this algorithm, the generated offspring does not replace the parents, but instead are added to the population while other individuals are removed via an age/lifetime process.

To address the generation of individuals for the initial population in an EA, the center of mass was proposed as a metric to measure diversity and detect any unwanted bias towards specific regions of the search space [42], thus highlighting potential issues with unsupervised initial populations, such as premature convergence. Another approach to generate the initial population is to use *quasi-random* sequences known for their good distribution properties [43]. Both approaches tackle the need for diversity on the initial population and demonstrate positive results in the performance of the Genetic Algorithm.

2.4 Ensemble Learning

Ensemble learning has gained significant popularity in the past few decades due to its ability to enhance classification performance [44]. The ensemble learning technique is, by definition, a composition of learners with the objective of solving a learning problem (see Figure 2.4). This approach is based on the idea that combining multiple models, each with its own strengths and weaknesses, can lead to more accurate and robust predictions [45].

2.4.1 Ensemble Learning Techniques

Ensemble learning encompasses various techniques, including bagging, boosting and stacking, each with its own unique approach to leveraging the strengths of multiple models [46].

- **Bagging**, or bootstrap aggregating, is a technique where multiple models are trained on different subsets of the training data and their predictions are combined using majority voting or averaging. Given a dataset with a number of m samples, the idea is to randomly pick samples from the original dataset m times building a sample dataset while keeping the copied samples in the original dataset. This whole process is repeated T times, building T sample datasets on which the base learners are trained. After training the base learners, they are combined through a simple voting method [45] [47]. The Bagging method is an efficient ensemble learning method, as its complexity increases with the complexity of the combination method, which is a voting method, and therefore its complexity is low. Because it uses a random selection of samples, the samples from the original dataset that were not used can be incorporated into the model as a validation set, though using randomness instead of a deterministic approach raises evident issues [45].

- **Boosting**, on the other hand, starts by training a base learner, and it will adjust iteratively the distribution of the training samples according to the results obtained by the base learner in order to give a higher importance to incorrectly classified classes, such that it converts a weak learner into a strong learner [45]. This process ends until the base learners reach a predefined value T. A prominent Boosting algorithm is AdaBoost [48] that focuses on minimizing the classification errors by maintaining a set of weights over the training samples and adjusting the weights over the Boosting iterations [49]. The key idea when adjusting the weights is to increase the weight of misclassified training samples while reducing the weight of correctly classified training samples. This grants the algorithm good performance and generalization capability, however, because the algorithm gives high importance to misclassified training samples over the iterations, it increases the probability of overfitting, making it very sensitive to noisy data [49]. XGBoost is another important Boosting algorithm based on Gradient-Boosted Decision Trees, focusing more on performance and speed, it includes Decision Tree techniques that improve the algorithm. It uses a regularization technique to avoid overfitting and a parallelization technique for feature selection for each tree. XGBoost first trains a tree on a training set and sample label, then uses this tree to predict the training set to obtain the predicted value of each sample, then subtracts the predicted value of the sample from the label to obtain the residual, that will be fit when training the second tree [50].

- **Stacking** is another popular ensemble technique where multiple models are trained, and their predictions are used as inputs to a metamodel that combines them to make final predictions with higher accuracy. By integrating the predictions of these base models as input features for a metamodel, stacking facilitates the synthesis of varied perspectives and predictive patterns. This process enables the metamodel to identify and exploit the complementary strengths of the base models while also mitigating their individual weaknesses. Moreover, stacking can effectively handle various challenges associated with prediction tasks, such as overlapping between different classes, class imbalance, the presence of noise, and non-linear boundaries between classes [51].

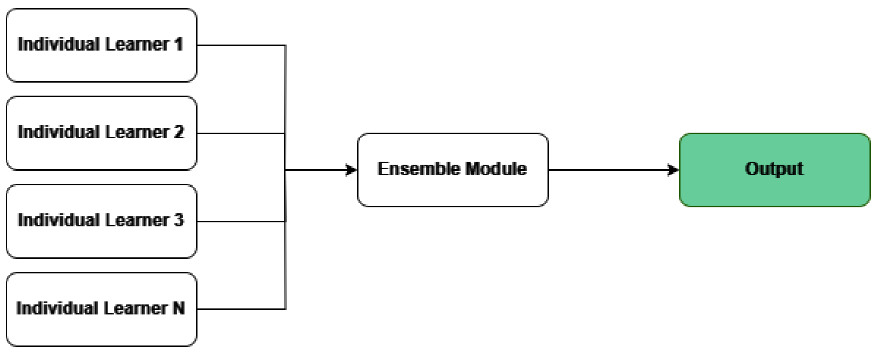


Figure 2.4: Control Flow of an Ensemble Model

2.4.2 Methods for Combining Ensemble Learners

Moreover, ensemble learning methods have different strategies when combining the individual learners to obtain the final output model. As aforementioned, Bagging commonly uses a voting method to combine the individual learners. Voting is a combination strategy where each individual learner can vote on a class label and the final output label will be chosen based on the voting. Voting itself can be divided into different strategies when it comes to accepting the final label of the class. Majority Voting outputs the class label that receives more than half of the votes, or rejects if none of the classes receive more than half of the votes. Plurality Voting outputs the class label with more votes and selects one randomly in a tie case. Weighted Voting uses the same idea as Plurality Voting, but assigns weights to each learner.

Another combination strategy is Averaging, where each learner attributes a probability for each class label. In a simple averaging approach, the final output class label is based on an average of the probabilities. In a weighted averaging, weights are assigned to each learner and an average is performed with weighted probabilities. Combining by learning is another prominent combining strategy, and it lays the foundation for the Stacking ensemble method. This combining strategy uses a meta-learner to combine the first-level learners, the individual learners [45].

2.4.3 Individual Learners

An ensemble can be of two types, depending on the individual learners that compose the ensemble. The ensemble can be homogeneous if all the individual learners are of the same type; e.g., an SVM ensemble contains only SVMs as individual learners, which in the case of a homogeneous ensemble are called base learners. If the individual learners are different from each other, the ensemble is heterogeneous, and the individual learners are called component learners [45]. The performance of the ensemble is intrinsically dependent on the individual accuracy of its individual learners and the diversity among them. The individual accuracy of an individual learner is the percentage of instances that are correctly classified by the learner in a given dataset, while diversity refers to the difference between the predictions of the individual learners [44]. The individual learner's accuracy and their diversity are interconnected in achieving a well-performing ensemble as the ensemble will inherit the accuracy of its learners and the diversity between them compensates for the individual weaknesses, reducing the risk of overfitting to specific patterns in the training data.

2.4.4 Application of Ensemble Learning

Bagging and Boosting are two of the most widely used ensemble learning techniques, each with its distinct approach to improving the performance of machine learning models, like SVMs. Bagging aims to reduce variance by training multiple models on different subsets of the data and then aggregating their predictions. Research was conveyed on the use of the Bagging ensemble learning technique on an ensemble of SVMs to enhance classification accuracy [52]. The authors investigated various aggregation methods, including majority voting, Least Squares Estimation based weighting, and double-layer hierarchical combining. Their findings demonstrated that Bagging significantly outperformed a single SVM in terms of accuracy, with the double-layer hierarchical combining technique yielding the best overall performance. This research underscores the importance of the aggregation method in maximizing the benefits of Bagging.

On the other hand, Boosting focuses on reducing bias by sequentially training a series of weak learners, each one correcting the mistakes of its predecessor. A Boosted SVM-based Ensemble Classifier was introduced for sentiment analysis of online reviews [53]. The approach involved weakening the SVM by reducing the number of features and lowering the regularization parameter, making it suitable for the Boosting process. The Adaboost algorithm was then applied to create an ensemble, using the size of the weights assigned by Adaboost as indicators of which data points were likely to become support vectors. This method resulted in a significant improvement in accuracy over a single SVM classifier, highlighting Boosting's capability to enhance model performance in complex classification tasks.

When comparing Bagging and Boosting, both techniques have proven effective in improving SVM performance, but they do so in different ways. Bagging is particularly useful for reducing variance and increasing the stability of models prone to overfitting [52], especially when an appropriate aggregation method is chosen. Conversely, Boosting, excels in tasks that require high accuracy by focusing on hard-to-classify instances [53], though it demands careful management of the base learners and their weights to avoid overfitting and complexity.

2.4.4.A Addressing Data Complexity

Ensemble Learning techniques are frequently applied to classification problems to enhance model performance and address the drawbacks of individual classification algorithms. Most complex classification problems have complex patterns associated with high data complexity, such as the stock market. To effectively manage the challenges posed by data complexity, Dynamic Ensemble Models have emerged as a solution that adapts to the evolving nature of complex patterns within the data.

A dynamic ensemble model leveraging SVM hyperplane distances is proposed to address unbalanced datasets and data complexity [54]. The approach begins with K-means clustering to create subsets from the larger class, followed by the calculation of SVM hyperplane distances using a linear kernel to determine signed distances. These local classification results are then integrated using Linear Regression. Building on this research, the solution was extended to support non-linear kernels and redefines the problem decomposition by incorporating a Minimum Spanning Tree (MST) approach with a k-NN algorithm [46]. The final prediction is generated through a dynamic ensemble technique that combines the component classifiers' outputs, taking into account both dataset characteristics and the proximity of the test instance to the separating plane in each sub-problem. In contrast, a strategy to address unbalanced datasets was adopted by constructing an SVM classification tree through data partitioning with Locally Linear Embedding [55]. The tree is developed until the child nodes consist entirely of homogenous classification groups, with each internal node representing an SVM classifier. However, this method has limitations when dealing with complex datasets, as the increase in membership group size can significantly reduce classification accuracy. Consequently, this approach proves less effective in addressing data complexity compared to the more adaptive and scalable solutions proposed in the other two works.

2.5 Stock Market Application

This section explores the application of SVMs in stock market prediction, focusing on strategies that enhance SVM performance in this context. A study introduced an SVM-based approach for predicting stock market trends using 53 technical indicators as features [56]. The study emphasizes the importance of feature selection by employing a correlation-based filter and ranking system. This method evaluates the relevance of each feature, selecting a subset that is highly correlated with the target outcome while remaining uncorrelated with each other, similar to the feature selection strategy in [28].

In a related study, [57] utilized a GA for feature selection, where the fitness function is based on the classification accuracy of the SVM model with the feature set represented by the chromosome. Both studies underscore the critical role of feature selection in improving stock market prediction accuracy. Ideally, combining hyperparameter tuning with feature selection would yield optimal accuracy in such predictive models.

Further advancing this field, an ensemble method was designed for stock market prediction [5]. The solution employs a modified majority voting ensemble to maximize final accuracy, integrating hyperparameter tuning via a grid-search algorithm with feature selection using a filter-based approach. While this method outperforms the baseline Buy & Hold strategy, individual classifiers, and a standard majority voting ensemble, it faces challenges such as the high computational cost of grid-search hyperparameter tuning and the potential accuracy loss associated with filter-based feature selection methods.

2.5.1 Label Design

When applying machine learning to stock market prediction as a classification task, one critical yet often overlooked step is label design. Unlike other machine learning problems, creating labels for stock market prediction is far from straightforward. In this context, labels typically represent buy, sell, or hold signals, the target outputs that the model aims to predict. Among the presented studies on stock market prediction, only one specifically addresses and proposes a label design method. This study employs an up-down labeling method based on daily price change thresholds to predict future trends. Exceeding the upper threshold signals an uptrend, breaking the lower threshold indicates a downtrend, and movements within the thresholds are considered insignificant price changes [5]. Another approach to an up-down labeling method is to use a crossover of a moving averages to determine the future trend. In this approach, the difference between a short-term moving average and a long-term moving average signals the future trend: a short-term average higher than the long-term average suggests an uptrend, while a short-term average lower than the long-term average indicates a downtrend [58]. Research focused on label design for stock market prediction introduces a more advanced alternative to traditional up-down labeling: the N-Period Min-Max (NPMM) labeling method, which focuses on identifying minimum and maximum values over a specified period [59]. This study underscores the importance of precise data labeling in developing effective trading systems and demonstrates that NPMM labeling outperforms traditional methods like the up-down approach in stock market prediction. However, due to the nature of the Min-Max algorithm, when labeling for long-trend strategies there is a substantial loss of instances that are neither minimum nor maximum and that therefore have no label.

Research	Year	Algorithm	Evaluation	Contributions
[5]	2022	HEC	2014 - 2020 Stock Exchange of Thailand (SET) 20 leading stocks	Heterogeneous Ensemble Classifier. Feature Selection with Grid-Search for Hyperparameter tuning
[30]	2005	GA-SVM	11 Classification Datasets (UCI repository)	Simultaneous Feature Selection and Hyperparameter Selection
[37]	2015	DEMOFS	20 Classification Datasets (UCI & LIBSVM repository)	Optimize the Inter-Class Separability and Intra-Class Nearness of Data Points using feature selection and weighting. Considering Conflicting Objectives in a MOEA.
[38]	1998	GA	Meta-GA	Dynamic Match Fitness Function based on Hamming Distance to provide a Rate of Change.
[39]	2000	DGA	14 Function Problems. 0/1 Knapsack Problem.	Dynamic Crossover and Mutation Ratios. Average Progress Value generated by Crossover Operation to evaluate its effectiveness.
[43]	2004	GA	52 global optimization problems	Importance of initial population. Uses *quasi-random* sequences in the initial population.
[46]	2023	MST-2NN-SVM-DESCC	28 Classification Datasets (KEEL repository)	Use of MST and 2-NN to create subproblems. Dynamic Ensemble of Component Classifiers (DESCC) based on the weighted distance of the test data and decision boundaries.
[57]	2011	GA-SVM	1993-2000 Swedish Stock Index (SXGE), Ericsson and Volvo	Use of GA-SVM for Stock Market Forecasting. Technical Indicators Feature Selection using GA. Feature Linear Mapping
[59]	2023	NPMM-XGBoost	2009-2020 Top 100 NASDAQ	Importance of labelling method in stock market. Introduces N-Period Min-Max (NPMM)

Table 2.2: Summary of Key Research Contributions

3 Architecture

Contents

This section provides an in-depth review of the structural design and specifications of the proposed solution. This includes detailing the various components, their interactions, and the underlying principles guiding the design choices. The architecture serves as the blueprint for the solution's implementation, ensuring that all components work together to achieve the desired functionality, which is to predict the stock market.

We begin by presenting an overview of the architecture of the solution, followed by a detailed explanation of each core component. We will also discuss the data flow between components, the technologies and tools employed, and the rationale behind key design decisions.

3.1 Architecture Overview

The proposed solution is the Dynamic Trend Ensemble Dual Support Vector Machine (DTE-2SVM), draws from Dow's theory of market trends [10] by dividing the market into long-term and short-term trends using different time frames. Each trend is modeled using specialized Support Vector Machines (SVMs) tailored to capture the unique dynamics of each trend. To further enhance the model's predictive accuracy, a novel data labeling technique based on the N-Period Min-Max (NPMM) is employed [59]. This method identifies critical market points, such as local minima and maxima, to define a trend between the critical points in each period, which is then used to label the remaining points that lead to the next critical point. The predictions from these models are then dynamically combined using an ensemble strategy that leverages an SMA as a trend technical indicator, in order to choose the model's prediction that better reflects the market's current behavior.

The DTE-2SVM's architecture adopts a widely-used multi-layered structure, consisting of three distinct layers: the data layer, the processing layer, and the interface layer (see Figure 3.1). The multi-layered architecture of the proposed system enhances scalability and maintainability by employing a modular approach. This design enables independent development and modification of individual components, allowing for seamless integration of improvements while simplifying issue resolution and bottleneck identification.

- **Data Layer**: This layer is responsible for collecting, storing and preprocessing historical stock market data. The data collection process involves gathering financial data, such as the high, low, volume, close and open price for each day, over a period of time. The collected data is then used to build the technical indicators, which will serve as features for the machine learning algorithms. Two distinct feature sets and labels are built for a long and short trend strategy, resulting in two datasets. The data is then stored, providing easy access for the subsequent layers.

- **Processing Layer**: This layer is the core of the system, where the main computational work occurs. The key components of this layer comprise hyperparameter optimization, model training and ensemble. The hyperparameter optimization process focuses on tuning both SVM's hyperparameters C and γ to achieve optimal performance. The model training process houses the implementation and fit of the SVMs. The ensemble process dynamically combines the predictions from the two SVMs to generate a final prediction.

- **Interface Layer**: This layer is responsible for making the system's results easily accessible, understandable, and actionable for end-users. It handles the display and visualization of the processed data and results, transforming raw outputs into user-friendly formats such as charts and tables. The user interacts with the program primarily through the command prompt, as the primary objective of this project is to demonstrate a functional trading system.

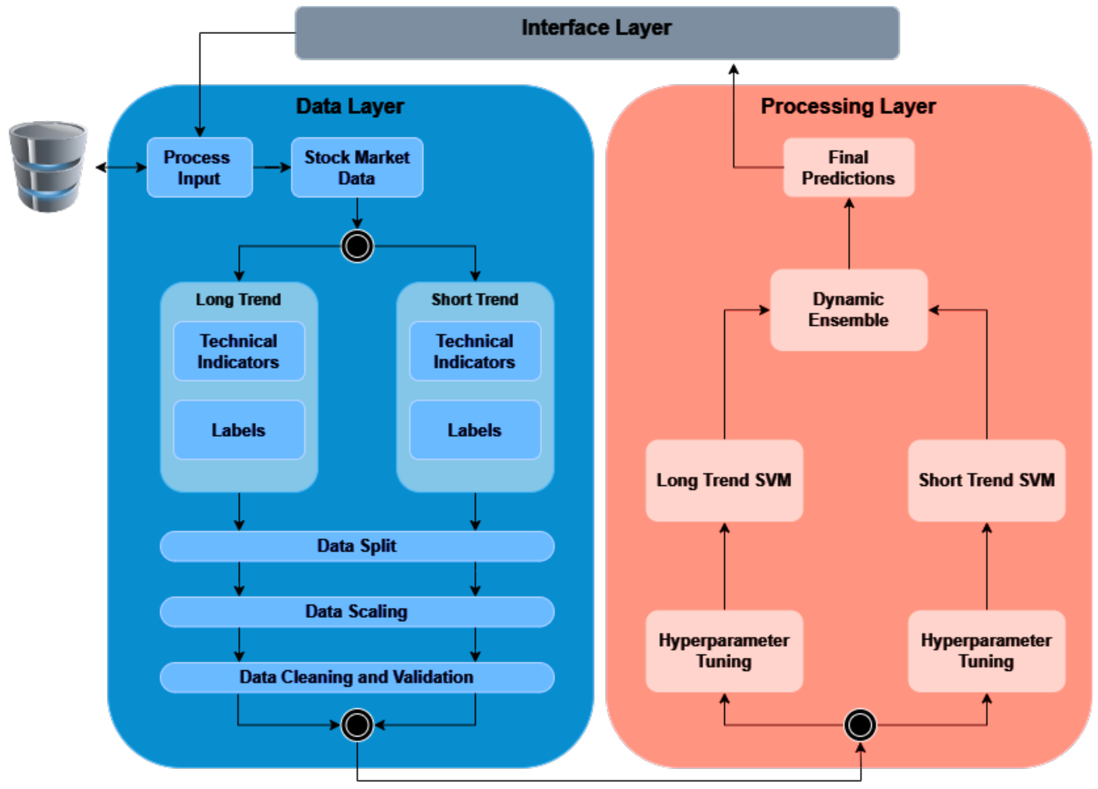


Figure 3.1: Architecture Overview

3.2 Data Layer

The Data Layer serves as the foundational entry point of the entire system, where raw data is collected, stored, and preprocessed. At its core, the Data Layer is responsible for data acquisition, which involves collecting financial data from the stock market.

