Department of Business and Management

Chair of Machine Learning and Artifical Intelligence

An approach to optimize long - short equities using Fuzzy Logic and metaheuristic techniques.

Supervisor:

Prof.

Giuseppe Italiano

Co Supervisor:

Prof.

Antonio Simeone

Candidate:

Alessandro Ivashkevich

276871

Academic Year 2023/2024

Alla mia famiglia, ai miei cari.

Contents

Introduction

Today, the financial world is immersed in data, making it crucial to comprehend the intricacies of the mechanisms driving financial processes. This includes understanding the complex correlations between technical indicators, ratios, and price variations. The rise of globalisation, technological advancements, and regulatory changes has elevated this complexity to a new level.

This paper aims to provide a potential strategy to obtain long-short equity in the analysis of financial time series, thanks to the synergy of Fuzzy Logic and metaheuristic algorithms. This combination can deal with the uncertain nature of the financial market, thanks to the nature of these instruments, obtaining a flexible solution for complex queries. In fact, Fuzzy logic is often applied in fields without defined levels or classes. On the other hand, metaheuristic algorithms are used to simplify and solve structured problems.

The widespread adoption of long-short equity strategies in several hedge funds, which were selected for their proofed risk-adjusted returns, proves the validity of these strategies. [1] This gives solid support to the aim of this document.

Chapter 1

Time Series and Algorithm Evaluation for Equity Optimization

1.1 Description of the chosen time series

1.1.1 EDA on time series

The subjects of this analysis are large and well-established companies, leaders in their respective industries with a substantive influence on market trends and economic conditions. The companies' stocks are AMGEN, American Express, Apple, Chevron, CISCO, and 3M. All these companies belong to stock indices such as the SP500 and Dow Jones Industrial Average. Vast investors and analysts very closely watch their stock performance. Mainly for this paper, the weekly time series of their stocks for the last 20 years is taken. Each data set consists of the following columns: Date, Open, High, Low, Close, and Volume.

The following chart of the weekly closing prices from 1993 to 2023 reveals

generally rising trends. Significant historical events, like the financial crisis in 2008 or the COVID-19 pandemic, are visible and affect almost all stocks. Apple is highly volatile and grows dramatically, especially in the middle part of the 2000s and then in the late part of the 2010s when it took over the market. AMGEN grows up to the peak around 2021 and starts falling almost X-wise down, maybe because of the industry-specific factors. Chevron grows slowly, being highly dependent on oil prices, while CISCO stays virtually the same in its value, slightly rising. 3M shows steady performance and keeps itself away from wild fluctuations.

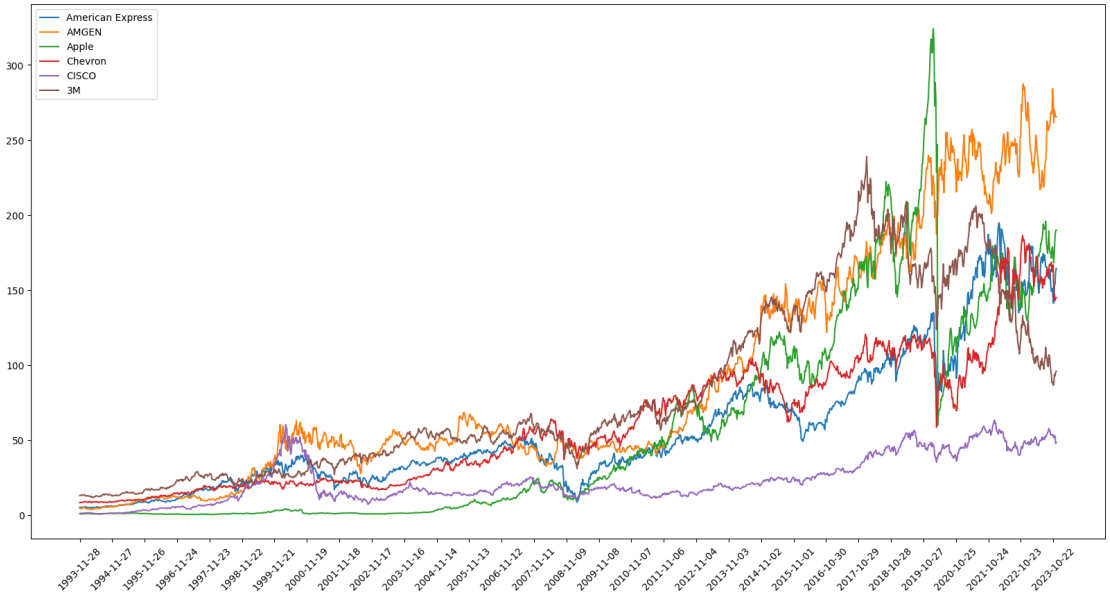

Fig. 1.1: Stocks' prices over time

Another crucial technical indicator is the *Exponential Moving Average* (EMA), which gives more weight and importance to more recent prices. This is typical for stock trading and represents an excellent way to smooth price data by creating a continuously updated average prices. This property makes the EMA significantly more reactive to current trends in the marketplace. It is, therefore, preferred by the majority of traders who need timely responses in their strategies. [15]

Below are the MA lines for American Express, AMGEN, Apple, Chevron,

CISCO, and 3M.

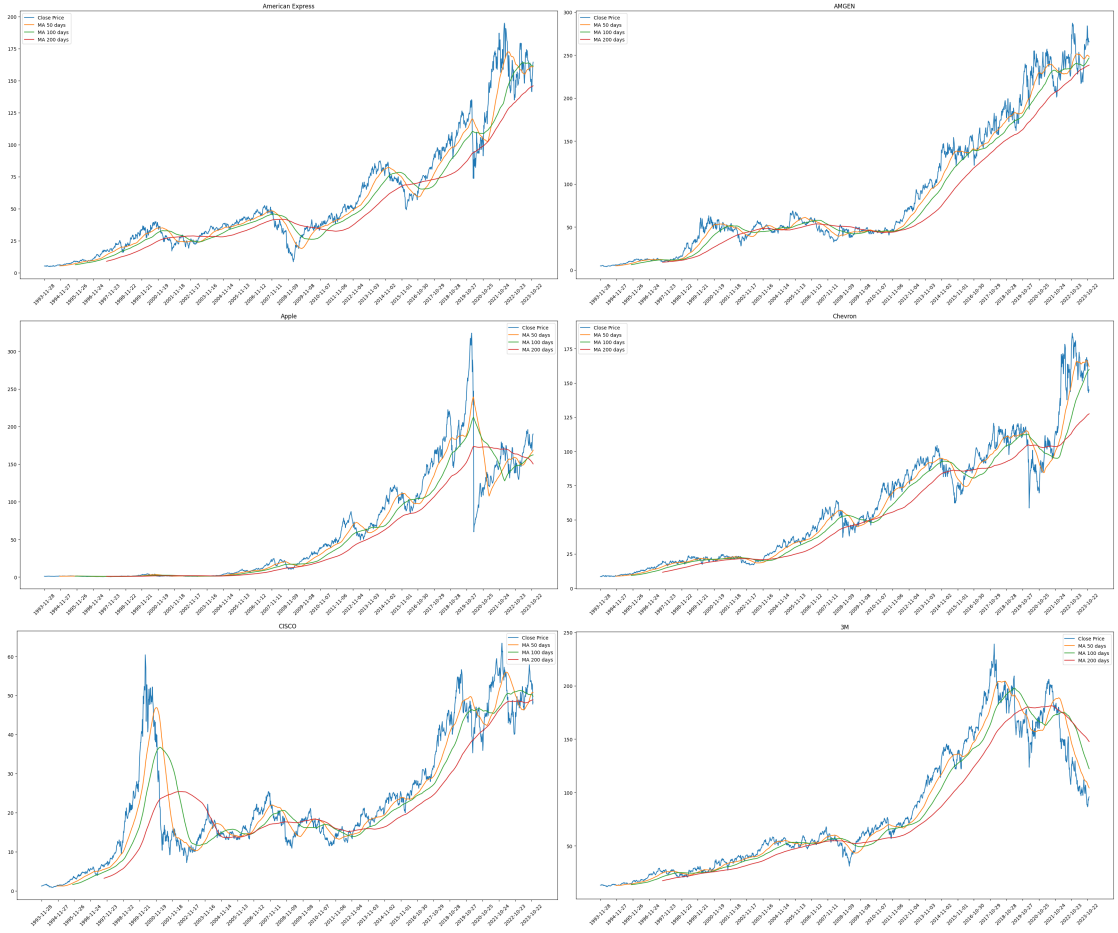

Fig. 1.2: MA curves of 50,100,200 days compared to the close line for each stock

These include the closing prices daily for 1993-2023, with a predominant blue line for "Close Price". Other features include the availability of three moving averages: *200-day (red line)* to gauge long-term trends, *100-day (green line)* for medium-term trends, and *50-day (orange line)* for short-term trends. The inclusion of these moving averages in the graph is for customers so that they would know how the stock performed over different time limits and, correspondingly, how to predict future movement through historical data.

