Mixture of Random Forest Experts in Options Portfolio Management

Rodrigo Manuel Lopes da Silva Lousada

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Rui Fuentecilla Maia Ferreira Neves

September 2020

Acknowledgments

I want to thank my dissertation supervisor Prof. Rui Ferreira Neves, who has been a great advisor and teacher for the past year. Since I can remember, I had the desire to learn more about Investment and the Financial Markets. Professor Rui made it possible to conciliate my Data Science thesis on this topic. I want to thank him for all the support, tips, lessons and availability that he showed while supervising my thesis. I would also like to share my gratitude with my thesis colleagues João Belo, Rodrigo Almeida, António Lourenço, Renato Henriques, Marco Montez, Pedro Bastos and the rest of the group for all the communication and discussions about what we have been learning in this journey.

A significant appreciation to my family, especially to my parents and siblings, for their support, encouragement, and caring over all these years. Thank you for pushing me to work more and better. Thank you for always being there for me and advise me. Thank you for teaching me to ever see the brighter side in life. The past five years would never be possible without you.

I would also like to thank family Aparício da Costa for all their support and tips about the research and thesis process. Their interest in research and learning is unlimited and contagious.

To all my friends and colleagues, that helped me grow as a person and were always there for me during the good and bad times in my life. I will always carry in my heart the long nights studying in Lab 15, the coffee breaks at Mónica's, as well as the BBQs to distress after a long project.

Last but not least, to my girlfriend and colleague, Sofia, for all the love, comfort and support. For always motivating me to be more productive and keep improving as a person and student. Both of us experienced the struggle of working on our thesis at the same time. I would never make it without her support.

This was a journey of emotions. The path was long and hard, but here we are!

Abstract

The financial markets have been studied in many research works. Although many Machine Learning solutions try to predict the stock market, the derivatives market still has a lot to explore. This Master thesis main goal is to develop an algorithm able to trade financial products. To achieve this main goal, a modern approach to trade in the Options Market was used, through an incremental validation, aiming a portfolio with high profits and associated controlled risk. The architecture includes an approach based on Technical Analysis and Machine Learning to develop a system that goes from collection to process, forecast and trade the options data. It uses a Mixture of Experts (MoE) composed of Random Forests (RaF) to optimize the forecasting performance of the S&P500 index (SPX). The final work overcomes the previous literature with an Annual Rate of Return (ROR) of 100% and Sharpe Ratio of 1.90, on the year of 2015 when a simple Buy&Hold on the SPX only achieves 0.22% of ROR and 0.09 of Sharpe Ratio. On the second half of 2017, when the market was bullish and the options market unstable, it also achieved more than 42%, and a single RaF was able to triplicate the invested money. The main implications of this work provide a solid strategy to accomplish higher profit margins in the options market, especially in periods of high volatility, providing one of the best Return Risk Ratios (RRR) reported, without resorting to Fundamental Analysis data.

Keywords

Machine Learning; Ensemble Learning; Technical Analysis; Options; Mixture of Experts; Portfolio Management;

Resumo

Os mercados financeiros têm sido estudados por diversos trabalho de investigação. Embora muitas soluções em Aprendizagem de Máquina tentem prever o mercado de ações, o mercado de derivativos continua com muito por explorar. O principal objectivo desta tese de Mestrado é o desenvolvimento de um algoritmo capaz de fazer negociação de produtos financeiros. Para atingir este objectivo, foi utilizada uma abordagem moderna para lucrar no mercado de opções, procurando um portfolio com altos retornos e baixo risco associado. A arquitetura inclui uma abordagem baseada em Análise Técnica e Aprendizagem de Máquina para desenvolver um sistema que vai da coleção ao processamento, previsão e negociação de opções. É utilizada uma Mistura de Especialistas (MoE) composta por Florestas Aleatórias (RaF) para otimizar a predição do indice S&P500 (SPX). O trabalho final supera a literatura com uma Taxa de Retorno (ROR) de 100%, e 1.90 de índice Sharpe, no ano de 2015, enquanto uma simples estratégia de ”Comprar e Manter” no SPX apenas consegue 0.22% de ROR e 0.09 de índice Sharpe. Na segunda metade de 2017, quando o mercado está a subir e o mercado das opções se encontra instável, o nosso algoritmo atinge os 42% de retorno, e uma única RaF triplicou o dinheiro investido. As principais contribuições deste trabalho oferecem uma estratégia sólida que atinge retornos altos no mercado das opções, especialmente em períodos de grande volatilidade, conseguindo um dos melhores Rácios Retorno-Risco (RRR) reportados, sem ter de recorrer à Análise Fundamental.

Palavras Chave

Aprendizagem de Máquina; Aprendizagem Conjunta; Análise Técnica; Opções; Mistura de Especialistas; Gestão de Portfólio;

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ANOVA	Analysis of Variance
API	Application Programming Interface
ATR	Average True Range
BBAND	Bollinger Band
B&H	Buy&Hold
CART	Classification And Regression Trees
CBOE	Chicago Board Options Exchange
CSV	Comma-separated values
CV	Cross-validation
DT	Decision Tree
DX	Directional Movement Index
EDA	Exploration Data Analysis
EMA	Exponential Moving Average
ETF	Exchange Traded Fund
FRED	Federal Reserve Economic Data
IMI	Intraday Momentum Index
ITM	In-the-Money
KNN	K-Nearest Neighbors
GA	Genetic Algorithm

HMoE	Hierarchical Mixture of Experts
LCS	Learning Classifier System
MDD	Maximum Drawdown
MACD	Moving Average Convergence/Divergence
MFI	Money Flow Index
ML	Machine Learning
MoE	Mixture of Experts
MOM	Momentum
MoRFe	Mixture of Random Forest Experts
MOEA	Multi-Objective Evolutionary Algorithm
MSE	Mean Squared Error
NN	Neural Network
OBV	On-Balance Volume
OHLC	Open, High, Low and Close
OI	Open Interest
OTM	Out-of-the-Money
PCA	Principal Component Analysis
PCR	Put-Call Ratio
PoC	Proof of Concept
RaF	Random Forest
ROE	Return on Equity
ROI	Return on Investment
ROR	Rate of Return
RRR	Risk Return Ratio
RSI	Relative Strength Index

S&H	Sell&Hold
SMA	Simple Moving Average
SMOTE	Synthetic Minority Oversampling Technique
SPX	S&P500 index
SVM	Support Vector Machine
SVR	Support Vector Regression
TI	Technical Indicator
UI	User Interface
VIX	Volatility Index
WILLR	Williams %R
WRDS	Wharton Research Data Services
XCS	Extended Classifier System
YAML	Yet Another Markup Language

Software List

Editors and Programs:

- **GitHub:** Distributed version control host.
- **Jupyter Notebook:** Platform for cell-level code execution.
- **Overleaf:** Online LaTeX editor.
- **PuTTY:** SSH Client with terminal emulator for connection with Server.
- **Visual Studio Code:** Code Editor.
- **WinSCP:** SFTP client for file transfer with Server.

Programming Languages:

- **Bash:** Unix shell and command language for scripting.
- **Markdown:** Markup language using a plain-text-formatting syntax.
- **Python:** Main Programming Language used.
- **YAML:** Ain't Markup Language used for the parametrization files.

Libraries:

- **Appmode:** Extension that creates interactive webapps from a Jupyter Notebook.
- **Imbalaced-learn:** Package that gives re-sampling techniques used in datasets with strong between-class imbalance.
- **JobLib:** Set of tools to provide lightweight pipelining.
- **Matplolib:** Library for creating static, animated, and interactive visualizations in Python.

- **Numpy:** Package for array processing. Produces a high-performance multidimensional array object, and mechanisms for working with these arrays.
- **Pandas:** Data Analysis and Manipulation tool.
- **Pickle:** Binary protocols for serializing and de-serializing a Python object structure.
- **Plotly:** Visualization Framework.
- **PyFolio:** Library for performance and risk analysis of financial portfolios.
- **QuantStats:** Library for performance and risk analysis of financial portfolios.
- **Scikit-learn:** Open source Machine Learning library to implement Supervised and Unsupervised Learning.
- **TA-Lib:** Tools for Market Analysis.
- **Tqdm:** Progress Bars for loop time estimation and progress visualization.

1

Introduction

Contents

Nowadays, financial freedom has become one of the primary goals of the young generations. Trading in the financial markets is one of the many paths they choose to seek fortune. Although all the warnings about the multiple challenges it presents and the risk of catastrophic loss, new fanatics arrive every year. Trading in the financial markets is hard, and it has so many factors to have in consideration that exceeds human capacity [2]. While many people choose to trade more straightforward assets like stocks and forex, bolder traders embrace the challenge of understanding more complex financial instruments and exploit their profits. The Options market is an example of a prosper market which, although famous, it is still underexplored. The field of Machine Learning (ML) has been researching this question for years, and developed solutions able to do automated trading in very short-term periods. These strategies, aligned with the usage of multiple technical indicators, allow us to extract more information and achieve a deeper understanding of the financial outcomes [3]. The usage of the Volatility Index (VIX) index was also referred to as an essential tool for forecasting improvement [4, 5]. Finally, in recent literature, Random Forest (RaF) has revealed to be an excellent constituent when ensembled in a combined solution [6–9]. The concept of a Mixture of Experts (MoE) is still growing, having very few applications on the financial markets [10]. The proposed architecture was, to the best of our knowledge, never used before in Options trading.

1.1 Motivation

Financial Markets exchange large quantities of capital every single day. Worldwide, there are cases of investors and traders who achieved large volumes of income after learning the art of investment [11]. However, that was never an easy task. As a ”zero-sum” game, more than 80% of traders lose money [12]. Even some of the most successful ones had times of significant losses [13]. Unlike most of the ML applications, values like precision and accuracy say little about the success of using an algorithm that will manage a trader's portfolio. Despite the forecasting performance of an algorithm, a single wrong prediction can make the trader lose all his money or even create debt. Besides that, the financial data is noisy, stochastic, and has a variety of unknown factors that can influence it [3]. It is essential to resort to indicators which can help us to manage risk and maximize profit.

Apart from that, research on Ensemble Learning methods is arising. The MoE method allows better performance on large datasets like the ones related to financial markets [10]. Extra motivation will be the exploration of a solution to distribute the price signal through experts, and the development of an approximation to an optimal gating function. Literature indicates the need to explore more in-depth the use of ML to predict the behaviour of the Options market [14].

1.2 Thesis Goals

During the development of the proposed work, this thesis achieved:

- Develop a solution based on Ensemble Learning methods, RaF and Technical Analysis to create a profitable and reliable system.
- Maximize profit and minimize risk in an autonomous trading simulation.
- Contribute with reference work for ML applications on the Options Market.
- Help to improve the current literature on MoE.
- Contribute with an Open Source project to the ML and Finance communities.
- Motivate future works on the Finance field to share the code and tools which help them generate profits and manage risk in the market.

1.3 Methodological Approach

This thesis developed a scalable trading system which exploits ML and Technical Analysis to interpret past data and forecast the direction of the market while simulating trading decisions to generate profit and reduce risk. It compares the usage of a single RaF algorithm, with a MoE [15] composed of a set of RaF experts.

The stated results were tested in a laboratory environment where the available data was split into training, validation and test periods. An iterative approach was used to maximize the performance of the algorithm by experimenting with multiple hyperparameter combinations on the validation sets. This validation follows the Cross-validation (CV) technique when applied to Time Series forecasting. The different strategies were tested based on the same evaluation metrics, which analyses their forecasting capacity and trading performance.

1.4 Thesis Results

The solution in hands achieved accuracy scores between 40% and 50% in both test periods and overcame the literature results in terms of trading performance. In a sideways period of high volatility, this work achieved an Annual Rate of Return (ROR) around 100% and Sharpe Ratio 1.89. On the other hand, on a period of very low volatility, the solution was able to exploit the unstable prices and achieve profit in all strategies, having a single RaF with 212% profit in only six months. In this period, the algorithm achieved profit even when the market was uncertain due to its overvalued price.

1.5 Thesis Contributions

This Master thesis has 5 contributions, both to Science and Industry. The main contributions are the following:

1. Generated an impartial and reliable system for choosing the best Call and Put Options to trade in the six months to expiration.
2. Developed an innovative and ground-breaking ensemble algorithm which gather predictions from different learners and weights their predictions based on their expertise.
3. Created personalized functions which split the data into different groups to focus each learner on a specific part of the dataset.
4. Developed a Trading System which simulates trading for a given period based on the predictions of a classifier.
5. Designed a ML system that is available for Finance specialists to give their input without requiring programming knowledge.

1.6 Document Structure

The document is structured as follows:

- **Chapter 1 (Introduction)** introduces the project, the authors' motivation, and the thesis goals and contributions.
- **Chapter 2 (Background)** gives support on the theoretical background necessary to fully understand the content of this thesis, followed by a description and analysis of the current state-of-the-art.
- **Chapter 3 (Architecture)** structures the complete architecture for the developed system, explaining the decisions made during the development process, along with deeper explanations of other theoretical concepts.
- **Chapter 4 (Results)** displays and analysis the achieved results, comparing the multiple approaches in two case scenarios, determining the level of success delivered by the solution.
- **Chapter 5 (Conclusions)** summarizes the main conclusions and achievements, followed by suggestions for improvements and future work.

2

Background

Contents

In this chapter, most of the theoretical background will be covered, starting by introducing the basics of Financial Concepts, followed by a detailed explanation of the Options Market. Secondly, the most popular methods for market analysis, and how this can improve the final results will be explained. Then, we will describe Machine Learning (ML) and its role in financial forecasting. Finally, we analyze the State of the Art on single predictive algorithms applied to financial markets, Ensemble Learning, Mixture of Experts (MoE) architectures, and applications over Derivatives Markets.

2.1 Financial Concepts

Financial Markets describe a variety of marketplaces responsible for trading securities or assets such as equities, currencies, derivatives, commodities or bonds. These markets set prices which allow buyers and sellers to trade globally in an open and decentralized system, with the intention of raising capital and relocate risk and liquidity.

Depending on the specific analysis, it is possible to characterize those markets by the categories in Table 2.1

Table 2.1: "Asset Classes and Financial Instruments" by Bodie. Source: [1]

Financial Markets				
Money Market	Bond Market	Equity Securities	Stock and Bond Indexes	Derivative Market
Treasury Bills	Treasury Notes and Bonds	Common Stock	Stock Market Indexes	Options
Certificates of Deposit	Inflation-Protected Treasury Bonds	Characteristics of Common Stock	Dow Jones Averages	Futures
Commercial Paper	Federal Agency Debt	Stock Market Listings	Standard & Poor's Indexes	
Bankers' Acceptances	International Bonds	Preferred Stock	Bond Market Indicators	
Eurodollars	Municipal Bonds	Depository Receipts	Equally Weighted Indexes	
Repos and Reverses	Corporate Bonds		Foreign Stock Indexes	
Federal Funds	Mortgages Securities		International Stock Indexes	
Brokers' Calls	Mortgage-Backed Securities		Bond Market Indicators	
The LIBOR Market				
Yields on Instruments				

The maximization of success prospects, while investing in financial markets, is firmly associated with a robust market analysis. Most of the strategies we will cover try to win better results, generally having as a common benchmark the Buy&Hold (B&H) strategy.

Investors typically own a financial portfolio, which intends that the money invested by them will not be allocated to a single asset, but rather, partially invested in several of them. Most portfolio management strategies consist of picking a variety of securities which will, as a combination, mitigate the risk while exploiting the profits [16].

Although the ideal manoeuvring would maximize the profit while suppressing the risk, higher profits tend to ask for higher risk tolerances [17]. Even though the core objective will be to produce the highest profit possible, there is always a limited amount of risk of losing that we are willing to take. Therefore, the biggest challenge of investment is knowing how to maximize both goals and choosing when to sacrifice one from the other.

2.1.1 Buy&Hold vs Sell&Hold

The Buy&Hold (B&H) is an approach confident that a certain asset will be more valuable in the future. Therefore the investor keeps this asset for a sustained period, ignoring possible short-term price fluctuations. After holding the asset for a determined period, the investor expects to take some profit from selling the asset. This B&H is the most straightforward and easy strategy possible because we only need to wait. As this strategy requires a low amount of effort, every new strategy, to be worthy, has to beat it in one of two ways: having more profit or less risk.

Similarly to the B&H strategy, the Sell&Hold (S&H) strategy is applied if the investor enters the market, expecting the asset to decrease its value in the future, regardless short-term price fluctuations. It is essential to understand that, as the relative price of an asset will range from zero to infinite percent, this position is an action of massive risk. When an asset is bought, we consider its value as 100% of the capital invested. A B&H in the worst-case scenario loses the money invested if the price reaches 0%, against the possibility of exponential profit if the price keeps growing. On the other hand, a S&H in the best case scenario can profit the total amount of money received in the initial sale if the asset's price falls to zero, among the possibility of limitless debt.

2.1.2 Investment Positions

The investor has the possibility of adopting one of three positions: long, short or neutral. If the investor considers that the market is currently unreliable and hard to predict, he could opt to stay out of it, without investing on the market. This decision is called taking a ”Neutral position”. In the previously explained B&H strategy, the investor tackles the market by buying and owning a specific asset, hoping for the price to go up, which can be described as an extended ”Long position”. The investor who takes a Long position, expects the asset to raise value while holding it so that he can sell it for a higher price.

On the other hand, if the investor expects the price to fall, it is possible to have a ”Short position”, selling the asset before actually owning it. By doing this, the investor sells an asset borrowed by an investor or broker. After a while, he buys this asset to the entity who borrowed it by the current market price, which he expects to be less than what he sold it.

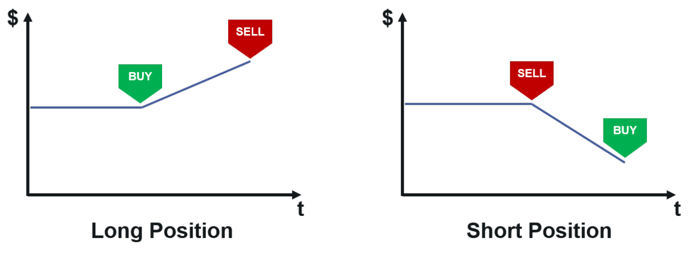


Figure 2.1: Long and Short position examples

Figure 2.1, represents two example of the goal for both, Long and Short, positions. It is easy to envision that a significant amount of strategies only invest in the expectation of price increase, due to the less risk associated with that position. In order to profit more than the B&H and S&H strategies, one should vary these positions following the direction of the market.

2.1.3 Trends

The previously explained investment positions are strongly related to the concept of ”trend”, which market analysts use to define the direction of the market. As illustrated in figure 2.2, an uptrend is described as a prevailing increase of peaks and throughs, revealing a direction of the stock price going up. A downtrend denotes that those peaks and throughs will have a descending direction. If the market is considered ”trendless” because of the horizontality of the peaks and throughs we call it a sideways trend.

The concept of a Bull market occurs when a particular market is envisioned to be consistently in an uptrend. If the market persists in a downtrend, it is labeled as a Bear market. [18] Those outlines will have a tremendous impact on the investors' confidence to buy the stock.

Investors tend to assume a Long position in a Bull market, as the price will eventually go up, while Short-selling in a Bear market, selling the security while the price is high, and ”covering” the sell after the price drops. The real mystery prevails on what position to adopt while the market is sideways. The most common decision is to stay neutral, as it is unpredictable which way will the price breakout.

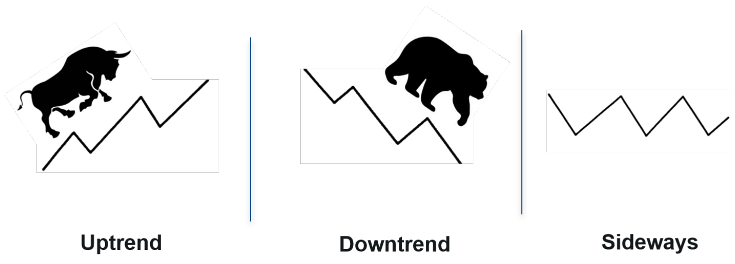


Figure 2.2: The three different market trends

2.2 S&P500 index (SPX)

A ”stock index” is a weighted average of the prices of a selected set of stocks. The most famous one is the S&P500 index (SPX), illustrated in Figure 2.3, which measures the performance of 500 large-capitalization companies exchanged on the American stock markets. The S&P500 is extensively quoted as the U.S. index, as it accurately portrays the American Economy. Therefore it is used as a benchmark for upcoming strategies. This index is directed to America's largest sector but is also a float-weighted

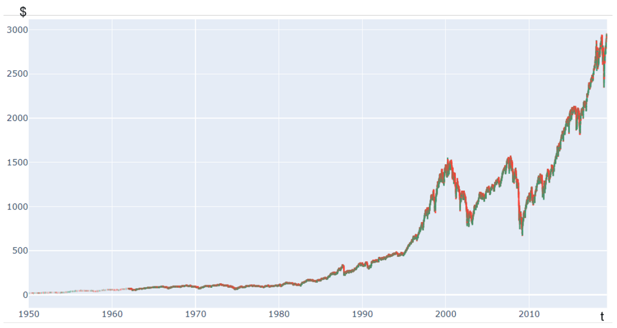


Figure 2.3: S&P500 Index (1950-2019)

index, indicating that market capitalizations adjust with the number of shares accessible to public trading. Other American market benchmarks include Dow Jones Industrial Average and the Russell 2000 Index.

2.3 Options Market

”Options” are a derivative product from the financial markets. Writing an Option designates an investment contract between an Option writer and an Option buyer (or holder). This exchange gives the buyer the right, but not the obligation, to buy (Call Options) or sell (Put Options) an underlying asset in a later date, for a certain fee. This fee will be higher for a more extended period. After the expiration date, the contract will have no value and will no longer exist.

An Option contract works as insurance for buyers, as the buyers will pay a small fee to safeguard their money if they take the wrong position in the market. In the worst case scenario, the investor would let the Option expire, ending up without the money spent on the fee. Three aspects determine this fee:

1. **Time to expiration** - as further in time the expiration date is, more time the investor has to profit. Therefore, more valuable is the Option. Due to the passage of time, the value of an Option tends to drop while we get closer to its expiration date.

2. **Underlying stock price** - both parties will look at the underlying price of a stock and agree on a fixed price for the contract. This price is known as the ”strike price”, which is how much will those shares cost if the Option is exercised.

3. **Volatility** - represents the risk for the stock owner, based on the magnitude of fluctuations on an asset’s price. An Option will be more worthwhile if associated with a volatile asset.

While buying an Option, the investor starts the venture with an expense of the fee cost. This cost implies that the investor, to take profit, should expect that the underlying price of the asset will rise or decrease more than the cost associated with this fee. Figure 2.4 illustrates the possible payoff of an Option depending on the intrinsic value of the Option.

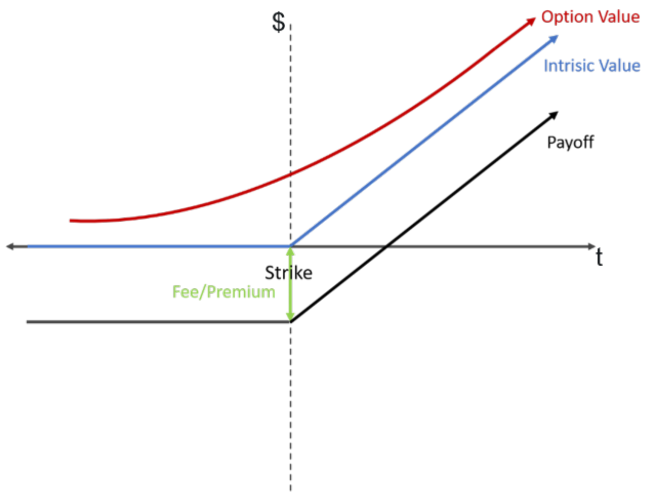


Figure 2.4: Option payoff range

The investor has three possible approaches: exercising the Option, letting the Option expire, or selling the valid Option to another entity.

In figure 2.4 it is possible to see an Intrinsic Value line (in blue) and an Option Value line (in red). The Option Value represents the price an Option is being exchanged. It is the sum of two values: the Intrinsic Value and the Extrinsic Value. The Intrinsic Value is the direct value of an asset at a certain moment, which in an Option, is how much an investor would profit from exercising the Option. The Extrinsic Value of the Option is a speculative calculation of how much more the Option is probable to value.

The largest and most famous US Options exchange market is the Chicago Board Options Exchange (CBOE). Most studies on Options explored data collected from this exchange market.

2.3.1 American and European Options

There are two kinds of agreements regarding the time to the expiration date, American Options and European Options. While European Options are only possible to exercise on the expiration date, the American Options may be exercised at one's convenience until the expiration date. Since the second ones can be exercised for the whole period that a buyer holds the Option, they will be more expensive than the European ones.

2.3.2 Call and Put Options

As explained before, an Option can be a contract that gives the right to buy or to sell an asset. When the investor foresees that the price will rise, he will buy a Call Option, which is the right to buy that stock further in time. On the other hand, if the investor envisions a recession, he will buy a Put Option, which is the right to sell that stock further in time. In both cases, the investor will buy or sell the asset by the strike price on the contract.

It is possible to compare this with the notions of short and long selling, explained in the previous section. Buying a Call Option is equivalent to take a long position on the market while buying a Put Option can be compared to take a short position on the market. The main difference here is that the investor pays a fee (or a premium price) to fix that price and only decide if he is going to take the move, at the expiration date. Acquire the Option will end up reducing the risk and possible loss taken. By that time, the investor will already know the outcome of having taken a short or long position.

Both types of Options for stocks are typically sold in batches of 100 shares. So, exercising an Option of $\$0.35$ will actually cost $\$0.35 * 100 = \$35$.

2.3.3 In-the-Money (ITM) and Out-of-the-Money (OTM) Option

While the strike price is a fixed price agreed between both entities, the underlying stock price will vary during the time that the Option is valid. The relation between both prices can fall within two situations: In-the-Money (ITM) or Out-of-the-Money (OTM), illustrated in Figure 2.5. ITM describes the situation where an Option holder will profit. This situation happens if the strike price of a Call Option is less than the underlying stock price, or if the strike price of a Put Option is more than the price of the underlying stock. Contrarily, OTM is the case when it was wise to sign the Option instead of solely invest in the asset. In those cases, the investor should let the contract expire. It can happen when a Call Option's strike price is above the underlying stock price, or when the Put Option's strike price is under the underlying price. Therefore, an Option will be more valuable if it is ITM.

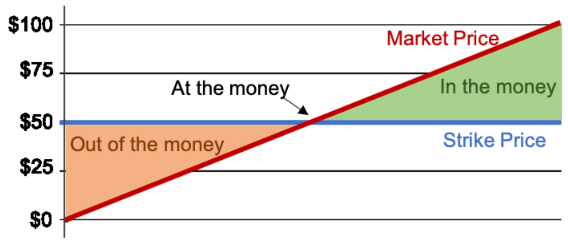


Figure 2.5: ITM and OTM representation

2.3.4 Liquidity, Volume and Open Interest

Liquidity represents the activity of an asset in the market, and how easy this asset can be transacted in the market, converting it into cash. The best way to measure Options liquidity is to analyze two important measurements: Open Interest (OI) and Volume.

OI is the number of active contracts that have not been exercised, expired or closed out, while Volume indicates the number of contracts being exchanged (or traded), between buyers and sellers, in a given period. Both metrics are indicators of the current interest in a specific asset, contract or extrinsic value.

Compared to other financial instruments, Options are traded far less frequently which can make it hard for investors to enter and close positions for a specific Option, at a specific time. For every buyer, there has to be a seller, and vice-versa. An investor may buy an Option, that Option can increase in value, but when the investor wants to sell the Option to take his profits without exercising the Option, he has to find a buyer for the same Option.

2.3.5 Volatility and VIX

Volatility regards the amount of uncertainty related to the dimension of change in an assets' price. Higher volatility indicates that a security's price will potentially vary over a larger range. If the Volume of an asset increases strongly, it is normal to expect higher volatility for the same asset, therefore it will present larger swings on price.