3.2.1 Data Collection and Storage Framework

The data collection process is done through the use of the python module *yfinance* which is an open-source tool that uses Yahoo's publicly available API's to retrieve stock market information, including stock prices, historical data and financial statements [60]. For this research, the purpose is to track and predict the S&P 500 index therefore we extract all the available S&P 500 data using the *yfinance* module through the *ˆGSPC* ticker that represents the S&P 500 index. The raw data is then stored using a Comma Separated Values (CSV) file providing quick and easy access to the data by being cached locally meaning that calls to the API are only made if the desired data is not yet in local storage. To efficiently export, import, and manipulate data from CSV files, the system leverages *pandas*, a widely-used Python library for data manipulation and analysis [61]. The *pandas* library offers robust data structures like *DataFrame* and *Series*, specifically designed to simplify working with structured data, such as tabular data found in CSV files. The tabular data organization is also a perfect fit for the raw data extracted. The extracted data includes key financial metrics for each trading day, such as the high, low, open, and close prices, as well as the trading volume. Therefore, using a tabular data organization, each data entry will consist of a row where the index is the date, and the columns will be the financial metrics: open price, high of the day, low of the day, close price and volume. As a result, the system uses the *DataFrame* data structure creating a perfect synergy between data storage, data collecting and data manipulating given the data we are working with.

3.2.2 Data Labeling Strategy

With the raw data in hand, the next step is to construct the labels that will serve as the foundation for training and evaluating the model. In supervised learning, these labels are critical, as they define the target outcomes that the models are designed to predict.

In the stock market context, designing labels for machine learning models involves creating buy/sell signals. The goal is for the model to forecast price movements, indicating whether to buy/hold in an anticipated upward trend or sell/stay out during a predicted downturn. Effective signal design is challenging due to market volatility, where frequent fluctuations and random events, or ”market noise,” can obscure true trends and produce false signals. Additionally, the time horizon affects label effectiveness; short-term signals often perform poorly for long-term strategies, and vice versa. While short-term strategies may offer higher returns, they involve greater volatility and noise, making predictions more complex and requiring a careful balance across time frames.

To address these challenges, the system divides the market into long and short trends, creating specialized models for each. This approach simplifies label construction by assigning separate labels for each trend, avoiding the need to merge short- and long-trend movements into a single label. The system uses binary labeling, where 0 signals a sell/stay out action and 1 indicates a buy/hold action.

3.2.2.A N-Period Min-Max Algorithm

A different labeling strategy was developed, based on the N-Period Min-Max (NPMM) algorithm [59] discussed previously (see Section 2.5.1). Likewise, the proposed labeling strategy identifies the minimum and maximum close price values over a period N. However, instead of only labeling the data points that correspond to the identified minimum and maximum values, the labeling strategy used in this system uses the identified values to assign a continuous label. The labeling strategy developed follows the following steps:

1. The N-Period Min-Max is applied to the stock market data close prices, and for each period N a Minima and a Maxima are located (see Figure 3.2).

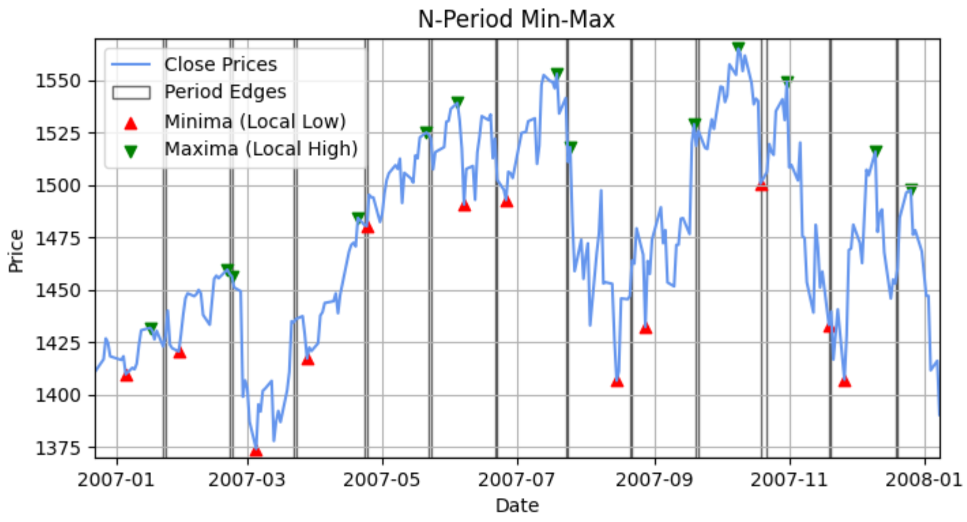


Figure 3.2: Data Labeling Process: NPMM. Short Trend Example 2007

2. Labels between every located Minima and a Maxima are assigned (see Figure 3.3), according to the following criteria:

$$\text{Label}_t = \begin{cases} 0(Sell), & \text{if the first located Min-Max is a Maxima} \\ 1(Buy), & \text{if the first located Min-Max is a Minima} \end{cases} \tag{3.1}$$

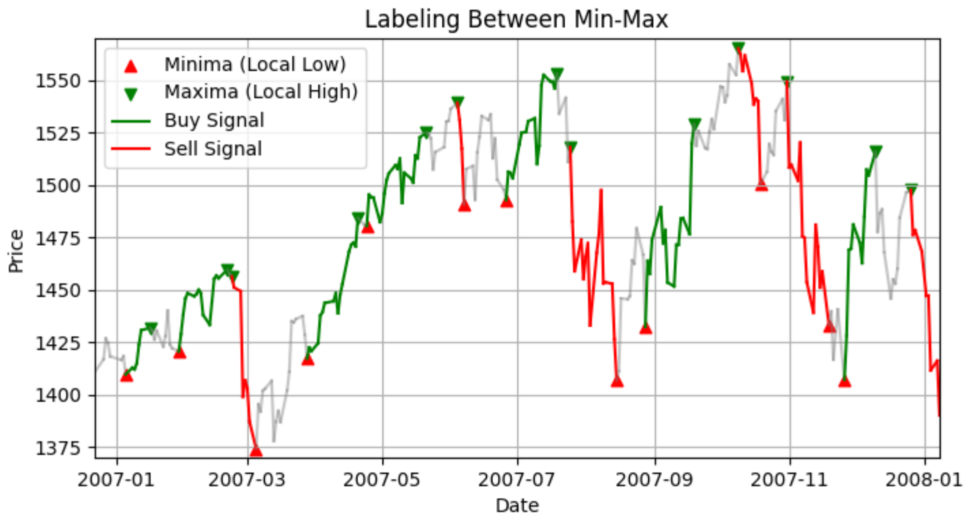


Figure 3.3: Data Labeling Process: Labels between Min-Max. Short Trend Example 2007

3. The remaining unlabeled data, have labels assigned (see Figure 3.4), according to the following criteria:

$$\text{Label}_t = \begin{cases} 0(Sell), & \text{if the last labeled data was } 0(Sell) \text{ and the next labeled data is also } 0(Sell) \\ 1(Buy), & \text{if the last labeled data was } 1(Buy) \text{ and the next labeled data is also } 1(Buy) \\ 0(Sell), & \text{if the last labeled data was } 1(Buy) \text{ and the next labeled data is } 0(Sell) \\ 1(Buy), & \text{if the last labeled data was } 0(Sell) \text{ and the next labeled data is } 1(Buy) \end{cases} \tag{3.2}$$

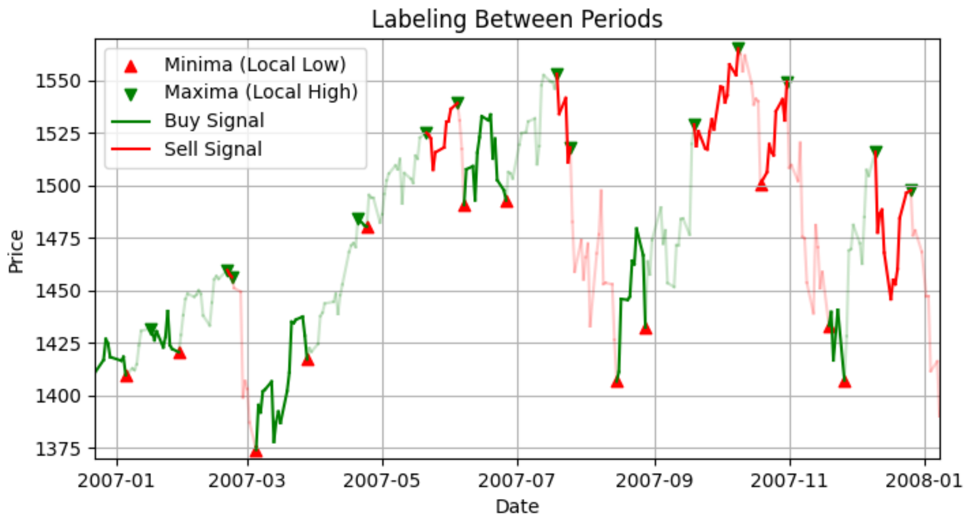


Figure 3.4: Data Labeling Process: Labels Propagation. Short Trend Example 2007

4. The data labeling process is finished, returning the final result (see Figure 3.5).

This approach allows the system to identify trend reversals by continuously monitoring and labeling the progression towards these critical points. The implemented labeling strategy is used to label both the long trend and the short trend. The difference in labeling the long and short trends lies in the chosen N period for each case (see Table 3.1).

Model	NPMM Period
Long Trend	$N = 126$
Short Trend	$N = 21$

Table 3.1: Period used in NPMM for each trend strategy

The long trend strategy focuses on capturing broader market movements by identifying key inflection points, minimums and maximums, over a six-month trading period. This approach aligns with the objective of capturing sustained, long-term trends, where more gradual price shifts are indicative of meaningful market direction. By using a larger time window, the long trend model minimizes the impact of short-term volatility and market noise, allowing for a clearer signal of overall market momentum.

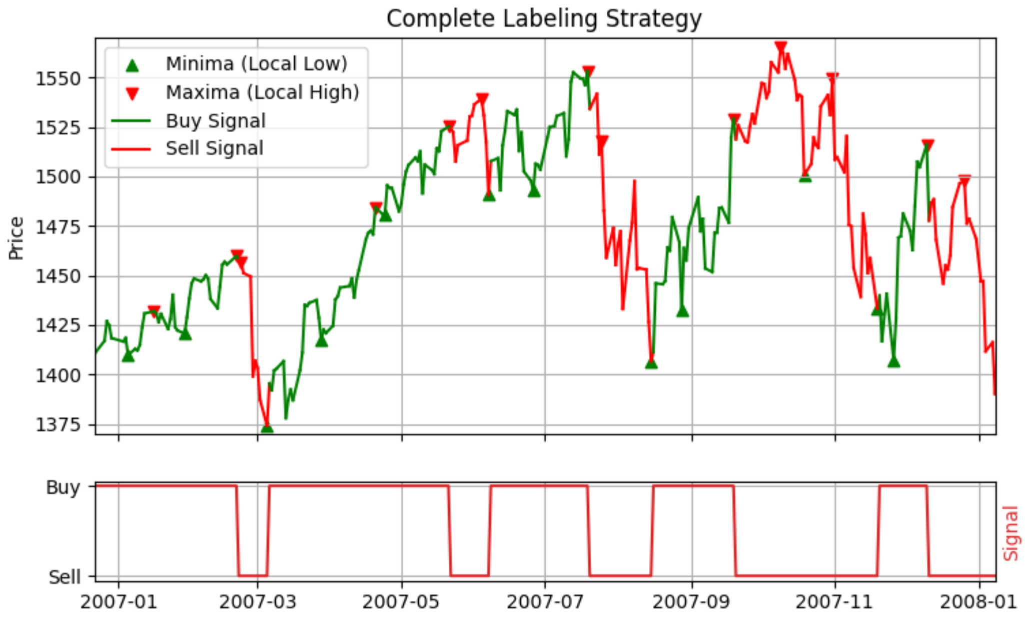


Figure 3.5: Short Trend Labels for 2007

In contrast, the short trend strategy operates on a much shorter time frame, identifying maximums and minimums over a month of trading days. This shorter window is designed to capture rapid fluctuations and exploit short-term opportunities. Given the higher frequency of price movements within this time frame, the short trend model is more reactive, seeking to quickly respond to smaller but potentially profitable market shifts in order to maximize gains or cut losses.

Together, these two approaches provide a comprehensive view of market dynamics. The long trend strategy acts as a stabilizing force, identifying significant market shifts, while the short trend strategy captures quicker opportunities without losing sight of the overarching trends. This combination allows the system to maximize returns while balancing the inherent risks and volatilities present in different market cycles. Ultimately, by dividing the market into distinct trend periods and employing specialized models for each, the system enhances predictive accuracy and adaptability.

By dividing the market into long and short trends, an ensemble strategy becomes essential to integrate the strengths of both models, resulting in a unified decision signal. This ensemble strategy will be discussed in detail in the later Section 3.3.3.

3.2.3 Build Features

The system is now prepared to build the technical indicators that will serve as the features providing information to the machine learning model. These indicators, derived from the raw financial data, provide insights into market trends, volatility, and potential price movements. By leveraging the structured data stored in the system, the necessary calculations for each technical indicator can be performed efficiently, ensuring that the model has access to accurate and up-to-date information. This process is crucial for transforming raw market data into actionable features that will feed into the machine learning algorithms, ultimately enhancing the model's predictive capabilities. To build the technical indicators, the system utilizes the *Pandas Technical Analysis (PandasTA)* library, a powerful and comprehensive tool specifically designed for technical analysis in financial data that leverages the *pandas* package [62].

The selection of technical indicators is grounded in the key literature review technical indicators (see Table 2.1) where a subset of technical indicators is selected (see Table 3.2). The selection of technical indicators was carefully made by prioritizing the most relevant and influential indicators for each category, with an emphasis on trend and momentum indicators. Furthermore, attention was given to ensure that all the raw data financial metrics: Open, Close, High, Low, and Volume, were reflected in at least one indicator within the final subset. All the selected subset of indicators are readily available within the *PandasTA* library.

The decision to opt for a fixed pre-selection of technical indicators, rather than employing a feature selection process, is based on the fact that these indicators have been extensively validated in the literature, demonstrating their effectiveness. Additionally, by using a consistent set of indicators, the system ensures generalization by avoiding overfitting to a specific training environment, a common risk in stock market prediction when feature selection is overly tailored to training data that may not fully reflect real-world conditions. This approach also simplifies model design, reducing the computational overhead associated with feature selection algorithms, and increases the interpretability of results by having fixed features.

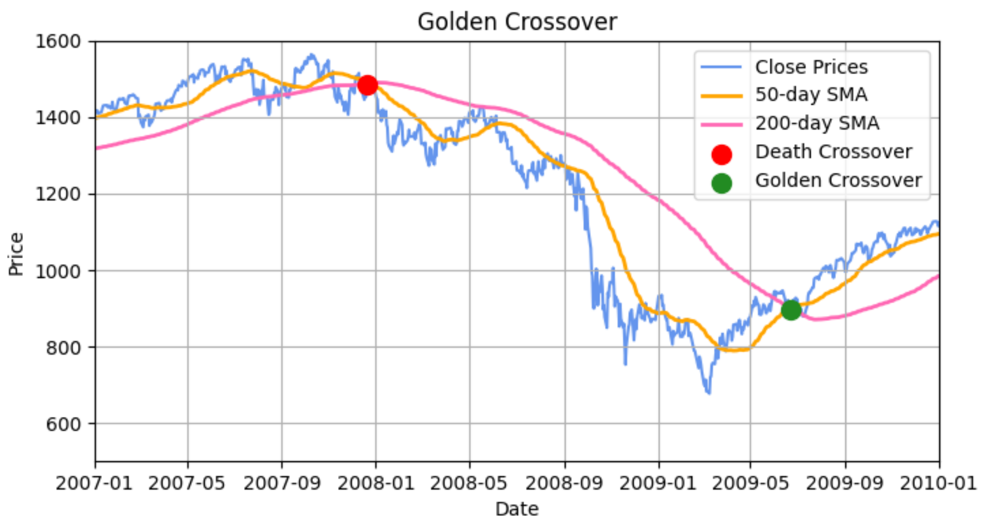


Figure 3.6: Golden Crossover Example

Furthermore, research was performed on more complex technical indicators and strategies, such as the Golden Crossover. The Golden Crossover is a particularly valuable addition to the system's indicator set. This trading strategy involves the crossing of a short-term moving average, such as the 50-day SMA, over a long-term moving average, like the 200-day SMA. When this crossover occurs, it signals a potential shift in market trends (see Figure 3.6).