The exponential smoothing method was first introduced by Robert Goodell

Brown in 1956. Exponential smoothing was then refined and developed by Charles Holt in 1957. Today it is one of the most popular forecasting techniques used to analyze time series.

The ES curves for American Express, Amgen, Apple, Chevron, Cisco, and the 3M stock price series display different trends and volatility paths. The core concept of exponential smoothing is to prioritize recent data points over older ones. This is done by assigning exponentially decreasing weights to older observations. These weights are then applied to calculate a weighted moving average, which serves as the forecast for the upcoming period. [29]

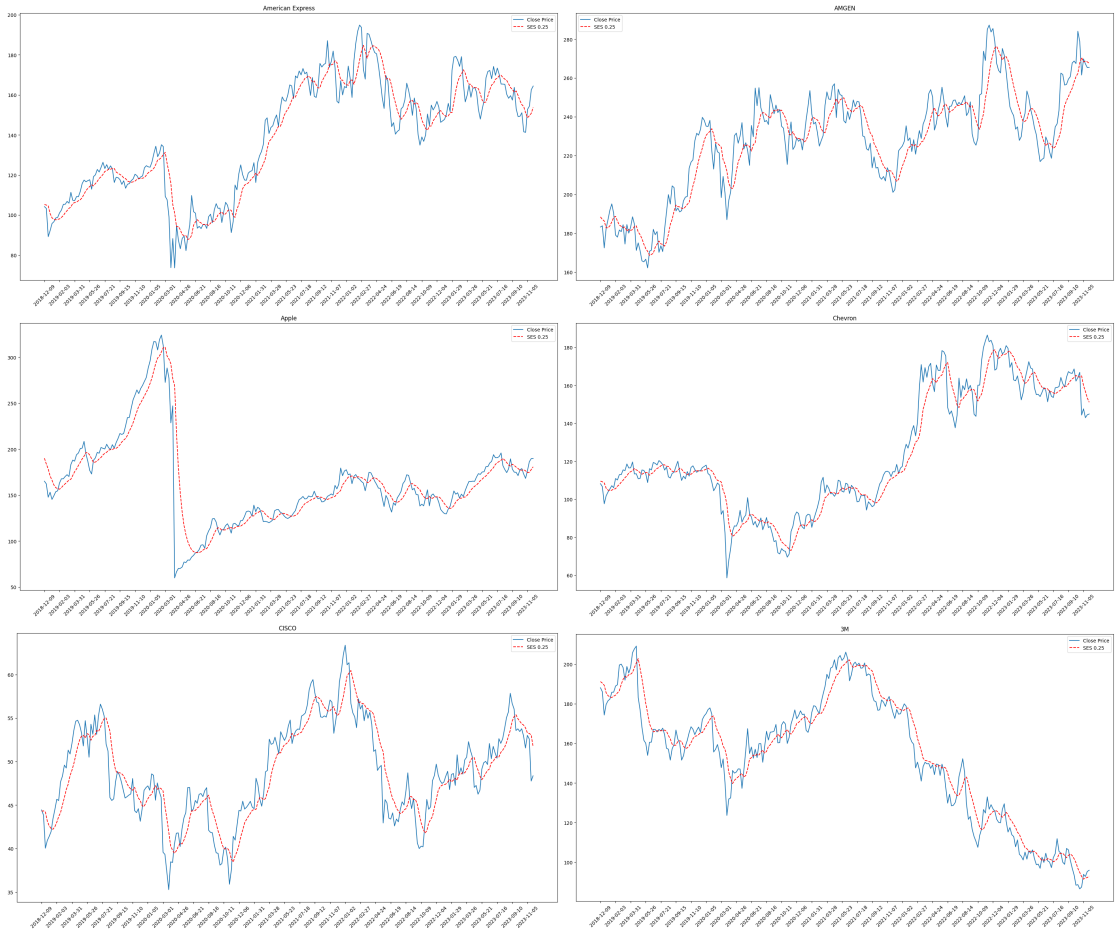

Fig. 1.3: Esp.Smoothing curves compared to the close line for each stock of the last 5 years

The analysis of the Exponential Smoothing (ES) curves for American Express, Amgen, Apple, Chevron, Cisco, and 3M reveals distinct patterns in their stock price trends and volatilities over time.

The graphs of American Express and Chevron are very volatile. American Express shows a balanced fluctuation with a significant downturn at the beginning of 2020, followed by recovery attempts. The same fluctuation is presented by Chevron, which identifies fluctuations most probably related to the global oil market. The ES curve of Apple would have shown healthy growth, being steady until the mid-point of 2021, with relatively low volatility. This shows steady performance and increases investor confidence in the company's prospects.

On the other hand, 3M does show consistency in the downtrend trajectory, or at least through early 2021; that is, a very long decline on its chart. Such a downtrend might be related to internal problems within the company or the industry. The curves for ES with Amgen and Cisco have a smoothness about them where growth is stable but for moderate to controlled volatility. Their curves capture general upward trends without significant fluctuations, reflecting steady performance and resilience in the face of market changes.

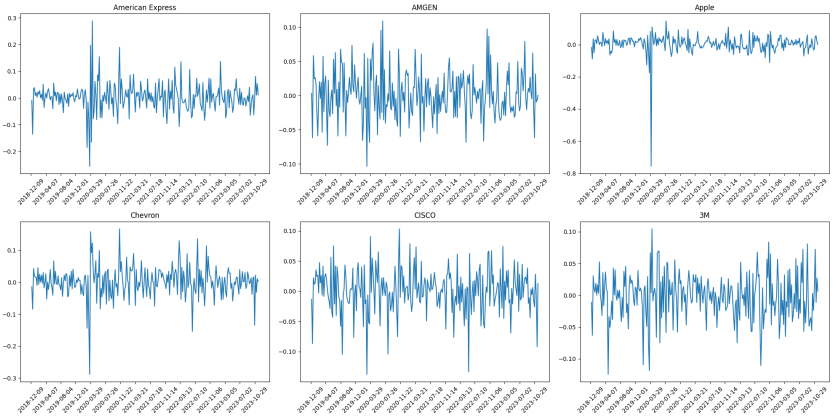

Fig. 1.4: Percentage change close price over time

Percentage change of close prices plots the variability of values over time, thus

making more straightforward observation of fluctuations and outlining outliers.

As for the percentage change in the close price, after 2020, the stock prices of American Express, Apple, and Chevron show higher stability in their variations and some phases of recovery, while the fluctuations keep happening. Under contrast, AMGEN and Cisco show relatively uniform high volatility, making their stock prices experience sharp increments and declines frequently. 3M, the third case showing regular fluctuations, sags a parallel trend of business volatility.

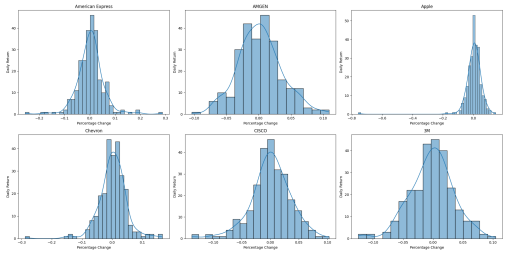

Fig. 1.5: Daily Percentage Distribution

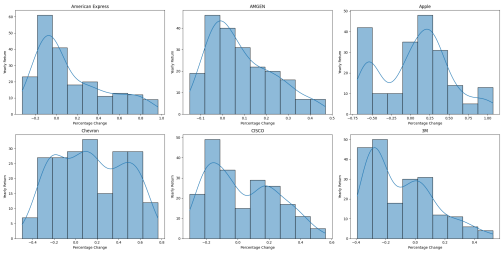

Fig. 1.6: Yearly Percentage Distribution

Above are the histograms that show the distribution of daily and yearly percentage changes in companies returns. Overall, most stocks have daily returns that center around 0 percent, indicating moderate to stable daily performance. Yearly returns, on the other hand, have more spread levels, and some of the distributions may tend to be positive-skewed with more comprehensive ranges, showing an increased volatility and mixed performances. These visualizations bring out the patterns of stability and variability in both daily and yearly returns among the different companies analyzed.