The Volatility Index (VIX), illustrated in Figure 2.6, was created by the CBOE. It is a market index which expresses the expectation of the volatility for the next 30 days. It is calculated through the price of the SPX Options, and presents a measure of the market risk and investors' reactions. Many researchers use VIX values to measure market risk, through the expected volatility on the VIX index [4, 5]. Equation 2.1 is provided by the CBOE.

$$\sigma^2 = \frac{2}{T}\sum_i \frac{\Delta K_i}{K_i^2} e^{RT} Q\left(K_i\right) - \frac{1}{T}\left[\frac{F}{K_0} - 1\right]^2 \tag{2.1a}$$

In equation 2.1a, σ is the $\frac{VIX}{100}$, which means that $VIX = \sigma \times 100$. The T is the expiration time, F is the forward index level desired from index Option prices, and K_0 is the first strike below F.

K_i is the strike price of the ith Option that is OTM. This Option will be a Call Option if $K_i > K_0$, or a Put Option if $K_i < K_0$. In case of $k_i = k_0$ the Option will be both, a Call Option and a Put Option.

ΔK_i gives the interval between strike prices, which can be represented as half of the difference between the strike price of both adjacent Ks of K_i.

$$\Delta K_I = \frac{K_{i+1} - K_{i-1}}{2} \tag{2.1b}$$

Finally, R is the Risk-free Interest Rate to expiration, and $Q(K_i)$ is the midpoint of the bid-ask spread for each Option with strike price K_i.

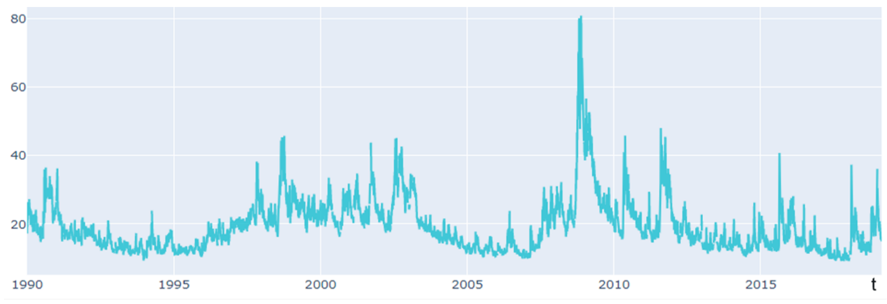


Figure 2.6: VIX (1990-2019)

2.4 Market Analysis

Sorts of examination about when and which assets to buy or sell may vary on the outlook and situation of each investor. However, regardless of the specific financial market, we can narrow most of the approaches in two branches: Fundamental Analysis and Technical Analysis. Although some investors tend to base their tactics on only one of these categories, recent researches exhibited good profits from combining them [19].

2.4.1 Fundamental Analysis

Fundamental analysis is an evaluation method with the intention to estimate an asset's ”true” value and predict its future price. The central aspiration consists in understanding whether the current price of the security represents an under or overvaluation. If the investor believes that a company is undervalued, it may be an excellent opportunity to take a Long position. On the other hand, if the investor analyses a company that is overvalued, he can decide to take a Short position.

This evaluation explores a variety of related financial, economic, quantitative and qualitative factors that can have an impact on the asset's price/success.

Most of the factors can be compiled in three essential layers: economic analysis, industry analysis, and company analysis. The first one relates to the examination of the national and international economic situation. The second describes the industry conditions related to customers, regulation, competition, market share, and industry growth. Finally, company analysis uses financial indicators and statements to measure liquidity, debt, growth, and profitability of a company.

While doing fundamental analysis, one can work in a top-down approach, which means starting to pick an economy and going down to select a bunch of companies underneath to analyze. However, it is also possible to follow a bottom-up approach by picking a bunch of attractive companies and further analyze the related industrial and macroeconomic factors.

Fundamentalists use historical and recent data to look at the market in a logical and emotionless perspective. This strategy does not take into consideration the investors' feelings or trends. They believe that despite short-term fluctuations, the market will eventually correct itself in the direction of the real value of the company. The fundamental approach is mostly associated with long-term investment as it seeks to profit from the previous knowledge of how valuable the company is. Through some fundamental indicators, it is also possible to get a good sense of how much will the company grow in the future. Warren Buffet, Philip Fisher and Benjamin Graham are high-grade examples of how powerful this examination can be [11, 16, 20].

2.4.2 Technical Analysis

Technicians believe there is no need to analyze companies fundamentals. They assume all the necessary information is represented on price, volume and open interest [3]. They study the market action through charts in order to forecast future price trends. Following a sort of techniques, technicians try to detect patterns and trends in the financial markets, which will help them to make a statistical decision about which action to take.

There are three assumptions on which the technical approach is based [3]:

1. **Market action discounts everything**

   The markets' price reflects all the factors that can change it, whether it is the fundamentals of a company, the economic sphere, or political and psychological reasons. It claims that the price action should reveal variations in supply and demand, which is represented by a price-volume relation. When demand surpasses supply, the price should rise. Contrarily, if supply exceeds demand, it should fall. The technicians also infer that if prices are rising, no matter the reason, demand must exceed supply and the fundamentals must be bullish. If prices fall, the fundamentals must be bearish.

2. **Price moves in Trends**

   The primary purpose of charting the price of a market is to identify the early stages of a tend development. Most of the methods are trend-following in nature, meaning that their purpose is to recognize and follow existing trends. Therefore, it is essential to believe that the market moves indeed in trends and that those trends tend to persist.

3. **History repeats itself**

   Technicians affirm that it is possible to understand the future by studying the past. They base their premise on human psychology, which tends not to change — the presupposition that humans will approach similar situations, with a similar attitude. Chart patterns have been identified and categorized through history, to reveal the bullish and bearish psychology of the market.

Knowing this, technicians focus on how the prices are moving, rather than looking for the causes. They believe that combining prices and technical indicators, framed in a chart as a function of time, will help them detecting patterns. Analysts also detect patterns in the charts, that are most likely to stick with a trend, informing the future direction of the price. In order to do that, they draw the support and resistance lines. The ”support line” attaches the lower values of the peaks and throughs, while the ”resistance line” joins the higher values. These two lines define the region where prices will assumably stay. The prediction happens when one of the peaks and throughs crosses one of the respective lines.

Those two concepts of chart patterns and technical indicators are the cornerstones for price forecasting in Technical Analysis. It is also possible to solely focus on one of the above. Charts do not trigger markets to move up or down, but solely display the bullish or bearish psychology of the marketplace.

Investors have available loads of indicators and patterns they can use for this approach. Due to real-time data that is easy to gather, technical analysis is perfect for short-time forecasting. However, it is based mostly on heuristics and not in a mathematical foundation.

Although there is some apprehension regarding technical analysis, technicians already proved how lucrative this approach can be [3, 8, 18, 21–23].

2.4.3 Technical Indicators (TIs)

Technical Indicators (TIs) are mathematical formulas applied to the asset's prices, volume and OI, which plot a new series of data points on the chart. This new information gives a statistical context about market development. If compared to the price and volume of the asset, it is possible to extract patterns with relevant information. Their primary goal is to help predict future prices or following trends. None of these indicators shows a perfect formula of success, and most of them require subjective interpretation. Although accurate in time, some TIs can deceive the investor as it gives delayed information about the asset's price. Therefore, most of the modern strategies combine a set of TIs in order to cross-check each other predictions.

Numerous indicators have emerged over time. TIs can be categorized in two kinds, depending on the state of the trend they envision:

- **Leading:** Lead the price movement by using a short period to predict price.
- **Lagging:** Determine a trend by using a signal after the start of a trend.

It is also possible to classify them in four different types: [3]

- **Trend Following:** measures the direction and intensity of a trend, using some averaging to set a baseline. It removes noise on data to ease the task of identifying the current behavior of the trend which corresponds to the investors' expectation on the market.

- **Momentum Oscillators:** identifies how investors are reacting to prices, and measures the directional speed of price movements. It helps to detect abrupt changes in those movements. It is also possible to detect if a trend is losing strength.

- **Volatility Indicators:** measures the degree of price movements, independent of the direction. Allows investors to detect points where the market may change direction.

- **Volume Indicators:** measures the robustness of the trend or confirms the direction. It helps associate wide variations in price with rising trading volume.

The stated types of indicators are meant to expand the level of understanding that an investor has from just looking at plain information, like price, volume and open interest. Each category tackles a concern at the moment of investing in the stock. As a whole, technicians believe that these formulas can give all the information necessary to predict the direction of the prices [3]. Appendix B explains in more depth some of the Technical Indicators used on this thesis.

2.5 Machine Learning Concepts

Machine Learning (ML) is a set of algorithms and statistical models that progressively boost their performance in a particular task. This performance is supposed to improve with experience [24]. ML algorithms receive large quantities of data and make it possible to produce better predictions and decisions about the possible outcome of a situation, without being explicitly programmed for it [25]. This strategy is wise when we have a complex problem which admits a lot of data and variables, without having any formula or instructions to solve it.

ML is present in everyone's lives, from recommendation systems to medical diagnosis. Companies incorporate it into their products to make better use of costumers' information. It is also possible to use it on image recognition, sentiment analysis, cybersecurity, and, of course, financial forecasting.

ML techniques are broadly classified as:

- **Supervised Learning:** Given a labeled set of input-output pairs, it learns to map from inputs x to output y. It infers a function from the label data used for training, which will be used for mapping new examples. These examples require the generalization of the data in order to predict unobserved events [25].

- **Unsupervised Learning:** Unsupervised Learning only receives output data without receiving any inputs. It tries to discover an unknown pattern in the data. These algorithms enable learning from data that is not labeled, categorized, or classified [25].

- **Reinforcement Learning:** This method uses a trial-and-error strategy based on an agent that has no knowledge about which actions to take. The agent will learn how to map states to actions, in order to maximize a numerical reward. While trying the actions, it is possible to know the direct reward associated with each action. It makes possible, in a given state, to associate each action to its immediate reward, but also all the subsequent rewards [26].

Due to the panoply of existing problems to solve, some challenges are better suited to a specific type of ML technique, while others can contemplate solutions from any of the three techniques. Multiple algorithms from these three techniques were applied to the financial markets, although there is still no ideal solution that shows the perfect formula for financial enrichment. The financial data has many uncertainties, making the prediction task challenging to do with high values of confidence. External factors like economic conditions, political events, investors' reactions, and media activity can have a meaningful impact on prices.

Financial markets analysis has been attracting academic and industrial attention, due to the challenges related to forecasting prices' behavior. The fact that achieving good results is most of the times synonym of accomplishing monetary profits [3] is also an attractive motive for those who follow this path.

It is possible to define which are the possible outputs in each algorithm. In most trading algorithms applied to financial forecasting, the outputs would be ”Buy”, ”Sell” or ”Hold”. Algorithms that specifically trade in options may use the three labels, or opt to be more detailed regarding which action to execute such as ”Buy Call Option”, ”Buy Put Option”, ”Sell Call Option”, ”Sell Put Option” or ”Hold”.

2.5.1 Training and Testing

Supervised algorithms typically divide the available historical data into two subsets: the ”training data” and the ”test data”. The training data is typically a larger percentage of the full dataset since it will be used for the algorithm to define its prior knowledge. The algorithms receive the training data and will build mathematical models fitted to this data. After the training period, the ”test data” is used to evaluate the capability of the algorithms to act on their own when applied to unseen data. The algorithm only receives the input and apply the adjusted statistical model to predict the output.

When examining the performance of the algorithm on our training and testing sets, it is common to read the terms ”in sample” and ”out of sample”. The sample indicates the data used to fit the model. Therefore ”out of sample” notes the data applied that is new to our algorithm, while ”in sample” is the data already used by the algorithm for training. Consequently, the out of sample error will be comprehensively higher, because this part of the dataset did not shape the algorithm.

The concept of ”time series” represents a sequence of data points listed in a timely order. Financial data is usually a time series because it has a time variable to be considered, for example when a particular stock has a price on day 1 and another price on day 2. These series require a set of special precautions and considerations.

If we are training an algorhtim to learn what is a cat, and our dataset has no features containing information of time, each sample would be independent in order, so each sample could be randomly assigned to the training or test data. Altough, shuffling the samples before splitting the data is a good strategy to avoid overfitting (covered in section 2.5.3), in time series the time order matters. Therefore the test data must be the last samples registered, otherwise the model, while training, could passively receive information about the test set, as the future data may reflect information about the past. For example, if our algorithm is trying to predict the direction of the market prices of the SPX, receives the data for 2008 as test, but was trained using data from 2007 and 2009, it would be overfitted to predict the 2008 market crash, as in 2009 the market would still be recovering from the crisis.

2.5.2 Evaluation Metrics

Following the prediction made on the test data, a set of metrics compares the predicted output with the correct output, scoring the algorithms' forecasting performance. The most common metrics to evaluate ML classification algorithms are precision, recall, accuracy and F-measure. In expressions 2.2 to 2.5, tp represents the True Positive predictions, fp the False Negatives, tn the True Negatives and fn the False Negatives.

$$\text{Precision} = \frac{tp}{tp + fp} \tag{2.2}$$

$$\text{Recall} = \frac{tp}{tp + fn} \tag{2.3}$$

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.4}$$

$$\text{F-measure} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.5}$$

The above formulas, however, only contemplate binary classification problems. When there is more than two labels, we have two kinds of classification problems. There are the multi-class classification problems, and multi-label classification problems. The multi-label classification can attribute more than one label to the same record, for example a movie can have multiple genres. On the other side, the multi-class classification presents more than two possible labels, however they are mutually exclusive, having only one "right answer" per record. When facing a multi-class problem with 3 possible labels ("Buy", "Sell" and "Hold"), the above formulas 2.2 and 2.3 change for the average result for each class.

Furthermore, for our case scenario, we also have to take into consideration that the number of times each of the classes appear will be uneven. If there is a label that has only one record in the sample, the result of that prediction will count as much as the predictions of all the records that had one of the another labels. Therefore, the formulas 2.2 and 2.3 should also weight our predictions regarding the number of times each label is represented in our sample. The resulting formulas will be equations 2.6 and 2.7, where L is the group of labels, l represents one label, F_l the frequency of the label in the sample, $Precision_l$ is the precision of the label, and $len(L)$ represents the total number of labels.

$$\text{Weighted-Precision} = \frac{\sum_l^L Precision_l * F_l}{len(L)} \tag{2.6}$$

$$\text{Weighted-Recall} = \frac{\sum_l^L Recall_l * F_l}{len(L)} \tag{2.7}$$

In our case scenario, the primary goal is not to be accurately predicting which direction will the price of the asset go but instead maximizing the profit with the minimum risk associated. While deciding which action to take, two correct answers can have different impacts. For example, one algorithm can buy an asset and sell it for a big profit. However, this algorithm can also choose to buy the asset but investing less money or sell it too soon, making less profit from the investment. Both actions were right, due to the valorization of the asset, but we can consider that the decision which gave us a more generous profit was indeed better. Another scenario is where our algorithm makes ten predictions in which 8 of them were right. Despite the accuracy of 80%, if the loss originated by the two wrong predictions is more than the profit made by the eight correct predictions, this algorithm should not be considered successful.

Following these examples, it makes no sense to evaluate our algorithm only by metrics like precision, recall or accuracy, without complementing them with specific metrics to calculate the accumulated profit and the risk associated to the investment made.

The most common measures of profit are the ones described in equations 2.8 and 2.9. While Rate of Return (ROR) only considers the percentage of gains over the capital invested, the Return on Investment (ROI) also discounts the price of the commissions taken in each transaction. These commissions are really important to be considered in high-frequency trading, where there is a lot of transactions happening. If an algorithm can correctly predict the direction of the price before investing in an asset, but for each transaction, the commission is higher than the difference between $Capital_f$ and $Capital_i$, it will get only losses.

$$\text{Rate of Return (ROR)} = \frac{Capital_f - Capital_i}{Capital_i} \tag{2.8}$$

$$\text{Return on Investment (ROI)} = \frac{Capital_f - Capital_i - Comissions}{Capital_i} \tag{2.9}$$

Most algorithms are trained and tested with different data and time spans. Regarding profits it is important to understand how much will our portfolio value but also the time span it took. In order to compare the stated algorithms, it is common practice to normalize the returns per day or year. Equations 2.10 and 2.11 describe how to get the normalized metrics.

$$\text{Annualized ROI (without accumulated returns)} = ROI/nrYears \tag{2.10}$$

$$\text{Annualized ROI (with accumulated returns)} = \left[(1+ROI)^{1/nrYears} - 1\right] \tag{2.11}$$

$$Profit_t = \begin{cases} \frac{close_t - close_{t-1}}{close_{t-1}} & \text{if } position_{t-1} = \text{Long} \\ \frac{close_{t-1} - close_t}{close_t} & \text{if } position_{t-1} = \text{Short} \\ 0 & \text{if } position_{t-1} = \text{Neutral} \end{cases} \tag{2.12a}$$

$$\text{Mean Daily Profit} = \frac{\text{Profit}_{t_1} + \ldots + \text{Profit}_{t_N}}{N} \tag{2.12b}$$

In the first two formulas, the difference is that the second takes into account that the returns on the second year, will be regarding the initial investment plus the returns of the first year. However this formula will only work for Long positions because it assumes that the investment value increases. However, in a Short position you already received payment and you are only trying to reduce your debt over that money, not accumulating your returns on your investment.

As mentioned before, although profit is the center aspiration to maximize, one should also try to minimize the risk of potential loss. To measure the risk of the investment, there are also a variety of metrics available. The Maximum Drawdown (MDD) (equation 2.13a) is the maximum drop in profits observed from a peak to a through of the portfolio returns. Larger Maximum Drawdowns (MDDs) imply higher volatility regarding the achieved profits. The Risk Return Ratio (RRR) (equation 2.13b) is a metric that allows to evaluate the achieved profits, expressed as a proportion of the MDD.

$$\text{Max Drawdown (MDD)} = Max_{ROR_{t_x}} - Min_{ROR_{t_y}} \tag{2.13a}$$

$$\text{Risk Return Ratio (RRR)} = \frac{ROR}{MDD} \tag{2.13b}$$

The Sharpe Ratio (equation 2.14) is the most famous measure of risk. It represents the relation of how much more profit you can get by having a more volatile position. It results from dividing the subtraction of the average ROR and the ”risk-free rate of return” R_f, by the Standard Deviation of ROR. The R_f is the theoretical ROR from an investment with zero risk. Although there is no investment with zero risk associated, the U.S. Treasury-bill, which represents a short-term government debt obligation, is often used as the R_f measure. The major obstacle of the Sharpe Ratio is assuming that the returns are normally distributed. There are other variations of the Sharpe Ratio such as the Sortino Ratio (equation 2.15) which uses the Standard Deviation of only the downside returns which drops below a minimum threshold.

$$\text{Sharpe Ratio} = \frac{ROR - R_f}{StdDev(ROR)} \tag{2.14}$$

$$\text{Sortino Ratio} = \frac{ROR - R_f}{StdDev(DownsideROR)} \tag{2.15}$$

2.5.3 Overfitting

Sometimes, analysts try to create a statistical model that corresponds too strictly to the training data, in order to present better forecasts. This model ends up having perfect results predicting ”in sample” queries, however, when inquired using the test data, it is most likely to fail. The model ends up having more features than necessary, fitting to all the data points used for training, even the ones that were considered noise or ”exceptions to the rule”. These kinds of model are considered overfitted (Figure 2.7). An overfitted model is too dependent on the training data, and it will eventually display a higher error rate on ”out of sample” data.

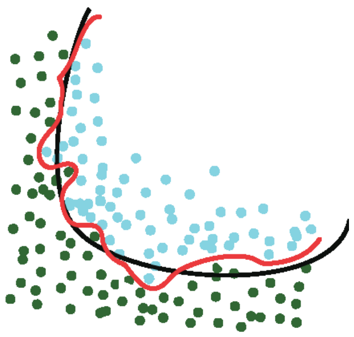


Figure 2.7: Example of overfitted model (red line) against not overfitted model (black line)

Figure 2.8 shows the error rate for an overfitted model. The blue line represents the in-sample error rate, and the yellow line represents the out-of-sample error rate. The dashed line outlines the parametrization where the model starts to be overfitted. It is possible to observe that while the error rate for already seen samples continues to decrease, the model is so fitted to the known data, that loses the generalization performance, rising the error rate for unseen data.

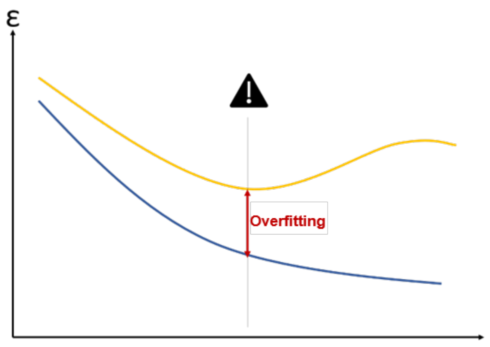


Figure 2.8: Error rate for Overfitted model

2.5.4 Cross Validation (CV)

Most ML algorithms have hyperparameters, which are parameters defined before the learning process starts. These parameters are not tuned by the training process as they specify the algorithms' characteristics. After knowing which metric will be used to evaluate the model, one may generate a score function that intends to be maximized using different configurations of the model. Using the test data for tuning the hyperparameters will be overfitting our model to the supposed "unseen" data, and losing the ability to see the generalization performance of the algorithm on the test set. Therefore we should generate a new test data inside the training data.

Cross-validation (CV) is a model validation technique that examines how a specific algorithm will perform on an out-of-sample dataset. It consists of splitting the training data into two subsets, one subset that is going to be used to train the model, and the other one to validate it and tweak the hyperparameters. Most CV methods generate different combinations of splits and combine the validation results to get a better estimation of the model prediction.

The k-fold CV, presented on Figure 2.9, splits the training data into k smaller subsets. For each iteration, one of the k folds is used as a validation set and the remaining $k-1$ folds are used for training.

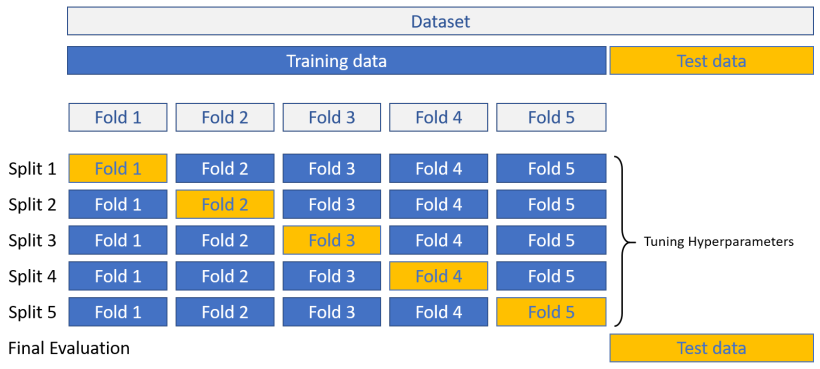


Figure 2.9: k-fold CV where $k = 5$

In a CV method like k-folds, one assumes that each sample is independent in order, however, in a time series the samples are dependent by time. As stated in section 2.5.1, the test set should only be used by models trained with past data. For this type of data, there is a special type of CV, which varies from the k-fold approach. This approach, illustrated in Figure 2.10 gets the first k folds as training set and the $k+1$ fold is used as validation set. In the next iteration, the previous training and validation sets become the training set, and the next fold becomes the validations set. Every fold further in time of the validation set is not considered in that specific iteration.

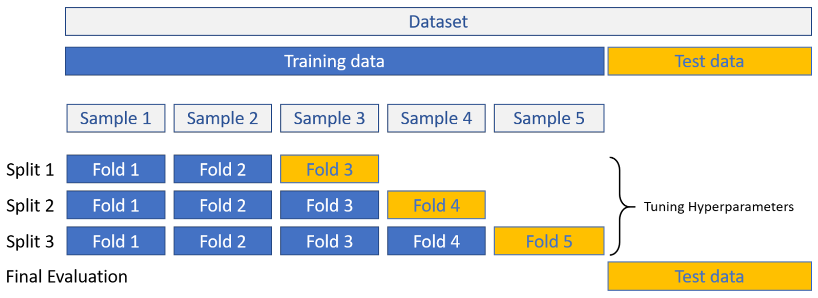


Figure 2.10: Time Series CV

2.6 Single Prediction Algorithms

2.6.1 Decision Trees (DTs)

The Decision Tree (DT) is an algorithm mostly used for classification and regression. This algorithm maps the features observed to conclude their predictive value. Its final model creates a classification in form of a tree, such as the one described in Figure 2.11.

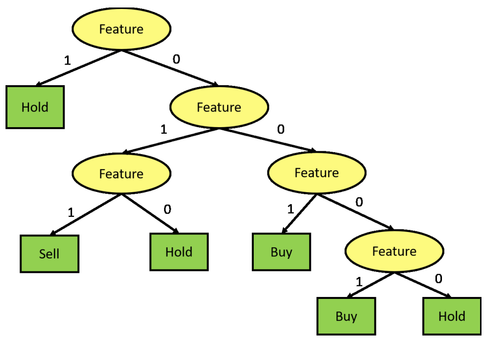


Figure 2.11: Example of a Decision Tree classifier

The inner nodes represent features or variables, which in the Figure have binary values, but can have multiple possible values to be considered. Each arch corresponds to a possible value for this feature and leads to the next feature, starting to build a path to the final prediction. The leaf nodes provide the prediction for the algorithm if the features respect the path taken.

It is possible to have two branches of tree-based methods: Classification Trees and Regression Trees [27]. The Classification Trees are produced for a finite number of unordered classes, while Regression Trees works with continual or ordered discrete values.

A set of statistical calculations must be made to construct the final Decision Tree, as described in formulas 2.16 [28]. For Classification Trees, the $\mathrm{InformationGain}$ measures the entropy reduction of the training dataset T by learning another variable's state. It means that it measures how much will a specific feature V help to separate possible predictions accordingly with its values. This gain can be calculated by equation 2.16a.

$$\mathrm{InformationGain}(T, V) = E(T) - I(T, V) \tag{2.16a}$$

Entropy represents the amount of uncertainty in a dataset. $E(T)$ is this entropy for the training data T. In equation 2.16b, C is the number of possible outcomes and $p(c)$ represents the proportion between the number of elements in class c and the number of elements in the total dataset T .

$$E(T) = -\sum_{c \in C} p(c) \log_2 (p(c)) \tag{2.16b}$$

$$I(T, V) = \sum_{i}^{values(V)} \frac{|T_i|}{T} \times E(T_i) \tag{2.16c}$$

Function I is the average amount of information required to identify the class of V in the training data T. In equation 2.16c, T_i is the subset of T where V has a specific value i.

After getting the $\mathrm{InformationGain}$ for each feature, the one with the greatest entropy reduction, displayed as the highest value, will be the next to add to the tree. Another metric which collaborates in selecting these features is specified by equations 2.16d and 2.16e.

$$\mathrm{GainRatio}(T, V) = \frac{\mathrm{InformationGain}(T, V)}{\mathrm{IntrinsicInformation}(T, V)} \tag{2.16d}$$

$$\mathrm{IntrinsicInformation}(T, V) = -\sum_{i}^{values(V)} \frac{|T_i|}{T} \times \log_2 \left(\frac{|T_i|}{T} \right) \tag{2.16e}$$

$\mathrm{GainRatio}$ is the ratio between the $\mathrm{InformatioGain}$ and the $\mathrm{IntrisicInformation}$, that represents the percentage of information after the split. The $\mathrm{IntrisincInformation}$ estimates the quantity of potential information generated by dividing the training set T into C parts.

For Regression Trees, instead of Entropy it is used the Mean Squared Error (MSE) as a criterion to determine the Information Gain of each feature. For equations 2.17, n represents the number of training samples, Y_i is the true target value, while $\hat{Y}_i$ represents the predicted target value.

$$\mathrm{MSE}(T) = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{2.17a}$$

$$\hat{Y}_i = \frac{1}{n}\sum_{i=1}^{n} Y_i \tag{2.17b}$$

Algorithms such as C4.5 [29], ID3, ID5 [30] and Classification And Regression Trees (CART) [31] combined both types of decision trees to achieve better performance.