3.2.3.A Handle Non-Stationary Data

In the context of financial markets, where data often exhibits non-stationary characteristics, addressing this issue is crucial for developing robust predictive models. Non-stationary data, characterized by mean and variance that change over time, can lead to misleading conclusions and poor model performance if not properly handled. To mitigate the effects of the non-stationary characteristics, the system employs a method that focuses on the slope of the technical indicators rather than their raw values:

$$\text{Indicator Slope} = \frac{\Delta\text{Indicator}}{n} = \frac{\text{Indicator}(t) - \text{Indicator}(t-n)}{n} \tag{3.3}$$

By using the indicator's slope the system captures the rate of change in the indicator over a specified period. This approach transforms the raw indicator values into a more stable, stationary form, allowing the model to focus on trends and momentum rather than being influenced by the fluctuating absolute levels of the indicators.

This method is particularly effective in financial markets, where sudden shifts and trends are more informative than the static values of indicators. By normalizing the indicators through their slopes, the model becomes more resilient to the inherent volatility and non-stationary nature of market data. This adjustment not only enhances the predictive power of the model but also improves its ability to generalize across different market conditions, making it a more reliable tool for forecasting and decision-making in the stock market.

For all the indicators used in the system, default parameter values are employed because these settings have been rigorously optimized for performance in previous analyses (see Table 3.2). This ensures that the indicators operate effectively within their intended contexts. Both the long and short trend strategies use the same technical indicators with the same parameters. The distinction between long-term and short-term trends is incorporated by calculating the slope for each indicator where n equals to the N period used in the NPMM labeling for each model, ensuring that the trend characteristics are appropriately captured.

Technical Indicator	Parameters
RSI	Period $n = 14$
WILLR	Period $n = 14$
MACD	$FastEMA_n = 12$, $SlowEMA_n = 26$, $SignalEMA_n = 9$
ROC	Period $n = 10$
BI	$MA_n = 26$
SMA	$MA_n = 10$
DEMA	$EMA_n = 10$
HMA	$WMA_n = 10$
ADO	No variable parameter
ATR	Period $n = 14$
BBANDS (Upper and Lower)	Period $n = 5$, $std = 2$
Golden Crossover (Long)	$FastEMA_n = 50$, $SlowEMA_n = 200$
Golden Crossover (Short)	$FastEMA_n = 5$, $SlowEMA_n = 20$

Table 3.2: Technical Indicators used and their parameters

3.2.4 Data Split

For the upcoming processes, it is crucial to maintain distinct training and testing sets, with the training data used to build and optimize the models, and the testing data reserved for evaluating their performance on unseen data. Although this might seem like a straightforward step, it is essential for ensuring the integrity and reliability of the model evaluation process. A custom function is developed to handle the data splitting, which takes as input the training period $[trn_s, trn_e]$ and the testing period $[tst_s, tst_e]$, where the condition $trn_e < tst_s$ must be satisfied. This condition is necessary given the time-series nature of the data, as it prevents data leakage from the future into the past. Based on these periods, the market data *DataFrame* is split by index, creating the appropriate training and testing sets for model development and evaluation.

3.2.5 Data Scaling

Data scaling is a critical preprocessing step in machine learning that significantly influences the performance of models. The primary goal of data scaling is to normalize the range of features so that they contribute equally to the model, preventing any single feature from disproportionately impacting the model's predictions. This process is especially important for distance-based algorithms like SVMs because these algorithms directly compute distances between data points to make decisions. Regarding the case of SVMs, this algorithm finds the optimal hyperplane that separates data points into classes. The position of this hyperplane is determined by the distances between data points and the hyperplane itself. If features are on different scales, the hyperplane could be biased toward features with larger ranges, leading to suboptimal classification.

When working with financial data, like stock market indicators, the feature values can vary widely. If left unscaled, the model might assign undue importance to features with larger ranges, leading to biased predictions.

To address this, Min-Max scaling is applied, scaling features to a predefined range $[-1, 1]$. This process involves subtracting the minimum value of the feature from each data point and then dividing the result by the range (maximum value minus minimum value) of the feature:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{3.4}$$

Where X is the original feature value, X_{min} is the minimum value of the feature and X_{max} is the maximum value.

This scaling method is advantageous because it preserves the relationships between the original data points, maintaining the structure of the data while ensuring that all features contribute equally during model training. Additionally, Min-Max scaling is straightforward to implement and computationally efficient, making it well-suited for large datasets typical in financial applications.

To implement the Min-Max scaling, the system utilizes the *MinMaxScaler* class from the *preprocessing* module of the *scikit-learn* library, which is an open-source machine learning library for Python that provides simple and efficient tools for data mining and data analysis, built on top of other scientific libraries [63]. A *MinMaxScaler* class is created and computes the minimum and maximum using the training dataset by applying the *fit* function of the class. Afterwards, the *transform* function is employed to scale both the training and testing datasets.

3.2.6 Data Cleaning and Validation

In the final stage of the data layer, the focus shifts to data cleaning and validation, ensuring that the dataset is both accurate and reliable before it is used for further analysis or model training.

The calculation of technical indicators often results in missing values, particularly at the start of the dataset, where insufficient historical data may exist to compute certain indicators. For example, moving averages require a specific number of prior data points, leading to null values in the early stages of the time series. The system systematically removes rows with null values to ensure that only complete and viable data records are used in subsequent analysis. This step is crucial for avoiding errors during model training and maintaining the integrity of the dataset.

After cleaning the dataset, the system performs data validation to confirm that the remaining data is both correct and consistent. This involves ensuring that after the cleaning process, no unexpected gaps remain in the data by confirming that all required columns are present and contain valid data for every entry. This is done through the use of the *pandas-market-calendars* library that provides access to market trading calendars for various financial exchanges around the world [64]. This allows the system to validate the alignment of the data with the real market sessions, making sure that there are no inconsistencies or data losses.

3.2.6.A Prevent Look-ahead Bias

In time-series forecasting, preventing look-ahead bias is essential to maintain realistic and reliable model performance. Look-ahead bias or data leakage occurs when information that would not have been available at the time of prediction is inadvertently used, leading to overly optimistic results. In the context of time-series data, such as stock market prediction, this can happen if the model has access to future data points, even indirectly. For instance, using today's indicators to predict today's stock price could unintentionally incorporate future information, creating a scenario where the model's performance appears better than it would be in real-world trading conditions.

To prevent look-ahead bias, a critical step is implemented in the data layer after every preprocessing and data management process. Each technical indicator, which corresponds to the features to be used by the models, is shifted one day forward. By shifting the indicators one day forward, we align the feature set with the actual decision-making process in trading:

- **Historical Data Alignment**: Each data point in the training set now represents the indicators from the previous day, ensuring that predictions are made using only past data, operating on the basis that we can only predict day N with the information from day [0, N-1].
- **Realistic Model Performance**: This approach reflects the real-world scenario where traders make decisions based on the latest available data, without the benefit of hindsight, providing a more accurate assessment of the model's predictive capabilities.

The decision to compute the minimum and maximum values for Min-Max scaling solely from the training data is also a data leakage prevention measure. This approach ensures that scaling parameters are derived without influence from the test data, preserving the integrity of the testing phase.

3.3 Processing Layer

The Processing Layer is the central component of the system where algorithms are fit to the preprocessed data from the Data Layer, ultimately producing a refined machine learning model. This layer encompasses several critical components, including hyperparameter tuning, the construction of individual models, and the ensemble of these models. Each of these steps is designed to optimize model performance, ensuring that the system effectively captures and handles the complexities of stock market prediction.

3.3.1 Hyperparameter Tuning

In this system, hyperparameter tuning is performed to ensure that each algorithm is operating at its optimal capacity. This process involves exploring a predefined range of hyperparameters to identify the combination that yields the best performance metrics, such as accuracy, precision, or other relevant evaluation criteria.

Given that the system will utilize an Radial Basis Function Support Vector Machine (RBF-SVM) to fit the preprocessed data, hyperparameter tuning is crucial for extracting maximum performance from this machine learning algorithm. The RBF-SVM relies on two key hyperparameters: the regularization/penalty parameter C, which controls the trade-off between achieving a low training error and a low testing error, and the width of the Gaussian kernel parameter γ, which controls the influence of individual training instances on the decision boundary. To identify the optimal combination of these parameters, a range of values is defined (see Table 3.3) based on prior research [65], where a GA is then employed to efficiently search through these possible values, ultimately selecting the combination that yields the best performance.

Hyperparameter	Minimum	Maximum
C	2^{-2}	2^{15}
γ	2^{-15}	2^{3}

Table 3.3: Hyperparameters' range of values

To implement the GA the system leverages Distributed Evolutionary Algorithms in Python (DEAP) which is an open-source Python library for evolutionary algorithms that provides implemented genetic operators and evolutionary processes [66]. Using this library, a GA algorithm is employed with the following genetic operators:

- **Uniform Crossover**: This crossover operator effectively explores the combinations of C and γ by allowing each gene (hyperparameter) to be inherited independently, which is crucial for fine-tuning the SVM's performance in a highly sensitive and interdependent search space. This approach maximizes the exploration of possible hyperparameter pairs, ensuring that the genetic algorithm does not prematurely converge on suboptimal solutions.

- **Random Resetting Mutation**: The value of a gene is replaced by a random value within the predefined range (see Table 3.3). This mutation operator is best suited for integer representations, reintroducing diversity by randomly altering the value of genes.

- **Roulette Wheel Selection**: It is a fitness proportionate selection, where each individual's chance of being chosen is proportional to its fitness score relative to the total fitness of the population. This means individuals with higher fitness have a greater probability of being selected, allowing their traits to be passed on to the next generation while maintaining diversity in the population.

The implemented GA adopts the Elitism strategy where the individual with the highest fitness value, achieved throughout the algorithm's execution, is saved. Each individual has two genes, and each gene represents an hyperparameter of the SVM. The genes are integer values that represent an exponent, and are within a predefined range of values according to the minimum and maximum exponents (see Table 3.3).

To determine the performance of individuals and consequently the combination of hyperparameters presented as solution by the GA, we chose the F1-score metric, which therefore defines the GA's objective as searching the combination of hyperparameters that maximize the F1-score. This metric was chosen it balances the ability to correctly predict positive instances (precision):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.5}$$

and the ability to detect all actual positive instances (recall):

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.6}$$

The F1-score is a more reliable metric for hyperparameter optimization because it provides a balanced evaluation of the model's ability to handle both classes, particularly in scenarios with imbalanced datasets. Unlike accuracy, which merely reflects the percentage of correct predictions, the F1-score captures the trade-off between precision and recall, ensuring that both false positives and false negatives are considered. Accuracy can be misleading in imbalanced cases, where a model could achieve high accuracy by always predicting the majority class.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.7}$$

A cross-validation method is used in the fitness function to ensure generalization in the evaluation process. As the stock market data is a time-series data where the order of the data is relevant, the cross-validation method is chosen accordingly. Thus, the *TimeSeriesSplit* class from the *model_selection* module of the *scikit-learn* library is used for this task. Unlike traditional k-fold cross-validation method, which randomly splits the data, the *TimeSeriesSplit* respects the temporal order of observations. The *TimeSeriesSplit* splits the time series data into training and test sets n times, where $n = 5$ for this system, where each split includes an increasing portion of the data for training while preserving the chronological order, ensuring that there is no data leakage.

For each split created, a SVM is created with the decoded individual's gene values, and it is fit to the split training data and tested on the split test data, where the F1-score metric is then calculated for that split, using the *f1_score* function from the *metrics* module of the *scikit-learn* library. Upon finishing the last split, the F1-score average of all the splits is computed and returned as the fitness value.

In order to ensure a more diverse and comprehensive exploration of potential solutions, we utilize Sobol sequences to generate the initial population, inspired in the aforementioned research that studied *quasi-random* sequences including Sobol sequences [43]. Sobol sequences are a type of low-discrepancy sequence, designed to cover the search space more uniformly than purely random sampling. To fully leverage the properties of Sobol sequences, especially their efficiency in covering the space when the population size is a power of 2, we set the initial population size to 8. This choice maximizes the sequence's ability to uniformly distribute individuals across the search space, providing a strong foundation for the evolutionary process that follows. By starting with a well-distributed and diverse population, the GA is better equipped to explore the parameter space effectively, potentially leading to faster convergence and improved optimization results. The Sobol sequences are implemented using the *qmc* class of the *stats* module featured in the *SciPy* Python library [67]. The Sobol sequence samples are scaled according to the exponents of the upper and lower bounds of the hyperparameters (see Table 3.3). Thus creating the initial population with the encoded hyperparameters into the genes.

When designing the number of generations for the GA, the population size must be taken into account to materialize the objective of striking a balance between exploring a diverse solution space while maintaining computational efficiency. The primary objective of using a GA instead of the traditional Grid-Search algorithm is to significantly reduce the time complexity, especially considering that the parametric grid for Grid-Search would require evaluating 18×19 combinations (see Table 3.3). Given the search space and the chosen population size, a number of generations of 35 is proposed, making up to 8×35 combinations, ensuring a computational complexity advantage over the Grid-Search algorithm. The parameters used in the GA and their corresponding values are detailed in Table 3.4.

Parameter	**Value**
Population Size	8
Generations	35
Crossover Probability	80%
Mutation Probability	40%
Independent Gene Mutation Probability	50%
Independent Gene Crossover Probability	50%

Table 3.4: Genetic Algorithm parameters

The hyperparameter tuning process is performed separately for each trend strategy, resulting in a distinct set of optimized hyperparameters for both the long trend and short trend strategies. Given the independence of these strategies, this process can be parallelized to enhance efficiency, allowing for concurrent optimization of hyperparameters for both strategies. Moreover, the evaluation process of the population can also be parallelized, to further reduce the overall computational time. This is especially important as the hyperparameter tuning process is the bottleneck of this system regarding the time complexity. This can be achieved using the *multiprocessing* Python module and the *joblib* library [68], which allow parallel evaluation of individuals in the genetic algorithm and parallel execution of the genetic algorithm itself, significantly reducing the overall computation time by distributing the workload across multiple CPU cores.

3.3.2 Build Individual Models

With the hyperparameter tuning process completed, we proceed to the construction of individual models for both long and short trend strategies. This step involves utilizing the optimized hyperparameters identified during the previous phase to build robust models tailored to each specific trend strategy.

For this task, the *SVC* class from the *scikit-learn* library's *svm* module is utilized to implement the Support Vector Machine (SVM) algorithm for classification. This implementation offers the kernel trick, with support for the Radial Basis Function (RBF) kernel, enabling the model to effectively manage non-linear classification problems. Additionally, it offers functionality for adjusting class weights, allowing for better handling of imbalanced datasets.

3.3.2.A Class Weights

In many classification tasks, especially in real-world applications like stock market prediction, data can be highly imbalanced. This means that one class might significantly outnumber the other(s), leading to biased model predictions where the algorithm tends to favor the majority class. To address this issue, we can adjust class weights, allowing the model to penalize misclassifications of the minority class more heavily than those of the majority class.

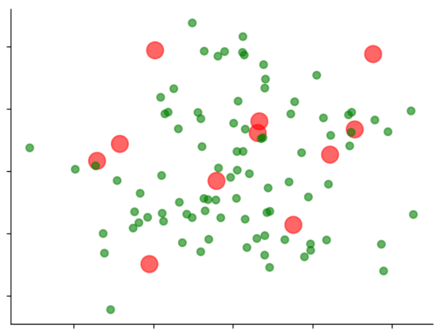


Figure 3.7: Class Weights Visualization Example

For instance, in a scenario where buy signals (class 1) greatly outnumber sell signals (class 0), the use of class weighting ensures that the model does not disproportionately favor the buy signals. By assigning a higher weight to the less frequent sell signals, the model avoids overfitting to the dominant class, leading to improved generalization and better handling of both classes in unseen data (see example Figure 3.7).

In *scikit-learn*, this can be handled using the *class_weight* parameter of the *SVC* class, which can be set to *balanced* or manually specified by the user. When set to *balanced*, the algorithm automatically adjusts the weights inversely proportional to class frequencies in the input data:

$$class\_weight_i = \frac{n\_samples}{n\_classes \cdot n\_samples\_class_i} \tag{3.8}$$

This ensures that both classes are given equal consideration, regardless of their occurrence rates. In this study, we apply the *balanced* class weights only to the short-trend model, as this trend is more volatile and requires an equal treatment of both buy and sell signals to capture rapid market movements effectively. For the long-trend model, however, we intentionally avoid using balanced class weights. This is because the long-trend follows the general upward growth of the S&P 500, where buy signals naturally dominate. By allowing this bias toward the buy class, we align the model with the overall market behavior, which tends to favor long-term growth.

To implement the models, an *SVC* class is instantiated using the RBF kernel, with the hyperparameters C and γ tuned for each trend strategy. Class weights are applied only to the short-trend model, while the long-trend model is left without class balancing. The feature and label data from the training split are then passed into the *fit* function of the respective *SVC* instances, training separate SVM models for both the long and short trends. This results in two distinct models: one for predicting long-term trends and another for short-term trends.

3.3.3 Ensemble Individual Models

In this system, an ensemble strategy is required to combine the outputs of the long and short trend models, in order to formulate a final prediction. By leveraging the strengths of both models while compensating for their individual limitations, the ensemble approach aims to deliver more robust and reliable predictions, enhancing the system's overall performance in dynamic market conditions.

The ensemble approach is based on an SMA crossover technique, to dynamically select which model prediction to follow. Therefore, a 10 day SMA is built on the Close Price, providing the information on the last 2 weeks of trading days (half of the period used for the short trend strategy). Note that for the SMA the same principles to avoid the look-ahead bias are applied (see Section 3.2.6.A). After building the SMA, a relationship is established with the daily Open Price.

$$\text{Prediction}_{\text{today}} = \begin{cases} \text{Long Trend Prediction} & \text{if } SMA_{10} \geq \text{Open Price}_{\text{today}} \\ \text{Short Trend Prediction} & \text{if } SMA_{10} < \text{Open Price}_{\text{today}} \end{cases} \tag{3.9}$$

Specifically, if the Open Price for the day crosses above the SMA, the system uses the long trend model's prediction, indicating sustained and stable market movements, typical of the S&P 500 index growing nature, which falls in the long trend model specialty. Conversely, if the Open price falls below the SMA, the short trend model is activated, indicating a possible period of a downtrend, which is usually associated with volatility in the context of the S&P 500 index. This is illustrated in the label construction example for the year 2007: when the ensemble strategy selects the short-trend model for a given period, the final labels reflect the short-trend labels for that timeframe. Conversely, when the ensemble strategy opts for the long-trend model, the final labels for that period are derived from the long-trend labels (see Figure 3.8).