1.2 Procedures to valuate the quality of an algorithm and analysis of time series

1.2.1 Common Performance Metrics: Confusion Matrix and F1 Score

Evaluating the actual performance of the implemented algorithms is crucial in machine learning and statistical modeling.

The Confusion Matrix, often used in classification problems, visualizes the performance of an algorithm on the classification of data.

		Actual Values	
		Positive (1)	**Negative (0)**
Predicted Values	**Positive (1)**	TP	FP
	Negative (0)	FN	TN

It labels predictions into one of its four key components:

- *True Positive* [TP] → the model correctly identifies an instance as positive, such as diagnosing a disease that is actually present
- *True Negative* [TN] → the model rightly signals no presence of the condition it is testing for, accurately indicating a patient does not have the disease
- *False Positive* [FP] → the model wrongly predicts a positive result, like mistakenly diagnosing a healthy person with a diseases
- *False Negative* [FN] → the model fails to detect something that is there, such as missing a disease in an afflicted patient

These values are crucial for calculating various performance metrics such as accuracy, precision, sensitivity , and F1-score, which help in assessing the quality of

classification algorithms. [2]

The F1 score is calculated as the harmonic mean of the precision and sensitivity scores. For the sake of clarity, in the following lines are recalled some important definitions:

Precision is a performance measure and is the fraction of all positive predictions that were actually correct according to a classification algorithm. It is calculated as the ratio of true positives to the total that was predicted as positives. The metric answers the question "Of all instances labeled as positive, how many are actually positive?", and it is calculated by the following equation:

$$Precision = \frac{TP}{TP + FP} \tag{1.1}$$

Sensitivity, also known as recall, on the other hand describes how effectively a classification model can identify all relevant cases in a dataset. It answers the question: "Of all the actual positives, how many did the model successfully identify?".

$$Sensitivity = \frac{TP}{TP + FN} \tag{1.2}$$

The *harmonic mean* is particularly useful because it gives a more meaningful average when dealing with ratios that vary widely, as it tends to be dominated by the smallest values.

$$F1 = 2 \cdot \frac{Precision \cdot Sensitivity}{Precision + Sensitivity} \tag{1.3}$$

[2] The *F1 Score* contributes in the optimization and evaluation of models when the balance between both precision and recall is to be reached. This makes it particularly valuable in a situation where uneven class distribution may bias the model towards the majority class or in a situation where both of the types of classification errors are similarly costly. [3]

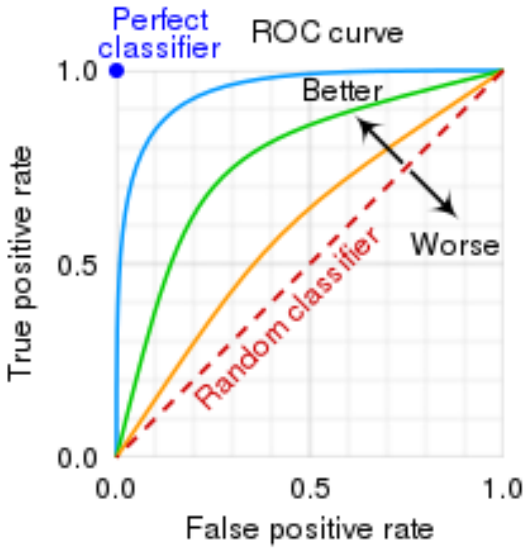

Fig. 1.7: An example of ROC Curve[20]

1.2.2 ROC and the AUC Score

ROC curve and its *AUC score* are the pivotal classifiers in the performance of binary classifiers in statistical learning. It is a plot that shows the TP rate against the FP rate. In other words, it demonstrates how well a classifier separates classes within a model, showing the power of discrimination of the classifier across numerous possible classification thresholds. The closer the score is to 1, the result is better, but on the other hand, an AUC 's value of less than 0.5 (level with no discriminative power) means the performance is worse than random guessing. [4]

However, using ROC's AUC as a performance metric for model evaluation has several notable limitations. For instance, consider a binary classification problem where the positive class is the minority. By evaluating the model's ability to distinguish between classes across all thresholds, AUC might give a misleading positive evaluation in the presence of a majority class. This serves as a clear indication that AUC does not directly represent the real accuracies or error rates for each class, which are crucial in real-world scenarios. It's important to recognize

that AUC also does not consider the implications of different error types, like false positives and false negatives, which can have varying impacts based on the application. This emphasizes the significance of taking into account the specific context of the application when selecting a performance metric.[4]

In imbalanced datasets, it has shown that the performance of AUC compared to the Precision-Recall Curve (AUPRC) can differ significantly. As it is demonstrated in other researches, AUPRC is more informative for such datasets as it focuses on the minority class and considers both precision and recall, making it sensitive to the model's ability to detect rare events. [5]

1.2.3 Sharpe Ratio

Important tools in the analysis of investments are *Sharpe Ratios*. The Sharpe Ratio shows the return on investment adjusted to the level of risk in the stock versus the returns from other types of investments. This ratio calculates the excess return to the unit of risk about its standard deviation. The Sharpe Ratio provides a lot of strength in the framework for the measurement of risk. A higher Sharpe Ratio demonstrates that an investment achieved more return for lower risk strategy.

A higher Sharpe Ratio indicates that an investment has yielded better risk-adjusted returns, pointing to a more successful investment strategy.

The formula for calculating the traditional Sharpe Ratio is as follows:

$$SharpeRatio = \frac{R_p - R_f}{\sigma_f} \tag{1.4}$$

Where R_p is the expected portfolio return, R_f the risk-free rate, and σ_f the standard deviation of the portfolio's returns.

Econometric studies show that Sharpe's Ratio is crucial in assessing portfolio

efficiency under different financial market conditions, guiding decisions about asset priority. Additionally, maximizing the Sharpe Ratio is often used as a strategy to outperform benchmarks like the SP500, providing a guideline for constructing portfolios that offer superior adjusted returns.[6]

However, if return distributions are not normal, the Sharpe ratio can lead to wrong conclusions. To address the limitations of the traditional Sharpe Ratio when dealing with non-normal distributions of returns, several modifications to the initial formula have been proposed. These modifications aim to better capture the risks associated with *skewness*[1] , *kurtosis*[2], and other aspects not accounted for by the standard deviation alone. One such approach is the modified Sharpe ratio which incorporates the Value at Risk (VaR) associated to a given stock or portfolio. [7]

VaR is a threshold that estimates the potential loss in value of a risky portfolio. It represents the maximum percentage loss likely to be incurred on a portfolio position over a specified holding period at a given confidence level. Specifically, for a selected portfolio and time period, with the confidence level α within the interval (0, 1), VaR is defined as the threshold value at which the probability of the mark-to-market loss in the portfolio exceeding this VaR level matches the preset probability of loss α, assuming no further trades occur. [8] Conventionally, this worst-case loss is always expressed as a positive percentage. Formally, if L denotes the loss, measured as a positive number, and α represents the confidence

[1]Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or data set, is symmetric if it looks the same to the left and right of the center point. [https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm]

[2]Kurtosis is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution. That is, data sets with high kurtosis tend to have outliers.[https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm]

level, then VaR is defined as the smallest loss(in absolute terms) such that:

$$P(L > VaR) \leq \alpha \tag{1.5}$$

[8] The most common methods used to estimate values for VaR include historical, parametric (variance, covariance), and Monte Carlo.[3]. The modified sharpe ratio is computed as follows[8]:

$$SR_\alpha = \frac{R_p - R_f}{VaR_\alpha} \tag{1.6}$$

Moreover, exists another approach that provides better performance metrics in the presence of outliers and non-linear return distributions: *the Cornish-Fisher expansion.*

It is a method used to adjust quantiles of a distribution when it deviates from normality. This adjustment allows for a more accurate assessment of the risk-adjusted return, particularly in portfolios with asymmetric risk profiles or fat tails.

$$z_{cf,\alpha} = z_\alpha + \frac{1}{6}(z_\alpha^2 - 1)S + \frac{1}{24}(z_\alpha^3 - 3z)(K - 3) - \frac{1}{36}(z_\alpha^3 - 5z_\alpha)S^2 \tag{1.7}$$

Where:

- $z_{cf,\alpha}$ is the adjusted z-score at the thereshold α
- z_α is the original z-score at the thereshold α, which is a measure of how many standard deviations an element is from the mean of a distribution.
- S is the skewness of the distribution
- K is the excess kurtosis

[8]

[3]Monte Carlo simulation uses repeated random sampling to compute and solve complex problems by simulating system behavior with random variables, particularly when analytic solutions are not feasible.