2.7 Ensemble Learning

Ensemble Learning starts from the idea that we do not need to rely on only one learner to predict a specific outcome. Ensemble methods train multiple learners to work on the same problem and try to combine their outputs [32]. These methods use the combination of learners to achieve better predictive performance than could be obtained from any of the single learning algorithms alone [33]. These single or constituent learners are often called base learners [32].

Consider we have four learners which receive the same input and try to predict the same kind of output. Each one of them has a slightly different predictive model, whether it is because it is a different algorithm, or because it received a different subset of training data. Despite the wrong prediction of one of the learners, we still have the prediction of 3 more to help us recover and return a correct prediction from the ensemble. Therefore, this approach allows us to minimize possible errors.

Ensemble methods also provide less overfitting in the overall model. Even if one of the ensemble's constituents is overfitted, as the final result will not depend entirely on the overfitted algorithm, the overall ensemble can be considered not overfitted.

There are three branches which originated the field of Ensemble Learning: combining classifiers, ensembles of weak learners, and MoE [32]. ”Combining classifiers” attempts to produce powerful combining rules, from strong classifiers, to get a even stronger combined classifier. ”Ensembles of weak learners” work on weak learners and seeks to create powerful algorithms to boost their overall performance (e.g. Bagging, Boosting etc.). Finally, a ”MoE” employs a divide-and-conquer strategy, trying to learn a mixture of parametric models simultaneously, and using combining rules to get a final solution.

Bagging and Boosting, illustrated in Figure 2.12, are two meta-algorithms designed from the concept of ”ensemble of weak learners”. The first one generates a defined number of training subsets, each one sampled from the original training set uniformly and with replacement. This means each subset will have a fraction of the training set, but the sum of these subsets will contain duplicates of the data. Each training subset will be given to a base learner which will be trained in parallel with the remaining ones and the combination of outputs will be averaged or voted. In Boosting, on the other hand, each model will be trained on the data that was generated taking into account the previous classifiers' successes. While in Bagging, any element has the same probability of appearing in a training subset, in Boosting the different samples will be weighted, giving to misclassified samples a higher probability of appearing in the next training subset. Each base learner output will also be weighted considering the number of errors in its predictions.

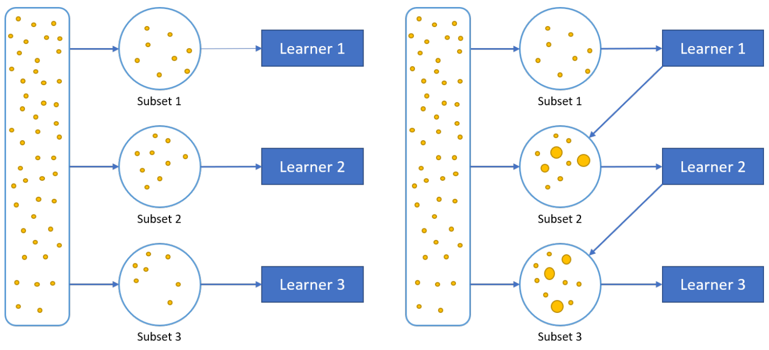


Figure 2.12: Bagging (on the left) and Boosting (on the right)

Multiple ensemble strategies have been developing over the years, each of them showing improvements compared to their constituents algorithms. The concept of an ensemble is so broad that can cover multiple branches. In the next subsections, some approaches will be explained in more detail.

2.7.1 Random Forests (RaF)

Random Forests (RaFs) [34], illustrated in Figure 2.13, is an "ensemble of week learners" which combines a set of DTs and outputs the mode (classification), or the mean (regression), of the outputs generated by the DTs [35]. It uses the Bagging technique which decreases the variance of the algorithm. [32]

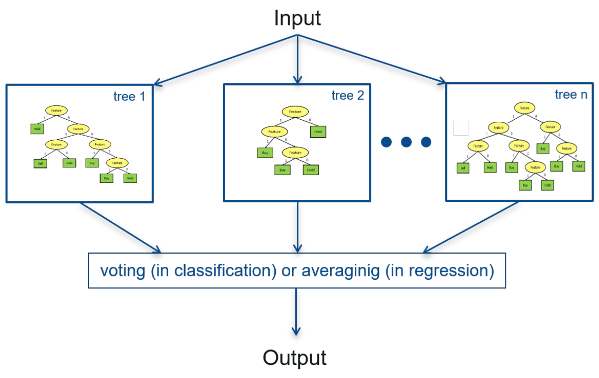


Figure 2.13: Example of RaF architecture

The RaF's original concept is based on the idea that constructing a small DT with fewer features has low computational weight. Knowing this, the algorithms run multiple DTs in parallel, taking advantage of their low computational demand. It generates each DT using a random sub-sample of the training data with a random subset of features being considered in each tree. In the end, it creates a strong learner based on the prediction of each DT.

RaFs allow a better prediction by avoiding the tendency of overfitting the training data as much as the DTs [36,37]. Algorithm 2.1 expresses the pseudocode of RaF.

Algorithm 2.1: Random Forests (Pseudocode)

```
Precondition: A training set $S := (x_1, y_1), . . . ,(x_n, y_n)$, features $F$, and number of trees in forest $B$.
 1: function RandomForest($S$, $F$)
 2:     $H \leftarrow \emptyset$
 3:     for $i \in 1, ..., B$ do
 4:         $S^{(i)} \leftarrow$ A bootstrap sample from $S$
 5:         $h_i \leftarrow$ RandomizedTreeLearn($S^{(i)}$, $F$)
 6:         $H \leftarrow H \cup \{h_i\}$
 7:     end for
 8:     return $H$
 9: end function
10: function RandomizedTreeLearn($S$ , $F$)
11:     At each node:
12:         $f \leftarrow$ very small subset of $F$
13:         Split on best feature in $f$
14:     return The learned tree
15: end function
```

2.7.2 Mixture of Experts (MoE)

The concept of Mixture of Experts (MoE) proposes to use the multiple learners at its disposal in a ”divide-and-conquer” strategy. It breaks up a complex task into various smaller and simpler subtasks. Each base learner is called an expert and is trained for a different subtask. A component called ”Gating” generally manages the connection between experts. Figure 2.14 displays the structure of a typical MoE.

MoE differentiates from the majority of ensemble methods since in other methods each base learner is trained for the same problem. A MoE, while distributing the problem in smaller problems, benefits the diversity of learners, preventing them from being too much correlated.

A good application for this method is in the field of Visual Recognition trying to classify a large number of objects. While the general problem would be the classification of multiple unrelated objects, it is possible to achieve better results by dividing the objects into groups with similar visual complexities, such as humans, animals, geological formations, plants or artifacts, and specializing each base learner in one group [38]. It is also possible to have a Hierarchical Mixture of Experts (HMoE) [39] where we

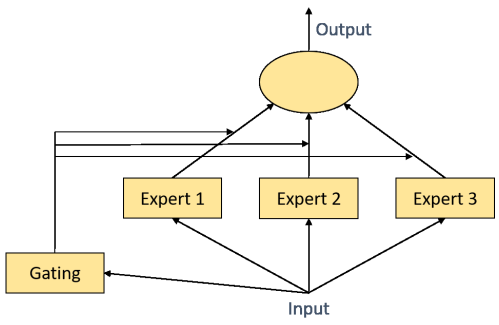


Figure 2.14: Example of MoE architecture (based on similar figure from Jacobs and Zhou)

have multiple layers of MoEs, each subgroup containing a more detailed subset of the data. For example, dividing the Animal subgroup into Mammals, Birds, Reptiles, Fishes, Amphibians, and Invertebrates.

The primary concern is finding the natural division of the task, and determine the final solution from the set of subsolutions. This distribution allows reducing the processing time due to the reduced data attributed to each expert. Researchers believe that, in order to achieve good performance, it is essential to turn the experts local, focused in only one subtask. A possible solution is to assign each expert to a distribution selected by the gating function instead of the original training data [32].

In a collection of N experts, given an input $\boldsymbol{x}$ and a set of parameters Ψ. The final output of a MoE is a weighted sum of all the local output E_i generated by each ith expert as described in function 2.18a. The gating function G_i returns the weight of the ith expert.

$$H(\boldsymbol{x}, \Psi) = \sum_{i=1}^{N} G_i(\boldsymbol{x}) E_i(\boldsymbol{x}) \tag{2.18a}$$

In order to accept the sparsity of experts important for a specific decision, it is possible to save significant computational time by not computing $E_i(x)$ whenever $G_i(x) = 0$ [40].

In the same level of generalization, it is possible to extend this function a little bit more [32], as shown in equation 2.18b. While θi is the parameter of the ith expert, α is the parameter of the gating function. Also, considering binary classification in which a possible output y is a discrete variable with possible values 0 and 1.

$$H(y|\boldsymbol{x}, \Psi) = \sum_{i=1}^{N} G_i(\boldsymbol{x}, \boldsymbol{\alpha}) \cdot E_i(y|\boldsymbol{x}, \boldsymbol{\theta}_i) \tag{2.18b}$$

In a simple approach, the gating function is modeled by the $Softmax$ function, as described in equations 2.19a and 2.19b

$$G_i(\boldsymbol{x}, \boldsymbol{\alpha}) = Softmax(\boldsymbol{x}, \boldsymbol{\alpha}) \tag{2.19a}$$

$$Softmax(\boldsymbol{x}, \boldsymbol{\alpha}) = \frac{\exp\left(\boldsymbol{v}_i^\top \boldsymbol{x}\right)}{\sum_{j=1}^{N} \exp\left(\boldsymbol{v}_j^\top \boldsymbol{x}\right)} \tag{2.19b}$$

The weight vector of the ith expert is represented as v_i, and α contains all the elements in every v.

There is a difference in the application of G_i if applied to the training or the test level. During the test stage, G_i defines how much each expert contributes to the overall forecast. However, while in the training step, this function declares the probability of an instance x appearing in the training set of each expert. The training period of a MoE not only trains each expert based on the distribution specified by G_i, but also seeks the optimal gating function for the overall output.

2.8 Related Work

In this section we summarizes the literature, encapsulating it in Single Algorithms, Ensemble Learning, MoE architectures, and Derivatives. Table 2.2 encapsulate these approaches and techniques.

2.8.1 Works on Single Algorithms

Rosillo et all. [5] used VIX with Moving Average Convergence/Divergence (MACD) and Relative Strength Index (RSI), indicators, as inputs to forecast the SPX. This study used a Support Vector Machine (SVM), which produced as output the direction of the price for the following week. The VIX seemed to improve the algorithm performance on bearish phases, but not as much the bullish ones.

Pinto et al. [4] introduced indicators based on VIX and other TIs, for their Multi-Objective Genetic Algorithm (GA). Their solution allowed them to avoid multiple falls in the stock market, and achieve more than 10% of annualized return in a period which included the 2008's crash.

Maragoudakis and Serpanos [6] exploited many technics, between them, a Markov Blanket RaF to forecast the direction of stock markets. Combined with multiple Indicators, this research used Textual Resources to apply a particular Data Mining technique to alleviate the issue of high dimensionality in volatile and complex domains. They were able to exploit returns of 48% in 7 weeks.

Silva et al. [19] used a Multi-Objective Evolutionary Algorithm (MOEA) with two objectives (return and risk control), using fundamental and TIs. They computed the Pareto Front and looked for solutions around it with TIs. They supported that the use of more indicators conducts to better selection of companies. The best chromosome in this paper achieved Return on Equity (ROE) of 50,24%.

Kumar and Thenmozhi [23] examined how predictable would be the direction of the S&P CNX NIFTY Market Index. This study concluded that the SVM and the RaF outperformed the remaining models, Neural Network (NN), Logit Model and Discriminant Analysis. The SVM performed slightly better than the RaF due to the ability to minimize the generalization error.

2.8.2 Works on Ensemble Learning

Ballings et al. [9] made a comparison between base classifier models (Artificial NN, logistic regression, SVM and K-Nearest Neighbors (KNN)) and ensemble methods (RaF, AdaBoost and kernel factory), for prediction of stock price movements. Ensemble methods proved themselves better than individual learners, having RaF as the best performer.

Research using MoE has been used for applications such as risk estimation [15], financial forecasting [10] [41] [42] and direction variation prediction [43].

Versace et al. [43] developed a mixture of NN algorithms to predict in the exchange-traded fund DIA. They also used a GA to choose the best mixture of NNs possible, the window size of the input-time series, and to determine the set of features. Their MoE consists of Recurrent Back-Propagation Networks, and Radial Basis function networks. It was possible to end up with an accuracy of 73,4% while determining the up/down predictions.

In a research made to trend forecasting of 37 companies of the Tehran Stock Exchange, Ebrahimpour et al. [44] offered a solution using a mixture of three NNs and an Adaptive-Network-Based Fuzzy Inference System. This approach outperformed the B&H and achieved a recognition rate of 86.35%.

Booth et al. [8] developed a performance weighted solution based on RaFs which only invested in specific seasons. Some of these events would be turn-of-the-month, exchange holiday and weekend effect. They stated that although same papers based on reinforcement learning [21], evolutionary bootstrapping [45] and Principal Component Analysis (PCA) [46] affirmed to beat benchmarks, they also proved to be overfitted showing large drawdowns in profits as well as unnecessary switching behavior. The authors developed an approach based on three layers: expert generalization, expert weighting, and risk management. In the first layer, they generated repeatedly RaFs to make predictions on the magnitude of stock prices fluctuations. Secondly, based on each current learner performance, the experts generated an overall output. Finally, the last layer analyzed the decisions made and eradicates the weak signals and liquidates positions which are challenging to predict.

Krauss et al. [7] used Deep NNs, Gradient-Boosted-Tree, and RaF, to build multiple ensemble algorithms. They developed three ensemble approaches: equal-weighted, performance-based, and rank-based. After this, they developed a statistical arbitrage strategy, using the previously evaluated algorithms. The stocks were ranked. The top k stocks should be used for Long-selling, and the last k stocks should be used for Short-selling. In the end, tests on the SPX showed higher results with k=10 for the equal-weighted ensemble. This approach uses one of each algorithm to achieve 0.45% of daily returns. If considering transaction fees, the first solution achieved out-of-sample returns of 0.25% per day and annualized returns of 0.73%. This study also confirms the RaF as the best performer.

2.8.3 Works on Mixture of Experts Architecture

Since Jacobs and Jordan [39, 47] published the MoE, much research has been emerging. There were proposals for new architectures based on SVMs [48], Gaussian Processes [49–51], and NNs [40, 52]. Eigen et al. [52] proposed the concept of multiple MoEs, each of them with a specific gating network as a component of a deep model.

Other investigation work converged to changing the expert's configurations. Rasmussen and Ghahramani [53] proposed an infinite number of experts, while Aljundi et al. [54] proposed to add experts sequentially. Jordana, Jacobs [55], and further Yao et al. [56] introduced a hierarchical structure for MoE. Waterhouse and Robinson [57] explained how to expand the tree structure of HMoE continuously. Waterhouse et al. [58] exploited Bayesian frameworks to infer the parameter of a MoE, while Bishop and Svenskn [59] did the same for HMoEs.

The most mediatic application was based on machine translation, with Garmash and Monz [60] using the gating network to pre-train an ensemble model, and Shazeers et al. [40] performing sparse and noisy gatting. Shazeers et al.'s [40] convolution application also granted different gating decisions at each specific position of the text.

2.8.4 Works on Derivatives

Liang et al. [61] introduced a non-parametric method of forecasting the Option prices in the Hong Kong Option market. This research employed NNs and Support Vector Regressions (SVRs) to decrease the forecasting error of the parametric methods. They also improved and modified parametric metric methods such as binomial, finite difference and Monte Carlo in order to forecast the Option price. The NN and the SVM displayed higher forecast accuracy, having the SVR outperformed the NN.

Chen et al. [62] argue that there are some disadvantages regarding the use of GAs and NNs. Both types of algorithms struggle to consistently fit the dynamic financial environment, decreasing their performance on out-of-sample testing. While GAs only use historically learned rules to forecast prices and trends, NNs patterns can be hard to understand. Chen et al. [62] explore an Extended Classifier System (XCS), a variation form Learning Classifier System (LCS), which uses sentiment indicators such as VIX, Put-Call Ratio (PCR) and Traders Index. The algorithm outputs if it should take a Long or Short position on the market. This approach allowed the authors to overcome the B&H, Mean Reversion and Trend-Following strategies.

Table 2.2: Overview over literature

Paper	Market Tested	Period of Simulation	Ensemble Method	Algorithms Used	Best Performance	Observations
[8]	DAX	2000 - 2012	Perfomance Weighted	Random Forests (RAF)	Annualised Return: 0.09 Sharpe Ratio: 1.27	Outperformed Models in profitability and accuracy
[61]	Hong Kong Options market	2 years	-	Neural Networks (NN) Support Vector Regressions (SVR)	Average Absolute Error: 0.0620 with SVR and L=12	SVR outperformed NN
[5]	S&P500	2000 - 2011	-	Support Vector Machine (SVM) using VIX, MACD and RSI	-	Overcomes Buy&Hold and SVM w/ VIX
[62]	S&P500 Futures	1997 - 2004	-	Extended Classifier System (XCS) using VIX, Put/Call Ratio and Traders Index	Accuracy: 62.28% Mean Profit: 2456.602	Overcomes profits of Buy&Hold, Mean Reversion and Trend-Following
[9]	5767 publicly listed European Companies	2009 - 2014	AdaBoost	Support Vector Machine (SVM) K-Nearest Neighbour (KNN) Logistic Regression Neural Networks (NN) Random Forest (RAF) Kernel Factory	Random Forest with AUC: 0.9037	All classifieres above 0.5 RF overcomes all others
[19]	S&P500	2010-2014	-	Multi-Objective Evolutionary Algorithms (MOEA)	ROE: 50,24%	Best chromosome with 50,24% of return
[4]	NASDAQ, DAX indexes	2006-2014	-	Multi-Objective Genetic Algorithm (MOGA) using VIX and RSI	Annualised Return: 10%	Outperform Buy&Hold and Sell&Hold
[43]	AMEX ticker: DIA	2001-2003	Mixture of Experts	Genetic Algorithms (GA) Neural Networks (NN)	Accuracy: 73,4%	73.4% correct up/down predictions
[44]	37 Companies on Tehran Stock Exchange	2005-2007	Mixture of Experts	Neural Networks (NN) Adaptive Network-Based Fuzzy Inference System (ANFIS)	Recognition Rate: 86.35%	
[7]	S&P500	1992-2015	Equal-Weighted, Performance-Based, and Rank-Based ensembles	Deep Neural Networks (DNN) (SVM) Gradient-Boosted-Trees (GBT) Random Forests	Annualised Basis: 73%	Equal-weighted ensemble outperformed base learners. RAF was the best base learner
[23]	S&P CNX NIFTY Market Index	2000-2005	-	Support Vector Machines (SVM) Random Forests (RAF)	SVM Hit Ratio: 68.44% RAF Hit Ratio: 67.40%	SVM and RAF outperformed NN and others

3

Architecture

Contents

In this chapter, we start by explaining the decisions that were made based on the Related Work. We will briefly explain the initial premises regarding dataset, strategy and the requirements of this thesis. Next, the Architecture will be briefly explained, followed by a specific explanation of each layer. Finally, we will specify how the thesis was developed, coded and shared. At the end, the expected results are explained based on the architecture strengths and weaknesses.

3.1 Decisions based on literature

Reviewing the literature, it seems not to exist a consensus over the benefits of using GAs and NNs. Although both algorithms displayed great results, they generally have large drawdowns on out-of-sample testing [62]. Although they appear to have great results on stocks forecasting, they might not be the best algorithms to rely on when applying for money, specially on a derivative product since its price changes based on the reaction to the index price. On the other hand, solutions based on RaFs and SVMs appear to show significant results [6–9], with RaFs showing off as the best base learner when ensembled [9] which indicates us the benefits of using two layers of ensembles, a MoE with RaFs as base learners. In the literature review, it was also clear that ensemble methods, when applied, always overperformed the individual learners' achievements [7].

Regarding the Market Analysis, TIs are reported to be used in every research project, therefore we pretend to feed them to our predictor in order to impact results. To have a complete analysis of the market situation, all TIs' types mentioned in section 3.4.1 should be generated with different Short and Long periods. Another remark was the impact of the VIX to perform on bearish signals [4, 5], as this index gives a useful estimation of the volatility for the following 30 days.

With this knowledge, we decided to develop a Mixture of Random Forest Experts (MoRFe) to trade SPX options. The market analysis will use Technical Analysis by applying TIs and using the VIX. We chose not to use any sort of Fundamental Analysis as the necessary data would be harder to collect in a real case scenario. This thesis also intends to confirm the veracity of the technicians' belief, stated in section 2.4.2, that looking at the price and Volume is enough to predict its future direction.

3.2 Dataset, Strategy and Requirements

This work was applied to Options from the SPX [63], in which the original dataset contained a time period of 5 years. Our solution started by only trading Put Options since Call Options have low liquidity and the two Option types work very differently. In a further phase of the project, we decided to check the system performance while trading both types of Options.

As explained in chapter 2, in most situations, Options tend to expire and reach value zero. This characteristic gives the Option a negative tendency, being on a slow downtrend while it gets closer to the expiration date. Some unusual changes on the index price can trigger a sudden uptrend on the Options price. Various observations revealed that whenever this change of trend happens, the prices tend to increase fast and with high volatility. Therefore, the algorithm should take fast long positions so that it can profit from the quick and brief uptrend. As explained in section 2.1.2, short-selling is an action of tremendous risk. A market that presents high trends in the opposite direction, which can happen abruptly as in Options, asks for extreme caution. Therefore, the algorithm shall have defined rules to control loss and take profit when the risk is too high. In order to have an ”à posterior” control over the learner performance, we decided to separate the process in two phases: prediction of the trend, and trading based on these predictions.

Consequently, the strategy mentioned above demands the algorithm to collect data regarding the SPX, SPX Options and the VIX. It will simulate the portfolio management by trading existing Option contracts, however it will not exercise the Options, neither sign new contracts. The reduction of non optimized hyper-parameters is also a goal of the project.

3.3 Architecture Overview

The architecture of our portfolio manager consists on a system that collects data from the CBOE options market and simulates trading SPX options to generate the maximum profit possible with less risk associated. It is based on three layers: the Data Layer, the Logic Layer, and the Presentation Layer. Figure 3.1 illustrates an overview of the overall architecture.

The bottom layer is responsible for managing the original data and preparing it for the Logic Layer, it does it while using the parameters defined by the user in the Presentation Layer. The second layer is responsible for the predictive process where it will train our model, predict the direction of the price and simulate the investments in the market. Finally, the presentation layer contains the User Interface (UI) for the user, with a set of parameters that can be twitched, and all the relevant information displayed, such as the success metric defined in 2.5.2 and the characteristics of the collected data.

The architecture represented in figure 3.1 is a bottom-up approach, which was developed accordingly. The first step was to collect past data from reliable sources. Data relative to the SPX and VIX, have high demand in the market, therefore it was possible to acquire them for free. However, options' trustworthy data is still short in supply due to the complexity of this instrument. Therefore we had to resort to paid database services. After gathering the data, we were ready to develop our solution. First, we created the Data Layer, starting by collecting the data from the different files, and joining all in one DataFrame, ordering it in records and features in the Data Collection module. As the two sources where we gathered the options' data had different schemas (name of features, and categories' formats), we

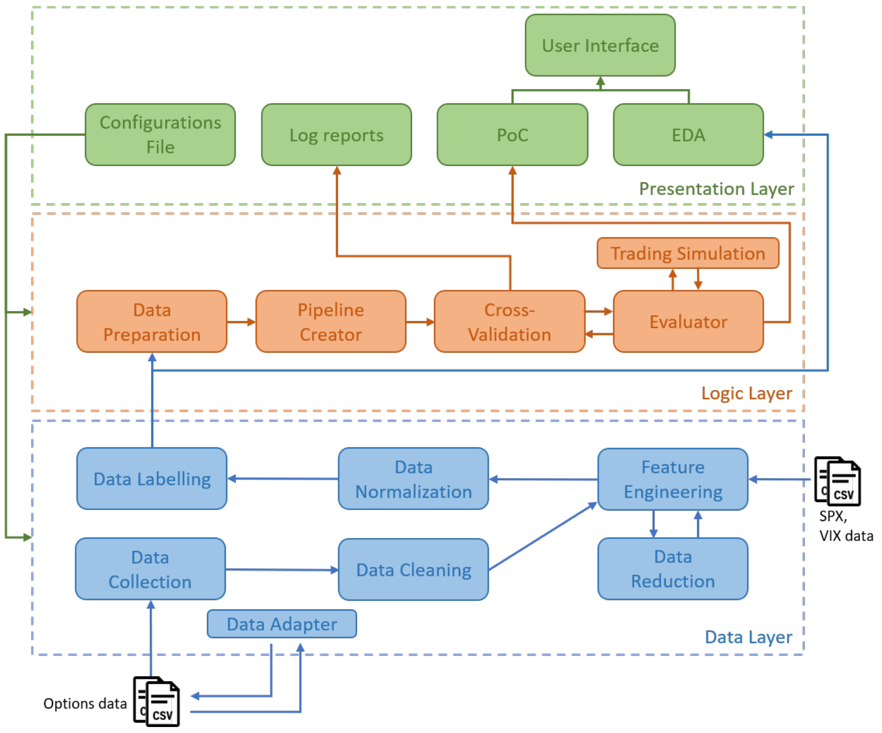


Figure 3.1: General View of the System

also developed a Data Adapter which converts a third party data into the same schema of our solution.

The Data Cleaning module receives the data and fixes problems of missing values or inaccurate data. After this, we apply the steps of Features Engineering and Data Reduction. The Feature Engineering process is responsible for generating new features with relevant information which may help in getting better predictions from the algorithm. The Data Reduction module applies a set of rules to filter the data into a chosen set of options. It uses rules related to the type of contract (put or call), liquidity of the option, and average price some months before expiration. The data Feature Engineering module is divided into two steps, one before, and another after the data reduction process. The first step generate features that need to be calculated before the Data Reduction module, such as the price and time to expiration of the option that will be used in that process, or the PCR in case we filter only for one type of contract. The second step, calculates the remaining features in a shorter dataset, avoiding unnecessary calculations for the excluded options. This second step is when the Feature Engineering module will read the SPX and VIX data to calculated Technical Indicators, Intrinsic and Extrinsic Value, if the option is ITM or OTM, as well as, the percentage change in price.

Most of the generated features are unlimited continuous values, which may be difficult to predict for our algorithm if unseen values appear. Therefore, the Data Normalization module grades most of the TIs based on a set of rules, and normalizes other features in a range between 0 and 1. Finally, the data is sent to the Data Labelling module which determines the labels that our classification algorithm will try to predict. In this case, the values will be -1 for selling, 0 for holding or staying neutral, and 1 for buying.

In the Logic Layer, the Data Preparation module will read the data from the Data Layer and arrange it as input of a ML algorithm, casting features types, enconding categorical variables, and splitting the data into train and test sets, following the specific rules for Time Series problems. After this, the Pipeline Creator will be responsible for assembling in a pipeline all the elements which will transform the received data into a set of predictions. These elements include processes like standardizing the data into a normal distribution, sampling the data to balance the number of occurrences of each label, ranking the features importance and select only the most relevant, and finally creating the ML algorithm. The ML algorithm will vary from a single RaF to a MoE with different gating functions, in order to compare the performance of the further explained classifiers.

Henceforth, the CV module splits the training data into multiple pairs for train and validation so we can determine the best hyperparameters for the Pipeline. Each set of training and validation will be sent to the Evaluator which will try to predict the defined label and send it's prediction to the Trading Simulator. The Trading Simulator uses those suggestions to test its trading performance. The Trading Simulator will send its results to the Evaluator which send the Classification and Trading performance results back to the CV module. This CV module generates a report in the form of a log, for further analysis in the Presentation Layer. After determining the best hyperparameters the same process is repeated with the Evaluator saving the model, predictions and results, and sends it to the Proof of Concept (PoC) module.