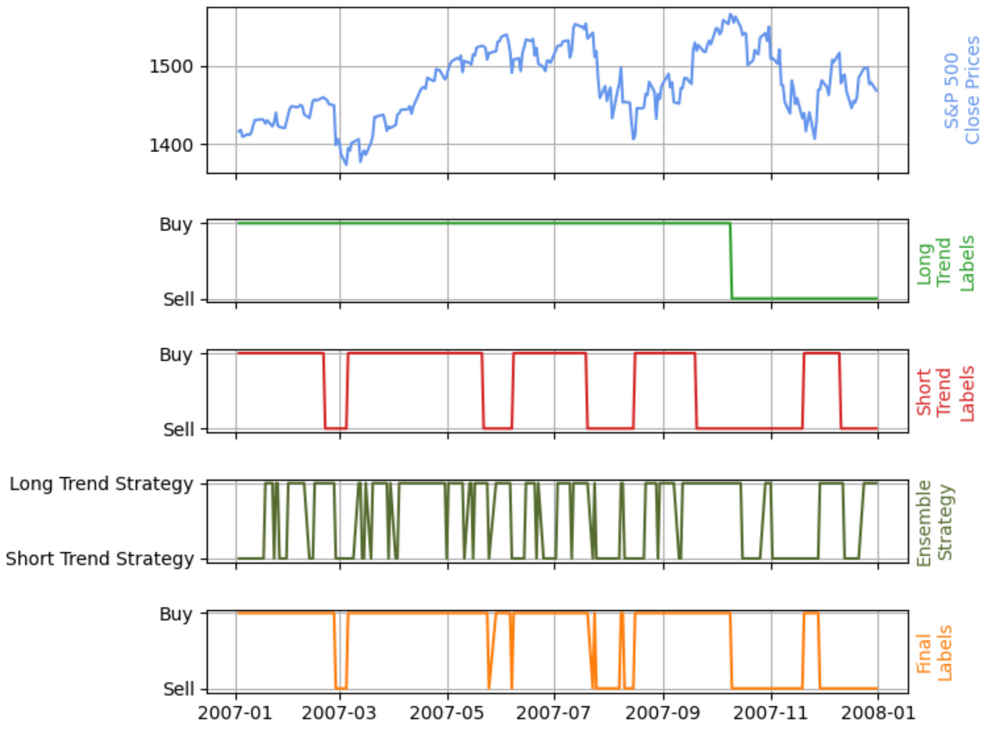


Figure 3.8: Long and Short-Trend Ensemble Data Labels for 2007

To interpret Figure 3.8: the green line represents the Long Trend Labels, indicating buy or sell periods for the long-term model, while the red line corresponds to the Short Trend Labels and the corresponding short-term model. The olive green line shows the Ensemble Strategy signal, specifying which model's predictions are used at each period. Finally, the orange line represents the Final Labels, combining the Short and Long Trend Labels according to the Ensemble Strategy.

This method dynamically switches between the two models based on market conditions. By using the SMA trend indicator, responsiveness to market shifts is added, capitalizing on each model's strengths in their optimal context.

4 Evaluation

Contents

In this section, we establish a test environment to rigorously evaluate the developed system, presenting and discussing the results obtained. The following subsections outline a series of case studies, each with clearly defined objectives. Before presenting the results and analysis for each case study, we provide a comprehensive description of the test environment setup. Following this, we present the results for each case study and conduct a thorough analysis. Finally, we offer an overall assessment and synthesis of the findings, highlighting the key insights.

4.1 Case Studies and Objectives

We present three case studies designed to cover the key features of the system, with the aim of assessing its effectiveness in predicting stock market movements. These studies not only measure the system's predictive performance but also explore potential enhancements, grounded in the key concepts and architectural principles that underpin the system. Through this comprehensive analysis, we seek to validate the system's design while identifying areas for future improvement.

- **Diversity Control in Evolutionary Algorithms**: This case study investigates the use of DGA to evaluate the results of incorporating diversity control mechanisms in the evolutionary process. By introducing dynamic mutation-crossover control and early stopping criteria, the goal is to improve GA's solution quality and execution time. The evaluation compares the performance of both the DGA and the proposed GA, benchmarked against a Grid Search algorithm, a greedy and more deterministic approach.
- **Dual Model vs Single Model Architecture**: This case study aims to validate the effectiveness of the proposed dual-model architecture, which employs specialized models for both long and short trends. The architecture is compared against more conventional and widely used single-model approaches in stock market prediction. The study highlights the advantages and limitations of the DTE-2SVM, presenting its results and performance in terms of predictive capability and trading profitability. Additionally, it also contrasts these outcomes with the performance of a Buy and Hold strategy, a competitive benchmark strategy when predicting the S&P 500.
- **Dynamic Ensemble Model for Long Trend Class Imbalance**: This case study addresses the challenge of class imbalance inherent in long trend strategies for the S&P 500, largely attributed to the index's continuous upward trajectory. While the imbalance has been leveraged to benefit the DTE-2SVM, its bias toward the buy class has, in certain scenarios, resulted in suboptimal performance, particularly when compared to the Buy & Hold strategy. As the influence of this bias increases, it creates a bottleneck in the overall predictive capacity. To mitigate these limitations, this study evaluates the use of a Dynamic Ensemble Model, specifically designed to handle data complexity and class imbalance, in comparison with the introduction of class weights into the SVM, aiming to provide a more robust solution to the imbalance challenge.

4.2 Test Environment Setup

Although each case study has its specific requirements and objectives, there are methodologies and processes that are shared between all of the case studies and that formulate the test environment setup that is presented in this section. A comprehensive evaluation framework was designed and implemented to assess the predictive performance of the developed system within the stock market using two key methodologies: Backtesting and Walk-Forward validation.

Backtesting is a key process in evaluating the effectiveness of a trading strategy, involving testing a trading strategy on historical data to see how it would have performed in the past. The rationale behind this methodology is that if a strategy had been profitable in the past, it might be profitable in the future as well. By applying a model to past data, we can evaluate how well it would have predicted actual market movements and the effectiveness of the trading strategies derived from these predictions. This retrospective analysis is essential for validating the model's accuracy, robustness, and potential profitability in real-world scenarios, as well as identifying potential overfitting issues or periods where the model might underperform.

To perform backtesting we use the collected S&P 500 historical stock market data from the *yfinance* module, detailed in Section 3.2.1. We collected historical data from the S&P 500 index, spanning from 1985 to 2024, in order to provide and analyze the maximum reliable information of the index. However, for the purposes of this study, we will focus on data from 1995 onwards. The decision to exclude data from 1985 to 1995 is based on the significant structural and behavioral differences in the market during that earlier period, which do not accurately reflect the conditions observed in more recent decades, as seen in Figure 4.1. By concentrating on data from 1995 to 2024, we aim to provide a more relevant and accurate representation of the current market dynamics.

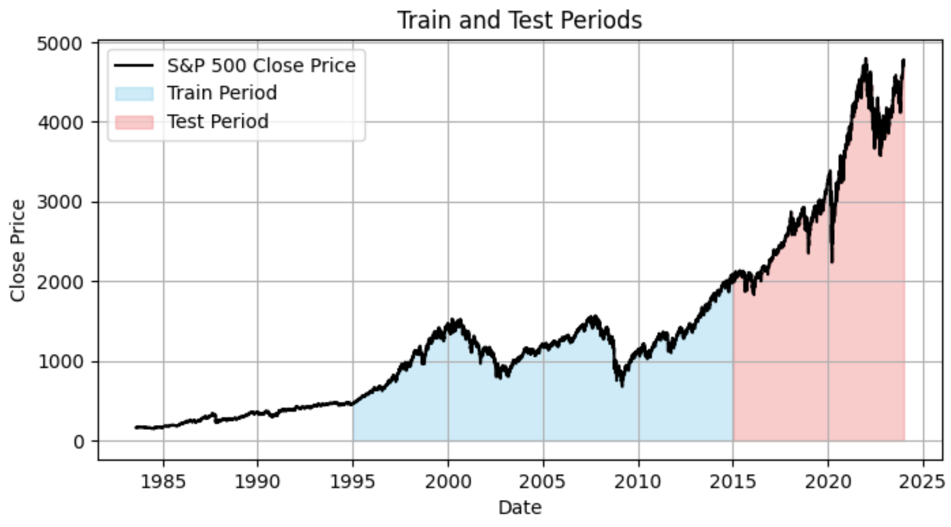


Figure 4.1: Test Environment's Train and Test Periods

In addition to backtesting, we employ Walk-Forward validation, a dynamic evaluation technique that involves training the model on a growing window of historical data and then testing it on the next sequential period. This approach is particularly well-suited for the stock market, where market dynamics are constantly evolving. By progressively expanding the training dataset with the most recent data while testing on subsequent periods, walk-forward validation helps maintain the model's relevance and adaptability to current market conditions. This method is crucial for evaluating how well the model can generalize to new, unseen data, reflecting its real-world predictive power.

For the purpose of this study, we will use a fixed sequential period for testing of 1 year, and will start with a window of 19 years starting from 1995 (see Table 4.1). This ensures that there is enough information (data points), for the first and subsequent tests, while the fixed 1-year testing period allows evaluating and analyzing the model in different market dynamics with greater detail.

To avoid look-ahead bias and considering the data labeling approach described in Section 3.2.2, it's crucial to acknowledge that the labeling strategy relies on future information. The longest period used in the labeling process is 126 trading days, used for the long trend strategy. To mitigate the risk of data leakage between the training and test sets, we remove the final 126 days from the training period, as these days could inadvertently contain information that pertains to the test period.

Train Period	Test Period
$[1995, 2014]$	2015
$[1995, 2015]$	2016
$[1995, 2016]$	2017
$[1995, 2017]$	2018
$[1995, 2018]$	2019
$[1995, 2019]$	2020
$[1995, 2020]$	2021
$[1995, 2021]$	2022
$[1995, 2022]$	2023

Table 4.1: Walk Forward Validation Set of Tests

In order to evaluate the performance of the machine learning models in predicting the stock market, we use two key and straightforward metrics:

- **F1-Score**: a fundamental classification metric that combines precision and recall, providing a balanced measure of a model's performance, especially in scenarios with class imbalance. It is particularly useful when both false positives and false negatives are critical in the evaluation (see Formula 3.7).

- **Return on Investment (ROI)**: a finance specific metric that measures the profitability of an investment, which is given by dividing the net profit by the initial investment cost. A higher ROI indicates a more profitable strategy, which is the ultimate goal of stock market predictions.

$$\text{ROI} = \frac{\text{Net Profit}}{\text{Initial Investment}} \times 100 \tag{4.1}$$

To contextualize the ROI metric, the ROI of the buy-and-hold strategy is provided for each test period. This comparison allows us to evaluate the effectiveness of the model's predictions, as the buy-and-hold strategy serves as the benchmark our model aims to surpass. Given that the S&P 500 index is inherently designed for this strategy due to its consistent growth, outperforming its ROI would indicate the success of our approach.

4.3 Case Study 1: Diversity Control in Evolutionary Algorithms

This case study focuses on evaluating the effectiveness of a Dynamic Genetic Algorithm (DGA) in tuning the hyperparameters of a Support Vector Machine (SVM) model. The primary aim is to assess how the DGA's dynamic control of mutation and crossover rates impacts the convergence speed and quality of solutions, compared to a traditional Genetic Algorithm (GA) and Grid-Search method.

4.3.1 Dynamic Genetic Algorithm Implementation

To achieve this, we develop a DGA, based on previous research [39], that introduces mechanisms for dynamically controlling and adjusting the population during the evolutionary process. By doing so, the algorithm can adapt to the optimization landscape, reducing the likelihood of premature convergence and fostering greater exploration of the solution space. This adaptive approach allows for a more efficient search process, with the potential for achieving improved solutions in a shorter timeframe compared to a traditional static GAs.

The developed DGA is essentially equal to the proposed, GA with 3 additional steps in the evolutionary process. The extra steps are placed after the genetic operations such that:

1. After the new offspring population is created, there is a validation of improvement. This validation is based on verifying if the current generation's best individual has a greater fitness value than the previous generation. In case of no improvement a counter of generations without improvement is increased, conversely if there is improvement, this counter is reset.

2. The counter of generations without improvement is validated to verify if it reaches a fixed patience value, in which case the mutation and crossover probability is increased to enhance exploration of the solution space.

3. The counter of generations without improvement is validated if it reaches a fixed value of maximum number of generations without improvement. In which case, the DGA is halted, indicating that there is no further improvement and there is no need to keep evolving the population. The values for the added variables can be seen in Table 4.2.

Parameter	Value
Patience	4
Max. Generations Without Improvement	10
Mutation Probability Increase	15%
Crossover Probability Increase	10%

Table 4.2: DGA Specific Parameters

4.3.2 Test Scenario Specifications

For this case study, we will exclusively use the training period from $[1995, 2022]$ to conduct the hyperparameter optimization processes under discussion. This choice is justified by the fact that it represents the longest training window within the proposed walk-forward validation setup. As such, it encompasses the most extensive dataset, providing a larger volume of information (data points), which is crucial for a meaningful comparison of time complexity and performance across different optimization methods. To effectively evaluate the DGA, we distinguish two distinct phases in this case study:

- **Comparison with Standard Genetic Algorithm**: where the objective is to evaluate whether introducing dynamic control over mutation and crossover rates improves the evolutionary process.

- **Benchmarking against Grid-Search**: where the objective is to validate if the DGA reaches solutions comparable to those found by exhaustive search methods (Grid-Search) while being more computationally efficient.

To validate the improvement of introducing dynamic control in the evolutionary process, we will compare DGA and GA regarding the fitness values (based on the F1-score) and diversity in the population. In the context of measuring diversity in a GA with integer-valued genes, the pairwise Manhattan distance (L1 distance) was the chosen metric to reflect the diversity. This metric involves calculating the sum of the absolute differences between the genes of all pairs of individuals in the population. By averaging these distances, we quantify the population's diversity, which reflects the spread of solutions in the search space. A higher average distance indicates greater diversity, indicating greater exploration of the search space, while a lower value suggests that the population may be converging, potentially to a local optimum.

In order to perform the comparison between DGA and GA, each algorithm undergoes the optimization process five times for both the long and short trend strategies, resulting in a total of 10 optimization runs per algorithm. This repetition is necessary due to the probabilistic nature of these algorithms, ensuring a more reliable comparison by accounting for variations in performance across different runs.

4.3.3 Discussion of Results

The following Figure 4.2 shows the mean diversity of the 10 optimization runs by generation (5 runs for each trend strategy), for each algorithm:

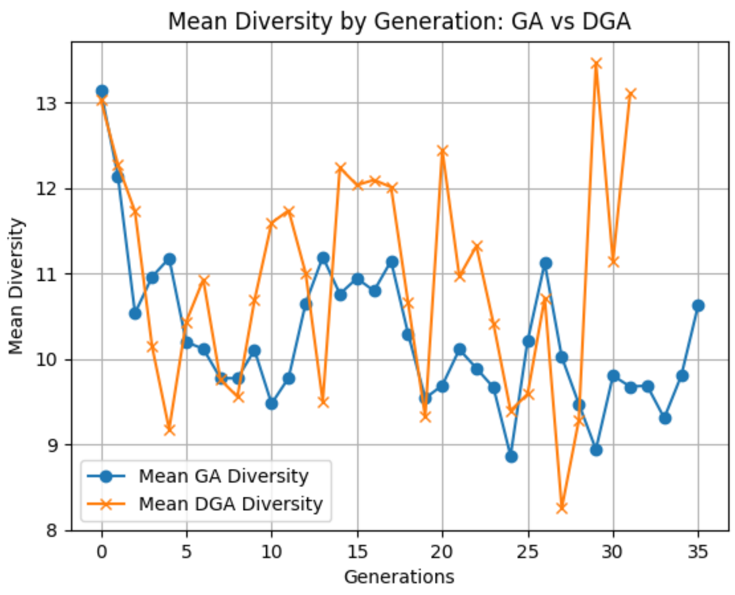
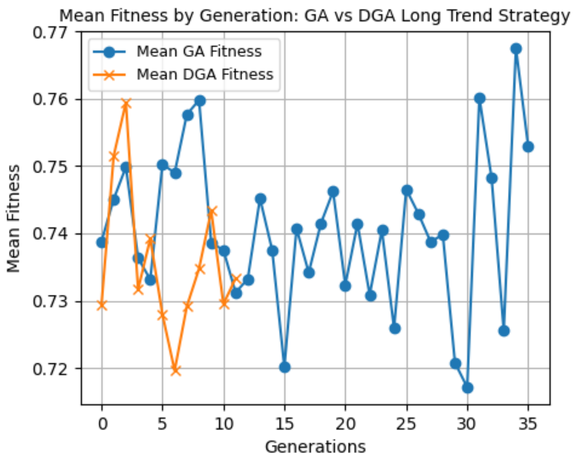


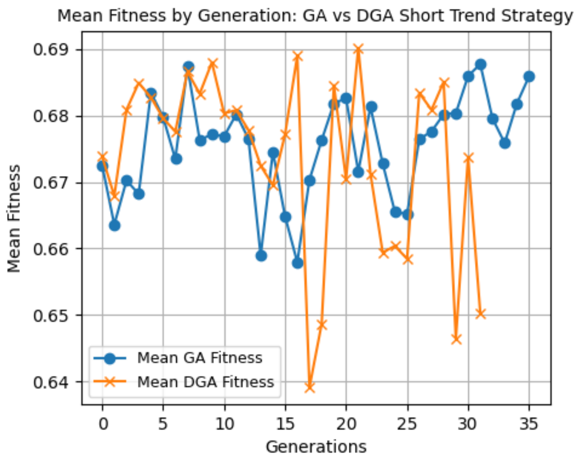
Figure 4.2: GA vs DGA Mean Diversity Comparison

We can validate that DGA effectively introduces greater diversity in the population, as the mean diversity line of DGA is mostly above of the GA line. This is especially relevant in the fourth generation, where the first chance of increasing mutation and crossover rates occurs, and a spike of increased diversity is seen for the DGA. When advancing the generations, we start to see a larger deviation on the diversity values of the DGA, this is explained as although some runs of the algorithm went further into the generations, some of the DGAs halt their execution earlier, and therefore there is a higher contrast between the diversities of DGA runs that are reaching the end of their execution and others that are exploring the search space. We can also validate that no DGAs needed the 35 generations to produce a solution which corroborates the DGA's potential in improving the GA's execution time. The high average diversity values on both algorithms validate the effectiveness of the Sobol sequences in creating a diverse initial population.