1.2.4 Backtesting

Backtesting is a pivotal method in algorithm optimization, particularly in financial and engineering domains, by assessing algorithm performance against historical data to predict future behavior.[6] It involves setting up conditions under which the strategy operates, applying it to historical market data, and analyzing the results to assess the quality of the algorithm. This approach assists in evaluating the advantages and limitations of a strategy prior to its implementation in actual trading, potentially preventing substantial loss of funds and resources.

On the other hand, it is showed that backtesting can sometimes suffer from *overfitting*: when a model is tuned too finely to historical data, leading to poor performance in real-world trading. Minimizing overfitting in financial model backtesting can be effectively achieved through several innovative techniques. As it will be described in the following chapters, one effective strategy is using *Generative Adversarial Networks (GANs)* combined with *Long Short-Term Memory* (LSTM) networks to generate synthetic data paths that mimic the distribution of historical data.[9] This approach helps to validate trading strategies against overfitting by ensuring they perform well on both historical and synthetic data, thus providing a robustness check against overfitting.[9]

Additionally, applying covariance penalties to adjust risk metrics based on the number of parameters and the amount of data used in a strategy can prevent misleading results in backtesting, offering a more conservative assessment of a strategy's viability.[10]

Chapter 2

Strategy and Methods

2.1 Fuzzy Logic

2.1.1 An introduction

Introduced by Lotfi Zadeh in 1965, *fuzzy logic* represents a transformative extension of Boolean logic, allowing several degrees of truth between 0 and 1, and because financial markets are characterized by volatility and ambiguity, fuzzy logic emerges as a critical tool for modeling and analyzing financial scenarios. While boolean logic operates on clear-cut true or false values, perfect for decisions that require binary outcomes, such as TRUE/FALSE or 0/1, fuzzy logic stands out. In the fuzzy process, precise input data are transformed into a decision through several interconnected stages. The process starts with fuzzification, where crisp (precise) inputs are converted into fuzzy sets characterized by degrees of membership. These fuzzy sets are processed within a knowledge base that includes a database of membership functions and a rule base containing fuzzy IF-THEN rules. The decision-making unit applies these rules to the fuzzy inputs to derive fuzzy outputs. Subsequently, the defuzzification stage converts these outputs back

into a single crisp output. [14]

2.1.2 Crisp Sets, Fuzzy Sets and Membership function

Crisp sets are the traditional sets we encounter in everyday mathematics, in these sets an element either belongs to them completely or does not belong at all. On the other hand, fuzzy sets allow for partial membership of a given object in a set. The membership function $\mu(x)$ of a fuzzy set gives as output the degree to which an crisp object x belongs to a fuzzy set, which outcome is a number between 0 and 1. Thanks to this process called fuzzification, the initial crisp values can be applied in the fuzzy logic rules to make decisions or control systems in various applications. [14]

Let's consider an example to better understand the fuzzification process, shown in the table (2.1). We define the temperature range from -10 to 45 degrees Celsius, dividing it into four sections: Cold, Mild, Warm, and Hot. Each section has a fuzzy set with a membership function indicating the degree of belonging. For instance, Cold might be from 0 to 20 degrees, Warm from 15 to 30 degrees, and Hot from 25 to 45 degrees. When we have a specific temperature, like 25 degrees, we assess its degree of belonging to each section. [11]

It is important to notice how these temperature ranges are overlapping to each other. This example is considerable valid since different individuals might perceive temperature differently, and what one person considers "warm" might be "hot" to another. With overlapping ranges, fuzzy sets can accommodate this variability in perception, allowing for smooth transitions between categories. This quality lead fuzzy logic to be a powerful mechanism for modeling complex and multidimensional phenomena where traditional crisp sets may fall short. Fuzzy Logic simplifies the complexity of reality, and this quality shows up in the analysis of

Temperature range in Celsius		**Temperature Section**
-10 $\leq x \leq$ 7	$\Rightarrow$	Cold: 0 $\leq y \leq$ 0.31
5 $\leq x \leq$ 18	$\Rightarrow$	Mild: 0.27$\leq y <$ 0.46
16 $\leq x \leq$ 28	$\Rightarrow$	Warm: 0.47 $\leq y <$ 0.69
25 $\leq x \leq$ 45	$\Rightarrow$	Hot: $y \geq$ 0.655

Table 2.1: Example Fuzzy Logic

long time series. In fact, it is well know that more complex model, more it has a risk of overfitting. [12]

Fuzzy sets can be united or intersect to each other, as the common sets. The union of two fuzzy sets, A and B , denoted as $A \cup B$, is defined mathematically for each element x in the universe of discourse[1]. The degree of membership of x in the union set $A \cup B$ is given by the maximum of the degrees of membership of x in A and in B. Formally, it is expressed as:

$$\mu_{A \cup B}(x) = max(\mu_A(x), \mu_B(x)) \tag{2.1}$$

This concept can be extended to the union of more than two fuzzy sets. For any number of sets $A_1, A_2, ..., A_n$ the membership function of an element x in the union of these sets is:

$$\mu_{A_1 \cup A_2 \cup ... \cup A_n}(x) = max(\mu_{A_1}(x), \mu_{A_2}(x), ..., , \mu_{A_n}(x)) \tag{2.2}$$

[14]

Using only one indicator may result in false signals, leaving the trader with uncertain trading actions, especially in high volatility markets. Therefore, a solu-

[1] the collection of all items under consideration

tion for traders can apply the intersection of fuzzy sets to combine signals from multiple indicators.

If A and B are two fuzzy sets with membership functions $\mu_A(x)$ and $\mu_B(x)$, respectively, then the membership function $\mu_{A\cap B}(x)$ of the intersection $A \cap B$ is defined as:

$$\mu_{A\cap B}(x) = min(\mu_A(x), \mu_B(x)) \tag{2.3}$$

This means that for any element x, its degree of membership in the intersection set $A \cap B$ is the minimum of the degrees of membership of x in sets A and B. [14]

2.1.3 Setting of the fuzzy rules

Modus Ponens and *Modus Tollens* are both types of deductive reasoning used in logic. Modus ponens (Latin for "method that affirms by affirming") asserts that if the antecedent A of a conditional statement is true, then the consequent B must also be true. On the other hand, Modus tollens (Latin for "method that denies by denying") asserts that if the consequent B of a conditional statement is false, then the antecedent A must also be false.

Generalized Modus Ponens (GMP)

In fuzzy logic, the modus ponens is applied in this form:

- If p implies q (with $p \Rightarrow q$ being a fuzzy implication relation).
- p is true to a degree (expressed as a fuzzy set).
- Then q is true to a certain degree (also a fuzzy set).

The generalized modus ponens takes into account the degree of truth of the premise p and computes the degree of truth of the conclusion q based on the fuzzy implication and the degree of the premise. The fuzzy implication often used is $q = \mu(p)$,

where μ is a membership function defining how much q is true when p is true to certain degrees. [15]

For example, in a fuzzy system:

- Rule: If it is very hot (degree x), then the air conditioner power should be high (degree
- Observation: Today is very hot with a degree of 0.8.
- Conclusion: The air conditioner power should be high, with a degree determined by applying the fuzzy implication function to 0.8.

Generalized Modus Tollens (GMT)

In fuzzy logic the modus tollens extends in a slightly different way than GMP:

- If p then q (with $p \Rightarrow q$ being a fuzzy implication relation).
- q is false to a degree (expressed as a fuzzy set).
- Therefore, p is false to a certain degree.

This involves using the fuzzy implication to determine to what extent the falsity of q impacts the truth of p, for instance:

- Rule: If it is raining (degree x), the street will be wet (degree y).
- Observation: The street is not wet (degree 0.2).
- Conclusion: It is likely not raining, with a degree determined by applying the inverse of the fuzzy implication function to 0.2 .

[14]

2.1.4 Defuzzification process

Defuzzification is a process used in fuzzy logic and fuzzy control systems to convert fuzzy values obtained as outputs from fuzzy logic into crisp, conventional values that can be understood and acted upon in the real world. There are many methods to apply defuzzification, but let's focus on those that are going to be implement in our case study.

One of the most useful approaches is *Center of Gravity* (COG) , which calculates the centroid of the area under the curve of the membership function of the aggregated fuzzy output set. It effectively finds a balance point, considering the shape and spread of the fuzzy set.

$$x^* = \frac{\int x\mu_A(x)d_x}{\int \mu_A(x)dx} \tag{2.4}$$

COG is the most commonly used method because it provides a balanced result reflective of the entire fuzzy set.