In the Presentation layer, the User has the Configuration File to determine and adjust some of the hyperparameters, rules, and ranges used in the full solution. Apart from having the Log reports mentioned before, it contains a UI based on Jupyter Notebooks that contains a set of visualizations for the Exploration Data Analysis (EDA) and PoC of our solution. This notebooks allow the user and the developer to further investigate the current results, interpret the data, and think on alternative solutions.

In the next sections each layer will be covered in a more detailed explanation.

3.4 Layer 1: Data Layer

The data layer is responsible for managing all the data required for the system. The raw Comma-separated values (CSV) files containing Option data will be primarily translated by the Data Adapter, if necessary. Then, the data will be retrieved by the Data Collection module. The Data Cleaning module will properly clean the errors in the data, which will be reduced to its meaningful records by the Data Reduction module. After the configuration of parameters in the Presentation Layer, the Feature Engineering module will generate new information about the data. The Data Normalization module will rearrange the values for a better predictive performance and send it to the Data Labelling module which will use future information to determine which classification should be predicted in the Logic Layer.

3.4.1 The Datasets

The main dataset used in this work contained the option prices for 5 years (from 2011 to 2015) of all the U.S. Equity options including stocks, indexes and Exchange Traded Funds (ETFs). This data was purchased to Delta Neutral [63], which was reported to provide data to hundreds of universitites, hedge funds, individual traders, and the Wall Street Journal [64]. The data generated over 67GBs of daily CSV files, organized by monthly and yearly folders. Table 3.1 illustrates the relevant columns of this dataset. Each entry on the dataset consists of an option end of day historical price.

Table 3.1: Option prices data from Delta Neutral

UnderlyingSymbol	UnderlyingPrice	OptionRoot	Type	Expiration	DataDate	Strike	Last	Bid	Ask	Volume	OpenInterest
SPX	1332.41	SPX110416C01415000	call	04/16/2011	04/01/2011	1415	0.1	0.05	0.15	9	2853
SPX	1332.41	SPX110416P01415000	put	04/16/2011	04/01/2011	1415	0	82.2	85.7	0	0
SPX	1332.41	SPX110416C01420000	call	04/16/2011	04/01/2011	1420	0.1	0.05	0.15	41	4904
SPX	1332.41	SPX110416P01420000	put	04/16/2011	04/01/2011	1420	0	87.2	90.7	0	0
SPX	1332.41	SPX110416C01425000	call	04/16/2011	04/01/2011	1425	0.1	0.05	0.1	61	13513
SPX	1332.41	SPX110416P01425000	put	04/16/2011	04/01/2011	1425	92.9	92.2	95.4	10	0
SPX	1332.41	SPX110416C01430000	call	04/16/2011	04/01/2011	1430	0.1	0.05	0.15	38	3089

In Table 3.1, the DataDate column is the date when the entry was recorded, the UnderlyingSymbol represents the asset covered by the option. The OptionRoot is a unique code for each option contract. It is the result of concatenating the UnderlyingSymbol, the Expiration date in the format ”YYDDMM”, ”C” or ”P” if it's a Call or a Put Option, and eight digits representing the Strike price with 3 decimal places.

Delta Neutral doesn't record Open, High, Low and Close (OHLC) for Options prices as they believe these records will give misguiding directions to investors unless the Option is heavily traded. For a non-liquid Option the first trade of the day could occur 1 hour after the market opened or 5 minutes before the market closed. Therefore the available data is the closing Bid, Ask and Last prices. The Bid is the highest price given by a buyer that still didn't meet a seller, and the Ask is the lowest price asked by a seller that still didn't meet a buyer. If these two prices meet, a transaction is made. Every time a transaction occurs, those offers exit the Bid and Ask books, and its price updates the ”Last” column as the last price the option was traded. For non-liquid Options, days can pass without any transaction.

Although the OHLC prices may not be so relevant for the Option prices, the SPX is very liquid, therefore the UnderlyingPrice value could give us too little information about the index's range of prices. This column represents the closing price of the UnderlyingSymbol. For this reason, it was extracted from the Yahoo Finance [65] website a CSV file containing the SPX daily OHLC prices as described in Table 3.2. We chose this source as it is free and accessible to everyone who may want to replicate our work. Although a free source may not be extremely accurate for very specific securities. The SPX is one of the most commonly used indexes in the world, therefore for this particular asset, we confirmed that it presents very similar registers as other payed sources such as Reuters and Wharton Research Data Services (WRDS). The CSV file contained the daily values of the OHLC prices and the Adj Close. The Adj Close represents the Adjusted Closed Price which corresponds to the closing price of a stock amended to accurately reflect the stock's value after accounting for any corporate actions. As we are only extracting prices for the SPX, there is no difference between the Close and Adjusted Close, as we don't have corporate actions to take into account.

Table 3.2: SPX data from Yahoo Finance

Date	Open	High	Low	Close	Adj Close	Volume
1950-01-03	16.660000	16.660000	16.660000	16.660000	16.660000	1260000
1950-01-04	16.850000	16.850000	16.850000	16.850000	16.850000	1890000
1950-01-05	16.930000	16.930000	16.930000	16.930000	16.930000	2550000

Instead of calculating the VIX with the already collected data, we chose to extract the index from another source as it would be more precisely calculated by experts in the field. These daily values, represented in table 3.3, are registered in a CSV file downloaded from the Federal Reserve Economic Data (FRED) website [66], which registers the VIX daily closing values. Sourcing this data is the data services of the Research Division of the Federal Reserve Bank of St. Louis, which is in the top 1% of all economics research departments all over the world [67]. This specific CSV file is also free and available to everyone at their website.

Table 3.3: VIX data from Fred

DATE	VIXCLS
1990-01-02	17.24
1990-01-03	18.19
1990-01-04	19.22

After developing the remaining architecture, an additional 2 years (2016-17) of daily Option prices were collected from the WRDS. The WRDS [68] are the data services of the Wharton University and also belong to the top 1% of all economics research departments all over the world [67]. The WRDS collects over 350TBs of data for its researchers. The collected data, represented in Table 3.4, generated a CSV file with 793 MBs. Compared to the data in Table 3.1, it doesn't have the columns corresponding to the UnderlyingPrice and Last, as the first one will be extracted from table 3.2, and the second one wasn't being used anymore at that moment.

Table 3.4: SPX data from WRDS

ticker	symbol	cp_flag	exdate	date	strike_price	best_bid	best_ask	volume	open_interest
SPX	SPX 160115C1000000	C	15/01/2016	04/01/2016	1000000	1007.4	1010.8	2000	42291
SPX	SPX 160115C1025000	C	15/01/2016	04/01/2016	1025000	982.3	985.8	0	0
SPX	SPX 160115C1050000	C	15/01/2016	04/01/2016	1050000	957.3	960.9	0	0
SPX	SPX 160115C1075000	C	15/01/2016	04/01/2016	1075000	932.3	935.9	0	0
SPX	SPX 160115C1100000	C	15/01/2016	04/01/2016	1100000	907.4	911.1	0	20
SPX	SPX 160115C1125000	C	15/01/2016	04/01/2016	1125000	882.3	886.1	0	4
SPX	SPX 160115C1150000	C	15/01/2016	04/01/2016	1150000	857.4	861.1	0	2

3.4.2 Data Adapter

The Data Adapter module is a simple module responsible for doing the schema mapping between datasets containing the same features with different names. This module will select and rename the given dataset column names in the Delta Neutral format as illustrated in table 3.1. It contains a dictionary giving the translation pairs from one format to another. In the context of this work it was used to select and rename the columns from the WRDS [68] to the Delta Neutral format, which was the one used to code all the remaining architecture. As there was no need for automation of this process, the schema mapping represented in Figure 3.2 was manually introduced.

Additionally from translating the column names, the Data Adapter module also translated the values of each column to match the format of the existing ones. It turned the cp_flag values, ”C” and ”P”, into ”call” and ”put”, and deleted the space in the column that contained the option root.

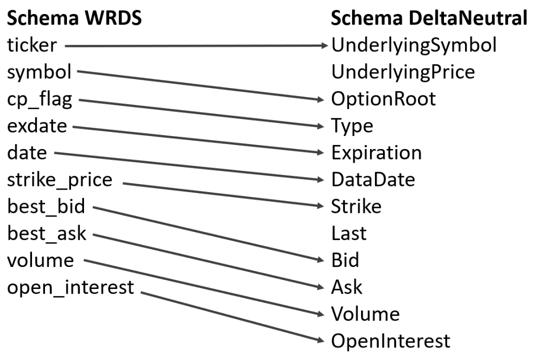


Figure 3.2: Schema Mapping from WRDS Schema to Delta Neutral Schema

3.4.3 Data Collection

The Data Collection module is responsible for reading all the Delta Neutral CSV files containing the option prices data, selecting only the rows that have the chosen Underlying Symbols, dropping all the unnecessary features, and merging all the data in one DataFrame.

As reading all the data into memory would exceed the limit of Memory usage allowed in Python, a few considerations were made. Each CSV file is read by chunks of 20000 entries. The module will then add to the dataframe only the entries that have the Underlying Symbols defined in the configuration file. After this, the current 20000 entries will be rewritten for the next 20000 entries.

In case of the SPX being one of the symbols, and the Last not being considered as the Option price, the WRDS data will be merged to the existing one.

3.4.4 Data Cleaning

The Data Cleaning module is responsible for detecting and fixing possible defects in records. The data may contain incomplete, inaccurate or incorrect values which have to be modified, replaced or deleted. In the financial sector, the accuracy and consistency of the extracted data is key to achieve true predictions regarding the profit of a portfolio. There are a variety of companies which have the sole purpose of providing the best and most accurate data possible. Nowadays, there is a lot of free data providers such as Yahoo Finance [65]. However, professional investors may want to purchase to specialized data services when searching for specific assets or fundamentals. In the context of this work, as explained in section 3.4.1, we collected data from sources which guarantee us to have reliable data. Examples of paid data services would be Delta Neutral [63] and WRDS [68]

Even though the data sources are reliable, the retrieval of data in financial securities is automatic and may have some ”dirty” data points. In the case of our dataset, the first inconsistency detected was that for the same asset in the same day, different Options should have the same UnderlyingPrice value. In some cases, the UnderlyingPrice was different by some cents. For this reason, instead of using the closing UnderlyingPrice of the Delta Neutral dataset [63], we chose to use the Close price of the SPX dataset extracted from Yahoo Finance [65].

Another problem in the Delta Neutral data was that some Option records which presented values of zero for Ask, Bid and Last, but not for Volume or OI. This corresponds to the registration of an Option lacking the information about those fields. It is possible to see the inconsistency in this data, as the previous and next values would be near the same price, having the zero value as a big drop, followed by a big rise in price. This sudden change is also not explained by the index, and similar Options do not present the same behavior. For these values we decided to apply a ”fill forward” approach, by assigning the same value as in the previous existing date.

A similar situation happened with the WRDS records. In this case, instead of having value zero on price, there was a lack of records for some days where parallel Options had registered activity. After analysing the data, we concluded that this missing records tended to happen too much times in the same Option, and the previous approach would result in more ”fabricated” dates than real ones. Therefore, in this case, we decided to drop all the Options that presented missing dates during their trading time.

3.4.5 Feature Engineering (pre-reduction)

The Feature Engineering Module is responsible to generate new features that may contain useful information for the algorithm. In our architecture, there are two Feature Engineering stages: before Data Reduction and after Data Reduction. The first one generates the features that may depend on data that could be erased by the reduction like the number of Puts and Calls to calculate the PCR. It also generates features that may contain information to be used in the reduction process.

The first two features generated are the difference between the Ask and Bid price, and the mean price between them. As discussed in section 3.4.1, the Option data doesn’t have a specific price for the Option, it only has the Ask, Bid and Last prices. The ideal concept would be using the Last price as the Option price, however, for less liquid Options the Last price may stay the same for more than a day, while the Ask and Bid prices are still moving. It can happen that the Last price would not be inside the range $[Bid, Ask]$, which means that this value can correspond to a value that we know it's not the correct price of the asset. Therefore, the middle price between the Ask and Bid values was chosen as the most reliable value to be considered as the Option price, as it would be a plausible agreement of both, seller and buyer, on a transaction made at that moment.

3.4.6 Data Reduction

The Data Reduction module is the module responsible for reducing the amount of entries in the dataset, so we extract only the meaningful parts of it. The module will start by filtering the data only by the Option types desired. As explained in section 3.2, we only wanted to consider Put Options. Theoretically, it should also be the module to only select the Options which had the desired Underlying Symbols. However, as explained in section 3.4.3, it was necessary to do it right on the collection process, to avoid memory corruption.

For the same Date and UnderlyingSymbol, there is a high amount of contracts with different strike prices and expiration dates. Some of these contracts are much more liquid than others. As the SPX Options prices should react to the SPX price, Options with low liquidity may not correctly reflect the index. Furthermore, trading in Options with low liquidity will give our positions a bigger impact on the Option market, which we cannot account for. Consequently, it was necessary to filter the available Options by their volume, in order to only get the most liquid Options.

This module presents two ways of filtering the most liquid Options. The first approach drops all the Options which present a volume below a certain threshold and gets the n Options with the higher average volume. This approach turn to be a too much aggressive as even the most liquid Options may have a day that, for a justified reason, don't have much liquidity. A part from that, this technique would not hold in a live example, as we must reduce the data before starting to train and test the model. It doesn't consider a time span and can choose Options not evenly distributed over time. Although this decision would not represent leakage of future on the predictive process, it would represent leakage on the trading simulation. The second approach, illustrated in figure 3.3, considers the last 6 months before an Option expire to give the ML algorithm. To behold on a live example, it uses the week previous to those 6 months in order to determine the most liquid Options. For a time frame of 1 month, it chooses the n Options, 6 months to expiration, with the higher average volume, as illustrated in Figure 3.3.

An additional process also reduces this choice to Options that, on the described week, had an average price over 20 dollars. As explained before, Options with low liquidity do not correctly correspond to the assets price variations. This can also happen with low price Options. Options with too low prices can also end up having too much volatility in terms of profit. Imagine if our algorithm choses to enter a Short position in a Option costing $\$0.5$, if the price rises to $\$0.6$ it doesn't look much, but the investment already had a 20% loss.

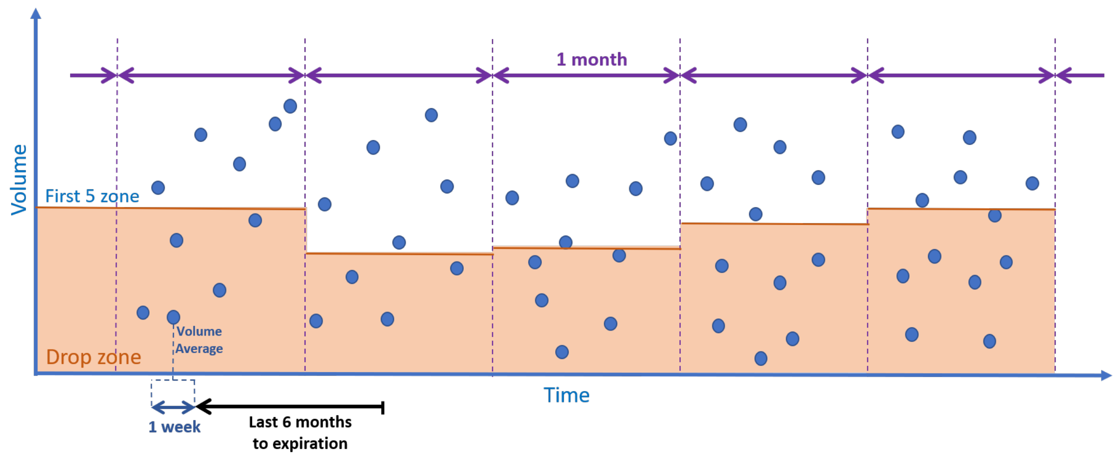


Figure 3.3: Option Selection based on fixed window with 5 Options per month

Although we could drop all the entries prior to the 6 months in this step, the calculation of most TIs require information about previous days. Only after the process of Feature Engineering, will it make sense to drop that extra data.

3.4.7 Feature Engineering (post-reduction)

The user gives the algorithm a list of periods that should represent Short and Long periods in time. After Data Reduction, these periods will be used to calculate the TIs. Our algorithm will apply TIs to three different price lines: SPX Option prices, SPX price and VIX value. Although most indicators have "default" periods when applied, the usage of such specific periods doesn't have any scientific justification. Our approach avoids using biased periods, and instead, applies a variety of periods for each TI. Therefore, the calculation for the TIs is repeated for each time period in the list.

Before calculating the TIs, the Feature Engineering Module reads the CSV files containing the VIX and SPX data. These two dataset will be merged to the main dataset, however, in order to ease the performance of our system, all time series calculations will take place before that. As we only have the closing value for the VIX we calculated the Exponential Moving Average (EMA), Momentum (MOM) and Bollinger Bands (BBANDs) for it. For the SPX, as we have the OHLC prices, we explored TIs such as Simple Moving Average (SMA), EMA, MOM, RSI, Directional Movement Index (DX), Money Flow Index (MFI), Williams %R (WILLR), Average True Range (ATR), BBANDs and On-Balance Volume (OBV). For the SPX, we also computed the logarithm of the closing prices, the logarithm of the daily difference in the closing price, and the daily percentage change of the closing price. Additionally, for each entry of the SPX price, we create a feature which gives its price x days ahead in time. The features which contain information of the future can't be used as features in our model, but they will be further used to determine the label of our supervised algorithm. After this, we merged both datasets into the Option prices by date.

After that, the program goes to each different Option and apply the SMA, EMA, MOM, RSI, BBANDs and OBV to its price. We also get the future price x days ahead, to calculate the target of the supervised models. Finally, the values for Intrinsic Value and Extrinsic Value are calculated, followed by a flag indicating if the Option is ITM or OTM.

3.4.8 Data Normalization

The Data Normalization Module is responsible for transforming the features' values into an aligned distribution between a range of values. It helps to reduce the number of new occurrences into a more compact distribution. In this application, all information containing prices has a too wide range of values, and it would be better to take those values into account in a percentage or grading format. To detect patterns in prices, it is necessary to use these operations otherwise the model would be overfitted to the specific prices.

For the Intrinsic and Extrinsic Values, as their sum would be the price of the Option, they were converted into a percentage over that price. We can take more information from "option contract that has a price 80% extrinsic" rather that "Option contract in which the Extrinsic Value is $20". The value 80% gives us information of the Extrinsic Value relative to the option price, while $\$20$ can be too vague as the option price can be $\$21$ or $\$20000$. The distribution of values will be much denser now, as the values will fit inside a range of $[0, 1]$ instead of $[0, +\infty)$.

Table 3.5: Scoring Rules based on Gorgulho et al.

	SMA, EMA
Very Low Score	Price line crosses below TI line
Low Score	TI line is decreasing
Null Score	TI line is horizontal
High Score	TI line is increasing
Very High Score	Price line crosses above TI line

	ATR
Very Low Score	TI variation below negative threshold
Low Score	TI variation is negative
Null Score	TI variation is 0
High Score	TI variation is positive
Very High Score	TI variation above positive threshold

	RSI, MFI
Very Low Score	TI line crosses above roof threshold
Low Score	TI line is decreasing
Null Score	TI line is horizontal
High Score	TI line is increasing
Very High Score	TI line crosses below floor threshold

	WILLR
Very Low Score	TI line below negative threshold
Low Score	TI line is decreasing
Null Score	TI line is horizontal
High Score	TI line is increasing
Very High Score	TI line above positive threshold

	OBV
Very Low Score	TI line is decreasing since last week
Low Score	TI line is decreasing
Null Score	TI line is horizontal
High Score	TI line is increasing
Very High Score	TI line is increasing since last week

	MOM, DX
Very Low Score	TI value is negative and the value is decreasing
Low Score	TI value is negative
Null Score	TI value is 0
High Score	TI value is positive
Very High Score	TI value is positive and the value is increasing

For the same reason, as the SPX is a bullish market and most of the time its price is completely new, also the TIs which follow this line will have the same problem. Thus, the first solution will be to calculate for each TI the percentage change for 1 day and for the number of x days we are trying to predict ahead. For this reason, we will see the daily shift in price as well as the shift which could or could not confirm the previous prediction of the algorithm. A second method of normalization was applied which consisted on grading the TIs. Gorgulho et al. [17] purposed a grading system based on crossing TIs with the price of the asset. This system would have rules determining if the grade for a specific TI should be 1 (Very High Score), 0.5 (High Score), 0 (Null Score), -0.5 (Low Score) or -1 (Very Low Score). Table 3.5, based in Gorgulho et al [17], specifies the grading conditions for each individual TI.

3.4.9 Data Labelling

The Data Labelling Module creates the target for our algorithm. As mentioned in the previous sections 2.6 and 2.7, there are two types of modeling: Classification and Regression. While Classification predicts discrete class labels, Regression predicts a continuous value. Both algorithms are suited for different problems, and they are evaluated by different metrics. Classification is evaluated by metrics like accuracy (percentage of times the prediction was right), and Regression is evaluated by metrics such as MSE (average squared difference between the true values and the predictions). The regression has the problem of trying to get a smaller error on average, even though it doesn't get a precise prediction of the price. The classification is evaluated by how many times it was completely right. We want our algorithm to predict the future price of each Option but in most cases we don't want an estimation, we want to be sure of which action to take. Imagine we have a Regressor with average error of $\$10$. If it estimates that a $\$50$ Option will fall to $\$15$ we will be confident to enter a Short position because it predicts a $\$35$ drop in price, however if every day it estimates that the same $\$50$ Option will fall only $\$5$ we wouldn't be so sure to enter this position, as the error could represent the difference between wining or losing money. To get a good performance on multiple Option trading it is fundamental to get the direction of the price right, as most of them slowly tend to zero, and we can't be sure if the regression metric would be a good or bad performance for each and every chosen Option. Although we could get a better perception of the dimension of change in price, we could lose a lot of profit on little errors near the 0% change. Therefore, we sacrifice the prediction of quantity over a better accuracy in the direction of the price. It is more valuable to know if the price is going to rise or fall, rather than how much it will.

It is common practice to convert a Regression problem in a Classification problem. In our case the most common approach is to create a label based on the direction of the Option price. If the price rises, the label will be 1, if it falls, the label will be -1. This module does it by checking the percentage change in the Options price for the next x days. The user also controls the range for which the percentage change is too little to be considered, and gives a label 0. This module also gives the user the possibility to change the signal labeled from the Option price to the index price, and vice-versa.

3.5 Layer 2: Logic Layer

The Logic Layer is responsible for all the prediction and evaluation processes. It starts by receiving the data from the Data Layer, and uses the User parameters from the Configurations File. Based on the requirements of the Pipeline, the Data Preparator works the data for the model. It sends the data for the Pipeline Creator which creates a pipeline containing our classifier. The pipeline will be used by the CV Module to optimize its parametrization. In each iteration of the CV, the forecast and the actual outcome for the tested data will be sent to the Evaluator Module, which will evaluate the algorithm performance using a set of metrics. The Trading Simulator will be used by the Evaluator in order to achieve the financial metrics. Finally, the resulting predictions and metrics will then be send to the Presentation Layer.

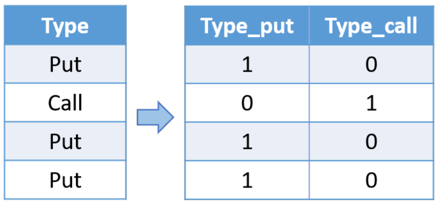
3.5.1 Data Preparation

The Data Preparator receives the data from the Data Layer. It converts the time features to numeric, and sets a MutiIndex of DataDate and OptionRoot. It replaces any infinite value by zero. This module is also responsible for handling the categorical variables and to split the data in training and test data.

3.5.1.A Categorical Variables

Categorical Variables are string variables that tend to be limited to a set of values, the so called categories. Most of ML algorithms only read numerical values, therefore, the Data Preparator encodes the categorical values into numerical ones. In our dataset, the Categorical features are the UnderlyingSymbol and Type. The method of enconding may differ based on the number of possible categories that will exist in each feature. While for Type there are only two categories (Put or Call), the UnderlyingSymbol will have as many categories as symbols chosen to be traded simultaneously. Due to the low cardinality of Type, it was converted to ”dummy variables”, which are mutually-exclusive features for each category, using the values 1 or 0 to indicate the presence of that value in each row. This method is called One Hot Encoding and it is illustrated in Figure 3.4.

If trading multiple symbols, this method would create a too large number of new columns from the UnderlyingSymbol feature. If so, the method of Label Encoding would be a more suitable option, by assigning to each category a numerical value, like illustrated in Figure 3.5. The problem with the last method is that those assigned values may be considered ordinal variables, representing a sort of rank between the different categories. For example, if we encode the categories ”No”, ”Maybe” and ”Yes” to 0,1 and

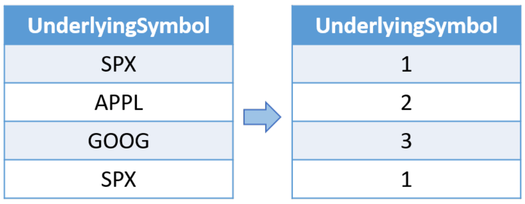


Type
Put
Call
Put
Put

Type_put	Type_call
1	0
0	1
1	0
1	0

Figure 3.4: One Hot Encoding example

2, for a feature $BossFeedback$, it makes sense to create assumptions such as $if(BossFeedback > 0)$. However, having our algorithm making assumptions about the order of the different symbols when there is no ranking encoded on them, would result in misguiding data to our model. As there is no clear winner between the two methods, our decision was to use the One Hot Encoding when using a low number of symbols and using the Label Encoding for higher variety of symbols.



UnderlyingSymbol
SPX
APPL
GOOG
SPX

UnderlyingSymbol
1
2
3
1

Figure 3.5: Label Encoding example

3.5.1.B Train Test Split

Before sending the data to the Pipeline Creator, the module splits the data into training and test sets. As mentioned in section 2.5.1, time series data has specific restrictions and requires a special type of ministration. In this case, we handle a set of options that share record dates but have different timeline ranges. To avoid data leakage, our dataset must be separated into the training data containing the earlier dates, and the test data with the most recent ones. Besides that, while defining a percentage of the dataset for the test period, it is important that records of the same date will all go to the same set, training or test data, but not separated into both. For this reason, our Preparator gets the date for the cut, and assigns all posterior dates for the test data.

3.5.2 Pipeline Creator

The Pipeline Creator generates a sequence of processing steps applied to the data, which are illustrated in figure 3.6. The constituent steps must be only executed in the training data as this should only be executed on the data that we are supposed to know the outcome. The prepared data starts by being processed in the Scaler which will scale the data properly, then goes to the Sampling Module which tries to improve the imbalance problem of our classification, followed by the Feature Selection module that reduces the number of features, and finally the MoE which will create the classifier.

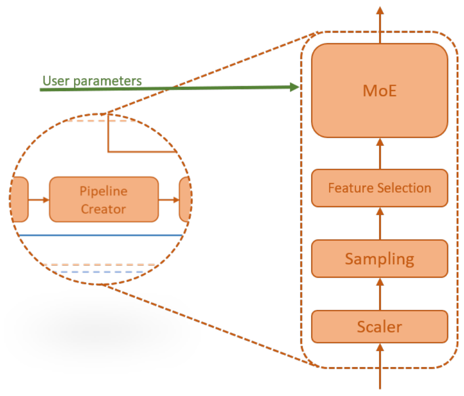


Figure 3.6: Pipeline Creator close-up

3.5.2.A Scaler

The Scaler is responsible for standardizing features by removing the mean and scaling to unit variance. ML algorithms tend to converge faster if the features are smaller or closer to a normal distribution. In section 3.4.8, we already normalized some of the features in order to squeeze the data between a defined range of values. The Standardization process will give that data a unit variance.