When analyzing the average fitness values of both the algorithms, we note that in the GA algorithm there is a tendency for the average to rise, indicating a convergence of individuals to a final solution (see Figure 4.3). However, when looking into the DGA we note that there is a large deviation of the average fitness values with a tendency to drop, especially visible in the short trend hyperparameter optimization which required, on average, more generations for the DGA to output a solution. This justified as the



(a) Long Trend Strategy



(b) Short Trend Strategy

Figure 4.3: GA vs DGA Mean Fitness Comparison

algorithm when reaching higher generations, it means it is being rewarded to explore the search space, and therefore it will have better solutions, but it may end up having also worse solutions as it does not continuously converge as the GA. This behavior further highlights the importance of using the Elitism approach on the DGA, ensuring that the best solution is not lost when expanding the search space.

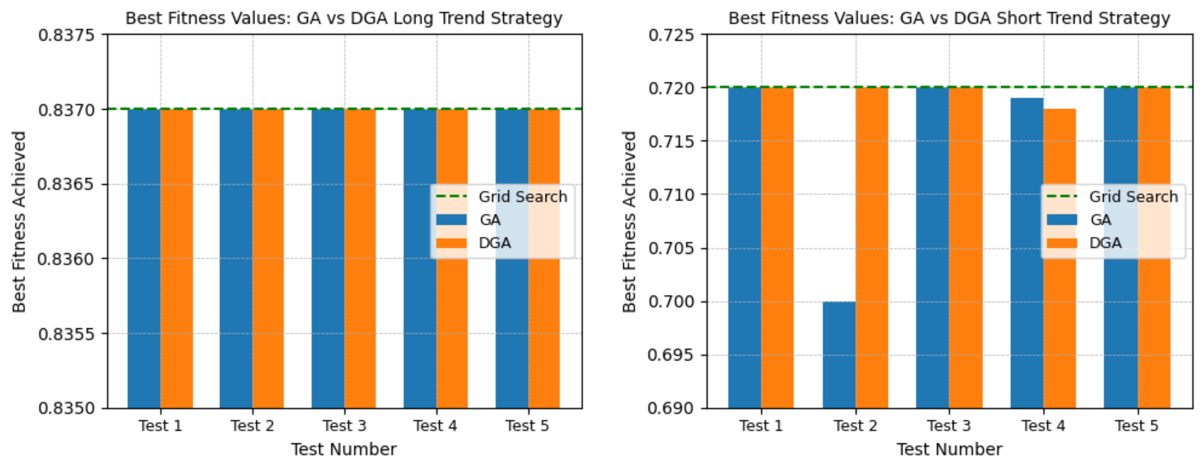
Employing the Grid-Search algorithm with the same cross-validation technique used in the GA and DGA we can obtain the optimal results for this hyperparameter optimization process as this algorithm follows a greedy approach to validate every combination possible at the cost of higher time complexity, being a greedy approach (see Table 4.3).

Long Trend Best Fitness	Short Trend Best Fitness
0.837	0.720

Table 4.3: Grid-Search Results

Both the DGA and GA approach reach the optimal result for the long trend strategy, however in the short trend strategy the optimal result was not obtained in every test (see Figure 4.4). In case of failing to reach the optimal result, the DGA demonstrates, on average, a better solution than the GA. This shows that there is a considerable probability that the Evolutionary Algorithms may fail to present the optimal results for the hyperparameter optimization process, which presents a liability when using these algorithms.

In terms of execution time, both Evolutionary Algorithms outperform the Grid-Search algorithm in finding a solution, as presented in Figure 4.4. This is particularly true for the DGA, which demonstrates a notable reduction in runtime compared to the GA, which itself already shows better efficiency than the



(a) Long Trend Strategy **(b)** Short Trend Strategy

Figure 4.4: GA vs DGA Solution Comparison

Grid-Search	GA	DGA
37m 48s	34m 18s	28m 41s
	-9.28%	-24.12%

Table 4.4: Hyperparameter Tuning Mean Execution Time Comparison

Grid-Search approach. This shows the effectiveness of the variable that controls the maximum number of generations without improvement, which stops the algorithm by indicating a convergence to a solution.

Model ROI %	2015	2016	2017	2018	2019	2020	2021	2022	2023	Cum.
Without Tuning	-0.93%	9.45%	$\mathbf{14.93\%}$	-8.27%	29.50%	8.76%	11.05%	-18.19%	$\mathbf{22.80\%}$	79.62%
Grid-Search	$\mathbf{0.35\%}$	$\mathbf{12.64\%}$	11.67%	$\mathbf{9.51\%}$	$\mathbf{31.52\%}$	$\mathbf{16.99\%}$	$\mathbf{23.78\%}$	$\mathbf{-2.71\%}$	18.81%	$\mathbf{204.31\%}$
DGA*	-0.06%	11.98%	12.97%	7.54%	30.11%	14.36%	18.67%	-4.78%	20.12%	174.59%

Table 4.5: DTE-2SVM ROI: Hyperparameter Tuning algorithm comparison
* Mean results of 5 optimization runs

To validate and draw conclusions regarding the hyperparameter tuning process, we compared the proposed DGA with the Grid Search algorithm and an approach that uses no hyperparameter tuning, to observe how hyperparameters affect results, results are presented in Table 4.5. The non-tuned approach relies on default values from the *scikit-learn* library, where $C = 1$ and $\gamma = scale$, with $scale$ being the inverse of the product of the number of features and their variance.

The results highlight the critical role of hyperparameter tuning, as both tuned approaches outperform the non-tuned method significantly. The Grid Search algorithm further proves its superiority over the DGA, achieving better returns, which also validates the use of F1-score as an objective function as effectively optimizing this metric led to superior financial results when compared to an approach without hyperparameter tuning.

4.3.4 Key Findings

This case study evaluates the hyperparameter tuning process with an implementation of a Dynamic Genetic Algorithm (DGA), to assess the benefits of diversity control in the evolutionary process as a potential improvement on the proposed GA, while also benchmarking both the Evolutionary Algorithms against the Grid-Search greedy approach.

The DGA improves upon the GA by maintaining greater diversity through dynamic mutation and crossover rates, allowing for early termination upon convergence, which enhances execution time and solution quality. However, the DGA did not consistently achieve optimal solutions, as confirmed by the Grid-Search, which, despite its higher time complexity, reliably finds optimal solutions through exhaustive search.

While the DGA offers advantages in execution time and diversity, Grid-Search remains the most reliable method in smaller search spaces, balancing solution quality with a manageable time cost for the designed search space. Therefore, Grid-Search is selected as the preferred algorithm for hyperparameter tuning in the upcoming case studies, as it guarantees precision and high solution quality. Its deterministic nature also enhances interpretability, making the analysis process clearer compared to the probabilistic outcomes of Evolutionary Algorithms.

However, the DGA demonstrated notable advantages, suggesting its potential for expanding the search space, adding finer granularity within the defined boundaries (see Table 3.3), by considering real numbers instead of only integers for the exponents. This would enable the search for even better solutions in scenarios where the exhaustive nature of Grid-Search would be computationally prohibitive.

4.4 Case Study 2: Dual Model vs Single Model Architecture

In this case study, we assess the effectiveness of a dual model architecture, which separates the prediction of long-term and short-term market trends, by comparing it against a more traditional single-model approach. The aim is to determine whether the specialization of models for different market behaviors offers measurable improvements and overall increased performance.

4.4.1 Test Scenario Specifications

To conduct this evaluation, we will compare our dual model system against a single SVM model as well as other widely used machine learning algorithms in stock market prediction, such as the k-NN

Classifier, Random Forest Classifier, and XGBoost. These models have been selected for their proven effectiveness in similar financial prediction tasks, providing a robust benchmark for the performance comparison. The tests will be performed using the defined set of validation periods outlined in Table 4.1, ensuring a comprehensive evaluation across varying market conditions.

This comparison will provide valuable insights into the ability of each approach to capture market trends, allowing us to determine whether the dual model's architecture, with its tailored long and short trend predictions, offers a significant advantage over the traditional single-model approach not only in terms of accuracy in predicting the outlined strategy but also in delivering a higher ROI for a potential user.

The proposed DTE-2SVM architecture utilizes two distinct sets of features and labels, corresponding to the two SVMs it comprises. However, for the benchmark algorithms, we must use a single set of features and labels to adhere to the single-model architecture approach. To ensure a fair comparison and accurately assess the results, we will conduct the tests outlined in Table 4.1 twice for each benchmark algorithm. One test will use the long trend strategy features (indicated by having "Long" after the algorithm name), and the other will use the short trend strategy features (indicated by having "Short" after the algorithm name), while both tests will rely on the ensemble label that the DTE-2SVM is designed to predict (see Figure 3.8).

To further ensure a fair and consistent comparison between all algorithms in this study, the hyperparameters of each algorithm were carefully tuned (see Table 4.6), using the same approach employed for the proposed DTE-2SVM. Using the insights from Case Study 1 (see Section 4.3), we applied a Grid-Search with 5-fold cross-validation to optimize the hyperparameters for each benchmark algorithm and the DTE-2SVM.

Algorithm	**k-NN**	**Random Forest**	**XGBoost**	**SVM**	**DTE-2SVM**
Parameters	n_neighbors; weights; distance_metric	n_estimators; max_depth; max_features; min_samples_split; min_samples_leaf	n_estimators; learning_rate; max_depth; subsample; colsample_bytree; min_child_weight;	C; γ	Long C; Long γ; Short C; Short γ

Table 4.6: Hyperparameters Tuned for each Algorithm

By applying the hyperparameter tuning process across all the algorithms we ensure that the comparison reflects the true potential of each algorithm when appropriately tuned under identical conditions, guaranteeing that the performance differences observed are attributed to the strengths and weaknesses of the models themselves, rather than differences in hyperparameter optimization strategies.

4.4.2 Discussion of Results

Model ROI %	2015	2016	2017	2018	2019	2020	2021	2022	2023	Cum.
SVM Long	−2.93%	−2.99%	**18.42%**	−7.01%	28.71%	15.29%	**28.79%**	−19.95%	24.73%	97.87%
SVM Short	−0.69%	11.24%	**18.42%**	−7.01%	28.71%	15.29%	**28.79%**	−19.95%	24.73%	132.13%
k-NN Long	−1.25%	−3.43%	11.88%	**13.94%**	8.69%	17.76%	13.13%	−13.96%	13.41%	71.76%
k-NN Short	−7.08%	1.34%	11.37%	−0.70%	28.78%	23.23%	11.48%	−17.45%	**26.75%**	92.77%
Random Forest Long	−4.23%	−3.61%	12.14%	5.37%	16.05%	23.10%	16.18%	−5.98%	15.11%	95.93%
Random Forest Short	−9.48%	4.93%	14.78%	0.69%	25.80%	29.06%	16.11%	−10.22%	19.03%	121.14%
XGBoost Long	−4.66%	−2.54%	10.70%	1.58%	13.38%	30.29%	14.67%	−2.83%	17.74%	102.49%
XGBoost Short	−12.56%	1.85%	14.59%	4.34%	25.42%	**37.60%**	9.77%	−13.69%	13.57%	97.73%
DTE-2SVM	**0.35%**	**12.64%**	11.67%	9.51%	**31.52%**	16.99%	23.78%	**−2.71%**	18.81%	**204.31%**
Buy & Hold	−0.69%	11.24%	**18.42%**	−7.01%	28.71%	15.29%	**28.79%**	−19.95%	24.72%	132.11%

Table 4.7: DTE-2SVM vs Benchmark Algorithms: ROI over all the test periods

The results presented in Table 4.7 highlight the strong performance of the proposed DTE-2SVM model, which consistently surpasses other strategies, including the well-established Buy and Hold approach. Achieving a cumulative ROI of 204.31%, DTE-2SVM demonstrates its ability to outperform traditional single-model classifiers such as SVM, k-NN, Random Forest, and XGBoost.

One of the most significant findings is that the DTE-2SVM surpasses the Buy & Hold strategy, a noteworthy achievement given the difficulty of outperforming this strategy in the S&P 500 over the long term (see Figure 4.5). Due to the inherent upward trend of the S&P 500, many strategies fail to outpace the Buy & Hold in bullish periods. This makes DTE-2SVM's overall outperformance particularly impressive, as it not only keeps up in high-return years but also excels in managing risk during downturns.

Model ROI %	2015	2016	2017	2018	2019	2020	2021	2022	2023	Cum.
Long Trend	−3.49%	9.98%	**18.42%**	−7.01%	28.71%	15.29%	**28.79%**	−19.95%	**23.73%**	121.24%
Short Trend	**1.00%**	12.37%	9.61%	−0.02%	26.83%	5.01%	12.92%	**0.78%**	18.36%	123.12%
DTE-2SVM	0.35%	**12.64%**	11.67%	**9.51%**	**31.52%**	**16.99%**	23.78%	−2.71%	18.81%	**204.31%**

Table 4.8: ROI of Long and Short Trend Models that compose DTE-2SVM

For example, in challenging market years like 2018 and 2022, where the Buy & Hold strategy recorded negative returns, the DTE-2SVM not only delivered a positive return in 2018 (see Figure 4.6),

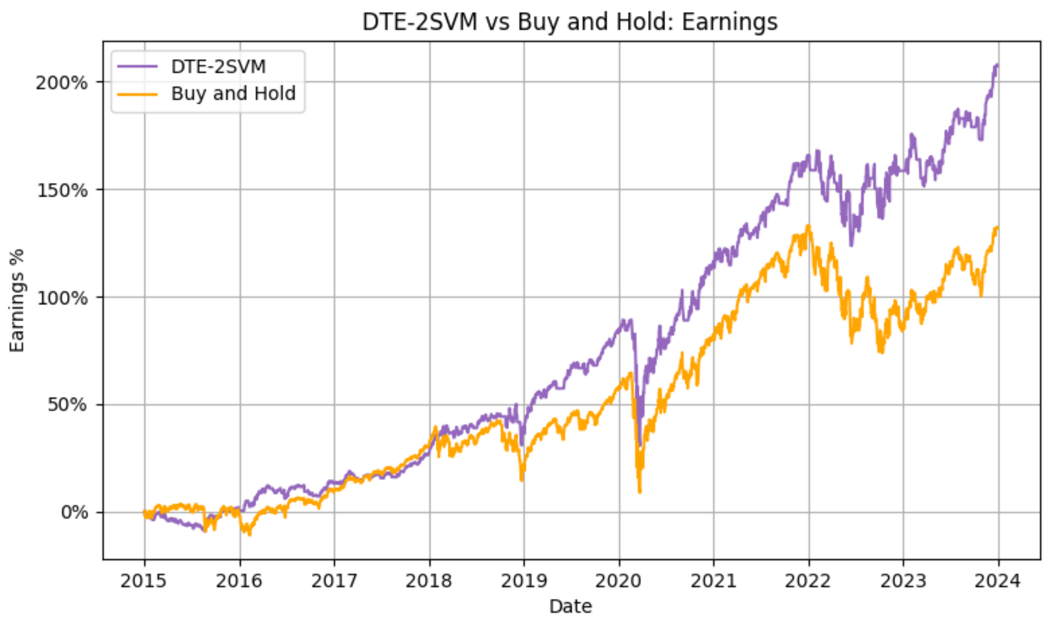


Figure 4.5: DTE-2SVM vs Buy and Hold: Earnings

but also significantly minimized losses in 2022 (see Figure 4.8). When looking at the results of each strategy separately (see Table 4.8) we can see that the key for the positive results of the DTE-2SVM is the action of its short trend model, which achieves great results in both years in contrast with the long trend model.

Not only reflects the effect of the dual model strategy, but also of the ensemble strategy, which is crucial to ensure that the models are used in the situations they are more optimized for. Analyzing more in detail the year 2018, we can see that the ensemble strategy has a very important role in the superior results of the proposed model.

Both the long trend and short trend strategies individually produced negative results in 2018, although the short trend model managed to significantly reduce losses. However, the final DTE-2SVM model achieved an impressive return of 9.51% for that year. This underscores the value of the ensemble strategy, as it successfully combined the strengths of both models to deliver positive results. While other models like XGBoost also performed similarly for this particular year, the dynamic ensemble approach proposed demonstrates its effectiveness in enhancing performance through specialized predictions.

This highlights the DTE-2SVM's capability to not only reduce losses but also capture gains during bearish markets, making it particularly beneficial for investors seeking more consistent returns across varying market conditions. Unlike many other strategies, which struggled during downturns like in 2018 and 2022, the ensemble approach proved resilient, especially in challenging market environments where the S&P 500 experienced significant draw-downs, creating a layer of risk management in the model.