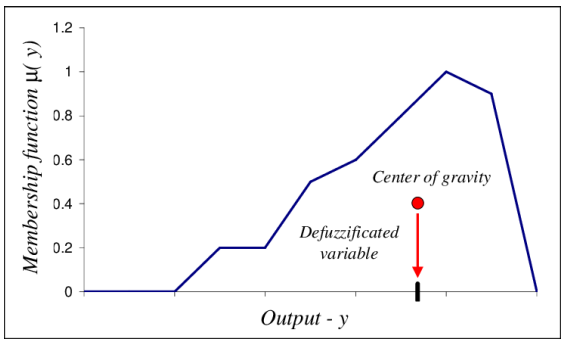

Fig. 2.1: Defuzzification using the COG method [23]

In addition to that, another optimal method is *Bisector of Area* (BOA), divides the area under the curve of the aggregated fuzzy set into two equal halves.

$$\int_{\alpha}^{x^*} \mu_A(x)dx) = \int_{x^*}^{\beta} \mu_A(x)dx) \tag{2.5}$$

$$\alpha = min\left\{x \mid x \in X\right\}, \beta = max\left\{x \mid x \in X\right\}$$

The output is the point where a vertical line would split this area equally. BOA is particularly useful when the symmetric distribution of the output is essential. [16]

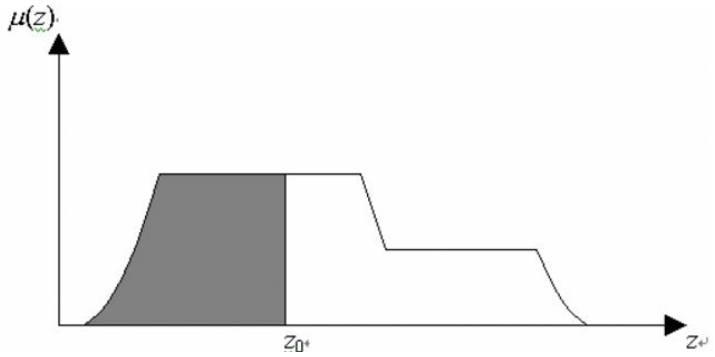

Fig. 2.2: Defuzzification using the BOA method[23]

2.1.5 Technical indicators as fuzzy variables

In technical analysis, traders often use various indicators to identify potential entry and exit points for trades. Each indicator generates signals based on specific criteria, such as price movements and momentum.

Market indicators are quantitative tools that assess current and future market conditions, providing insights into market trends, trading volumes, price movements, and overall economic health, guiding investment decisions and strategy formulation.

Market indicators are pivotal in trading and investment decision-making, and each type serves unique purposes.

- *Chart patterns* provide a visual analysis of price data combining both quantitative and qualitative methods. They are based on historical price movements and are used to predict future market behavior. Unlike other indicators, chart patterns rely heavily on visual interpretations of price shapes and

trends, such as head and shoulders or double tops, and are less formulaic.

- *Price Trends* involve mathematical manipulation of time series data to identify and understand price directions and oscillations over time. This analysis can forecast future price movements by observing historical data trends.

- *Volume and Momentum Indicators* combine price data with trading volume to gauge the market's sentiment and momentum, unlike price trends and chart patterns, which focus more on price data alone.

[18]

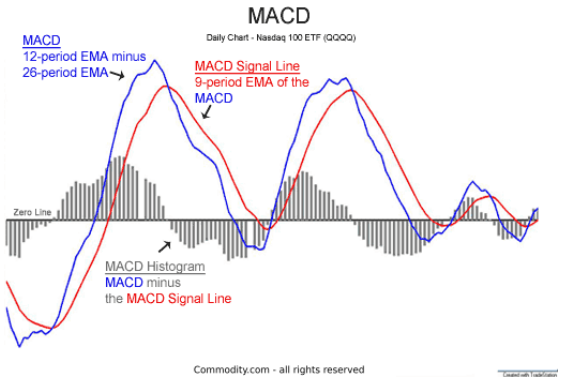

Fig. 2.3: An example of MACD Curve [21]

The *Moving Average Convergence Divergence* [MACD] is a widely used technical indicator in trading. It helps identify momentum and trend changes in the price of an asset. The MACD is calculated by subtracting the 26-period EMA from the 12-period EMA, which produces the MACD line. A signal line, which is the 9-period EMA of the MACD line, is then plotted on top of the MACD line. [15]

$$MACD = EMA_s(P) - EMA_l(P) \tag{2.6}$$

$EMA_s(P)$: EMA's price in the short period

$EMA_l(P)$: EMA's price in the long period

Traders look for signals generated from the crossover between these two lines as well as divergences and the rise or fall of the histogram, which represents the difference between the MACD line and the signal line.

Fig. 2.4: An example of applied RSI[22]

The *Relative Strength Index* [RSI] is a momentum oscillator that measures the speed and change of price movements. It oscillates between zero and 100, typically used to identify overbought or oversold conditions in a market. [15] Its formula is stated above:

$$RSI = 100 - \frac{100}{1 + RS} \tag{2.7}$$

where $RS = \frac{(Total\ Gain)/(n\ days)}{(Total\ Loss)/(n\ days)}$

For a long-short equity strategy, where the goal is typically to capitalize on both upward and downward market movements by taking long positions in undervalued stocks and short positions in overvalued stocks, an optimal strategy can be the implementation of the technical indicators MACD and RSI as fuzzy variables.

RSI and MACD are both valuable for identifying market trends, and in addition to that their combined use can enhance prediction accuracy and provide clearer signals for executing long and short positions. An example of Fuzzy rules applied:

1. If (RSI is High) and (MACD is Positive), then (Position = Sell)

2. If (RSI is Low) and (MACD is Negative), then (Position = Buy)

3. If (RSI is Medium) or (MACD is Zero), then (Position = Hold)

The *Stochastic Oscillator* is very similar to RSI, capturing the information about overbuying or overselling. It is one of the most renowned momentum indicators used to identify whether a stock is overbought or oversold. Its clarity has allowed the stochastic oscillator to be one of the most used indicators since its invention in the 1950s. It has two threshold lines to distinguish if the given time series is overbought or oversold, usually those values are set at 20 and 80. The Stochastic Oscillator is composed by:

- K (also known as Fast Stochastic) $\rightarrow$ This represents the most recent value in relation to the highest and lowest values within the lookback period.

- *Lookback Period* $\rightarrow$ This refers to the number of previous periods considered for calculating the

- D (also known as Slow Stochastic) $\rightarrow$ This is a lagging momentum indicator that smooths out trendlines by averaging the K values. It is used to generate trade signals and typically involves a Simple Moving Average (SMA) of 3 periods.

- *Overbought Line* $\rightarrow$ This is the level at which the Stochastic Oscillator (SO) indicates a strong bullish market trend. It is generally set at 80, but sometimes at 70.

- *Oversold Line* $\rightarrow$ This is the level at which the SO signals a strong bearish market trend. It is typically set at 20, though it can also be set at 30.

- *Min/Max Lines* → These lines represent the upper and lower boundaries of the SO chart, indicating the range of possible values from 0 to 100.

The Fast Stochastic is determined based on the highest and lowest values observed within the specified lookback period:

$$k = (\frac{close - low_n}{high_n - low_n}) \cdot 100 \tag{2.8}$$

The value of d is a smoothed version of k, calculated over a shorter lookback period, typically 3 periods. It is a simple moving average (SMA) of k and provides a slightly slower response to changes in relative price over a longer lookback period.

$$d = \frac{1}{w} \sum_{i=n-w+1}^{n} \tag{2.9}$$

[19]

2.2 Model Optimization

2.2.1 An introduction

Optimization algorithms are fundamental computational techniques that aim to achieve the best solution within a feasible set of solution options. Fundamental in many areas of engineering and financial analysis, optimization plays a big role in the fields where constraints force either a maximization or minimization of the objective function value.

All optimization problems have three principal ingredients: an objective function, decision variables, and constraints. The objective function defines quality or value with respect to the problem goals for a candidate solution, this function in optimization problems is either maximized or minimized. The decision variables

are parameters that the process can change in order to achieve the optimal solution. Constraints are the conditions that must be satisfied by a candidate solution in order to be considered a valid solution.

A meta-heuristic algorithm is a large subset of optimization algorithms. These are very high-level strategies by which algorithms are guided in solving problems that are very complex, usually involving randomness. Examples of these are simulated annealing, genetic algorithms, and particle swarm optimization. [25] In this chapter will be explored the basic principles of these algorithms, the applications, and how these can be used in optimization problems.