For the Standardization process we used the Scikit-learn's StandardScaler which transforms the data so the distribution has mean as 0, standard deviation as 1, and about 68% of the values are between -1 and 1. This is a process independent for each feature of the data. It generates the new data through equation 3.1 where μ is the mean of the data, and σ is the standard deviation.

$$StandardScaledData = \frac{Data - \mu}{\sigma} \tag{3.1}$$

Tree-based classifiers (algorithms based on DTs) do not require standardization as they are invariant to this changes. The generated rules will be the same, even though after this process, instead of generating the rule $1 < X < 10$ it will generate a rule with the standardized values of 1 and 10 on its place. Although tree-based classifiers don't need standardization, this process may still be helpful for plotting, correlations or measures of association.

3.5.2.B Sampling

After exploring the labeled data coming from the Data Layer, we found an extremely imbalanced dataset. Figure 3.7 illustrates the classes count coming from the dataset. In this case, we can see that most classes were labeled as -1 which represents a downtrend. This confirms our previous statement that most Options follow a downtrend to zero. However, as we explained, some of the few counts of 1 can generate massive losses to our trading simulation.

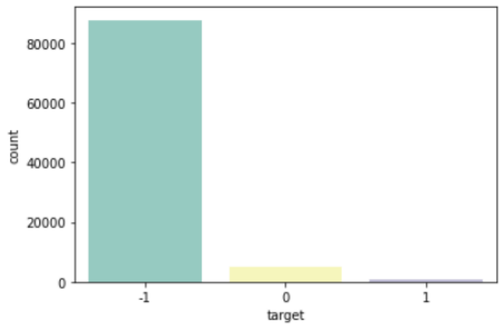


Figure 3.7: Imbalanced classification

The challenge of working with this kinds of datasets is that most ML techniques will ignore the minority classes, having poor performance while predicting them. Another problem is that, when we find these cases, the minority class tends to be the most important. Examples of this are fraud detection, market crashes, spam filtering or medical diagnosis [69]. In our case, it is also very important as otherwise we would always assume a S&H position. As mentioned in section 2.5.2 ignoring the minority classes would result in bad performance in terms of profit and risk metrics. It could also increase the out-of-sample error, with market crashes and other events that didn't occur in our 2011-2015 time window.

Our solution was based on combining two approaches. The first one is to oversample the minority class using the Synthetic Minority Oversampling Technique (SMOTE). This approach is a type of data augmentation that synthesizes new examples of the minority class, from the existing ones [70]. The second approach undersamples the majority class by using a technique called Tomek Links [69]. This technique removes pairs of opposite classes in close vicinity, called Tomek Links. For every Tomek Link the majority class element is removed undersampling the majority class and creating a more defined border between classes. Figure 3.8 illustrates how this two sampling techniques work when combined.

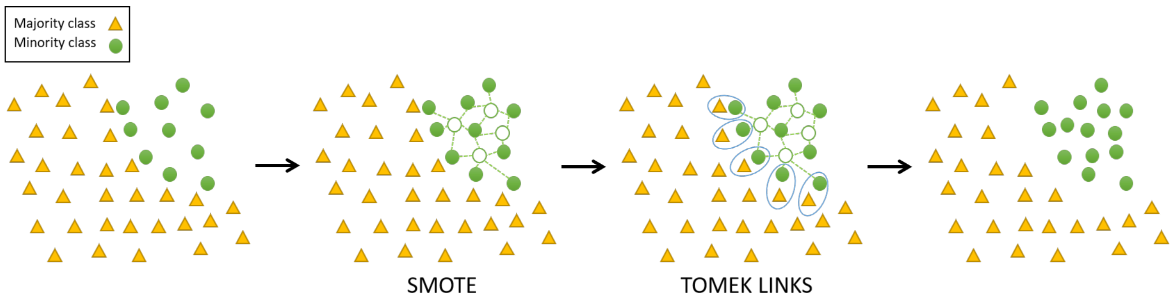


Figure 3.8: Example of SMOTE and Tomek Links techniques combined

3.5.2.C Feature Selection

After the Feature Engineering process in section 3.4.7, one can see that the number of generated features is too high. This is a problem as it increase the probability of having a system overfitting the training data, as it has too many details to accomplish a good generalization when applied to new unseen data. On that account, there are techniques that allow us to only select the more important features for our model.

To accomplish the Feature Selection we used the Scikit-learn's SelectKBest, which ranks the features by a determined scoring function and selects only the K best features to be used in the classification process. For our classification problem, we tested two scoring functions: f_classif and mutual_info_classif. The first one uses the Analysis of Variance (ANOVA) F-test [71] by comparing the linear relationship between the features and labels, and scores each feature by the square error of a model composed by that feature. On the other hand, the second method determines how much information can be obtained about one variable by knowing other variable, and scores each feature by how much information about the label we get by knowing it. Although scikit-learn also offers the chi2 scoring method, this one only works for non-negative features and classes.

3.5.2.D Principal Component Analysis (PCA)

The Principal Component Analysis (PCA) is a transformation technique used for dimensionality reduction, to remove redundancy in our dataset. This method rearranges the data by generating new features (or axis) that may see the data in a more separable perspective. Given a number of points in a dimensional space, it determines the best-fitting line by calculating the line which has minimum average squared distance from a point to that line. The next best-fitting line is chosen using the same method on directions perpendicular to the first. It repeats this process creating an orthogonal basis with uncorrelated individual dimensions called Principal Components. The resulting of determining two Principal Components on a two dimentional dataset is illustrated in Figure 3.9.

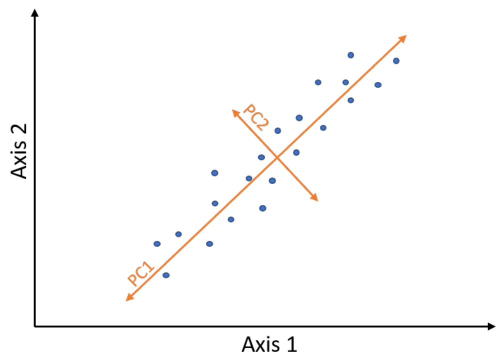


Figure 3.9: Example of PCA

3.5.2.E Single Random Forest

Before implementing the MoRFe, our system was developed using a single RaF. We used the Scikit-learn library that creates a RaF Classifier similar to the one explained in section 2.7.1. The algorithm required the optimization of hyperparameters which are important to better understand the MoRFe. As supported by the literature a simple RaF should give us already good results. Only after achieving the performance reported in the literature, would it be worth it to implement the MoE and explore possible improvements.

Before using the algorithm, it is necessary to understand its advantages and disadvantages and what impact those points can have in our results. Although we already used several pre-processing steps, this algorithm has many advantages for this type of problem. First, the RaF algorithm allows classification as well as regression. It can handle missing values, avoid overfitting, handle high dimentionality datasets, provide feature importance, mitigate outliers impact, and perform well on imbalanced datasets. Finally, one of the main advatages of using RaFs is to collect the best performing trees and extract meaningful rules from there. On the other hand, the RaF have also disadvantages which we took into account. The first one is the fact that it generally presents worst performance on regressions, which also contributed for our decision on choosing to opt for its Classification capabilities. Secondly, the developer has little control on what the model does, it is possible to tweak the hyperparameters but no much more than that. This algorithm also presents high computational and memory costs, which may slow the process and compromise the speed of predictions in a live case scenario, or the range of hyperparameter combinations to use for tuning. It can also overfit for some datasets with noisy classification tasks. In this context, noise in the price over time is also something that a human may not know how to distinguish, therefore the TIs used also help the algorithm to understand what is (or not) a noisy value. The predictions of the trees should be uncorrelated. Finally, variables with different number of levels may bias the RaF as the probability of selection of its levels is higher than other features, justifies the necessity of using the process of Feature Selection.

3.5.2.F Mixture of Experts

After testing the system with the single RaF, we developed a MoE from scratch. As the concept of MoEs is still under-explored, there are still no implementations of the algorithm in any notable ML library, but only in scientific research implementations. Although most implementations of MoEs are based on NNs architecture [40, 72], its definition doesn't require one. Our implementation was developed as an extension of the Scikit-learn library, based on a branch of a VotingClassifier.

The inside view of the MoE Module is represented in figure 3.10. The developer should specify which experts should be created and the gating function. In our implementation, the Experts Generator creates a set of N experts, which can be a base predictor or another pipeline per Expert. Each of these Experts

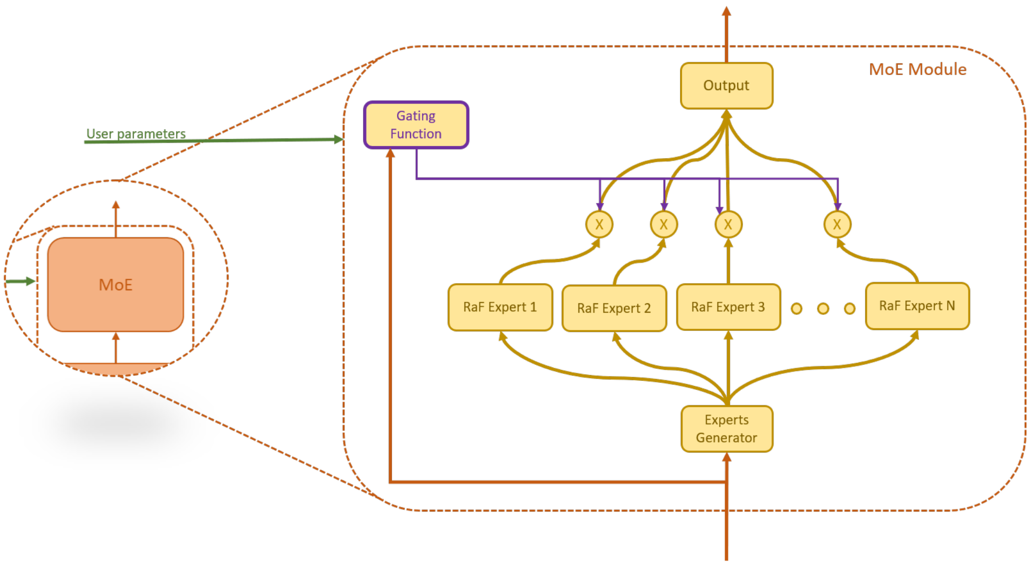


Figure 3.10: Mixture of Experts Module

will receive a part of the data from the previous module. The Gating Function defines a matrix returning, for each record, the weight of each expert for the decision. If an expert has weight zero for a specific record, it will not receive the record for training, as it doesn't belong to it's expertise. If the weight is above zero, then it receives the record data. When testing, the algorithm uses the same gating function to calculate the final prediction based on the weight of each base prediction. Finally, the Output module will return the overall prediction of the MoE.

This implementation allows to create a gating function for the model, or send a pre-created matrix with all the required weights. This can be done as the user may want to generate the weights based on intel that is lost on the pre-processing steps. Based on the VotingClassifier architecture, we also allowed the developer to give a static bias array of weights for each classifier. Imagine you are giving the MoE three DTs and one RaF. Your gating function will be blind to which kind of predictor is each expert, but you may want to give more weight to the RaF rather than an equal weight for each classifier.

We developed two gating functions. The first one, based on formula 3.2 gives an equal weight for each classifier, and doesn't differ from the VotingClassifier. The results of having the same classifier in each expert would be exactly the same as having only that classifier one single time. However, in order to have some degree of diferentiation between experts, the Random Seed of the MoE is introduced in a random number generator, which generates the random seeds for each base learner.

$$ExpertWeight = \frac{1}{nrClfs} \tag{3.2}$$

The second function sorted the test data by a continuous feature, discretized its continuous values into a specified number of bins. The number of bins is equal to the number of experts. Finally, the algorithm attributes each bin to an expert, which will receive only the data that falls into that bin. Each expert will then receive only the data that as weight higher than zero in its knowledge base, following equation 3.3. The gating function returns a matrix containing the weights that every record has for each expert.

$$ExpertWeight_i = \begin{cases} 0 & \text{if } data(i, feature) \text{ not in } Bin \\ 1 & \text{if } data(i, feature) \text{ in } Bin \end{cases} \tag{3.3}$$

In equation 3.3, $ExpertWeight_i$ represents the weight given to an Expert by a specific record, $data$ represents the data received by the gating function, i represents the index of that record, and $feature$ represents the index of the feature chosen to be the discretized into bins. As this algorithm is being used alongside techniques as Feature Selection and PCA, we chose the first feature, considered the most relevant, as the split feature.

3.5.3 Cross-Validation

The Cross-validation (CV) Module is responsible for cross-checking the system's performance and to tune hyperparameters of our algorithm. This process was explained in section 2.5.4. As we are dealing with a Time Series, our cross-validation will have to only use past data for every validation data, so leakage of future data is avoided. We had a set of multiple hyperparameters to tune, and a set of possible values for each of those hyperparameters. In order to get the best combination of parameters for our algorithm, we used a method called Grid Search. This method calculates all the possible combination of values, and, in each fold of the CV process, it scores each combination using the specified CV method. In this case, we selected the accuracy score as our score function. The best set of parameters is chosen by averaging the folds' scores, and picking the best performer. Table 3.6 lists the tested values for our pipeline. In order to speed up the process we also used the Randomized Search CV as an alternative method for the Grid Search. The Randomized Search defines a fixed number of iterations. For each iteration, it selects a random configuration of hyperparameters (without reposition) from the list of alternative. This allows to test the same range of values, having an approximate global maximum in an optimized time.

Although there was an existing Grid Search function for our pipeline, we developed the same strategy for the Trading Simulator. This allow us to test multiple combinations of the created strategies, and a range of thresholds for the orders. Table 3.7 lists the trading parameters which were optimized for the highest ROR.

Table 3.6: Values tested for each algorithm hyperparameter

Pipeline Item	Parameter	Range of Values
SMOTETomek	random_state	60,80,100
SelectKBest	score_functions	f_classig, mututal_info_classif, chi2
SelectKBest	k	from 10 to 100, every 20
PCA	n_components	from 50 to number of features, every 30
MoE	n_experts	3, 4, 5
RaF	n_estimators	from 50 to 220 every 30
RaF	criterion	gini, entropy
RaF	max_features	auto, sqrt
RaF	max_depth	None, from 10 to 110 every 11
RaF	min_samples_split	2, 5, 10
RaF	min_samples_leaf	1, 2, 4
RaF	bootstrap	True, False

Table 3.7: Values tested for each trading hyperparameter

Parameter	Range of Values
Trading Window	from 1 to the time window for prediction
Holding Method	Continuous, Individual
Holding Percentage for Long Positions	0.01, 0.1, 0.3, 0.5, 0.7
Holding Percentage for Short Positions	0.01, 0.1, 0.3, 0.5, 0.7
Trading Method	B&H, S&H, Target Predicted, Delayed Target Predicted
Stoploss Method	all the combination of stoplosses activated and not
Stoploss Percentage	None, -0.15, -0.2 , -0.5, -0.7
Stoploss Percentage for Peaks	None, -0.15, -0.2, -0.5, -0.7
Stoploss Penalty	0,1,3,5,15,30
Chandelier Long Period	13
Chandelier Short Period	22
Take Profit Method	[], ['Max Profit for Option']
Take Profit Percentage Take	0.1
Take Profit Percentage	0.5
Absolute Take Profit Percentage	2.8

3.5.4 Evaluator

The Evaluator is the module that makes the complete evaluation of our model. It receives the generated predictions and calculates the metrics established in Table 3.8. Therefore, after predicting the labels, the Evaluator is called to displays the performance of the predictions in terms of ML and Financial metrics. To do so, it also sends the predictions to the Trading Simulator to extract the financial metrics results.

During the elaboration of this thesis, commission-free brokers invaded the market. For this same reason, we decided to not consider commissions as we strongly believe that, in the short future, traders will only be commissioned by withdrawals and not for transactions. Therefore, the values of ROR and ROI will be the same.

In the end, the Evaluator module returns a table comparing the results of the Best Case Scenario, the Worst Case Scenario, the B&H strategy, the S&H strategy, and the results of the algorithm applied to the training set and to the test set. It also displays the comparison between the counts of each class

Table 3.8: Metrics used to evaluate predictions

Machine Learning	Financial Portfolio	
	Profit	Risk
Accuracy	Rate of Return	Sharpe Ratio
Precision	Return on Investment	Risk Return Ratio
Recall	Annualized ROI	Max Drawdown
F-measure	Mean Daily Profit	Sortino Ratio

and the counts of each prediction class.

3.5.5 Trading Simulator

The Trading Simulator module is the module responsible for simulating the trading activity of our algorithm based on the predictions made on the MoE module. This simulator is used by the Evaluator to access the financial results based on the profit and risk metrics defined in section 3.5.4. It receives the test samples with the predictions, and trades each Option individually. Instead of assuming an initial value, our implementation considers a percentage based approach, where the algorithm starts with 100% of the initial investment money. The algorithm will simulate an investment application, by the following the same steps each day. These steps will be explained in following subsections.

3.5.5.A Trading Strategies

First the simulator applies the trading strategy which could be a B&H, a S&H, to follow the predictions of the MoE, or to take the inverse position of those predictions. If its following the predictions, when the target is 1 the algorithm will assume a Long position, if the target is 0 a Neutral position, and if the target is -1 it will enter a Short position. It is also possible to assume the last two strategies in a delayed decision. This means that when the algorithm is taking a Long or Short position, and the target predicted is suggesting the opposite direction, the strategy will be to take a neutral position first, assuming the position of the next prediction. While only changing positions every x days, this can prevent us from taking a Short position every through of an uptrend, or a Long position every peak of a downtrend. If the next prediction was the same as the previous one, the strategy will follow what seems to be a new trend. In this case, if the prediction is a neutral position, the trading simulator will maintain the current position, as it could be a more horizontal period of the same trend.

As mentioned before, the predictions are relative to x days ahead in time. In case of $x \neq 1$, our algorithm shouldn't change positions every day, because it would trade on shorter time fluctuations than the prediction is suggesting. However, this doesn't mean that we should only consider to change the position every x days, it could happen that on the $x - 1$ day our prediction fails. Therefore, our trading simulator only apply the strategy every y days which could be equal to x, $x - 1$ or $x - 2$. This value will be defined by the user as well.

3.5.5.B Stop Losses

Although it would be a perfect scenario to only trust in our investment predictions, a successful trading strategy must have a Stop Loss. A Stop Loss is an exit strategy, that establishes a condition where we want to exit our position in case of a specific loss. The first Stop Loss order was called ”Only My Money”, as illustrated in Figure 3.11, as it stops investing in any Option that generated at least -100% in cumulative returns. The reason for this decision is assuming that our algorithm is not making a good job predicting the direction of the specific Option. It means that it never lets an Option generate more loss than the money invested in it. This Stop Loss prevents the algorithm from keep investing in a misguiding Option and losing possible profits from other Options.

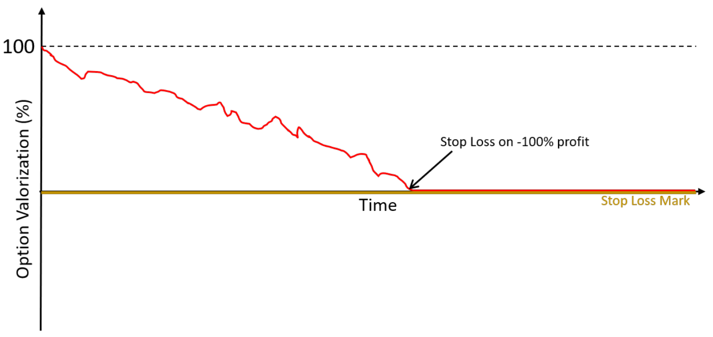


Figure 3.11: Stop Loss on -100% of cumulative returns

In Section 3.4.6, it was mentioned that we only chose Options which had an average price below a certain threshold, as low priced Options can have too much volatility in terms of profit. For the same reason, there is also a Stop Loss order called ”Per Price”, which allows the user to define a price considered to be too low to continue investing in the Option.

It is also possible to have Trailing Stop Losses. These orders are Stop Losses which will grow along the generated profits, the Option price or the UnderlyingPrice. Our simulator has 3 varieties of these. The first one is a Stop Loss based on the daily percentage change over the Option returns, called ”Daily Returns”. If it exceeds a certain threshold, the algorithm will exit the current position. Figure 3.12 illustrates this Stop Loss applied to a threshold of -20%.

Although, this approach helps the algorithm escape sudden changes in profit, it doesn’t account for continuous losses that may be under the daily threshold. This may happen very often with the Option Price. As explained before, an Option tends to have a subtle downtrend. If our algorithm decides to take a Long position in a Put Option, there is the necessity to create a Stop Loss that stops the price of the Option is continuously falling by a daily threshold lower than the one we defined. The second Trailing Stop Loss, called ”Percentage Peaks”, takes into account the percentage change from the Maximum

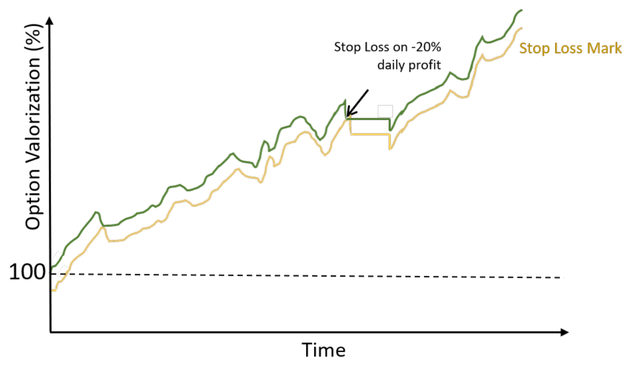


Figure 3.12: Stop Loss on -20% of daily returns

High (for Long position) or the Maximum Low (for Short positions) of the Option price. Figure 3.13 illustrates the Stop Loss signals of this approach when in a Long Position.

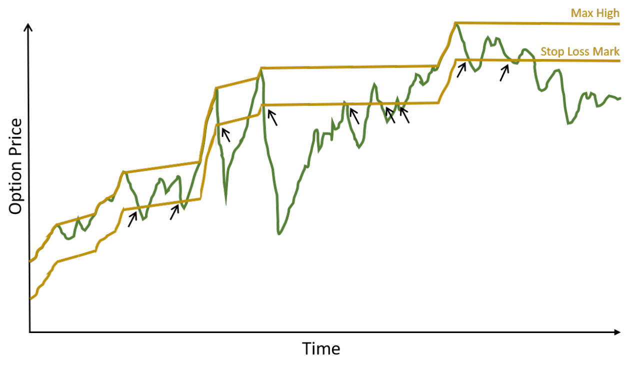


Figure 3.13: Stop Loss signals based on percentage over the Max High

The previous explained Stop Losses may have a positive impact on the overall ROR, as those orders stop a particular investment of losing too much money. However, these signals have fixed threshold which do not account for the existing Volatility. As explained in section 2.3.5, the Volatility determines the probability of a wider range of prices, therefore an higher uncertainty regarding the asset's price. Our third approach is the ”Chandelier Exit” that is a Trailing Stop Loss signal based on the volatility of the asset. This approach was applied to the SPX directly. The following formulas describe the math behind the Chandelier Exit signal for Long or Short positions. In the implemented architecture, the user may determine the number of days n and the $Multiplier$.

$$ChandelierExit(Long) = High(n) - ATR(n) * Multiplier \tag{3.4a}$$

$$ChandelierExit(Short) = Low(n) - ATR(n) * Multiplier \tag{3.4b}$$

Figure 3.14 illustrates the two Chandelier Exist signals. The Chandelier Signal for a Long position on the left, and, on the right, the Chandelier Signal for a Short position. In a Long position, the Chandelier Exit signal avoids every price below the $ChandelierExit(Long)$, while in a Short position, the signal avoids every price above the $ChandelierExit(Short)$ line.

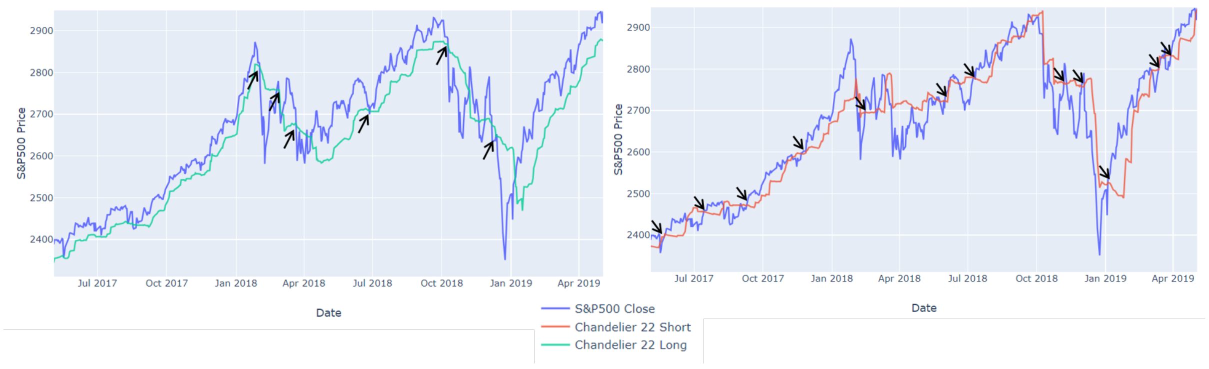


Figure 3.14: Chandelier Exit signals for Long and Short positions

The SPX and Put Option prices have an inverse correlation, therefore our system applies the Stop Loss in reverse for this case. The Chandelier Long signal triggers a stop signal for Short positions, as the Chandelier Short signal triggers a stop signal for the Long positions.

3.5.5.C Stop Loss Penalty

The Stop Loss Penalty is the number of days the trader is suspended of entering a Long or Short position for a specific Option, after a Stop Loss was activated. For the ”Only My Money” Stop Loss, the penalty is infinite, because there is no more allocated money to that Option. For the Chandelier Exit there is no penalty, the Stop Loss only intends to exit the positions over or below the specific lines. For the remaining Stop Losses, the number of days can be defined by the user.

3.5.5.D Take Profit

If we need Stop Losses to control our losses, it is also important to know when some of the profit generated, when we are winning, is enough. These actions are called Take Profit strategies. As mentioned in Section 3.2, Put Options can take a too quick bullish trend while our algorithm is in a Short position. A Take Profit strategy allow us to save a portion of our current profits when we are gaining money, so we don’t lose all our money when an abrupt rise in prices happens. It also allow us to think how much return is enough for us, while continuing investing in the money would represent too much of a risk.

Our Take Profit strategy works on thresholds over the percentage of cumulative returns. The user defines a maximum threshold where the totality of the profits are taken. For lower profits, it is established a certain percentage that is taken for every sub-threshold passed. This process is illustrated in Figure 3.15.

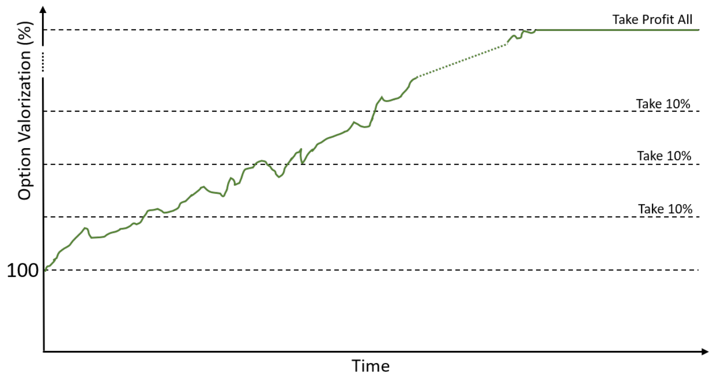


Figure 3.15: Take Profit Strategy

3.5.5.E Volume Check

Although our algorithm may want to take a determined position, if no one is interested to be on the other side of it, it would not be possible to make that transaction. Therefore, our simulator checks the Volume and Open Interest value in order to verify the transaction. In case the transaction would not be possible, the previous position in the Option will be maintained.