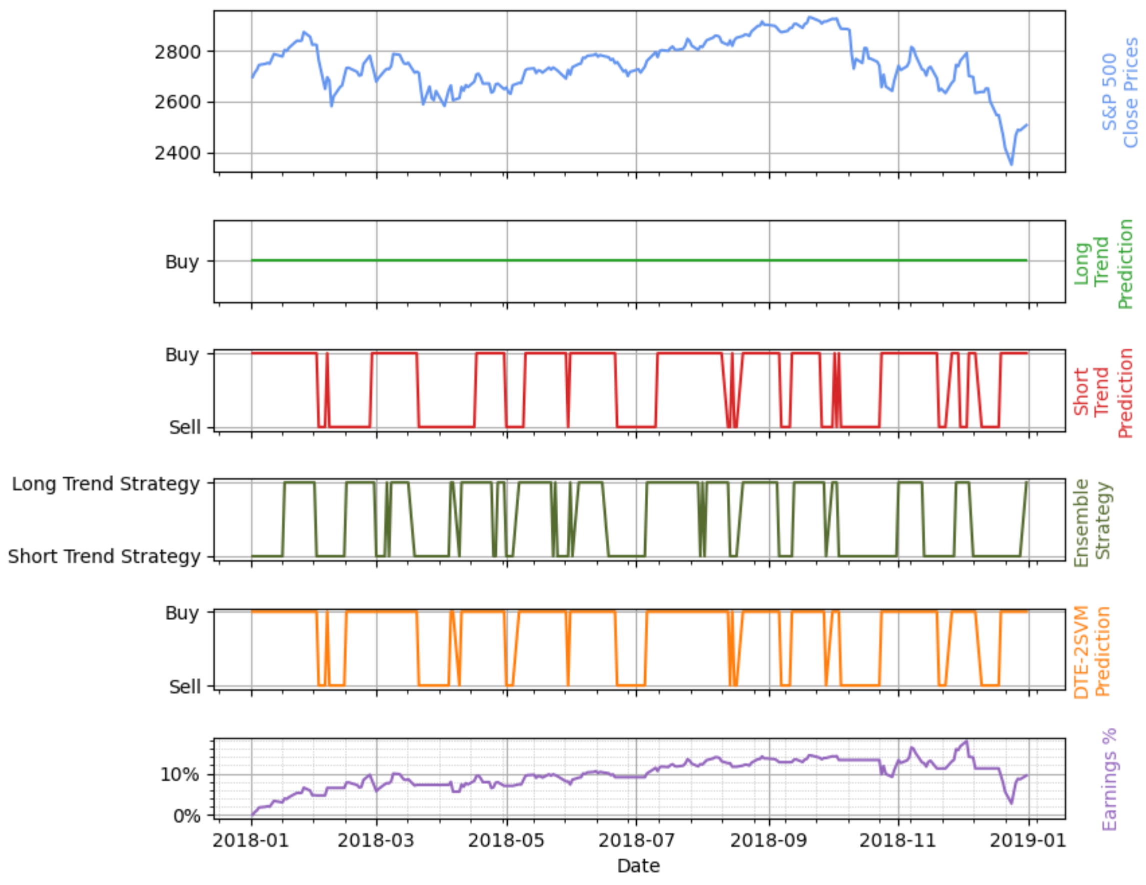


Figure 4.6: DTE-2SVM 2018 Results

Moreover, DTE-2SVM does not sacrifice performance in bullish years. In 2019 and 2021, which were strong years for the S&P 500, the model posted returns of 31.52 % and 23.78%, respectively, effectively keeping up with or outperforming Buy & Hold's high returns of 28.71% and 28.79%.

In periods like 2019 (see Figure 4.7), both the long and short trend strategies heavily favor buy actions, with the long trend adopting a Buy & Hold approach that captures the overall upward momentum. The short trend strategy, being more reactive, signals sell actions during brief periods of price decline. Since the ensemble strategy incorporates a 10-day SMA, it reacts to these fluctuations and tends to align with the short trend during downturns, such as the period between May and July 2019.

However, during broader upward trends, the market typically rebounds quickly from these brief downtrends. In such cases, the Buy & Hold strategy, recovers strongly and becomes preferable. The ensemble strategy's ability to quickly adapt by shifting focus back to the long trend allows it to benefit from the short trend's loss-cutting capability while capitalizing on the larger upward movement. This balance between reactivity and long-term focus is crucial, as the long trend's buy bias closely mirrors the behavior

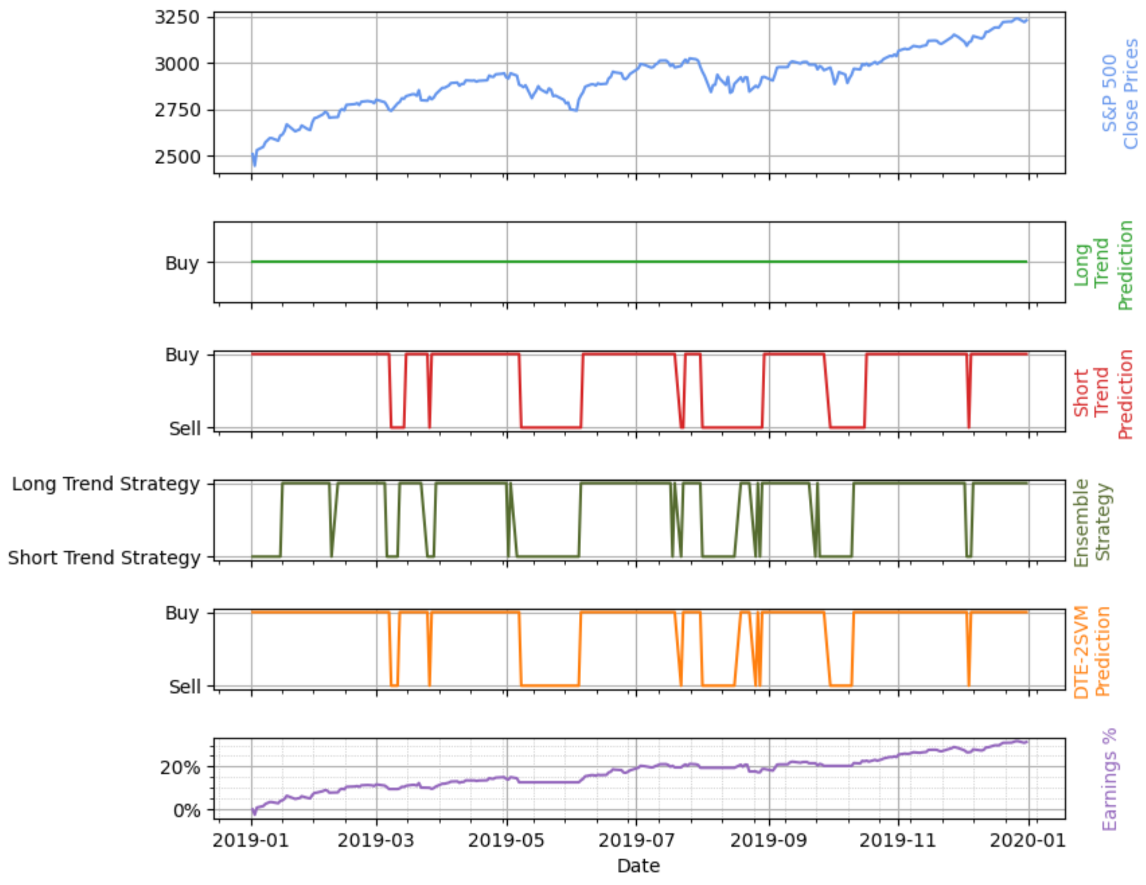


Figure 4.7: DTE-2SVM 2019 Results

of the S&P 500, enabling the DTE-2SVM to both mitigate short-term losses and capture long-term gains, which is essential to keep-up with the Buy & Hold strategy in these periods of upward trends where it is very strong.

Even so, the long trend model's bias towards the buy class, due to the absence of class weights, doesn't always lead to optimal outcomes. For example, when examining the performance in 2022, we observe that while the DTE-2SVM surpasses the Buy & Hold strategy, its results are inferior to those of the short trend strategy alone, and therefore it does not compliment it beneficially.

The 2022 S&P 500 exhibited a clear downtrend throughout the year, which the long trend strategy failed to capture due to its strong bias toward buying. This led to suboptimal predictions for that period. While the ensemble strategy primarily favored the short trend model, mitigating some of the negative impact as shown in Figure 4.8, the long trend's occasional influence still resulted in inaccurate predictions, harming overall performance. In a year with sustained bearish conditions like 2022, it is unrealistic for a long trend strategy to maintain a buy signal consistently.

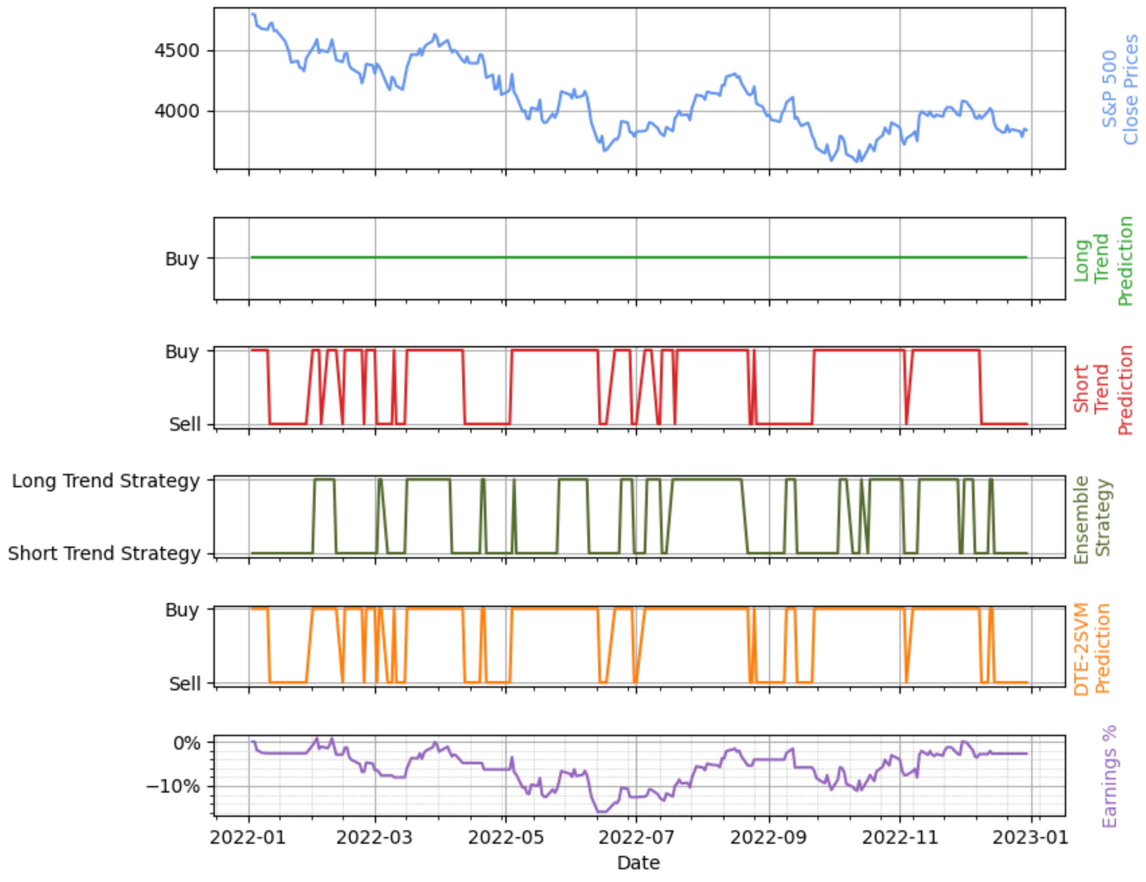


Figure 4.8: DTE-2SVM 2022 Results

The short trend model does also have some limitations. When analyzing the year 2020, we observe that DTE-2SVM outperforms the Buy & Hold strategy, however when comparing with the other benchmarked algorithms, the other algorithms and strategies not only outperform the Buy & Hold but also outperform DTE-2SVM by a substantial margin.

Between March and April 2020, a clear downtrend associated with the Covid-19 crisis is observed. During this period, the ensemble strategy appropriately shifts to the short trend model as the market enters a volatile downturn. However, the short trend model predicts a market recovery too early, which negatively impacts results during the sustained downtrend (see Figure 4.9). This highlights a key insight: the short trend model is adept at entering the market following a severe crash, which is beneficial, but in cases of extreme crashes, it may enter too soon and miss opportunities to further mitigate losses. Balancing reactivity to capture gains during a recovery with stability to avoid premature entry is a challenging task. The DTE-2SVM model was designed to address such scenarios and performs well in most cases. However, during this particular severe crash, it fell short, as it still experienced losses and did not

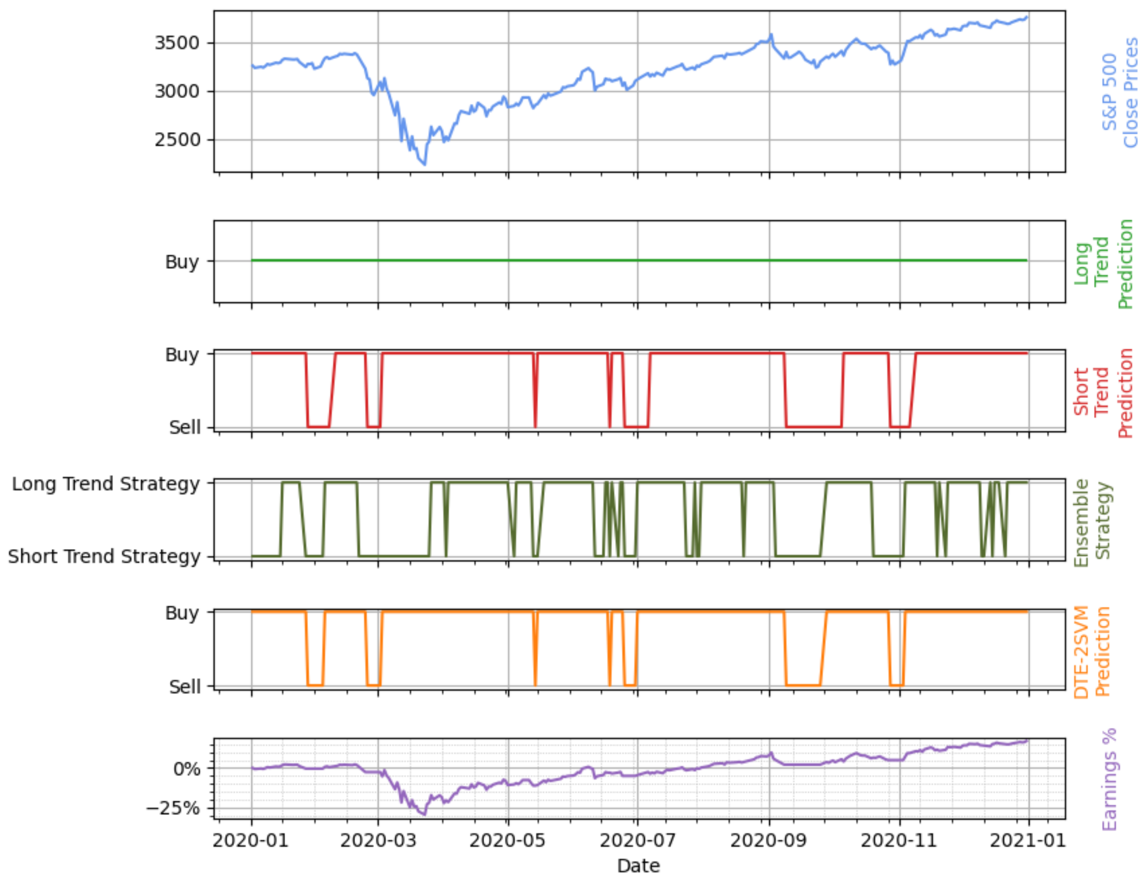


Figure 4.9: DTE-2SVM 2020 Results

capitalize as effectively as other strategies.

The Table 4.9 and Table 4.10 tables provide valuable insights into the performance of various algorithms, particularly highlighting the limitations of accuracy as a metric in stock market prediction tasks. Despite relatively high accuracy across most models, the results demonstrate that accuracy does not correlate well with actual performance, as the models present very similar accuracy values and yet their profitability results are very different. This reinforces the decision to use the F1-score as the primary metric during hyperparameter tuning, as it better captures the balance between false negatives and false positives, which is crucial in financial contexts where misclassification can have significant consequences.

Interestingly, while the DTE-2SVM algorithm shows superior performance in terms of ROI, it does not achieve the highest F1-scores. This discrepancy can be primarily attributed to specific years like 2018 and 2022, both characterized by strong downtrends. During these years, the biased long-trend SVM in DTE-2SVM produces outcomes misaligned with the true labels, leading to a significant drop in

Model Acc. %	2015	2016	2017	2018	2019	2020	2021	2022	2023	Avg.
SVM Long	**63.89%**	78.57%	**81.67%**	33.47%	82.14%	78.26%	75.00%	49.00%	81.60%	69.29%
SVM Short	55.16%	76.19%	76.49%	34.66%	78.97%	71.54%	73.41%	51.39%	78.80%	66.29%
k-NN Long	59.13%	75.40%	73.31%	35.06%	72.22%	73.12%	67.46%	**56.97%**	74.40%	65.23%
k-NN Short	59.92%	74.60%	79.68%	29.88%	79.76%	**81.42%**	75.40%	48.21%	81.20%	67.79%
Random Forest Short	54.76%	76.98%	78.88%	**39.44%**	79.37%	76.28%	75.40%	51.39%	80.80%	68.14%
Random Forest Long	61.90%	78.57%	78.49%	34.26%	76.98%	71.15%	61.90%	52.99%	79.60%	66.20%
XGBoost Long	58.73%	78.17%	77.29%	35.46%	76.98%	71.15%	65.08%	49.40%	75.60%	65.32%
XGBoost Short	57.14%	75.79%	77.29%	38.65%	80.16%	75.89%	74.60%	48.21%	82.00%	67.75%
DTE-2SVM	57.94%	**82.94%**	80.08%	24.70%	**87.70%**	79.84%	**82.14%**	42.63%	**86.00%**	**69.33%**

Table 4.9: DTE-2SVM vs Benchmark Algorithms: Accuracy over all the test periods

F1-score. This suggests that while the DTE-2SVM may excel in optimizing returns and controlling risk in high volatility periods, its predictive balance is compromised during these extreme downward trend market conditions, as reflected in its lower F1-scores for those periods, consequence of the lack of class balance in the long trend.

Model F1 %	2015	2016	2017	2018	2019	2020	2021	2022	2023	Avg.
SVM Long	71.47%	86.84%	94.76%	**44.10%**	93.22%	86.55%	96.30%	46.01%	86.62%	78.43%
SVM Short	**81.41%**	87.25%	94.76%	**44.10%**	93.22%	86.55%	96.30%	46.01%	86.62%	**79.58%**
k-NN Long	69.35%	82.22%	85.65%	24.89%	83.86%	81.54%	84.09%	42.51%	82.83%	70.77%
k-NN Short	68.34%	85.08%	88.15%	21.62%	88.12%	84.72%	85.38%	45.13%	88.24%	72.75%
Random Forest Long	70.34%	85.03%	87.96%	22.94%	87.08%	81.82%	83.37%	**47.53%**	86.86%	72.55%
Random Forest Short	65.58%	85.48%	85.30%	22.33%	86.72%	84.86%	85.17%	43.06%	87.74%	71.80%
XGBoost Long	70.87%	87.56%	88.99%	25.42%	88.73%	80.41%	86.71%	47.11%	87.43%	73.69%
XGBoost Short	69.33%	84.62%	86.12%	23.74%	87.28%	84.55%	87.06%	42.45%	86.03%	72.35%
DTE-2SVM	74.31%	**94.12%**	**98.38%**	3.14%	**99.00%**	**87.56%**	**99.00%**	35.95%	**100.00%**	76.83%

Table 4.10: DTE-2SVM vs Benchmark Algorithms: F1-score over all the test periods

4.4.3 Key Findings

The case study assesses the dual model architecture, DTE-2SVM, which addresses separately long-term and short-term market trend predictions, comparing it to traditional single-model approaches like SVM, k-NN, Random Forest, and XGBoost. Results show that DTE-2SVM significantly outperforms these models and the well-established Buy & Hold strategy. Its strength lies in effectively managing risk and capitalizing on market movements, particularly during challenging periods like 2018 and 2022, when the Buy & Hold strategy recorded negative returns, while also being able to follow the typical S&P 500 upward trend, designed for the Buy & Hold strategy. This highlights the model's versatility in both favorable and unfavorable market conditions, proving it to be a robust and adaptive solution compared to more static single-model approaches.