2.2.2 Genetic Algorithms

In essence, a genetic algorithm is an iterative sequence of steps that starts with a population of a certain size and looks something like the following:

1. Initialization: An initial population of individuals, also referred to as "solution candidates," or "chromosomes," generated either randomly or by a heuristic approach.

2. Evaluation: At every generation, the current population is evaluated and fitness values are associated with each individual.

3. Selection: Selection is the process through which a new population is formed by selecting the individuals, with respect to their fitness values. This is done with a probability proportionate to fitness, meaning that the members of the population with a higher fitness value have a higher probability of selection.

4. Reproduction: Selected parents reproduce and produce new offspring.

This process ensures that the number of times an individual is chosen is roughly proportional to its performance relative to the rest of the population. Genetic

algorithms use two main operators to generate new solution candidates: *crossover* and *mutation*.

Crossover takes two individuals, known as parents, and creates one or two new individuals, called offspring, by merging segments from the parents. In its simplest form, this involves exchanging substrings before and after a randomly chosen crossover point.

Mutation involves making arbitrary changes to help prevent premature convergence by introducing new points in the search space randomly. For bit strings, this means flipping bits randomly in a string according to a probability known as the mutation rate. Genetic algorithms are stochastic iterative processes that cannot guarantee convergence. Termination usually occurs after a maximum number of generations, finding an acceptable solution, or meeting more sophisticated criteria that indicate premature convergence. Here below is showned a pseudocode process of a so-called standard genetic algorithm (SGA):

Algorithm 1 Basic workflow of a genetic algorithm.

Produce an initial population of individuals

Evaluate the fitness of all individuals

while termination condition not met **do**

Select fitter individuals for reproduction and produce new individuals (crossover and mutation)

Evaluate fitness of new individuals

Generate a new population by inserting some new ”good” individuals and by erasing some old ”bad” individuals

end while

[26]

2.2.3 The importance of data mutation

Mutation is an operation used in genetic algorithms in order to maintain diversity within the population and avoid early convergences. It introduces some randomness in the algorithm such that it can modify some gene 's values of the chromosome, to find and discover new candidate solutions. The basic mutation operator for problems that are binary-coded is the *bitwise mutation*. Mutation occurs randomly and very rarely at a very low probability p_m: which is almost always less than ten percent. In some cases mutation is interpreted as generating a new bit and in others it is interpreted as flipping the bit. [27] So, here are some crucial types of the mutation approaches:

- *Adaptive mutation*: a set of genes is ordered and shuffled randomly in the search space, this helps in keeping diversity and exploring new solutions

- *Inversion Mutation*: a sequence of genes on a chromosome is selected and reversed in order. That shows an inversion of the sequencing of the gene. It may change the genetic makeup of the offspring and can help the algorithm explore other parts of the search space.

- *Random Mutation*: This will be mutate a subset of genes to value totally at random. In general, these new values are obtained by multiplying the original gene values by some random factor, which generally, is close to 1. Random mutation will be very effective in bringing new genetic materials into the population. This can result in the discovery of optimal or near-optimal solutions to the problem.

- *Swap Mutation*: it picks any two genes in a chromosome and swaps the values stored in them. This is quite a simple useful operation because it

will introduce new gene value mixes; therefore, giving room to diversity and letting go the genetic algorithm from a local optima.

2.2.4 Limitations of Genetic Algorithms

However, despite the wide application and success of genetic algorithms in the solution of diverse complex optimization problems, they have a few striking drawbacks.

First of all, the computation of the fitness function is rarely straightforward to compute and often resource-intensive; this is particularly true in multi-objective optimization scenarios that usually require trade-offs among goals. Another characteristic is that genetic algorithms usually suffer from quick convergence. In fact, they usually found local optima rather than the global best, and this is due to the lack of genetic diversity over the generations. This feature is further enhanced by its dependence on the choice of several initial parameters, such as the population size, mutation rate, and crossover rate, which can typically be adjusted just by trial and error.

It is also showed a problem of scalability in these algorithms: if the dimensionality of the problem increases, the search space increases exponentially.

Therefore, the efficiency of the search decreases with the growth in this exponential. Furthermore, genetic algorithms are heuristic methods; thus, they do not have a global guarantee of finding the best solution in all possible cases, but they will provide good-enough solutions within reasonable running times.

2.2.5 Swarm Intelligence approaches

Swarm Intelligence is a new paradigm in Artificial Intelligence that is based on the collective behavior of social animals such as ants, bees, and birds. Swarm intelligence is a methodology in which simple agents interact among each other

and with the environment they are placed in, enabling the system to solve significantly complex problems in an efficient way. These internal interactions lead to the emergence of "intelligent" global behavior, optimized to find solutions more efficiently than could be done by a single agent or a centrally controlled system. [28]

Mark Millonas (1994), at Santa Fe Institute, who develops his kind of swarm models for applications in artificial life, has articulated five basic principles of swarm intelligence:

- *The proximity principle*: The population should be able to carry out simple space and time computations.
- *The quality principle*: The population should be able to respond to quality factors in the environment.
- *The principle of diverse response*: The population should not commit its activity along excessively narrow channels.
- *The principle of stability*: The population should not change its mode of behavior every time the environment changes.
- *The principle of adaptability*: The population must be able to change behavior mode when it is worth the computational price.

[27]

Here below are reported some important SI models.

- *Ant Colony Optimization (ACO)*:

  As the name suggests, is inspired from the behavior of ants and applied to solving optimization problems. It emulates the way that ants locate the shortest path between their nest and food sources. A more practical approach

of application of ACO has found in the fields that include routing, scheduling, and network design.

- *Particle Swarm Optimization (PSO)*:

  Particle Swarm Optimization was inspired by the social behaviour in birds, called flocking. In PSO, the population of particles moves around in a search space while adjusts its individual positions and velocities toward optimal solution for the objective function. These particles learn how to do better searches or improve the solutions by communicating and sharing information.

- *Bee Colonization Optimization (BCO)*:

  The BCO algorithms are modeled on the bee 's forage behavior, used to solve questions such as the traveling salesman problem, job scheduling, and other optimization problems. Honeybees forage for food sources and inform other bees where the locations are, such that they optimize the search.

[28]

Chapter 3

Implementation

3.1 Environment Setting

3.1.1 Functions

In the section of the code named "*Functions*", 3 functions are worthy of note:

The *fuzzy_variables* function will instantiate fuzzy logic variables with their relative membership functions for different trading indicators. It creates these membership functions using a list of specified parameters, which determines the center and spread of Gaussian membership functions of terms like "Low," "Medium," and "High" for each of the indicators. It initializes the range of potential values (universe) for each of the indicators and the membership functions within the scope, by using the Gaussian and trapezoidal shapes for each of them, making these variables ready for decision-making processes using fuzzy logic in financial trading systems.

The *Defuzzification* function takes two parameters:

```
def Defuzzification(frame: pd.DataFrame, fuzzy_parameters: List):
```

It computes a decision value and appends it to the DataFrame as a new column.

Lastly, it returns new crisp decisions which can be used for trading strategies.

The *total_gain* function is developed to determine the potential financial gains obtained from trading with the trading signals derived from the outputs of the fuzzy logic. The function initially creates trading directions derived from the defuzzification scores relative to threshold levels. It then comes up with the entry prices for the trade initiations and identifies the periods in which a long and short holding is maintained. At that point, it calculates the gain from each; overall gains for periods of a long and short trade are aggregated to provide cumulative figures and then added to get the total potential from the trading strategy.

3.1.2 Fuzzy Rules definition

In this section, inside a list named *rules* are collected all the fuzzy rules that will be included in our model. Here below is reported one of those:

```
FuzzyRule(
        premise=[
            ("MACD", "High"),
            ("AND","RSI", "Low"),
            ("AND","STO", "Low"),

        ],
        consequence=[("Decision", "Buy")],
    ),
```

Then, the variable model applies a *DecompositionalInference* model, a type of fuzzy inference system. This system uses as defuzzification_operator the "cog" (center of gravity), the method used to convert the fuzzy output back into a crisp value by calculating the center of the area under the curve of the fuzzy set.

3.1.3 Variable definition

In this code section, we define our variables and initial setups for the training of our model. We do these operations to all our portfolio stocks through a *for loop*; here, the technical indicators (RSI, MACD, STO) are attached to their respective data series, and then the data is split into training and testing sets. It establishes the threshold levels for buy and sell deals and calculates low, medium, and high ranges for RSI, MACD, and STO using a defined coefficient. Moreover, the genetic algorithm parameters is configured in this code section, such as the number of generations, the number of solutions per population, and the level of the initial solution configuration.