3.5.5.F Biased Restrictions

After an impartial decision made by the trading simulator, which took into account the predictions of the MoE, before applying this positions, the algorithm checks if the user may have some biased restrictions for those decisions. To avoid too risky predictions, our algorithm also applies these imposed restrictions. The user may have decided that Shorting is too risky and decides to only take Long positions, or vice-versa.

3.5.5.G Holdings

The last decision for the trading simulator is to decide how much should it invest in each position. Our trading simulator begins with 100% of the money in cash. For every new position it invests a percentage of its cash, equally, in each new position. This percentage follows formula 3.5, where the $InvestmentPercentage$ is set by the user with separate values for Long or Short positions.

$$InvestedAmount = \frac{Cash * InvestmentPercentage}{NewPositions} \tag{3.5}$$

For safety reason, the simulator, when entering a Short position, saves for that position the equivalent amount of the -100% Stop Loss, in order to secure the loss. Some brokers only allow traders to enter a short position if this condition is met.

The user can also define if he wants to consider different positions in the same Option ”Individual” or ”Continuous”. In the first approach, every action will be considered independent, therefore if the algorithm exits one positions and assumes another, the generated profit will be cashed out, and the invested amount for the new position will be calculated using formula 3.5. In the second case, formula 3.5 will only be used if the invested amount in that Option is zero, otherwise it will use the money generated by the previous positions. This allow the algorithm to invest more money in Options that are acting as predicted, and less money with outliers or misguiding Options.

3.6 Layer 3: Presentation Layer

The presentation layer is the direct contact with the user. Through the UI, it is possible to configure some parameters which may influence the algorithm performance. It will also provide the results for the current settings, to let the user make a better analysis of the system configurations. The information regarding all the trading signals will be displayed in a graphical chart as well as the ROI.

3.6.1 Configurations File

The Configuration File is the file for the user whom we assume not having any technical skills. This file contains all the previously referred variables with their default values to be change by the user. This is a good scenario for a Financial Analyst which may want to test our algorithm. The file is formatted in Yet Another Markup Language (YAML), which is easy and intuitive to read and edit. This file is commented in order to explain to which group of the code does each variable belong. Appendix A describes what each variable represent in Configurations file.

3.6.2 Exploration Data Analysis (EDA)

The Exploration Data Analysis (EDA) intends to help the user and the developer understand the data and analyse it through some helpful visualizations. In this module, it is possible to analyze the raw data collected as well as the data coming from the Data Layer. We developed interactable graphs using the python library Plotly so it was possible to explore the data in more detail by zooming, dragging and selecting the relevant information. The visualizations are displayed in Jupyter Notebooks along the plugin *Appmode* to present it as informative dashboards.

The first dashboards present charts of the multiple signals with their TIs. This signals can be VIX, SPX price or the Options price. Examples of this visualizations were already displayed in the previous sections 2.2, 2.3.5 and 2.4.3. Other visualizations are the Option Chain table per day and a simulation of Option Spreads. The Option Chain is a listing comparing Call and Put orders, organized by Strike price and Expiration date. The Option Spreads are strategies when you simultaneously exchange two or more Options that, combined, limit the possible ROR to a range of values. Although these strategies can contain the loss of our algorithm, they also limit the possible reward of a good prediction from our algorithm. For that reason they were not implemented in our simulation. Nevertheless, while exploring the data, this visualization may give usefull information if analysed by a experienced technician. Technicians use this kind of visualization to find patterns in data that may not be easy to perceive by only analyzing the numbers, however if seen in a graphical way can give us useful insight to improve our profits.

After the data being process by the DataLayer, the User can also go to the EDA to see the dataset features, shape and description. Another useful visualization, is the balance of the target class, illustrated in Figure 3.7, to explore imbalance classification.

The developer may also use this EDA to explore the correlations between features, and the distribution of the different target classes by a feature's values. Figure 3.16 represents the multiple charts representing the distribution of the target classes by the features values. This analysis can help us to better understand which variables would be more useful for the split process of the MoE.

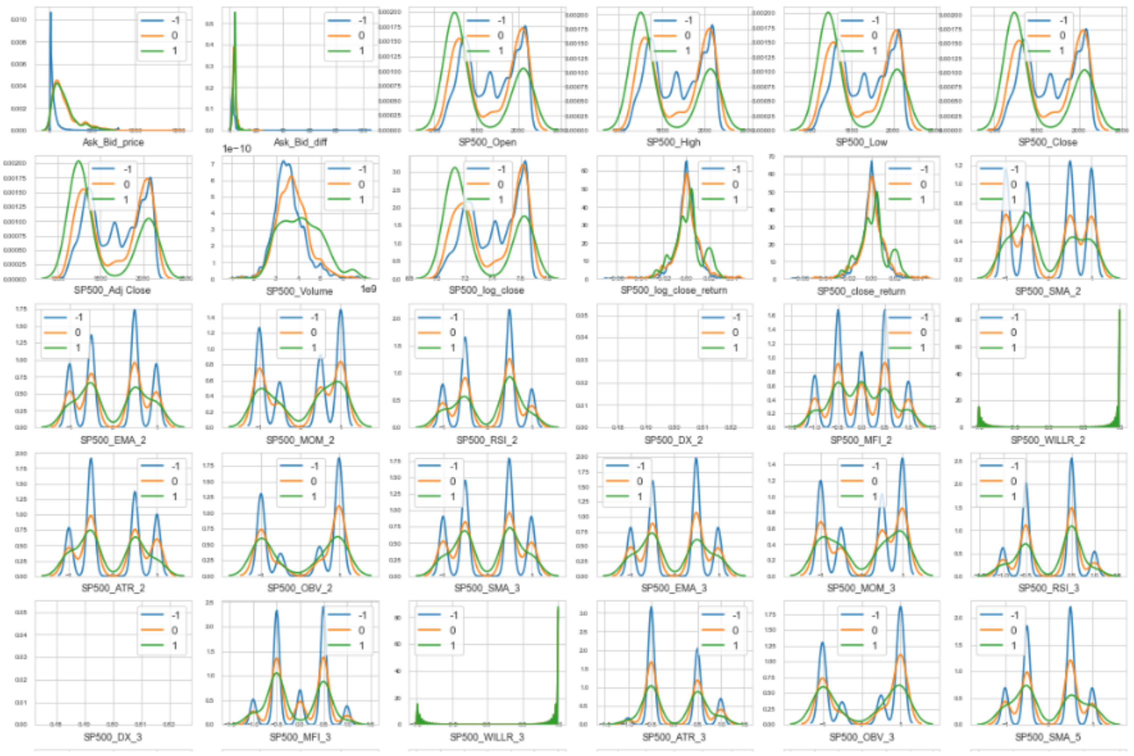


Figure 3.16: Features Distribution by target example

3.6.3 Log Reports

The Log reports are text files that save information regarding the execution of the algorithm. This log saves the configurations chosen for the execution, the execution time, the algorithm, the results for the multiple validation sets, the metrics result after testing, and the comparison with the baseline, benchmarks and best case, and worst case scenarios.

3.6.4 Proof of Concept (PoC)

A PoC is a popular way to showcase a system's viability. It ensures if the strategic requisites were met. Our PoC displays relevant information about our algorithm and Trading Simulation in order to better evaluate the systems predictive and trading capacity. The figures in Chapter 4 were generated in this PoC as both have the same intention to display the results and performance of the solution. It allows the user to analyse the results produced by the configuration files after tweaking some parameters. Figure 3.17 illustrates an example of what is possible to analyse. Beyond the Cumulative Returns compared with the SPX benchmark, it also displays how many Long and Short positions are being taken over time, how many of those positions are giving profit and how many are losing money. In the last graph it is also possible to see how many options have a penalty and how many Take Profit orders were executed.

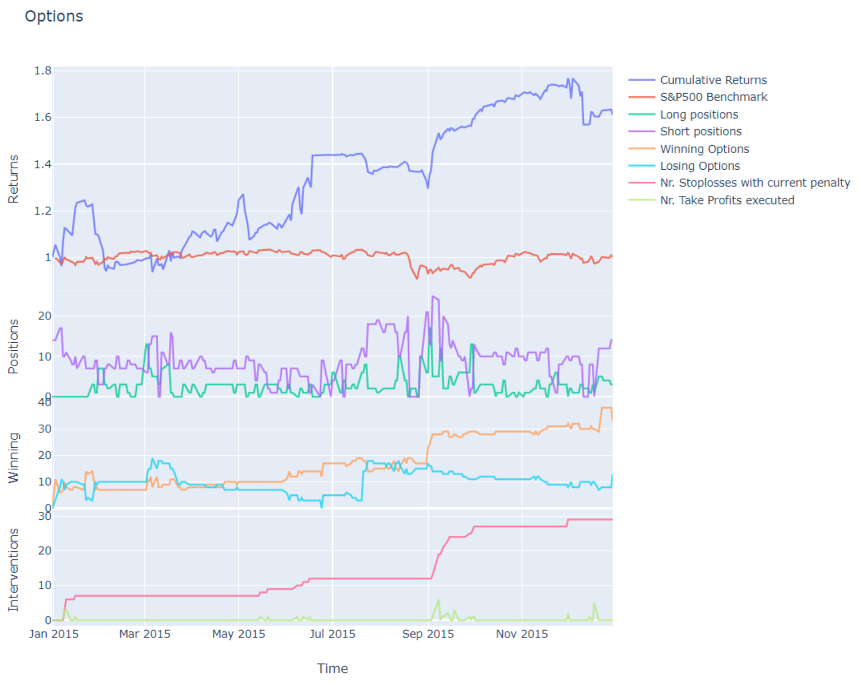


Figure 3.17: Example of visualization for returns analysis over time

We also used the python library *QuantStats* to generate a complete report of the Portfolio as an HTML file. This file is a tearsheet with some key perfomance metrics to properly analyse our Strategy.

3.7 Implementation

After publishing this work, the developed solution will be shared in the open-source platform GitHub [73]. Knowing this, the previously explained architecture was structured as described on Figure 3.18

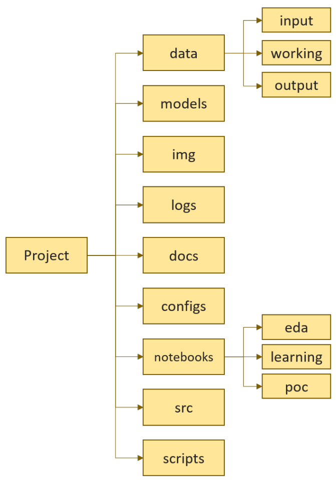


Figure 3.18: Folder organization in project development

The data folder contains all the data files, which are separated by three sub-folder: the ”input” folder for the retrieved dataset files, the ”working” folder for backup data files which are generated along the different phases, and the ”output” folder for the files containing the final data of our system. This files will be CSVs and pickle files. After evaluation, our program also saves the pre-trained predictor in the ”model” folder. The models are saved using the joblib library.

The ”img” and ”logs” folders are the target folders for any generation image and log reports. The log reports may contain useful information about the algorithm evaluation during CV, grid-search, and time execution. The ”docs” folder contains all the necessary documentation and detailed explanation of our algorithm, containing PDFs and Markdown files.

The ”configs” folder contains the parametrization files for the User, which contain YAML files with the parameters detail in Section 3.6.1.

The ”notebook” folder contains all the Jupyter Notebook and UI of our model. These visualizations are divided by three folders, the ”eda” and ”poc” folders which contains all the files for the EDA and PoC, and the ”learning folder” which contains basic and simple examples of the used libraries. The ”src” folder stand for ”source” and has all the developed code. This solution was programmed using the paradigm of Object-oriented programming. Inside the ”src” folder, there is a ”sklearn_MoE” sub-folder which contains the basic MoE solution using the Scikit-learn contribution rules and programming rules.

Finally, the ”scripts” folder has Python scripts which allow us to run the system as one program and select the specific layers we want to run.

4

Evaluation and Results

Contents

This section reports and examines the results accomplished by the developed system. First, an explanation of the Methodologies of Work Evaluation is made, followed by the presentation of the starting point for our system, as well as which benchmarks were defined. Next, we present the overall results of our solution. After that, we study our solution in two different Case Scenarios, looking in detail to each case. Finally, we make a comparison between our results and the established benchmarks to get a sense of the relative success of the solution.

4.1 Methodologies of Work Evaluation

All the metrics stated in Section 2.5.2 will evaluate the performance of the solution implemented in this thesis. Results for each algorithm implementation will be compared with each other to determine the best system. These results are the approximation of an optimized configuration based on a finite number of combinations for the hyperparameters. Due to time complexity, we can't assure that the combination used for the classifier will be the best for this problem but at least the best for a set of reasonably considered combinations. After achieving the best results for our algorithm, its profits, risk and accuracy will be compared against strategies such as the ones described in the following section.

For the stated metrics, we will focus our analysis in the accuracy metric to evaluate the classification performance. In terms of trading performance, we will firstly focus on the Annual ROR as the first accomplishment. The Sharpe Ratio will be the main metric for risk, which we want to keep above 1. The RRR will then be used in case our system fails to improve in ROR or Sharpe Ratio, as a metric that allows us to understand if the balance between profit and risk surpasses our benchmarks.

4.2 Starting Point and Benchmarking

To better understand the dimension of the solution, this section states the starting point for our challenge, noting the benchmarks and goals that this project intents to overcome.

The results of the previously explained literature can be summarized in Table 4.1. The first analysis we make is that most works do not share their complete report but only the 2/3 metrics that achieved the best results. Although the work of Ballings et al. [9] achieved such good accuracy, we can't assure that our results will be able to get around 90% as it sounds like a ludicrous result for such a complex problem. We aim to achieve an accuracy around the 73.4% stated by Versace et al. [43], which is still a very high score. Regarding Profit and Risk we expect that our algorithm can get similar results to the maximum values stated of 73% of annualized ROR and 1.27 Sharpe Ratio.

Regarding benchmarks for the testing time periods, our first benchmark will be to adopt a B&H strategy in the SPX. This strategy is the simplest and most common approach when people invest in the

Table 4.1: Literature Results Summary

Metric	[8]	[62]	[9]	[19]	[4]	[43]	[44]	[7]	[23]
Accuracy	-	0.6228	**0.9037**	-	-	0.734	0.8635	-	0.6844
ROR	-	-	-	0.5024	-	-	-	-	-
Annualized ROI	0.09	-	-	-	0.10	-	-	**0.73**	-
Sharpe Ratio	**1.27**	-	-	-	-	-	-	-	-

market. The SPX represents the US economy, which presents a long term bullish trend. If the economy is healthy, the price should be in an uptrend. It is a strategy that takes very low effort, and in some cases can represent a better return than the retirement pension.

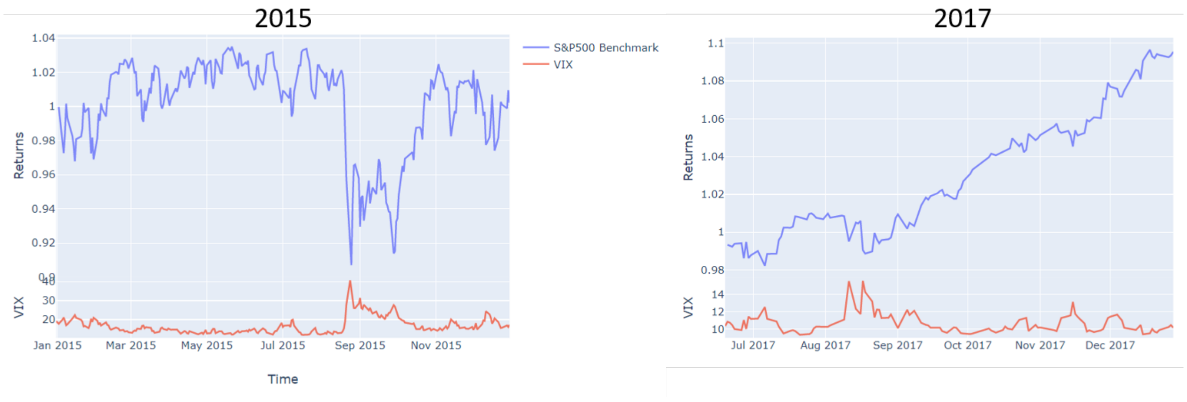


Figure 4.1: SPX and VIX in 2015 (on the left) and 2017 (on the right)

Figure 4.1 illustrates the application of a B&H for the two tested periods. As it is possible to analyse, these two periods are very different, which help us explore the algorithm's performance in multiple situations. The first period is from 31st of December of 2014 to the 30th of December of 2015. The second period is from the 20th of June of 2017 to the 28th of December of 2017. The first period corresponds to the 30% of the test of the Delta Neutral data, while the second one is the resulting data from the WRDS data after applying it to the Data Layer. For simplicity reasons, we will refer to them as 2015 and 2017, respectively. In terms of trend, the year 2017 is in a clear uptrend, ending with ROR around 9.5%, and 2.84 of Sharpe Ratio. On the other end, 2015 is sideways, ending with ROR around 0.22%, and a 0.09 Sharpe Ratio. While in 2017 the highest value of the VIX was 15.55, in 2015, the minimum value is 11.95, and the highest is 40.74. Both periods represent the full spectrum of hypothesis, with a sudden downtrend near September of 2015, having a volatile sideways period before that, and a fast recover around October. In the year 2017, we have a less volatile period with a clear uptrend. This period has a slow uptrend during most of the time, except, around late July and early August were it had a low volatility sideways period, followed by a fall when the VIX increased. These two periods are very contrasting, which will allow us to understand the performance of our solution when the SPX is in these kinds of trends. Although very contrasting, we recognise that both periods suffer the Max Drawdown, and the most volatile moment around the same time, late August beginning of September. The results for both periods are stated in Table 4.2.

Table 4.2: B&H applied to both test periods

Metric	B&H 2015	B&H 2017
ROR	0.00216	0.09541
ROI	0.00216	0.09541
Annualized ROI	0.00217	0.18150
Mean Daily Profit	5.80332	0.00080
Sharpe Ratio	0.09260	2.84230
RRR	0.00583	0.17905
Max Drawdown	-0.1235	-0.02110
Sortino Ratio	0.13112	4.46496

In an apriori analysis, we expected to exceed strategies such as the B&H for 2015 and 2017, as well as some of the metrics stated in the literature. We will focus our comparison in the work of Booth et al. [8], which achieved 9% of Annualised Return and 1.27 of Sharpe Ratio, as the architecture will be the most similar to ours. Not considering the Data Layer part of our solution, we want to compare the Logic Layer results with simple B&H and S&H for the given data. We expected the relevance of the VIX index to feed our algorithm as a good indicator of price volatility, and explore the biases direction of the Options to reduce the risk associated with each position taken. We also expected that the MoE architecture would improve the overall results from a single RaF.

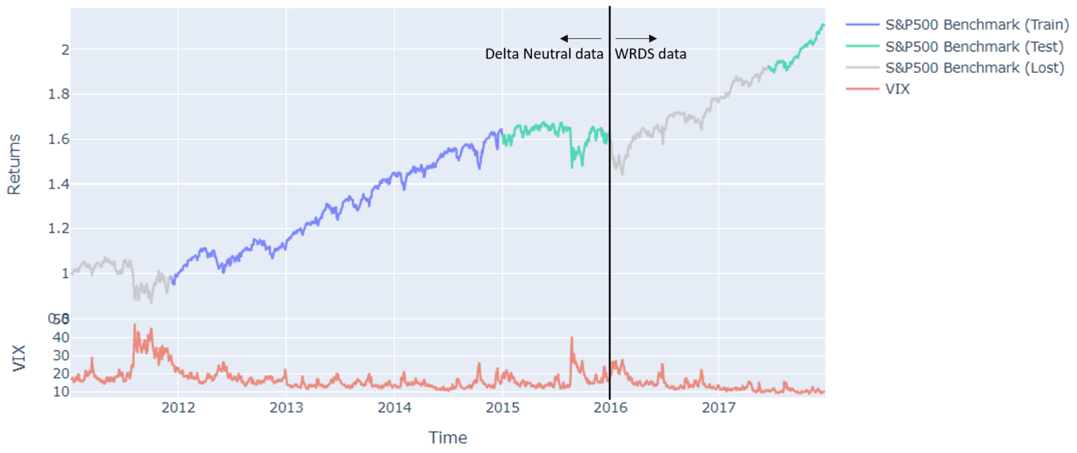


Figure 4.2: SPX and VIX in the complete time frame of the available data

Figure 4.2 illustrates the SPX and the VIX over all the data we had available. The periods were the SPX line is grey, represent the time where the data was "lost" by the Data Layer when creating the TIs and applying the Data Reduction process. This lost happens since, to calculate a TI with period X, we are only able to compute it with X days of data before the day we want to compute it. To test a real case scenario, and to avoid compatibility issues, we consider the WRDS data as a separated dataset only for testing, therefore we had to lose the first X days on the Data Layer. In this case, both datasets were complementary, which would allow us to use the Delta Neutral data to compute the first TIs without losing the WRDS data. However, we chose not to do it as it would only fulfil this specific case scenario, and not a situation where a User would download its own data, from any time period (after 2015), and test our system on it.

The purple line represents the period used for training, while the two green periods are the test periods, represented in Figure 4.1. As we can see, the training period diverges from the two testing periods. While the training period shows a moderated VIX and a bullish price with the peaks and throughs successively rising, the period of 2015 is sideways with high levels of VIX. At the same time, the year of 2017 represents very low values for the VIX, and, mostly, a strong bullish trend on price almost without any loss days. The year of 2017 was justifiably uncommon with the market rising too quickly. Although it is not represented in Figure 4.2, from January to March 2018, this price had a correction, with a depreciation around 10%. A correction is when the price of an asset drops, from its last peak, 10% or more. This happens when the asset is overvalued at that time, and a drop in demand occurs.

To sum up, the available data alongside the single properties and requirements of Time Series creates a situation were both test periods were never seen in the training period. Therefore, we expect lower accuracy values than in the literature. Nevertheless, we expect those same predictions to have satisfactory results in the trading metrics.

4.3 Results Overview

As explained in the previous section, this thesis has 2 case studies which are represented by two different time windows: 2015, and the second half of 2017. Each case study tests the following approaches:

- Benchmark - Trading Results if a B&H strategy was applied to the SPX.
- Best Case - Trading results if the algorithm used had 100% accuracy on the selected data.
- Worst Case - Trading results if the algorithm used had the inverse decision of the Best Case.
- B&H - B&H to the selected set of SPX Options.
- S&H - S&H to the selected set of SPX Options.
- RaF - Single RaF
- MoE (Equal) - MoRFe with Equal Weight as Gating Function
- MoE (Bin Split) - MoRFe with Bin Split as Gating Function

The SPX prices are used as the main benchmark for comparison on how could we benefit from just using a B&H strategy in the market. The Best Case, Worst Case, B&H and S&H give us a glance of the quality of the data selection, and will give us more context to understand the trading performance of our algorithms. After running all simulations, Table 4.3 joins all the metrics for these approaches. It is important to remember that the Best and Wort Cases represent the best and worst cases in term of predictions, which doesn't mean that it applies in terms of return or risk.

Table 4.3: Results for all case scenarios

		Precision	Recall	Accuracy	F Measure	ROR	ROI	Annualised ROI	Mean Daily Return	Sharpe Ratio	Risk Return Ratio	Max Drawdown	Sortino Ratio
Train	RaF	**77.06%**	**75.19%**	**75.19%**	**75.80%**								
	MoE (Equal)	**77.68%**	**75.74%**	**75.74%**	**76.38%**								
	MoE (Bin Split)	**77.22%**	**75.35%**	**75.35%**	**75.96%**								
2015	Benchmark					0.22%	0.22%	0.22%	0.01%	0.0926	0.58%	-12.35%	0.1311
	Best Case	100.00%	100.00%	100.00%	100.00%	6170.92%	6170.92%	6192.11%	18.56%	1.0245	6.45%	-13.16%	207.9627
	Worst Case	0.00%	0.00%	0.00%	0.00%	-702.03%	-702.03%	-704.44%	-20.98%	-0.9365	-5.90%	-3342.85%	-0.9764
	S&H	30.12%	54.88%	54.88%	38.90%	-42.07%	-42.07%	-42.21%	-0.84%	0.9194	5.79%	-94.26%	1.6838
	B&H	17.06%	41.30%	41.30%	24.15%	229.61%	229.61%	230.40%	1.24%	0.9424	5.94%	-28.92%	16.0484
	RaF	**50.28%**	**49.26%**	**49.26%**	**49.63%**	**91.49%**	**91.49%**	**91.80%**	**0.35%**	**1.3706**	**8.63%**	**-45.94%**	**1.9980**
	MoE (Equal)	**50.26%**	**49.10%**	**49.10%**	**49.52%**	**95.11%**	**95.11%**	**95.44%**	**0.26%**	**1.6904**	**10.65%**	**-19.28%**	**2.8591**
	MoE (Bin Split)	**50.19%**	**48.34%**	**48.34%**	**49.01%**	**100.20%**	**100.20%**	**100.55%**	**0.28%**	**1.8983**	**11.96%**	**-19.76%**	**2.9298**
2017	Benchmark					9.54%	9.54%	18.15%	0.08%	2.8423	17.90%	-2.11%	4.4650
	Best Case	100.00%	100.00%	100.00%	100.00%	87.02%	87.02%	165.54%	0.45%	2.5023	15.76%	-1.22%	85.4127
	Worst Case	0.00%	0.00%	0.00%	0.00%	9.02%	9.02%	17.16%	0.05%	0.6547	4.12%	-38.23%	1.3437
	S&H	38.41%	61.98%	61.98%	47.43%	16.42%	16.42%	31.24%	0.09%	0.9128	5.75%	-53.04%	2.4444
	B&H	11.75%	34.27%	34.27%	17.49%	99.17%	99.17%	188.65%	0.52%	1.9946	12.56%	-18.59%	6.3167
	RaF	**48.34%**	**43.97%**	**43.97%**	**45.74%**	**212.37%**	**212.37%**	**404.00%**	**1.11%**	**2.4665**	**15.54%**	**-12.22%**	**15.2212**
	MoE (Equal)	**48.12%**	**43.97%**	**43.97%**	**45.69%**	**52.92%**	**52.92%**	**100.68%**	**0.28%**	**1.4011**	**8.83%**	**-18.11%**	**3.2661**
	MoE (Bin Split)	**49.93%**	**47.09%**	**47.09%**	**47.57%**	**42.51%**	**42.51%**	**80.86%**	**0.22%**	**-1.6387**	**-10.32%**	**-110.08%**	**-0.1700**

A first analysis of the table allows us to see a big difference between the predictive results in the train data and the test periods, which may indicate some overfitting. This overfitting is already mitigated by choosing an algorithm which avoids it and applying some techniques such as Oversampling, Under-sampling, Feature Selection and PCA. As referred in section 2.5.2, this calculated accuracy is weighted for each class. Therefore we think these lower values on out-of-sample data may result from the less frequent labels as the number of examples is lower, causing a possible underfitting.

In all cases, the precision value for each algorithm is higher than the recall and accuracy. This means that the number of times a positive identification was actually correct is higher than the number of times an actual positive was identified correctly. Higher Precision is good when we want to enter a position, while higher Recall is good when we want to exit a position. As our trading algorithm has Stop Loss and Take Profit rules to exit a position, in the predictive process, it will be a priority to have better Precision than Recall. The F-measure represents the mean of both metrics (Precision and Recall), and we can see a significant improvement of our values from choosing to always use one of the positions at all times. It is possible to see that the MoE algorithms tend to have solid returns and better risk management. Therefore the RRR shows solid improvements.

4.4 Case Studies

In this section, we will cover both testing periods in more detail. We start by describing each period. Then we review the period results of Table 4.2, followed by a comparison of the benchmarks applied to the data coming from the Data Layer. Finally, we analyse the performance of each algorithm closely and compare it to the previous ones. This review of results will frequently mention values from Table 4.3.