The DTE-2SVM's success is attributed to its dynamic ensemble strategy, which optimally shifts between the long and short trend models based on market conditions. However, while the long trend model's bias towards buying, derived from the absence of class weights, does have the benefit of creating a reactive effect to market recover uptrends together with the ensemble strategy, it can also lead to suboptimal performance during sustained downtrends, as seen in 2022 and validated by the f1-scores (see Table 4.10).

Despite these limitations, the DTE-2SVM model balances short-term loss mitigation and long-term gain capture, presenting high profitability while delivering consistent results across various market environments.

4.5 Case Study 3: Dynamic Ensemble Model for Long Trend Class Imbalance

In this case study, we will evaluate an algorithmic approach to further address the inherent data complexity of the stock market and class imbalance that is predominant in the S&P 500 context due to its growing tendency, specifically, for a long trend strategy. For this purpose, we propose an algorithm based on the previously presented study that proposed a Dynamic Ensemble Model to address data complexity and class imbalance issues for classification tasks [46].

4.5.1 Dynamic Ensemble Model Implementation

Taking the preprocessed stock market data as input D, the goal of this algorithm is to decompose D into simpler sub-problems. Thus, the Euclidian MST of D is computed. The system leverages the *scikit-learn* module to compute the Euclidean distances between the data points, and the *scipy* module that computes the MST on top of the computed Euclidean distances. With the Euclidean MST built, we now have a network of the whole dataset with the minimum total distances, meaning we can now estimate decision boundaries by finding, in the MST, the edges joining two different classes, thus creating sub-problems.

To build the sub-problems, we add both edges to the sub-problem and use the k-NN algorithm on each edge to elect the k nearest neighbors of the same class as the edge used as the argument of the algorithm. An example of a sub-problem formulation is shown in the following Figure 4.10.

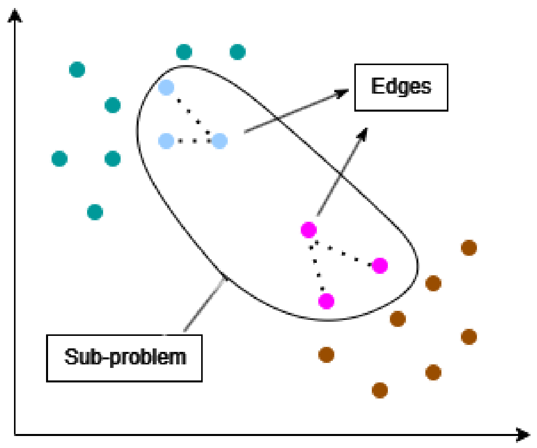


Figure 4.10: Sub-problem Formulation Example for k = 2

Whenever there is at least an instance in both classes that belongs to two different sub-problems, we can estimate that they belong to the same decision boundary, and therefore we merge the sub-problems [46].

Upon constructing the sub-problems, a RBF-SVM is fit in each sub-problem, to construct the component classifiers, building a model for each sub-problem.

After formulating the component classifiers, an ensemble technique is required to combine the predictions. Therefore, a dynamic ensemble is employed by assigning distance based weights to each classifier to predict the class value of each test data point, t giving a higher influence to component classifiers closer to t.

1. The class value for t is predicted by each model m, where $i = 1 \ldots N$ and $N =$ number of models generated.

2. The weight of each model m_i for test data point t is given by the inverse distance method:

$$W_{i,t} = \frac{1}{Distance_{i,t}^{\beta}} \tag{4.2}$$

Where, $W_{i,t}$ represents the weight of the class value predicted by model m_i for the test instance t; $Distance_{i,t}$ is the distance between t and model m_i; β is the power parameter that controls how quickly the influence of each point decreases as the distance increases. Smaller distances produce higher weights

3. The final predicted class value for t is determined by finding the class c that maximizes the sum of weights Wi_t across the classes in $C1$ and $C2$:

$$ClassValue(t) = \{c \mid \max(\sum_{c \in C1, C2} W_{i,t})\} \tag{4.3}$$

An hyperparameter tuning process is included in the algorithm, to search for the optimal k and β, while also tuning each base classifier hyperparameters, which for this case, since the base classifiers are SVMs, searches for optimal C and γ hyperparameters.

4.5.1.A Parameter Research

The selection of the k value, for the k-NN algorithm, plays a critical role in determining the algorithm's performance and behavior. The k value essentially controls the number of sub-problems created as a higher value aggregates more data points to the initial sub-problems and with the step of merging sub-problems it creates a snowball effect with higher values creating less and less sub-problems until reaching the start point which is one sub-problem (the whole dataset D). Therefore, because the stock market has particularly high data complexity and class imbalance, a study on the sub-problems formed for each k value was performed using the train period $[1995, 2014]$ with the long trend features and labels (see Table 4.11).

k	2	3	4	5	6	7	8
Sub-problems	263	203	130	73	49	37	20

Table 4.11: Sub-problems relation with k value

The study revealed that for the stock market context, the number of sub-problems is very high, which was expected due to the fact that higher data complexity leads to more sub-problems as the classes are very difficult to separate [46].

When analyzing the sub-problems for a k value of 5, the issue becomes evident: one sub-problem contains the vast majority of elements from D, while the remaining sub-problems hold significantly fewer elements (see Figure 4.11).

For the purposes of this approach, the creation of sub-problems is essential to address both class imbalance and data complexity within the dataset. To ensure each sub-problem's relevance, a threshold-based method is employed to assess their significance. This involves identifying the sub-problem with the maximum number of elements and evaluating whether the other sub-problems meet a predefined element percentage threshold. In this case, a sub-problem is considered relevant if it contains at least

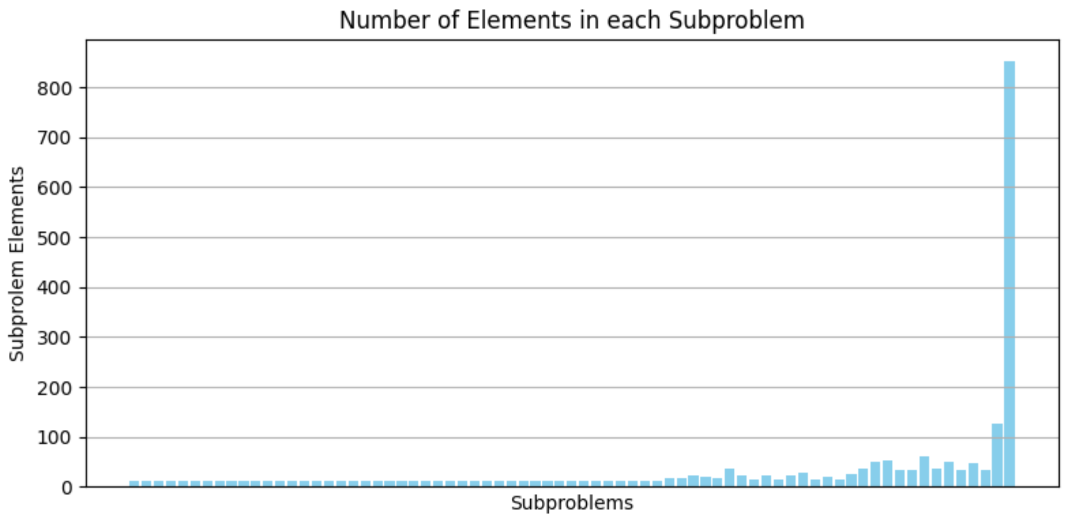


Figure 4.11: Number of Elements in each sub-problem for k = 5

20% of the elements of the largest sub-problem. sub-problems that do not meet this criterion are merged with their nearest sub-problem to maintain the integrity of the model while reducing the impact of less significant clusters.

k	2	3	4	5	6	7	8
Sub-problems	263	67	18	3	3	3	2

Table 4.12: Sub-problems with 20% relevance threshold

The threshold-based method significantly reduces the number of sub-problems for each k value, highlighting the imbalance among the elements of these sub-problems. For $k = 2$, however, this imbalance does not occur, indicating a balanced set of sub-problems. Nonetheless, the total number of sub-problems remains substantial, (see Table 4.12).

This is further confirmed by analyzing the number of elements in each sub-problem for the example k value of 5, where a balance in number of elements is identified, confirming that the threshold successfully identified and merged the sub-problems (see Figure 4.12).

As such, we choose the values 5,6,7 and 8 as candidates for k value, which will be evaluated and tested in the hyperparameter tuning process in order to choose the optimal k value for the test scenarios using the training period.

4.5.2 Discussion of Results

On Table 4.13 we present the results for the Dynamic Ensemble Model (DEM), SVM with balanced class weights and the SVM model used for long trend prediction in the DTE-2SVM. These results represent

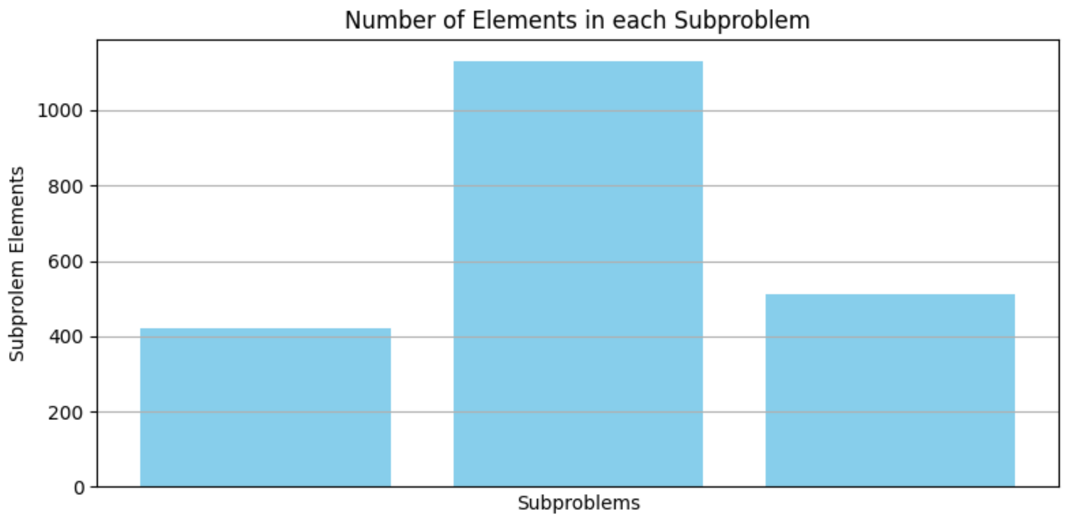


Figure 4.12: Number of Elements in each sub-problem for k = 5 using 20% relevance threshold

the prediction of S&P 500 index using the long trend features (see Table 3.2) and labels (see Table 3.1), formulating candidates for the long trend model used in the DTE-2SVM.

Model ROI %	2015	2016	2017	2018	2019	2020	2021	2022	2023	Cum.
SVM (Class Weights)	-2.28%	2.45%	15.99%	$\mathbf{-2.18\%}$	$\mathbf{28.71\%}$	$\mathbf{57.90\%}$	24.69%	$\mathbf{0.00\%}$	0.00%	$\mathbf{187.85\%}$
DEM	$\mathbf{-0.69\%}$	4.67%	$\mathbf{18.42\%}$	-7.01%	$\mathbf{28.71\%}$	30.66%	28.51%	-10.59%	2.28%	126.23%
SVM (DTE-2SVM)	-3.49%	$\mathbf{9.98\%}$	$\mathbf{18.42\%}$	-7.01%	$\mathbf{28.71\%}$	15.29%	$\mathbf{28.79\%}$	-19.95%	$\mathbf{23.73\%}$	121.24%

Table 4.13: ROI of Long Trend Models with Class Imbalance Strategies

Both class imbalance mitigation strategies—Dynamic Ensemble Model (DEM) and class-weighted SVM, demonstrate significant improvements in the long trend model's ROI. Notably, the class-weighted SVM achieves an impressive final result of 187.85%. The DEM, on the other hand, shows superior profitability during years characterized by sideways or uncertain market conditions, such as 2015 and 2016, where trends were not clearly defined. This highlights DEM's strength in handling data complexity, as it better distinguishes subtle patterns in these ”indecisive” years compared to the class-weighted SVM.

In contrast, during years with well-defined market trends, such as 2020 and 2022, both class imbalance strategies, particularly the class-weighted SVM, outperform the biased SVM utilized in the DTE-2SVM. For example, in 2020, a year marked by a significant downward trend followed by a strong upward recovery, and in 2022, a year dominated by a clear downtrend with fluctuations, the class-weighted SVM not only captured the large price decline but also avoided the downward trend entirely.

This ability to effectively capitalize on sharp market movements gives the class-weighted SVM a substantial edge over the biased SVM and also the DEM.

However, in 2023, a year marked by a consistent upward trend, both the DEM and the class-weighted SVM struggled to accurately predict the market's direction, resulting in missed profit opportunities. The upward trend of 2023 should have triggered a strong buy signal for the long trend model, a situation in which the biased SVM used in the DTE-2SVM regained its competitive edge, capitalizing on the clear upward movement where the other approaches faltered.

Model ROI %	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**	**2022**	**2023**	**Cum.**
SVM (Class Weights)	0.44%	7.60%	9.68%	6.25%	30.33%	-1.25%	21.83%	-14.15%	-6.02%	59.33%
DEM	$\mathbf{1.25\%}$	8.78%	$\mathbf{11.67\%}$	$\mathbf{9.51\%}$	$\mathbf{31.52\%}$	12.53%	21.35%	$\mathbf{-0.84\%}$	-6.02%	125.43%
DTE-2SVM	0.35%	$\mathbf{12.64\%}$	$\mathbf{11.67\%}$	$\mathbf{9.51\%}$	$\mathbf{31.52\%}$	$\mathbf{16.99\%}$	$\mathbf{23.78\%}$	-2.71%	$\mathbf{18.81\%}$	$\mathbf{204.31\%}$

Table 4.14: ROI of DTE-2SVM with the Class Imbalance Strategies integrated in Long Trend Strategy

When replacing the models in the analysis with the long trend model within the DTE-2SVM architecture, the results reveal several important insights (see Table 4.14). Notably, the performance of the class-weighted SVM experiences a significant decline compared to its solo application and in contrast to the DTE-2SVM. Upon closer examination, this underperformance can be attributed to the model's inability to replicate the exceptional results observed in 2020. This issue stems from a bottleneck in the short trend model, where the ensemble strategy signals a short trend entry, but the predictions fail to capitalize on the downtrend during this period (see Figure 4.9).

The 2022 results are similarly underwhelming, offering further insights into the DTE-2SVM's mechanics and explaining the poor performance of the class-weighted SVM. In 2022, the class-weighted SVM identified a downtrend and recommended staying out of the market (see Table 4.14). This means that whenever the ensemble strategy requested long trend predictions, there was a high probability of an upward momentum, where the long trend model should have signaled a buy. The negative returns in 2022 suggest that the long trend model failed to capture these uptrends, leading to overall poor performance. This also reveals the underlying mechanics of the DTE-2SVM: the short trend strategy navigates volatile downward periods, while the long trend model stabilizes performance by capturing the SP500's high-growth phases with its inherent bias toward buy signals, which aligns with the higher probability of upward trends signaled by the ensemble strategy and therefore, capitalizing on these periods of growth.

When we examine the DEM model within the DTE-2SVM architecture, we find that while its overall performance is similar to its standalone results, the year-by-year outcomes vary significantly. As a single model, the DEM excels in years with strong upward trends and delivers solid performance in sideways markets. However, it struggles during periods of extreme downtrends, failing to effectively limit losses. This contrasts sharply with its performance within the DTE-2SVM, where the DEM shines during downtrend years, mitigating risks and even turning volatility into gains, lightly improving the target year of 2022 where the proposed long trend strategy failed to capture the year's trend. However, the trade-off is a reduced ability to match the high returns of strong upward trend years, highlighting the reliance of the DTE-2SVM's approach on a long trend signal with a high probability of buy signals.

Overall, while the DTE-2SVM is adept at handling volatility and downturns, it misses opportunities for maximum gains in years like 2020, where the class-weighted SVM capitalized on these opportunities to generate far higher returns. This indicates that market conditions like 2020 act as a bottleneck for the DTE-2SVM strategy. Despite that, the DTE-2SVM still surpassed the Buy & Hold strategy for this year, thus highlighting its robustness. As such, the class imbalance remains a valuable feature for the long trend model, as it captures the high growth potential of the SP500. When the long trend model is applied in favorable conditions, this bias toward buy signals increases the likelihood of profitability. By having an ensemble strategy that enhances the probability of the long trend model being used in this favorable conditions, the model capitalizes on the bias towards the buy signal.

4.5.3 Key Findings

The case study revealed the effectiveness of both class imbalance mitigation strategies in predicting the stock market as a standalone solution, demonstrating superior cumulative returns than the proposed long trend model. However, when incorporated into the DTE-2SVM framework, both these strategies failed to improve on the proposed long trend model within the developed dual model approach. This revealed the effectiveness and importance of capturing the bias towards a buy signal for the long trend signal within the context of the DTE-2SVM framework, as the ensemble strategy favors the long trend predictions in periods of increased probability of higher returns.