3.2 Data Splitting,Training, and Model Optimization

The core of this optimization process is the *fitness function*, which serves as the evaluation metric that guides the genetic algorithm towards achieving the most accurate solution. The fitness function returns the total gain achieved, focusing on the gain of the last row in the defuzzified training frame. The GA initializes some critical parameters: the number of generations, the initial population, and the number of parents for the mating process. It uses the *fitness_func* to approach the evolutionary process using near-optimal solutions. Finally, after executing this GA, it will find the best solution, its fitness, and the index to which it belongs. In the testing phase, the optimal solution is applied to the test dataset, which is formed by splitting the input data into training and testing sets. The optimal parameters act on the test data using the *defuzzification* function. At the end of the section, the training and test data are concatenated to calculate the total gain further and,

hence, give a complete overall assessment of how the model performs. This process is a demonstration of a rigorous approach to the optimization of parameters for predicting financial models, using evolutionary algorithms to improve predictive accuracy and resulting profits. The plots are shown in the Chapter 4.

3.3 Hyper Parameter Tuning

Hyperparameter tuning is essential to enhance the performance of an algorithm, because it can dramatically impact the performance of the model. Tuning is what makes the model adaptive toward different market conditions. The importance of this step is underlined by its direct influence on the profitability of the system as it refines the conditions generating buy and sell signals.

In our approach, we first modified the hyperparameters such as *num_generations* and *sol_per_pop*, in order to see how the results could be varied from those. We also implemented the rate of mutation and the rate of crossover. The mutation rate, which is equal to 10%, increases the rate of genetic variation, preventing premature convergence and enabling exploration of new solutions, while the *crossover_rate* of 0.5 ensures that half of the offspring result from recombination. These adjustments are essential for optimizing the balance needed between exploration and exploitation, enhancing the model's performance, adaptability, and profitability in generating robust trading signals.

We implement some separate empirical changes on the model, to determine how it changes behaviour on different settings, in particular:

- Modified Model I : increased the number of generations and sol_per_pop, and implementing mutation and crossing portions.
- Modified Model II: implementation of a multiple-objective fitness function to maximize gain and optimize the Sharpe Ratio

Here belowe are reported the main modifications on the code:

Model I:

```
num_generations = 6
sol_per_pop = 2

mutuation_rate = 0.3
crossover_rate = 0.5
```

Model II

```
def fitness_func(ga_instance, list_param,RSI_range):
        train_frame = Defuzzification(train, list_param)
        total_gain_value = total_gain(
                                    train_frame,
                                    list_param[-1])['Gain'].iloc[-1]
        # Calculate daily returns
        returns = train_frame['Close'].pct_change().dropna()
        # Calculate the Sharpe ratio
        sharpe_ratio = calculate_sharpe_ratio(returns)
        # Define weights for the objectives
        weight_gain = 0.7
        weight_sharpe = 0.3
        # Combine the objectives into a single fitness value
        fitness_value = (weight_gain * total_gain_value)
        + (weight_sharpe * sharpe_ratio)
        return fitness_value
num_generations, sol_per_pop = 2,1
crossover_probability,mutation_probability = 0.8, 0.02
```

Chapter 4

Interpretation of the results

An important notice to make about fuzzy rules (**Fig. 4.1**) which is expressed in this project is that in certain ranges (e.g from less than 60 to around 70) a number can be associated with multiple membership functions, and this represents one of the main characteristics of the fuzzy systems.

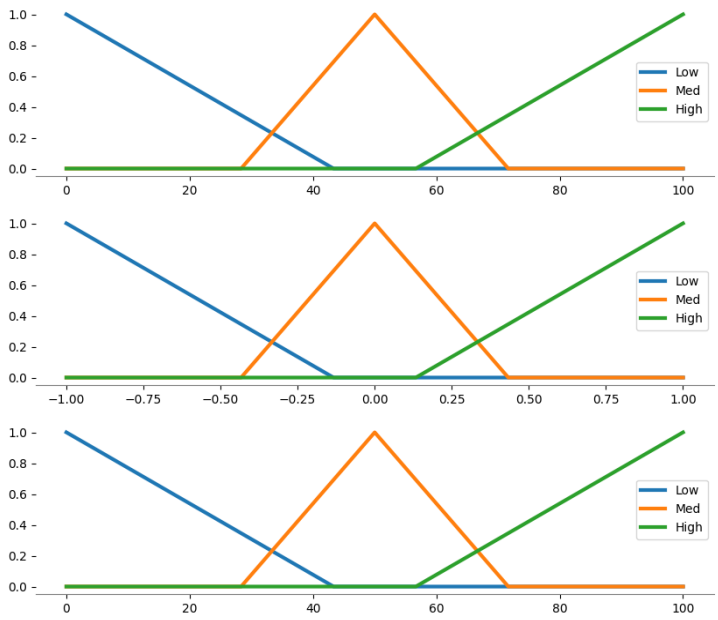

Fig. 4.1: Fuzzy Sets of RSI, MACD, and STO, respectively.

The outputs of the inference systems can be visualized graphically (**Fig. 4.2**). In this plot, each line represents a distinct set of fuzzy rules that evaluate the values of the technical indicators by comparing them against them respective fuzzy sets: High, Medium, or Low. These rules process the inputs, while the final column reflects the Buy-Sell Decision.

It is relevant to notice that the red vertical line marks the specific input values, while the size of the shaded area indicates how much the input fits into the fuzzy category. In this example, the values should be good enough to buy equity, since *Decision = 3.95*.

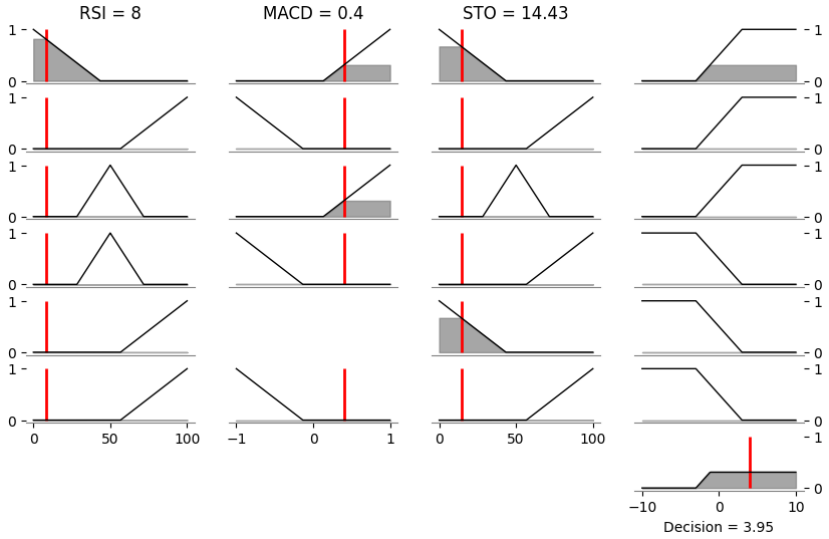

Fig. 4.2: An example of inference systems' results on Chevron performance

Code Outcome

Analysing the final graphs, the general and maximum equity lines across all stocks show consistent upward trends, reflecting robust profit generation predominantly driven by long positions, as indicated by the closely aligned long equity lines. Conversely, the short equity lines remain relatively flat in each graph, indicating proportionally low contributions from short positions.