4.4.1 Testing on 2015 data

This test period corresponds to the range between the 31st of December of 2014 and the 30th of December of 2015. The training period for our algorithm corresponds to the previous four years. Therefore we can expect the results being slightly better than for the 2017 period because some events affecting this period may have been affecting the training set in some sense. This period is sideways, ending with a low ROR of around 0.22%, and an even lower Sharpe Ratio of 0.09, when applying a B&H on the index. The VIX indicates values between 11.95 and 40.75, which represent very high volatility. High volatility in both directions will result in very volatile Options prices, as most traders will opt to stay out of the market or to assure a maximum loss on their investment by buying an Option.

In terms of selected data from the Data Layer and the applied labels, we can find outstanding results. Although very hard to accomplish in a period like this, if it was possible to predict all the daily directions correctly, our trading algorithm could profit an ROR of 61 times the money invested. On the other hand, if our algorithm would buy when it should sell, and sell when it should buy, our trading algorithm would lose seven times the money invested. This difference happens not from the forecasting performance, but from the account of the Trading Simulator, which uses Stoploss and Take Profit strategies to stop wrong positions and assure profits in good ones. This 61 to 7 ratio give us great confidence to test our algorithm. It is essential to highlight that high volatility periods provide very profitable opportunities, but are equally hard to predict the direction and time of its swings. This 6170.92% in ROR is only possible due to the high volatility of the SPX index. However, the accuracy of the ML algorithm will be understandably lower, and, realistically, will never get near this profit.

4.4.1.A Single RaF

The majority of the architecture for this thesis was developed using a single RaF. We pretended to establish a baseline based on the RaFs stated in the literature [8, 9, 23]. In this sense, some of the preprocessing steps were chosen due to the necessities of this single RaF. For this reason, we expect to accomplish optimised results solely with this classification algorithm. After running a Random Search over a high dimensionality hyperparameter set, and a Grid Search over a narrower set around the first best configuration, we accomplished a final set of hyperparameters that gave us the best results for the training dataset. When applied to the period of 2015, we got the following test results. Figure 4.3 shows us the performance of the Trading Simulator over time.

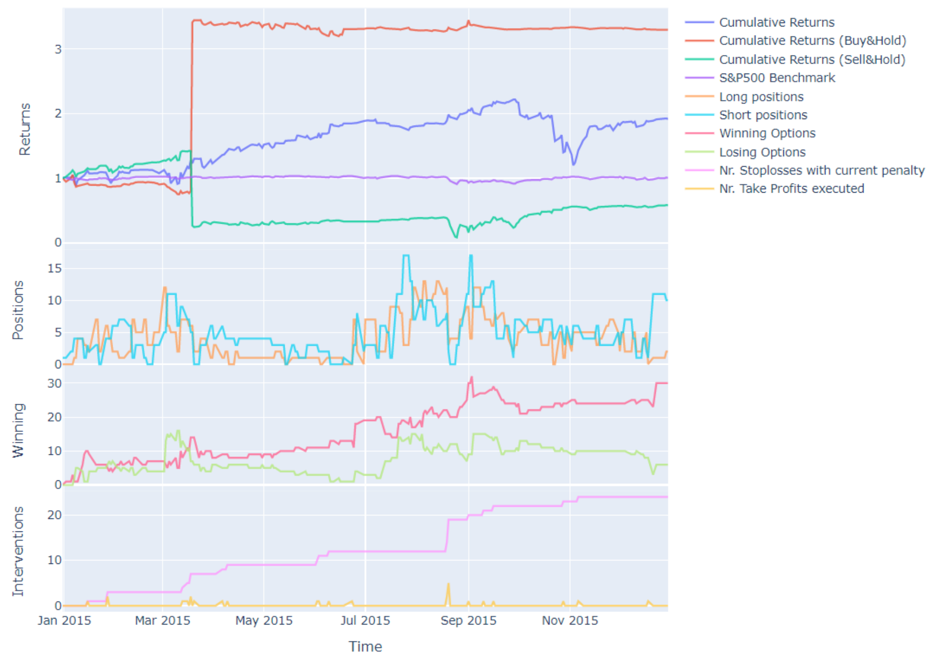


Figure 4.3: Single RaF results over 2015 testing period

In the previously stated Table 4.3, the RaF achieved an accuracy of 49% which exceeds both the B&H (24%) and S&H (38%), however its results do not get close to Versace et al. [43] (73.4%). Beyond the reasons stated in Section 4.2, this may happen because of the increased volatility of this period, which will comprehensively be harder to predict.

Regarding ROR, it achieved 91% of return, which represents an investor almost doubling the invested money every year. This result already overcomes all the RORs reported in the literature, by a good margin. Although in Table 4.3 we conclude that the B&H strategy had better ROR we can see in Figure 4.3 that the S&H and B&H suffered a peak over their returns around March. Although the B&H returns jumped over 200% on that time, the rest of the time, it was losing money slowly. Contrarily, the S&H strategy was getting profit most of the time, except when it had sudden losses which we warned about in Section 3.2. That event happens as both strategies have a position on the 19th of March, where the Option *"SPX150320P01900000"* rises from $\$1.5$ to $\$40$ in a day, as we can see in Figure 4.4. Before that event, our algorithm was performing a little bit under the S&H and better than the B&H. During the event, it was neutral for the Option which originated the peak. After the event, it got better performance than both strategies until October, when it had a big Dropdown, while the other strategies were not as volatile as this one. Looking to Figures 4.3 and 4.1, it is possible to infer that this loss happened during the fast recovery of the SPX during the month of October, which the algorithm failed to predict.

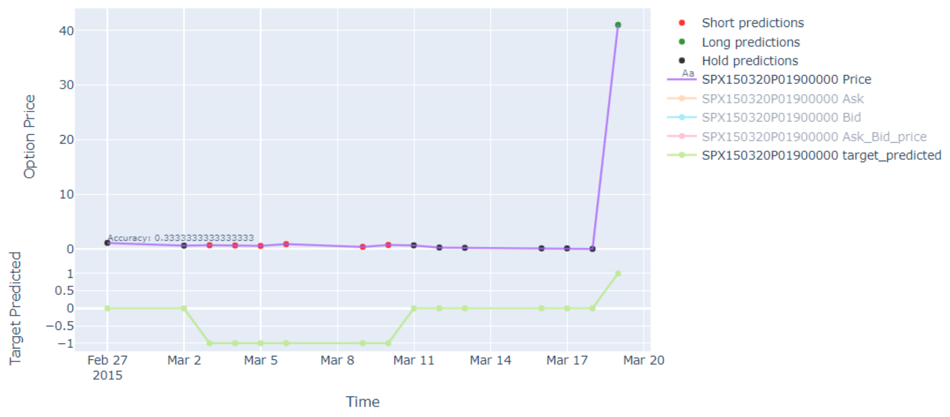


Figure 4.4: Option with sudden change in price

Comparing to our Benchmark (B&H on the SPX), it is possible to analyse how much leverage we can take from using Options instead of only buying the index. Compared to our Option returns, the index returns look almost flat. The Options volatility is increased by the volatility of the index, allowing our solution to exploit higher swings in prices.

In terms of risk, the previous approaches presented Sharpe Ratio levels below 1, while the RaF gets 1.37, which is already a good indicator of risk management from this algorithm. This indicator also surpasses the 1.27 reported in Table 4.1. As stated in Section 2.5.2, the difference between the Sharpe and Sortino Ratios is based on the Sortino Ratio only discounting the ROR based on the standard deviation of the Downward ROR to calculate the risk. In this case, the B&H achieved better performance as its consistent losses are not very volatile, as we can see by comparing the maximum Drawdowns. Although it has a higher maximum Drawdown, using the RaF algorithm makes it achieve a stronger RRR of 8.63%, which gives us a notion of achieving better profit/risk ratio.

In a deeper examination, we can see that our algorithm uses a similar share of Long and Short positions at the same time, only presenting an extended period of having more Short positions than Long positions around April, when it was continuously rising the ROR. We can also see an increase in winning Options over time, and some periods where the number of losing Options were higher than the average. The number of Take Profit orders seems to be more distributed over time, having a higher number on August, overlapping the most significant drop in the SPX price.

4.4.1.B MoRFe with Equal Weight as Gating Function

For comparison purposes, we chose to maintain the configuration of the RaF for the MoRFe algorithms, as it would allow us to see, for the same hyperparameters, the progress of results following the improved architecture. In this case, the only parameter that changed on the base learner was the Random Seed which was used to generate new seeds for each RaF, instead of going directly to the RaF algorithm.

In terms of predictive performance, the results were pretty close to the single RaF, which was expected. The whole dataset was sent to each expert, giving them exactly the same expertise. Therefore, we basically had almost the same RaF over and over. The only difference was the random factor of each expert. Even though the results were almost the same, this solution had a slightly worse prediction as it had slightly better results in the training period and slightly worse results in the test set.

On the other hand, the trading results showed some improvements from the single RaF. The ROR improved from 91% to 95%, while the Sharpe Ratio went from 1.37 to 1.69.

Figure 4.5 displays the ROR of our algorithm over the testing period. Some of the insights we take from this figure are similar to the conclusions from using a single RaF. Until the breaking point of the B&H and S&H strategies, the MoRFe (Equal) was performing better than the B&H and under the S&H. After that, the algorithm had a good ROR momentum until June, when it started to stabilise with a slower uptrend.

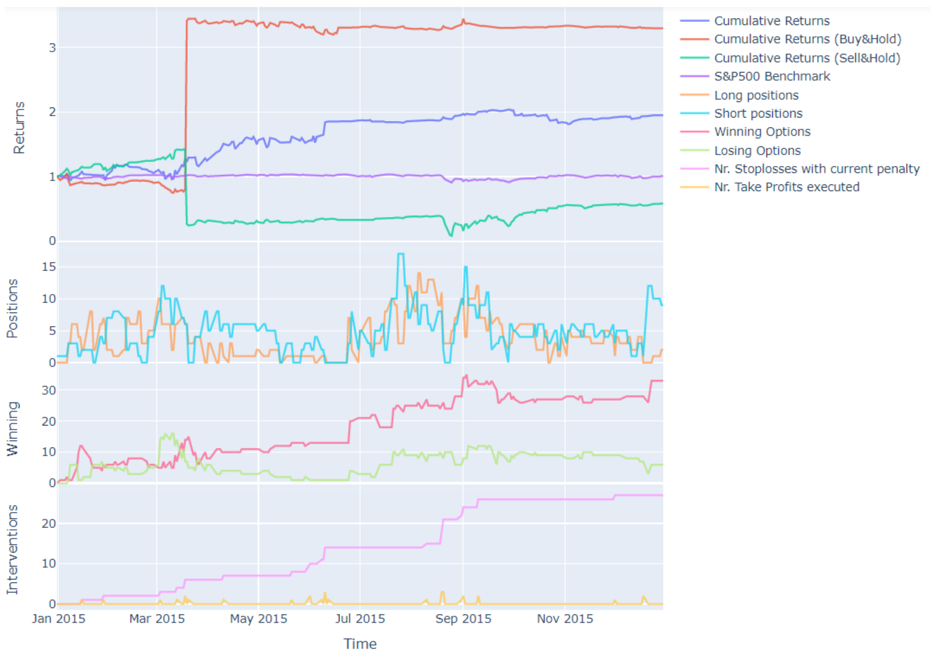


Figure 4.5: MoRFe (Equal) results over 2015 testing period

If we compare Figures 4.3 and 4.5, we can observe more stable returns from this algorithm. While the single RaF had a higher maximum ROR than the MoE around September, if we examine the months of February and October, the major losses have decreased to lower values. A higher number of Stop Loss orders seem to be activated during the month of June, which may explain the slower growth in ROR during the following month. The number of losing Options from July to November reduced from using only one base learner. Thus, the major benefit of using the MoE (Equal) over a single RaF seems to be the improvement in risk management. It increased the Sharpe Ratio, from 1.37 to 1.69, the Sortino

Ratio, from 1.99 to 2.86, and the RRR from 8.63% to 10.65%.

4.4.1.C MoE with Bin Split as Gating Function

Finally, we tested the MoE with the Bin Split function, which enabled the creation of experts with different subsets of data in their knowledge base. This split used the most relevant feature, determined by the SelectKBest, to generate equally ranged bins. The test was made with more features, however, the results seemed to get worse the less important the feature was.

Using this gating function achieved a drop of 1% on the accuracy, registering 48%. This decrease may happen as the base learners lose the most relevant feature to train the RaF. However, if the feature used to split the data were discarded entirely, we would have a much higher impact on the results. Hence, we understand that using a feature to break the data into different learners may have a beneficial effect on long term fitting. A possible improvement for this function would be to reference a biased feature such as the VIX or a TI. Nevertheless, that capability might limit the usage of some preprocessing steps such as sampling or PCA.

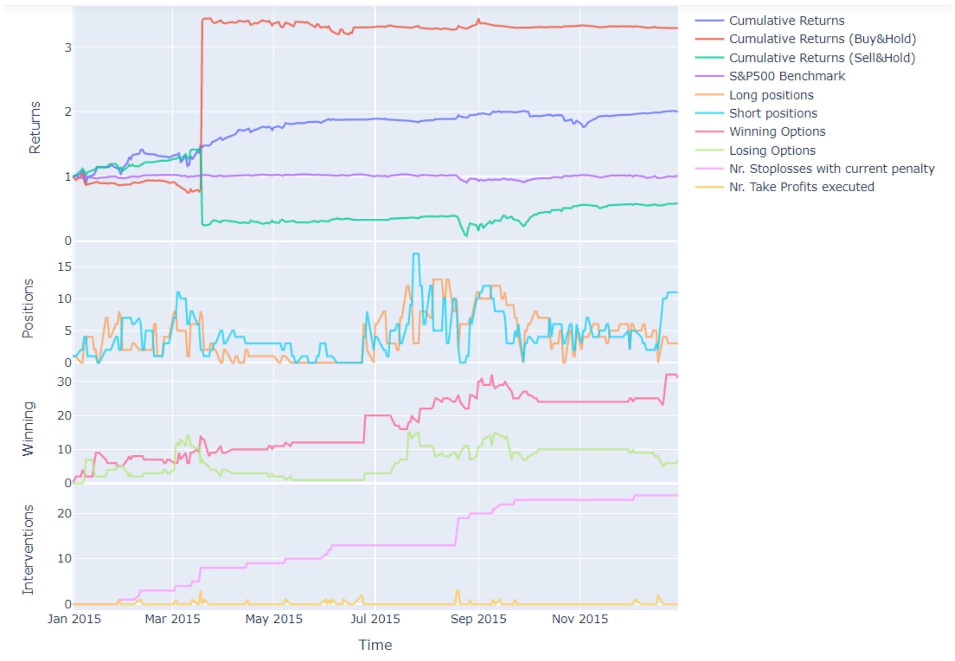


Figure 4.6: MoRFe (Bin Split) over 2015 testing period

If we explore Figure 4.6 and the results in Table 4.3, it is possible to verify an improvement in the trading performance by using this gating function. The ROR hits the 100% of annual ROR, and registers the highest Sharpe Ratio, increasing from 1.69 to 1.90. Also, this algorithm is the first to beat the S&H strategy during the month of February, and hits a very high ROR during the month of June, when its gains slowed down. This function accomplishes the highest RRR (11.96%). Although it registers higher peaks of Losing Options than the Equal Weight gating function, we can perceive the higher comeback from the Drawdown of the beginning of November on Figure 4.7, what let us compare the RORs of all the strategies together.

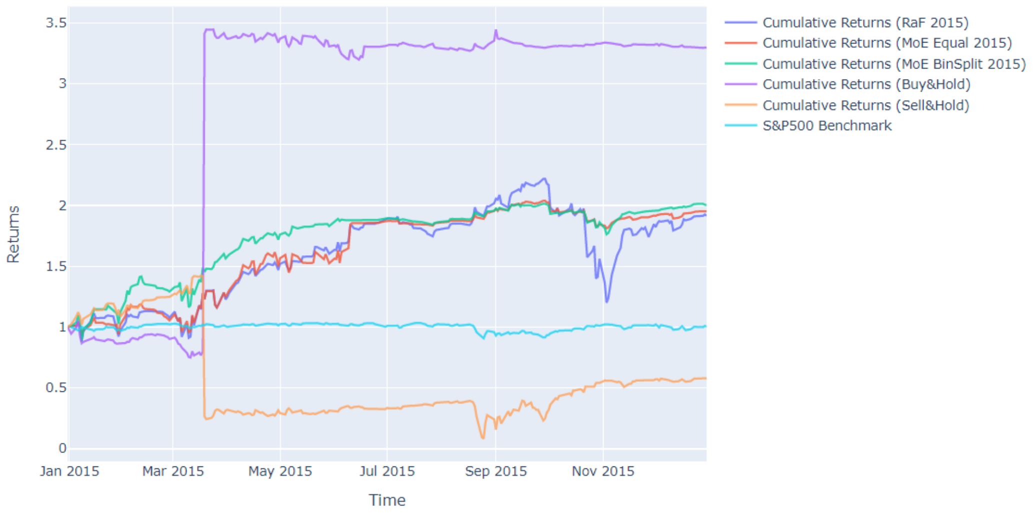


Figure 4.7: Returns of all the strategies during the 2015 testing period

4.4.1.D 2015 Conclusions

We can conclude that the three algorithms performed very well during this period of high volatility. The B&H strategy only works if we catch these big swings in price, that should compensate for the remaining losses, while the S&H works if we avoid those same swings. On the other hand, the the RaF and MoRFes, performed better over time, only losing from the money invested in short periods of time. These losses only occurred during the first three months, where the cumulative profit couldn't compensate for a possible loss. From the single RaF we understand improvement in risk management, where the months of September to November were less volatile. On the MoRFe gating functions, we can see an improvement in the capability of recovering from losses, since we see higher profits after a Drawdown. Three events when this ability was evident, were the recoveries from February, March and November.

Although less accurate, the MoRFe presented a substantial improvement in profit/risk ratio, as we can see by examining the RRR values. This algorithm also beats the reported metrics in the literature, in a period where investing in the SPX directly would result in hardly any profit. Most of the return was achieved until the month of June where it diminished the number of positions, but that fact can also be related to the decisions made on Section 3.5.5.G. As the amount of money is not evenly distributed over time, and the amount of cash available to invest is lower for the MoEs because we already allocated some on the most profitable Options, the amount invested in those Options is lower, therefore less expressive. Figure 4.8 proves that around September, the amount of money in cash is vastly more than for the MoEs. With that, we can comprehend that in the beginning of September, the RaF performed better for investing more money in a set of positions. However, those same positions are responsible for the significant loss that followed, which took it to almost equal the SPX B&H.

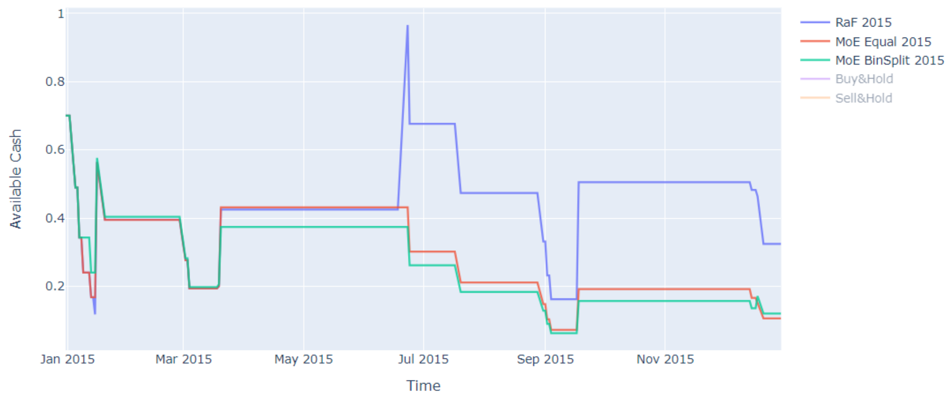


Figure 4.8: Cash available for each 2015 trading simulation over time

4.4.2 Testing on the second half of 2017 data

The period from the 20th of July to the 28th of December of 2017 was our second test period. This period corresponds to the data collected from the WRDS, which was preprocessed separately. This period has an extremely bullish trend with low volatility, and seems like one of the moments when investors would be more confident to invest with a B&H. Nonetheless, this period is probably overvalued, as described in Section 4.2. Even though its price could continue rising, it would eventually, sooner or later, suffer a correction. The VIX fall is a reflection of the optimism of the investors. The price could drop in the next day, or only in the next year. Although the B&H has an exemplary Sharpe Ratio of 2.84, it presents a low ROR of 9.5% in six months, compared to the results reported by our algorithms in the previously studied period. The VIX indicates values between 9.36 and 15.55, which are lower than before. Only August has a peak in the VIX values. These values overlay the most significant Drawdown of the period.

Regarding the selection of Options from the Data Layer and the classification of each position, we can find less opportunity from this period. This is a result of generating a longer frame to reconsider the positions in this low volatility periods. This decision was made to avoid high swings from a low set of orders. As said before, an investor tends to buy an Option in times of uncertainty. Those are the moments were paying a fee to limit possible losses looks more attractive. Thus, in this time of uncertainty, the Option prices show a wide range of values, which results from the rise and fall of demand. Despite being in an uptrend as the training period, the VIX values may have an impact on accuracy on the moments of highest evaluation. Thus, the expectation would be a very high or a very low ROR and a similar accuracy to the 2015 period.

For this period, if all the algorithm predictions were correct, our trading algorithm could profit an ROR of 1.87 times the money invested. The big surprise here is the change in strategy for low volatility, allowing us to also make a profit in case of switching the trading decisions. Our trading simulator would make around 9.02% in profit. This situation may symbolise that the prices are too unstable but changing inside the same range. This variation allows us to make a profit in either direction. Therefore, the Take Profit order makes the difference so we can exit at the right time. In case of a S&H strategy, our trading simulator would generate a ROR around 16.42%, and the B&H would generate around 99.17%. In terms of Sharpe Ratio, the S&H and B&H present values of 0.91 and 1.99, respectively.

4.4.2.A Single RaF

The single RaF performed exceptionally well for this period. However, the accuracy dropped to 43.97%, which seems to be more affected in terms of precision. Thus, affecting the ability to enter a position correctly. However, the algorithm achieved an incredible 212.37% of ROR and 2.4665 of Sharpe Ratio. The ratio between those metrics is represented in the 15.54% of RRR, which almost equals the result of the Best Case. The reason for not overcoming it relies on the Max Drawdown of -12.22%.

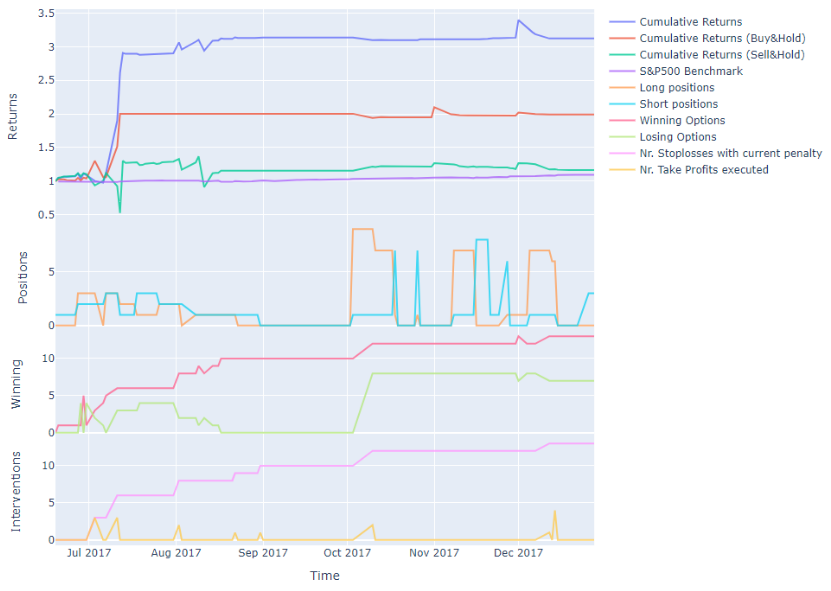


Figure 4.9: Single RaF results over 2017 testing period

Looking at Figure 4.9 we immediately identify a much lower amount of Option positions taken in this period. This happens as there is a higher amount of Options which do not fulfil the requirements of both the Data Layer and the Trading Simulator. The number of Stop Losses rise much sooner than the 2015 period, as well as the number of Take Profit orders.

We can also comprehend that the reason for this high profit relied on the interval between 6th and 13th of July when the SPX price was starting a bullish trend. Nevertheless, the algorithm continued to make a profit, rising its profit from 190% to the final 212%. Once more, we also prove the leverage of trading Options, from a simple B&H on the index.

4.4.2.B MoE with Equal Weight as Gating Function

The first application of the MoE had slowly changed the prediction results, similarly to the 2015 period. However, in this case, instead of improving results, the MoE didn't profit from the 6th to the 12th of July. It has worse performance than the S&H before July, and a close valorisation until mid-August, when it increased the number of winning positions significantly and achieved around 52.92% of ROR, in half the time of the 2015 period. This may be related to the fact that the market is not reacting as during the training period. Even though it didn't profit so much as the B&H on its peak, after that period its profit raised from 29.85% to the final 52.92% which is an excellent result. Even though, it is achieving good results with Sharpe Ratio around 1.40, we can see in Figure 4.10, the number of losing positions is rising on three occasions, what flattened the ROR progress.

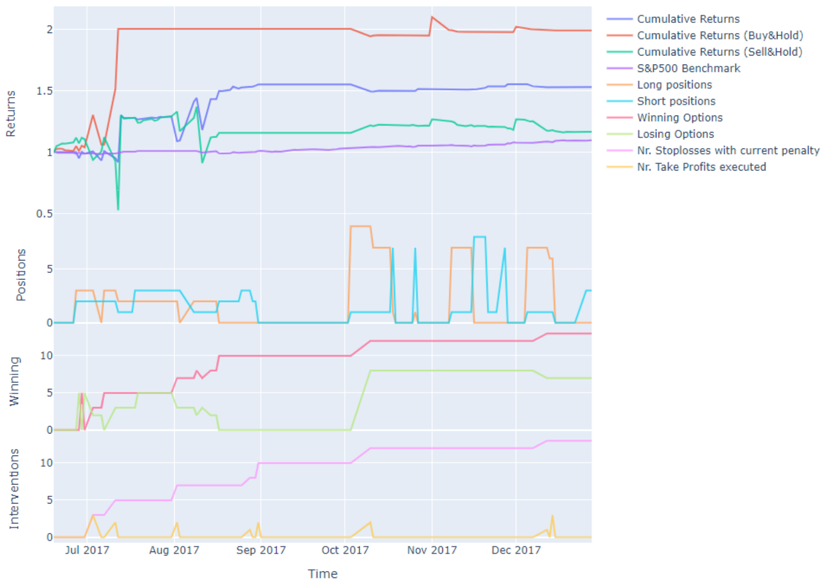


Figure 4.10: MoRFe (Equal) results over 2017 testing period

4.4.2.C MoE with Bin Split as Gating Function

When using the BinSplit gating function, the results improved in accuracy up to 47.09%, what shows us that the divide-and-conquer strategy may improve our performance. However, the predictions didn't have a better result in terms of trading performance, as the previous algorithms. This may be normal as the hyperparameters were tuned for the single RaF during the CV period.