Even so, these strategies highlighted the bottleneck of the proposed system as it failed to capitalized gains in 2020, identifying these market conditions as a profitability bottleneck for this system as the class imbalance mitigation strategies achieve much higher returns in these conditions. Thus indicating that in order to improve the model a joint solution considering ensemble strategy and class imbalance for long trend strategies must be considered

4.6 Evaluation Summary

The evaluation process was performed in three case studies, chosen due to their relevance and impact in the proposed solution, while also showcasing and discussing the solution's advantages and limitations, while actively proposing and discussing mitigation strategies.

- In Case Study 1, the Dynamic Genetic Algorithm (DGA) for hyperparameter tuning showed notable advantages over the standard Genetic Algorithm (GA), particularly in maintaining greater population diversity and reducing execution time through adaptive mutation and crossover control. However, the DGA did not consistently reach the optimal solutions achieved by Grid-Search, which, despite its significantly higher computational demands, provides guaranteed precision and interpretability. As a result, Grid-Search is selected as the preferred tuning method in subsequent case studies, especially suited for smaller search spaces requiring optimal and interpretable results. Nonetheless, the successful implementation of the DGA opens up opportunities for exploring a larger search space with finer granularity within defined boundaries, potentially leading to improved solutions.
- In Case Study 2 the proposed DTE-2SVM architecture was evaluated, a dual model strategy designed to separate long-term and short-term market trend predictions with a dynamic trend-based ensemble. This model significantly outperformed single-model approaches like SVM, k-NN, Random Forest, and XGBoost, while also surpassing the S&P 500 benchmark Buy & Hold trading strategy. The DTE-2SVM proved particularly effective in mitigating risk during challenging market periods while capitalizing on the typical S&P 500 uptrend, showcasing its adaptability and providing stable and continuous returns with risk control capabilities. However, its reliance on a bias toward buying in the long trend model led to underperformance during sustained downtrends, indicating room for improvement in handling bearish markets.
- In Case Study 3, class imbalance strategies, such as the class-weighted SVM and a DEM, were shown to be effective standalone solutions, achieving considerable cumulative returns. However, their incorporation into the DTE-2SVM framework did not outperform the proposed long trend model. While the DEM enhanced the long trend strategy's effectiveness and predictions during clear downtrends, it compromised the model's responsiveness and ability to capitalize on high-return periods. This highlighted the importance of the buy signal bias in the ensemble strategy within the DTE-2SVM framework, while showing the trade-off between approaches. Additionally, these class imbalance strategies exhibited potential for enhancing profitability, revealing a bottleneck in the system's capacity to leverage gains in specific market conditions, such as those experienced in 2020.

5 Conclusions and Future Work

Contents

In this chapter, we summarize the key findings and achievements of the research, focusing on the results of the proposed DTE-2SVM. Finally, we discuss the system's limitations and outline potential directions for future work.

5.1 Conclusions

The presented work proposes a novel approach to stock market prediction, by dividing and identifying long and short trends within the market, creating a specialized model for each with a dynamic ensemble approach based on a trend technical indicator, the DTE-2SVM.

The proposed solution and its architecture were validated through case studies, designed to address the key aspects of the system: hyperparameter tuning, overall model effectiveness, and the stock market's inherent class imbalance. A meaningful test framework was developed, utilizing backtesting and walk-forward methodologies, and evaluating the solution over a significant period $[2015, 2023]$ using the S&P 500 index, accounting for various market dynamics and behaviors.

The proposed solution effectively identifies and labels market trends through the novel NPMM data labeling strategy, which not only identifies local minima and maxima as critical points but also establishes a relationship between these critical points generating labels that are then used to label the data ”outside” the critical points. This is reflected in the presented results, where the proposed DTE-2SVM effectively identifies the market trends, capturing the different dynamics of long and short-term trends, resulting in improved profitability results when compared to commonly used single model machine learning approaches. Moreover, its superior profitability compared to the benchmark Buy & Hold strategy, a strong benchmark for the S&P 500 index, is particularly noteworthy. The results highlight the robustness of the DTE-2SVM, especially in its risk management during adverse market years, where it successfully avoided major losses (greatly improving on the Buy & Hold strategy), offering a stable and profitable environment that builds confidence for potential end-users.

5.2 System Limitations and Future Work

The following section discusses the identified limitations of the proposed system, highlighting areas that could be improved, suggesting a path for future work.

- **Improve Feature Engineering**: The proposed system uses a fixed set of features, based on literature review, for both long trend and short trend models. Although this approach ensured stability and improved model interpretability throughout the development and assessment processes, a more dynamic feature engineering process may enhance the overall system’s performance. For instance, a dedicated feature engineering process for each trend, may improve each model’s performance by using dedicated and more relevant features (technical indicators) that have better performance in short or long trend.

- **Increase Hyperparameter Tuning Search Space**: Given the positive indicators of the Dynamic Genetic Algorithm implementation, a study on enhancing the literature-based search space by increasing granularity during the search process shows potential to uncover better solutions within the defined boundaries, leveraging the DGA’s adaptive capabilities to optimize hyperparameter tuning further in search spaces where the Grid-Search would be computationally prohibitive.

- **Address Long Trend Class Imbalance**: The proposed system leveraged class imbalance to its advantage, particularly in the long trend strategy. However, this approach proved to be imperfect, leading to some suboptimal results and diminished predictive performance. This decision was based on the premise that the model is designed specifically for predicting the S&P 500 index, which limits its scalability to other markets which, altogether, formulates a key system limitation. Striking the right balance for the long trend is crucial, while it can remain biased toward the predominant class (reflecting the dataset), the model must not compromise its ability to effectively detect clear downtrends.

- **Improve Dynamic Ensemble Robustness**: The proposed system uses a dynamic ensemble based on a 10-day SMA trend indicator. While this approach was largely successful, it shows potential for improvement. When tested with other base models, the final performance degraded, highlighting issues with its adaptability. This decline can be attributed to the different behaviors of the base models clashing with the ensemble strategy, suggesting that the current ensemble method may be too closely tailored to the specific models and strategies used in this project, thus limiting its scalability and generalization capabilities. Additionally, the ensemble relies on a threshold involving the SMA and open price, which tends to cause excessive model switching in indecisive market conditions, creating prediction spikes. Future enhancements could involve developing a more robust ensemble, possibly incorporating a metamodel that adapts to current market conditions, improving stability and performance across varying scenarios.
- **Interface Layer for Real-Time Trading**: The interface layer of the system was not a primary focus of this project and, as a result, is not fully optimized for real-time trading applications. Future work should aim to enhance this layer in order to prepare the system for direct client use, enabling seamless integration with trading platforms and ensuring it meets the demands of live trading environments.

Bibliography

[1] M. Kumbure, B. Wickramatillake, A. Wijekoon, A. Karunananda, and H. Jayasekara, "Machine learning techniques and data for stock market forecasting: A survey," *Expert Systems with Applications*, vol. 197, p. 116659, 2022.

[2] M. Bansal, A. Goyal, and A. Choudhary, "Stock market prediction with high accuracy using machine learning techniques," *Procedia Computer Science*, vol. 215, pp. 247–265, 2022.

[3] G. Bonanno, N. Vandewalle, and R. N. Mantegna, "Taxonomy of stock market indices," *Phys. Rev. E*, vol. 62, pp. R7615–R7618, Dec 2000. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.62.R7615

[4] A. Picasso, S. Merello, Y. Ma, L. Oneto, and E. Cambria, "Technical analysis and sentiment embeddings for market trend prediction," *Expert Systems with Applications*, vol. 135, pp. 60–70, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419304142

[5] C. Worasucheep, "Ensemble classifier for stock trading recommendation," *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2001178, 2022. [Online]. Available: https://doi.org/10.1080/08839514.2021.2001178

[6] Z. Prohaska, I. Uroda, and S. Suljić, "Sp — a computer program for fundamental analysis of stocks," in *2011 Proceedings of the 34th International Convention MIPRO*, 2011, pp. 1051–1055.

[7] A. Silva, R. Neves, and N. Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2036–2048, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417414006113

[8] P. Uhr, J. Zenkert, and M. Fathi, "Sentiment analysis in financial markets a framework to utilize the human ability of word association for analyzing stock market news reports," *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 912–917, 2014.

[9] M. Phillips and T. Lorenz, ““dumb money” is on gamestop, and it’s beating wall street at its own game,” *The New York Times*, Jan 2021. [Online]. Available: https://www.nytimes.com/2021/01/27/business/gamestop-wall-street-bets.html

[10] K. Shan, “The dow theory.” *Aweshkar Research Journal*, vol. 13, no. 1, pp. 77 – 82, 2012. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&amp;db=bsu&amp;AN=75059305&amp;lang=pt-pt&amp;site=eds-live&amp;scope=site

[11] J. J. Murphy, *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.

[12] Y. Peng, P. H. M. Albuquerque, H. Kimura, and C. A. P. B. Saavedra, “Feature selection and deep neural networks for stock price direction forecasting using technical analysis indicators,” *Machine Learning with Applications*, vol. 5, p. 100060, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266682702100030X

[13] K. jae Kim and I. Han, “Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index,” *Expert Systems with Applications*, vol. 19, no. 2, pp. 125–132, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417400000270

[14] Y.-S. Chen, C.-H. Cheng, and W.-L. Tsai, “Modeling fitting-function-based fuzzy time series patterns for evolving stock index forecasting,” *Applied Intelligence*, vol. 40, no. 1, pp. 54–73, 2020.

[15] C. K.-S. Leung, R. K. MacKinnon, and Y. Wang, “A machine learning approach for stock price prediction,” in *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning*, 2014, pp. 27–35.

[16] C. Cortes and V. Vapnik, “Support-vector networks,” in *Machine Learning , vol. 20, no. 3*, 1995, p. 273–297.

[17] C. J. Burges, “A tutorial on support vector machines for pattern recognition,” *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[18] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines,” *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[19] I. Carmichael and J. Marron, “Geometric insights into support vector machine behavior using the kkt conditions,” *arXiv preprint arXiv:1704.00767*, 2017.

[20] A. Rosales-Pérez, S. García, J. A. Gonzalez, C. A. Coello Coello, and F. Herrera, “An evolutionary multiobjective model and instance selection for support vector machines with pareto-based ensembles,” *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 863–877, 2017.

[21] D. S. Chouhan, “Svm kernel functions for classification,” in *2013 International Conference on Advances in Technology and Engineering (ICATE)*. IEEE, 2013, pp. 1–9.

[22] H. Bhavsar and M. H, “A review on support vector machine for data classification,” *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 10, 2012.

[23] R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, B. Bischl, and A. C. P. L. F. de Carvalho, “Effectiveness of random search in svm hyper-parameter tuning,” in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.

[24] A. H. Ali and M. Z. Abdullah, “A parallel grid optimization of svm hyperparameter for big data classification using spark radoop,” *Karbala International Journal of Modern Science*, vol. 6, no. 1, p. 3, 2020.

[25] Y. Gong and L. Jia, “Research on svm environment performance of parallel computing based on large data set of machine learning,” *The Journal of Supercomputing*, vol. 75, no. 9, pp. 5966–5983, 2019.

[26] F. Cheng, J. Chen, J. Qiu, and L. Zhang, “A subregion division based multi-objective evolutionary algorithm for svm training set selection,” *Neurocomputing*, vol. 394, pp. 70–83, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220302034

[27] S. Saha, P. S. Sarker, A. A. Saud, S. Shatabda, and M. Hakim Newton, “Cluster-oriented instance selection for classification problems,” *Information Sciences*, vol. 602, pp. 143–158, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025522003826

[28] P. Moradi and M. Gholampour, “A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy,” *Applied Soft Computing*, vol. 43, pp. 117–130, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494616300321

[29] H. Banka and S. Dara, “Hamming distance based binary pso for gene expression data classification and feature selection,” *Pattern Recognition Letters*, vol. 52, pp. 94–100, 2015.

[30] C.-L. Huang and C.-J. Wang, “A ga-based feature selection and parameters optimizationfor support vector machines,” *Expert Systems with Applications*, vol. 31, no. 2, pp. 231–240, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417405002083

[31] A. Fatima, R. Maurya, M. K. Dutta, R. Burget, and J. Masek, “Android malware detection using genetic algorithm based optimized feature selection and machine learning,” in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019, pp. 220–223.

[32] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, “Evolutionary algorithms,” *WIREs Data Mining and Knowledge Discovery*, vol. 4, no. 3, pp. 178–195, 2014. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1124

[33] P. A. Vikhar, “Evolutionary algorithms: A critical review and its future prospects,” in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, 2016, pp. 261–265.

[34] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[35] A. V. Phan, M. L. Nguyen, and L. T. Bui, “Feature weighting and svm parameters optimization based on genetic algorithms for classification problems,” *Applied Intelligence*, vol. 46, pp. 455–469, 2017.

[36] S. Ghorbanpour, V. Palakonda, and R. Mallipeddi, “Multi-objective optimization of svm parameters for meta classifier design.” Daegu, Republic of Korea: Kyungpook National University, 2019. [Online]. Available: https://www.researchgate.net/publication/345317426

[37] S. Paul and S. Das, “Simultaneous feature selection and weighting–an evolutionary multi-objective optimization approach,” *Pattern Recognition Letters*, vol. 65, pp. 51–59, 2015.

[38] S. A. Stanhope and J. M. Daida, “Optimal mutation and crossover rates for a genetic algorithm operating in a dynamic environment,” *ERIM International Advanced Information Systems Group*, 1998.

[39] T.-P. Hong, H.-S. Wang, W.-Y. Lin, and W.-Y. Lee, “Evolution of appropriate crossover and mutation operators in a genetic process,” *Applied intelligence*, vol. 16, pp. 7–17, 2002.

[40] T. Chen, K. Tang, G. Chen, and X. Yao, “A large population size can be unhelpful in evolutionary algorithms,” *Theoretical Computer Science*, vol. 436, pp. 54–70, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397511001368

[41] C. Fernandes and A. Rosa, “Self-regulated population size in evolutionary algorithms,” in *International Conference on Parallel Problem Solving from Nature*. Springer, 2006, pp. 920–929.

[42] P. A. Diaz-Gomez and D. F. Hougen, “Initial population for genetic algorithms: A metric approach.” in *Gem*. Citeseer, 2007, pp. 43–49.

[43] H. Maaranen, K. Miettinen, and M. M. Mäkelä, “Quasi-random initial population for genetic algorithms,” *Computers & Mathematics with Applications*, vol. 47, no. 12, pp. 1885–1895, 2004.

[44] C. Xu and S. Zhang, "A genetic algorithm-based sequential instance selection framework for ensemble learning," *Expert Systems with Applications*, vol. 236, p. 121269, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423017712

[45] Z.-H. Zhou, *Ensemble Learning*. Singapore: Springer Singapore, 2021, pp. 181–210. [Online]. Available: https://doi.org/10.1007/978-981-15-1967-3_8

[46] S. B., M. Gyanchandani, R. Wadhvani, and S. Shukla, "Data complexity-based dynamic ensembling of svms in classification," *Expert Systems with Applications*, vol. 216, p. 119437, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422024563

[47] P. Bühlmann and B. Yu, "Analyzing bagging," *The Annals of Statistics*, vol. 30, no. 4, pp. 927 – 961, 2002. [Online]. Available: https://doi.org/10.1214/aos/1031689014

[48] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002200009791504X

[49] X. Li, L. Wang, and E. Sung, "Adaboost with svm-based component classifiers," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 5, pp. 785–795, 2008, constraint Satisfaction Techniques for Planning and Scheduling Problems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197607000978

[50] Z. H. Wang, Y. F. Liu, T. Wang, J. G. Wang, Y. M. Liu, and Q. X. Huang, "Intelligent prediction model of mechanical properties of ultrathin niobium strips based on xgboost ensemble learning algorithm," *Computational Materials Science*, vol. 231, p. 112579, 2024.

[51] J. Gu, S. Liu, Z. Zhou, S. R. Chalov, and Q. Zhuang, "A stacking ensemble learning model for monthly rainfall prediction in the taihu basin, china," *Water*, vol. 14, no. 3, 2022. [Online]. Available: https://www.mdpi.com/2073-4441/14/3/492

[52] H.-C. Kim, S.-B. Cho, and S.-H. Kim, "Support vector machine ensemble with bagging for iris data classification," in *International Workshop on Support Vector Machines*. Springer, 2002, pp. 402–408.

[53] A. Sharma and S. Dey, "A boosted svm based ensemble classifier for sentiment analysis of online reviews," *Expert Systems with Applications*, vol. 40, no. 8, pp. 2622–2629, 2013.

[54] J. Bektaş, "Eksl: An effective novel dynamic ensemble model for unbalanced datasets based on lr and svm hyperplane-distances," *Information Sciences*, vol. 597, pp. 182–192, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025522002535

[55] S. Pang, D. Kim, and S. Bang, "Face membership authentication using svm classification tree generated by membership-based lle data partition," *IEEE transactions on neural networks*, vol. 16 2, pp. 436–46, 2005.

[56] Y. Lin, H. Guo, and J. Hu, "An svm-based approach for stock market trend prediction," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–7.

[57] V. Khatibi, E. Khatibi, and A. Rasouli, "A new support vector machine-genetic algorithm (svm-ga) based method for stock market forecasting," *International Journal of the Physical Sciences*, vol. 6, no. 25, pp. 6091–6097, 2011.

[58] R. Kardile, T. Ugale, and S. N. Mohanty, "Stock price predictions using crossover sma," in *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. Noida, India: IEEE, Sep 2021, pp. 1–5.

[59] Y. Han, J. Kim, and D. Enke, "A machine learning trading system for the stock market based on n-period min-max labeling using xgboost," *Expert Systems With Applications*, vol. 211, p. 118581, 2023.

[60] R. Aroussi. (2024) yfinance: Yahoo! finance market data downloader. GitHub repository. [Online]. Available: https://github.com/ranaroussi/yfinance

[61] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445. Austin, TX, 2010, pp. 51–56.

[62] K. Johnson. (2024) Pandas ta - a technical analysis library in python 3. GitHub repository. [Online]. Available: https://github.com/twopirllc/pandas-ta

[63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[64] R. Sheftel. (2024) Market calendars to use with pandas for trading applications. GitHub repository. [Online]. Available: https://github.com/twopirllc/pandas-ta

[65] R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, B. Bischl, and A. C. P. L. F. de Carvalho, "Effectiveness of random search in svm hyper-parameter tuning," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[66] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.

[67] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[68] G. Varoquax. (2024) Joblib: running python functions as pipeline jobs. GitHub repository. [Online]. Available: https://github.com/joblib/joblib