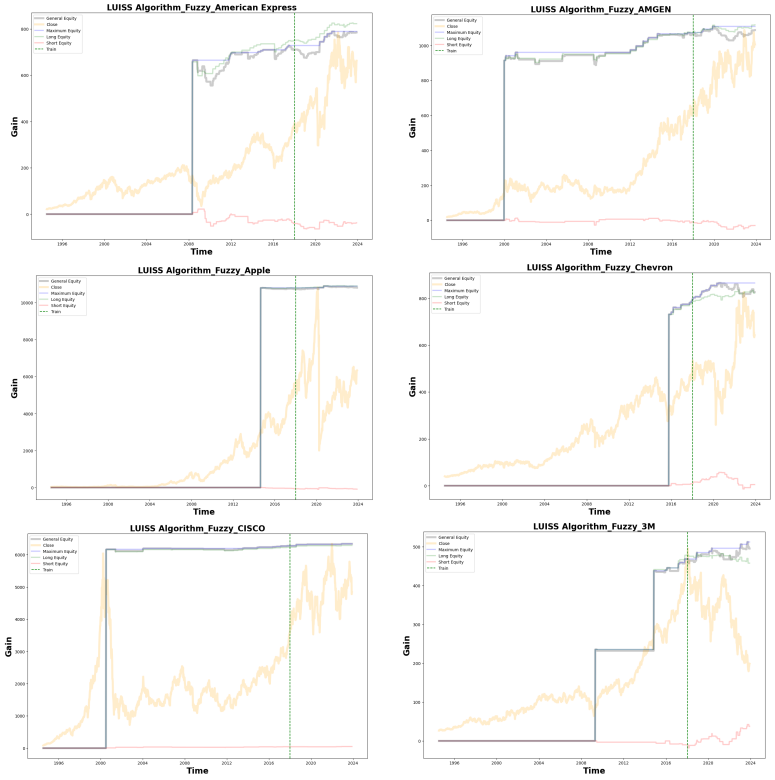

Fig. 4.3: Model plots

The vertical dashed green line again represents the transition from training set to the test set, in which the algorithm is performed on new unseen data. In the test set, which represents the most recent period of the time series, the model's performance has been relatively solid and stable. It makes robust, stable performances across different market conditions. Cisco had speedy growth in the early stages, followed by solid volatility around the 2000s; on the other hand, 3M and American Express maintained a smooth trajectory, recovering quite well after 2008. The algorithm consistently captures substantial gains on upswings and is resilient during downturns, maintaining mostly long positions.

HyperParameter Tuning results

The overall reason for the differences in fitness values, observable in the below tables, between the *Original* and *Model II* is the inclusion of risk (through

the Sharpe ratio) in the latter. While the Original model focuses solely on maximizing returns, Model II provides a risk-adjusted performance measure, leading to lower but more balanced fitness values. This approach ensures that both returns and risks are considered, offering a more comprehensive evaluation of stock performance. On the other hand, The similarity in fitness values between the Original model and Model I suggests that the original optimization method is already quite effective for the given stock data. The genetic algorithm enhancements (additional generators, mutation, and crossover) do not significantly alter the results, likely due to the saturation of performance, algorithm efficiency, limited impact of parameters, local optima, or the characteristics of the stock data.

This indicates that for this specific set of data and optimization goal (maximizing gains), the additional complexity introduced in Model I does not provide substantial benefits over the Original model.

Am.Express	Fitness Value
Original	708.6
Model I	708.58
Model II	495.83

AMGEN	Fitness Value
Original	1063.8
Model I	1063.79
Model II	744.47

Apple	Fitness Value
Original	10730.45
Model I	10752.79
Model II	7511.2

Chevron	Fitness Value
Original	795.1
Model I	795.1
Model II	556.31

CISCO	Fitness Value
Original	6290.821
Model I	6296.3
Model II	4403.427

3M	Fitness Value
Original	468.522
Model I	468.5216
Model II	327.7

Conclusion

This thesis explored the optimization of long-short equities using Fuzzy Logic and metaheuristic techniques, addressing the complexities and uncertainties of financial markets. By integrating these methods, it has been created a robust framework for strategic decision-making.

Overall, the paper successfully achieved its objective by demonstrating the capabilities of Fuzzy Logic and metaheuristic techniques for optimizing long-short

equities, both conceptually through detailed analysis. The implementation demonstrated promising returns and stability across different markets, highlighting the practical potential of the strategy.

This work aims to provide valuable insights for future research and real-world applications. While the availability of larger datasets, higher computing power, and more resources could potentially enhance model performance in the future, our analysis still yielded appreciable results within the given constraints. Consequently, this thesis serves as a solid foundation for next researches in the domain of financial optimization.

Bibliography

[1] Fung, W., Hsieh, D. A. (2011). The risk in hedge fund strategies: Theory and evidence from long/short equity hedge funds. Journal of Empirical Finance, 18(4), 547-569.

[2] Larner, A. J. (2024). The 2x2 matrix: contingency, confusion and the metrics of binary classification. Springer Nature.

[3] Riyanto, S., Imas, S. S., Djatna, T., Atikah, T. D. (2023). Comparative Analysis using Various Performance Metrics in Imbalanced Data for Multi-class Text Classification. International Journal of Advanced Computer Science and Applications, 14(6).

[4] James, Gareth. author. (2013). An Introduction to Statistical Learning with Applications in R / (Vol. 103). Springer New York: https://doi.org/10.1007/978-1-4614-7138-7

[5] Ozenne, B., Subtil, F., Maucort-Boulch, D. (2015). The precision recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases. Journal of clinical epidemiology, 68(8), 855-859.

[6] Qu, J., Zhang, L. (2023). Application of maximum Sharpe ratio and minimum variance portfolio optimization for industries. Highlights in Business, Economics and Management, 5.

[7] Zakamouline, V., Koekebakker, S. (2009). Portfolio performance evaluation with generalized Sharpe ratios: Beyond the mean and variance. Journal of Banking and Finance, 33(7), 1242-1254.

[8] Amedee-Manesme, C. O., Barthelemy, F. (2022). Proper use of the modified Sharpe ratios in performance measurement: rearranging the Cornish Fisher expansion. Annals of Operations Research, 313(2), 691-712.

[9] Sun, A., Lyuu, Y. D. (2022). Backtesting Trading Strategies with GAN To Avoid Overfitting. arXiv preprint arXiv:2209.04895.

[10] Koshiyama, A., Firoozye, N. (2019). Avoiding Backtesting Overfitting by Covariance-Penalties: an empirical investigation of the ordinary and total least squares cases. arXiv preprint arXiv:1905.05023.

[11] Singhala, P., Shah, D., Patel, B. (2014). Temperature control using fuzzy logic. arXiv preprint arXiv:1402.3654.

[12] Pik,J.,Ghosh,S.(2021).Hands-On Financial Trading with Python: A Practical Guide to Using Zipline and Other Python Libraries for Backtesting Trading Strategies.(n.p.):Packt Publishing.

[13] Broz Z., Dostal P. (2013). Fuzzy logic decision support for long-term investing in the financial markets. In Nostradamus: Modern Methods of Prediction, Modeling and Analysis of Nonlinear Systems (pp. 113-121). Berlin, Heidelberg: Springer Berlin Heidelberg.

[14] Klir, G., Yuan, B. (1995). Fuzzy sets and fuzzy logic (Vol. 4, pp. 1-12). New Jersey: Prentice hall.

[15] M.M. Gupta, J.B. Kiszka, Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems, Editor(s): Robert A. Meyers, Encyclopedia of Physical Science and

Technology (Third Edition), Academic Press, 2003, Pages 355-367, ISBN 9780122274107, https://doi.org/10.1016/B0-12-227410-5/00270-2. (https://www.sciencedirect.com/science/article/pii/B0122274105002702)

[16] Dongrey, S. (2022). Study of market indicators used for technical analysis. International Journal Of Engineering And Management Research, 12(2), 64-83.

[17] Jain, N., Mittal, S. (2022). A computational model for driver risk evaluation and crash prediction using contextual data from on-board telematics. Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science), 15(4), 620-635.

[18] https://www.alpharithms.com/trading-indicators-493009/

[19] https://www.alpharithms.com/stochastic-oscillator-574218/

[20] https://commons.wikimedia.org/wiki/File:Roccurves.png

[21] https://commodity.com/technical-analysis/macd/

[22] https://commodity.com/technical-analysis/relative-strength-index/

[23] Handling uncertainties in the seismic analysis using fuzzy theory - Scientific Figure on ResearchGate.

[24] Lee, K. H. (2005). Fuzzy Control and Fuzzy Expert Systems. First Course on Fuzzy Theory and Applications, 253-283.

[25] Saber, M., Abdelhamid, A. A., Ibrahim, A. (2023). Metaheuristic Optimization Review: Algorithms and Applications.Journal of Artificial Intelligence and Metaheuristics,3(1), 21-1.

[26] Affenzeller, M., Winkler, S., Wagner, S., Beham, A. (2009).Genetic algorithms and genetic programming: modern concepts and practical applications(p. 394). Taylor Francis.

[27] Eberhart, R. C., Shi, Y., Kennedy, J. (2001). Swarm intelligence. Elsevier

[28] Chakraborty, A., Kar, A. K. (2017). Swarm intelligence: A review of algorithms.Nature-inspired computing and optimization: Theory and applications, 475-494.

[29] https://www.influxdata.com/blog/exponential-smoothing-beginners-guide/

[30] https://github.com/aresio/fst-pso

[31] https://www.intechopen.com/chapters/75445