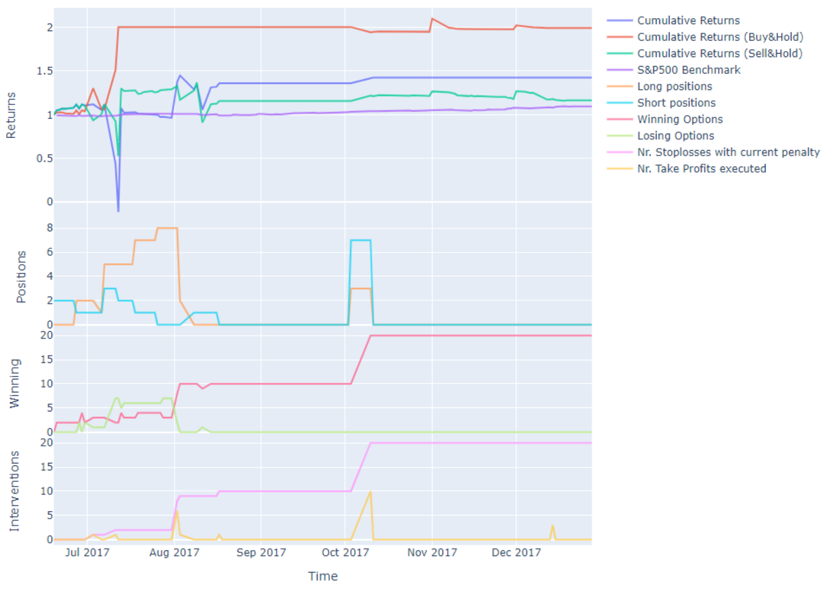


Figure 4.11: MoRFe (Bin Split) over 2017 testing period

Notwithstanding, the algorithm achieved 42.51% of return in six months. The main problem with its performance was in terms of risk. The Sharpe Ratio presents a negative value of -1.6387 with the max Drawdown of -110.08%. This means that at a specific point, the algorithm was not only losing the money invested but also in debt. We can see in Figure 4.11 that this Drawdown only happened in one day of July, which bounces back right after. Even so, this loss represents a significant threat for a portfolio, affecting the RRR with -10.32%.

During this period when the RaF increased dramatically, this algorithm registered a spike downwards, as it invested in the same Options in the opposite direction. If we inspect it further, as it is possible to do in Figure 4.12, this results from a Short position which has a spike in price. However it normalises in the next day, with the algorithm not exiting the position in loss, but actually making a profit. Figure 4.12 illustrates a common problem accurately in the Option prices, which present sudden spikes with growth higher than 100% from time to time. Something that didn't happen so regularly during the training period.

4.4.2.D 2017 Conclusions

In conclusion, the three algorithms had very different results, as one single position could change drastically the profit generated. One of the main events for this difference is the period from 6th to the 13th of July, where the right prediction could completely change the outcome. For this reason, a lot of Take Profit and Stop Loss orders were activated in response to this high swings in price, justifying why all

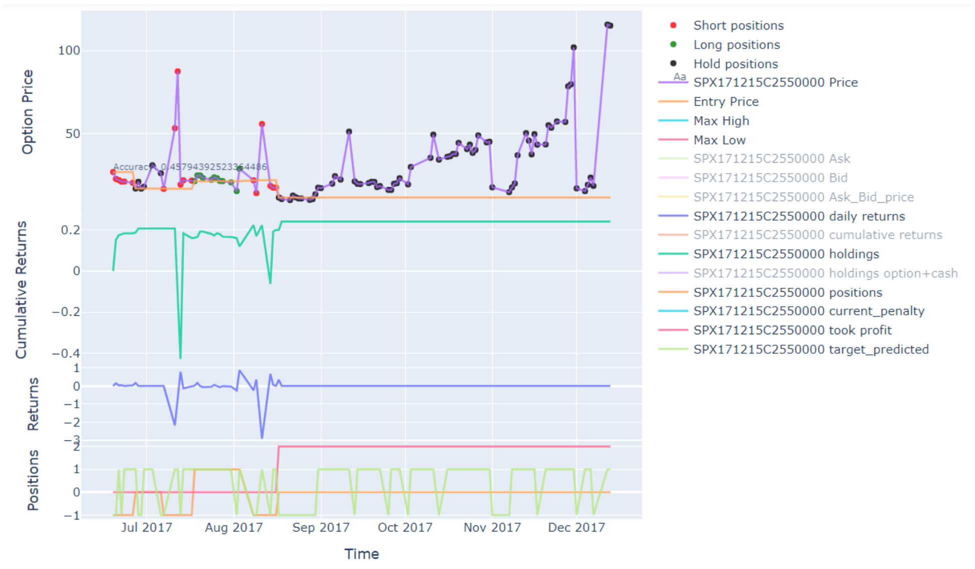


Figure 4.12: Option with sudden spike on price during 2017

strategies had much flatter ROR differences after September. One common issue is the few amount of Options considered to be traded for this period, leaving some parts of the period with the algorithm in a neutral position in any Option.

An excellent indicator was that our Trading Simulator was able to have profit, either adopting a B&H, or a S&H strategy. The single RaF was the top performer as it got one spike on the right direction. For this reason, there was some Take Profit and Stop Loss orders activated in the early beginning. On the other hand, the MoE with the Bin Split caught one of those. Nevertheless, this system was more profitable in this period, as it was applied to a shorter time span.

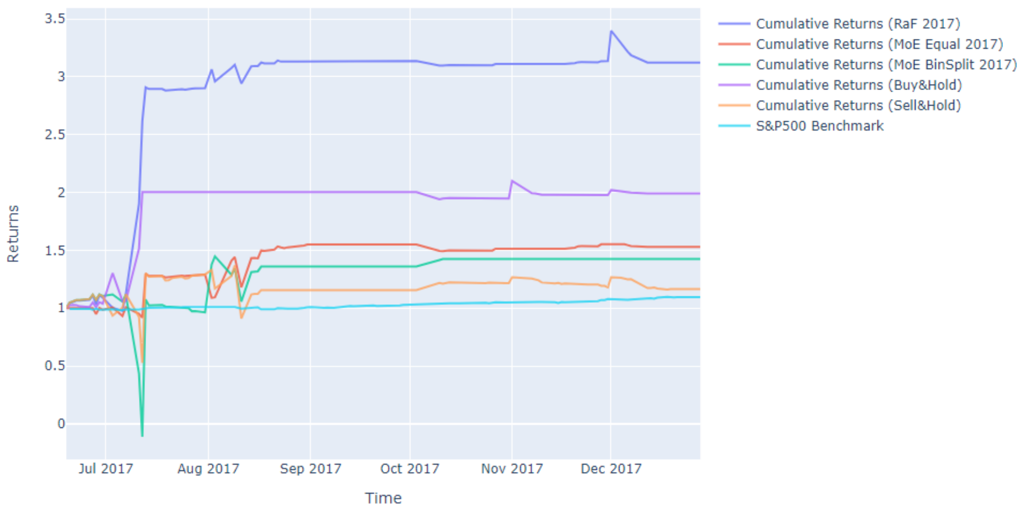


Figure 4.13: Returns of all the strategies during the 2017 testing period

5 Conclusions and Future Work

Contents

This work delivers a complete Machine Learning (ML) trading solution, from the collection of data to the trading simulation, based on forecasting the price directions. We hope this work come to help future researchers to learn more about Options and explore its possibilities. It was developed to become an Open Source project and to establish a base for a future thesis on ML applied to Finance. Although most theses on this topic end up not sharing the code online, we firmly believe that sharing our project would enable others to go further on their research, without the necessity of coding the whole system from scratch. Since this thesis is related to two fields, ML and Finance, the input from specialists in both areas would be welcome. Therefore, our architecture delivers a straightforward interface which allows people with a finance background, to collaborate without a Computer Science degree or any code skills. Our solution also presents state-of-the-art results for the options market. The application on S&P500 index (SPX) options, instead of common stocks, adds significant progress on Options research, which we proved to be a reliable source of income if combined with ML algorithms. To the best of our knowledge, the Mixtures of Experts (MoEs) architecture based on the scikit-learn architecture would possibly evolve to a contribution for the most famous ML library. We offer a different approach from most MoE literature by not choosing to implement it as Neural Networks (NNs). We also prove that the application of this methodology accomplishes better predictions than its base learners, mostly on topics that simple ML metrics may not mirror. Lastly, this thesis applies an innovative strategy and completed the primary goal: achieving high profits with controlled risk, reflected on the stated Risk Return Ratio (RRR) values. The next chapters will sum up the conclusion on the positive and negative points of our solution, followed by our recommendations for future work on this topic.

5.1 Conclusions

After a full analysis of Chapter 4, we can conclude that the thesis goals were attained. Besides using Options to leverage profits which beat the SPX exceedingly, our trading simulation overcame the benchmarks stated in the literature in profit and risk. The developed architecture of MoRFes proved to determine useful insights which were reflected on the metrics. Although the ML metrics were below the literature, the disparity between the training and test sets justifies it, achieving pretty satisfactory results.

The Data Reduction process surpassed our expectations as it created a reliable foundation to trade in the selected Option, even with a simple B&H strategy. The Trading system proved to exploit the ML predictions properly, allowing to mitigate the risk and achieve higher profits. The system amazed through its performance on high volatility periods, assuring stable and high returns. Regarding risk, it also mitigated the effect of volatility, ensuring low risk. On the other hand, in times of weak volatility, Options with high swings in prices should be avoided as it represents a very high risk on our portfolio. The low volatility values should be further studied and understood to create a more mindful rule which would allow our system to stay out of the market during these moments. Overall, we believe this work can encourage others to continue to explore the Options Market and possible MoE applications.

5.2 Limitations

This work is not without limitation. The gathered data is from 2011 to 2017, and it would be interesting to analyse real-time data. Also, similarly to the literature, it was considered a mid-point price between Ask and Bid, instead of creating two isolated Classifiers. In case of applying this strategy on a non-commission-free broker, the commission rate should also be considered.

5.3 Future Work

At the end of this thesis, many decisions were made, and it is inevitable to think what results would be obtained, if something was made different. With such results, we also envision more iterations to make the project even more interesting. On the other hand, we also understand possible flaws that if corrected could possibly improve our solution. Therefore, this section is divided in three different lists: challenges to overcome, possible variations and incremental work.

5.3.1 Challenges to overcome

- Increase the amount of Options data to cover all kind of trends in the training set.
- Improve time complexity by reducing the amount of hyperparameters. This may happen by finding the optimal value for some hyperparameters, or assume one based on previous literature.
- Improve grading rules for each TI. If the rules are not properly made, our algorithm could lose quality information, or even be induced in a biases imposed by them.
- Optimize cash allocation for each investment. As mentioned in Section 3.5.5.G our Trading Simulator is allocating more money on the begin as it has more cash and less open positions.

5.3.2 Possible Variations

- Using Regression instead of Classification. Possibly combine both type of experts and create a gating function that manages both kind of predictions.
- Select Options with time to expiration higher than six months, instead of with less than six months.
- Use Genetic Algorithm for tuning Hyperparameters.

- Adopt a S&H strategy with the classifier only detecting when to switch for a B&H strategy.
- Instead of defining a Close price for Options, define a multi-label prediction with Bid and Ask price individually, If using regressions, one could use two regressors, one for each price.

5.3.3 Incremental Work

5.3.3.A Incremental work on the Data Layer

- Automaticate the extraction of the assets data from the data sources. For common assets, it would be possible to use the python library *yfinance* to extract the data from Yahoo Finance [65], however Options would require a paid service with an available API.
- Create a labeling function that would also consider the asset price. For example, we could have the labels ”Sell”, ”Hold”, ”Buy” and ”Exercise” for trading the Option and also exercising it.

5.3.3.B Incremental work on the Logic Layer

- Design a Scoring Functions to be used in the Grid Search, balancing Accuracy, ROR and RRR.
- Generate more gating functions. Some of the functions used in other MoE projects are WTA, Inverse Distance, Softmax and Gaussian Transformations.
- The current architecture only buys and sells the Options, so it would be interesting to have the Trading Simulator also considering exercising those Option.
- We noticed, that the highest losses of our algorithms overlap the highest values of the VIX. Although wise, creating a Stop Loss based on the VIX would contradict our intention to study its impact as a feature. Nevertheless, it may be a good increment for next theses.
- Implement advanced Option strategies such as Options Spreads.

5.3.3.C Incremental work on the Presentation Layer

- Improve the current User Interface (UI) from Jupyter Notebooks to a customizable Dashboard. A good framework to build ML and Data Science web apps is the *Dash* framework which is based in the already implemented *Plotly*. Other Options in Python may include the framework *Django*.
- Create two Application Programming Interfaces (APIs) using the *Flask* framework. One for forecasting, and other to simulate Trading.
- Join all the user commands into the Dashboard, with all the configurations from the configuration file available to be change and tweaks using interactive filters, selectors, tabs, and menus.

Bibliography

[1] Z. Bodie, A. Kane, and A. J. Marcus, *Essentials of Investments 9th Edition*. McGraw-Hill, 2012.

[2] A. Gabriela ğiĠan, "The efficient market hypothesis: Review of specialized literature and empirical research," *Procedia Economics and Finance*, vol. 32, pp. 442–449, 2015.

[3] J. J. Murphy, *Technical Analysis of the Futures Markets: A Comprehensive Guide to Trading Methods and Applications, New York Institute of Finance*. Prentice-Hall, 1999.

[4] J. M. Pinto, R. F. Neves, and N. Horta, "Boosting trading strategies performance using vix indicator together with a dual-objective evolutionary computation optimizer," *Expert Systems with Applications*, vol. 42, no. 19, pp. 6699–6716, 2015.

[5] R. Rosillo, J. Giner, and D. de la Fuente, "The effectiveness of the combined use of vix and support vector machines on the prediction of s&p 500," *Neural Computing and Applications*, vol. 25, no. 2, pp. 321–332, 2014.

[6] M. Maragoudakis and D. Serpanos, "Towards stock market data mining using enriched random forests from textual resources and technical indicators," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2010, pp. 278–286.

[7] C. Krauss, X. A. Do, and N. Huck, "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500," *European Journal of Operational Research*, vol. 259, no. 2, pp. 689–702, 2017.

[8] A. Booth, E. Gerding, and F. Mcgroarty, "Automated trading with performance weighted random forests and seasonality," *Expert Systems with Applications*, vol. 41, no. 8, pp. 3651–3661, 2014.

[9] M. Ballings, D. Van den Poel, N. Hespeels, and R. Gryp, "Evaluating multiple classifiers for stock price direction prediction," *Expert Systems with Applications*, vol. 42, no. 20, pp. 7046–7056, 2015.

[10] Y. M. Cheung, W. M. Leung, and L. Xu, "Application of mixture of experts model to financial time series forecasting," in *Proc. Int. Conf. Neural Netw. Signal Process*, 1995, pp. 1–4.

[11] M. Buffett and D. Clark, *Buffettology: The Previously Unexplained Techniques that Have Made Warren Buffett the World's Most Famous Investor*. Simon and Schuster, 1997.

[12] B. M. Barber, Y.-T. Lee, Y.-J. Liu, and T. Odean, "Do individual day traders make money? evidence from taiwan," *University of California, Berkeley, working paper*, 2004.

[13] R. Dalio, *Principles: Life and Work*. Simon and Schuster, 2017.

[14] G. Kou, X. Chao, Y. Peng, F. E. Alsaadi, and E. Herrera-Viedma, "Machine learning methods for systemic risk analysis in financial sectors," *Technological and Economic Development of Economy*, vol. 25, no. 5, pp. 716–742, 2019.

[15] M. S. Yumlu, F. S. Gurgen, and N. Okay, "Financial time series prediction using mixture of experts," in *International Symposium on Computer and Information Sciences*. Springer, 2003, pp. 553–560.

[16] B. Graham, W. E. Buffett, and J. Zweig, *The intelligent investor: the definitive book on value investing*. Harper Collins Publishers, 2015.

[17] A. Gorgulho, R. Neves, and N. Horta, "Applying a ga kernel on optimizing technical analysis rules for stock picking and portfolio composition," *Expert systems with Applications*, vol. 38, no. 11, pp. 14 072–14 085, 2011.

[18] R. D. Edwards, J. Magee, and W. C. Bassetti, *Technical analysis of stock trends*. CRC press, 2007.

[19] 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.

[20] P. A. Fisher, *Common stocks and uncommon profits and other writings*. John Wiley & Sons, 2003, vol. 40.

[21] O. Jangmin, J. Lee, J. W. Lee, and B.-T. Zhang, "Adaptive stock trading with dynamic asset allocation using reinforcement learning," *Information Sciences*, vol. 176, no. 15, pp. 2121–2147, 2006.

[22] J. W. Wilder, *New concepts in technical trading systems*. Trend Research, 1978.

[23] M. Kumar and M. Thenmozhi, "Forecasting stock index movement: A comparison of support vector machines and random forest," in *Indian institute of capital markets 9th capital markets conference paper*, 2006.

[24] T. M. Mitchell *et al.*, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997.

[25] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[26] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 2011.

[27] W.-Y. Loh, “Classification and regression trees,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[28] C.-J. Huang, D.-X. Yang, and Y.-T. Chuang, “Application of wrapper approach and composite classifier to the stock trend prediction,” *Expert Systems with Applications*, vol. 34, no. 4, pp. 2870–2878, 2008.

[29] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[30] P. E. Utgoff, “Id5: an incremental id3,” in *Machine Learning Proceedings 1988*. Elsevier, 1988, pp. 107–120.

[31] T. Waheed, R. Bonnell, S. O. Prasher, and E. Paulet, “Measuring performance in precision agriculture: Cart—a decision tree approach,” *Agricultural water management*, vol. 84, no. 1-2, pp. 173–185, 2006.

[32] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[33] D. Opitz and R. Maclin, “Popular ensemble methods: An empirical study,” *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[34] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[35] T. K. Ho, “Random decision forests,” in *Document analysis and recognition, 1995., proceedings of the third international conference on*, vol. 1. IEEE, 1995, pp. 278–282.

[36] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, “Random forests and decision trees,” *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 272, 2012.

[37] T. Hastie, R. Tibshirani, and J. Friedman, “The elements of statistical learning, ser,” 2001.

[38] T. Zhao, Q. Chen, Z. Kuang, J. Yu, W. Zhang, and J. Fan, “Deep mixture of diverse experts for large-scale visual recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 5, pp. 1072–1087, 2018.

[39] M. I. Jordan and R. A. Jacobs, “Hierarchical mixtures of experts and the em algorithm,” *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[40] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” *arXiv preprint arXiv:1701.06538*, 2017.

[41] A. S. Weigend and S. Shi, "Predicting daily probability distributions of s&p500 returns," *Journal of Forecasting*, vol. 19, no. 4, pp. 375–392, 2000.

[42] A. L. Coelho, C. Lima, and F. J. Von Zuben, "Hybrid genetic training of gated mixtures of experts for nonlinear time series forecasting," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 5. IEEE, 2003, pp. 4625–4630.

[43] M. Versace, R. Bhatt, O. Hinds, and M. Shiffer, "Predicting the exchange traded fund dia with a combination of genetic algorithms and neural networks," *Expert systems with applications*, vol. 27, no. 3, pp. 417–425, 2004.

[44] R. Ebrahimpour, H. Nikoo, S. Masoudnia, M. R. Yousefi, and M. S. Ghaemi, "Mixture of mlp-experts for trend forecasting of time series: A case study of the tehran stock exchange," *International Journal of Forecasting*, vol. 27, no. 3, pp. 804–816, 2011.

[45] B. LeBaron, "An evolutionary bootstrap method for selecting dynamic trading strategies," in *Decision Technologies for Computational Finance*. Springer, 1998, pp. 141–160.

[46] N. Towers and A. N. Burgess, "Implementing trading strategies for forecasting models," in *Y. Abu-Mostafa et al.(eds.) Computational finance. Proceedings of the Sixth International Conference on Computational Finance*, 1999, pp. 313–325.

[47] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[48] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of svms for very large scale problems," in *Advances in Neural Information Processing Systems*, 2002, pp. 633–640.

[49] V. Tresp, "Mixtures of gaussian processes," in *Advances in neural information processing systems*, 2001, pp. 654–660.

[50] M. P. Deisenroth and J. W. Ng, "Distributed gaussian processes," *arXiv preprint arXiv:1502.02843*, 2015.

[51] L. Theis and M. Bethge, "Generative image modeling using spatial lstms," in *Advances in Neural Information Processing Systems*, 2015, pp. 1927–1935.

[52] D. Eigen, M. Ranzato, and I. Sutskever, "Learning factored representations in a deep mixture of experts," *arXiv preprint arXiv:1312.4314*, 2013.

[53] C. E. Rasmussen and Z. Ghahramani, "Infinite mixtures of gaussian process experts," in *Advances in neural information processing systems*, 2002, pp. 881–888.

[54] R. Aljundi, P. Chakravarty, and T. Tuytelaars, “Expert gate: Lifelong learning with a network of experts.” in *CVPR*, 2017, pp. 7120–7129.

[55] M. I. Jordan and R. A. Jacobs, “Hierarchies of adaptive experts,” in *Advances in neural information processing systems*, 1992, pp. 985–992.

[56] B. Yao, D. Walther, D. Beck, and L. Fei-Fei, “Hierarchical mixture of classification experts uncovers interactions between brain regions,” in *Advances in Neural Information Processing Systems*, 2009, pp. 2178–2186.

[57] S. R. Waterhouse and A. J. Robinson, “Constructive algorithms for hierarchical mixtures of experts,” in *Advances in neural information processing systems*, 1996, pp. 584–590.

[58] S. R. Waterhouse, D. MacKay, and A. J. Robinson, “Bayesian methods for mixtures of experts,” in *Advances in neural information processing systems*, 1996, pp. 351–357.

[59] C. M. Bishop and M. Svenskn, “Bayesian hierarchical mixtures of experts,” in *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 57–64.

[60] E. Garmash and C. Monz, “Ensemble learning for multi-source neural machine translation,” in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 1409–1418.

[61] X. Liang, H. Zhang, J. Xiao, and Y. Chen, “Improving option price forecasts with neural networks and support vector regressions,” *Neurocomputing*, vol. 72, no. 13-15, pp. 3055–3065, 2009.

[62] A.-P. Chen and Y.-H. Chang, “Using extended classifier system to forecast s&p futures based on contrary sentiment indicators,” in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3. IEEE, 2005, pp. 2084–2090.

[63] Delta neutral. [Online]. Available: http://www.deltaneutral.com/

[64] Wall street journal article using delta neutral data. [Online]. Available: https://www.wsj.com/articles/SB10001424127887323300004578559333319695070

[65] Yahoo finance. [Online]. Available: https://finance.yahoo.com/quote/%5Espx/

[66] Fred. [Online]. Available: https://fred.stlouisfed.org/

[67] Top 10% economic institutions, as of december 2019. [Online]. Available: https://ideas.repec.org/top/top.inst.all.html

[68] Wrds - wharthon research data services. [Online]. Available: https://wrds-www.wharton.upenn.edu/

[69] M. Zeng, B. Zou, F. Wei, X. Liu, and L. Wang, “Effective prediction of three common diseases by combining smote with tomek links technique for imbalanced medical data,” in *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*. IEEE, 2016, pp. 225–228.

[70] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority over-sampling technique,” *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[71] L. St, S. Wold *et al.*, “Analysis of variance (anova),” *Chemometrics and intelligent laboratory systems*, vol. 6, no. 4, pp. 259–272, 1989.

[72] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi, “Modeling task relationships in multi-task learning with multi-gate mixture-of-experts,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1930–1939.

[73] Code - mixture of random forests experts in options portfolio management. [Online]. Available: https://github.com/rodrigolousada/Thesis

A

Configuration Variables

- **OPTION_CLOSE:** Column name to be use as the Close price of the option (e.g.: 'Ask_Bid_price' or 'Last')
- **SYMBOLS:** List of UnderlyingSymbols that should be included for the algorithm (e.g.: 'SPX', 'AAPL', etc.)
- **TYPES:** Available option types to consider (e.g.: 'put' or/and 'call')
- **BENCHMARK_FILE:** Benchmark file for a B&H strategy (e.g.: file for SPX)
- **TIME_TO_EXPIRATION:** Number of days to expiration that we want to consider the option for investment. (e.g.: 1, 2, 5, 10, 183, ..)
- **OPTION_FILTER_METHOD:** Option filter method described in section. 3.4.6 (e.g.: 'FIXED', 'SLIDING')
- **SLIDING_FREQ:** Time frame for each window of option selection. (e.g.: '1M', '6M', '1Y')
- **NLARGEST_OPTIONS:** Number of options to be collected in each time frame. (e.g.: 10, 40, ...)
- **PRICE_THRESHOLD:** Minimum threshold for mean price of an option if filter method is 'SLIDING' (e.g. 100, ...)
- **MAX_DURATION:** Number of days before expiration that we intend to start trading an option. (e.g.: 183, ...)
- **VOLUME_THRESHOLD:** Minimum threshold for option Volume, if filter method is 'FIXED' (e.g. 100, ...)

- **LABELLING_METHOD:** Method to use for class generation. Available two method, the first labels based on Option prices direction, and the second one on the Symbol prices direction. (e.g. 'SINGLE_OPTION', 'SINGLE_STOCK')
- **FUTURE_PERIOD_PREDICT:** Number of days ahead that we are trying to predict.(e.g. 1, 2, 5, 10, 20, ...)
- **IGNORE_RANGE_UP:** Percentage of positive change that we ignore, considering a "Hold" move. (e.g. 0.0002, 0.002, ...)
- **IGNORE_RANGE_DOWN:** Percentage of positive change that we ignore, considering a "Hold" move. (e.g. 0002, 0.002, ...)
- **TRADING_GRID_SEARCH:** Switch of Grid Search for Trading hyperparameters. (e.g. True or False)
- **TRADING_GRID_SCORING:** Name of the evaluation metric to use in the Trading Simulator's Grid Search (e.g. 'ror', 'roi, 'sharpe', ...)
- **SINGLE_STOCK:** Debug parameter. If List with Option Roots, only those specified options will be considered in the Trading Simulation. If False, parameter doesn't have impact. (e.g. False, ['SPX150117C01000000', 'SPX150918P01900000', 'SPX150918C02150000'], ...)
- **LONG_BLOCKED:** Boolean to activate prohibition on Long positions. If activated and the Trading Simulator decides to take a long position, this block will cancel that order. (e.g. True or False)
- **SHORT_BLOCKED:** Boolean to activate prohibition on Short positions. If activated and the Trading Simulator decides to take a short position, this block will cancel that order. (e.g. True or False)
- **TRADING_WINDOW:** Time frame for new decision on positions to adopt. Although the Tradings Simulator will only reconsider the positions every X days, the Stop Loss and Take Profit orders still have a daily time frame. (e.g. 1, 2, 5, 10, ...)
- **HOLDING_METHOD:** (e.g. 'Individual' or 'Continuous')
- **HOLDING_LONG_PERC:** Percentage of available cash used in the next Long positions.
- **HOLDING_SHORT_PERC:** Percentage of available cash used in the next Short positions.
- **TRADING_METHOD:** Investment strategy for decisions regarding the positions to assume on each options. In case of a BuyHold, the trader assumes the decision of buying all, for a SellHold the trader chooses to sell all new positions. The trader can also follow the decisions of the prediction, or invert those decisions. (e.g. 'BuyHold', 'SellHold', 'Target Predicted', 'Delayed Target Predicted', 'Target Predicte Inverse' or 'Delayed Target Predicte Inverse')
- **STOPLOSS_METHOD:** List of Stop Loss strategies used in the trading process (e.g. [], and combinations with 'Only My Money', 'Daily Returns', 'Percentage Peaks', and 'Chandelier Exit')
- **STOPLOSS_PERC:** Percentage of loss to activate the 'Daily Returns' Stop Loss (e.g. None, -0.5, -0.2, ...)
- **STOPLOSS_PERC_PEAKS:** Percentage of loss, from the maximum evaluation of a position, to activate the 'Percentage Peaks' Stop Loss (e.g. -0.8, -0.5, -0.2, ...)
- **STOPLOSS_PENALTY:** Number of penalty days for an option which activated a Stop Loss order. (e.g. 1, 2, 3, 5, infinite)

- **STOPLOSS_PRICE_THRESHOLD:** Minimum price allowed for trading a specific option. (e.g. 0, 2, 5, ...)
- **CHANDELIER_LONG_PERIOD:** Number used as Chandelier period for Long positions. (e.g. 13, 22, ...)
- **CHANDELIER_SHORT_PERIOD:** Number used as Chandelier period for Short positions. (e.g. 13, 22, ...)
- **CHANDELIER_MULTIPLIER:** Number used as multiplier in both Chandelier indicators (e.g. 3, 5, ...)
- **TAKEPROFIT_METHOD:** List of Take Profit strategies used in the trading process (e.g. [], and combinations with 'Max Profit for Option' and 'Percentage over Returns')
- **TAKEPROFIT_PERC_ABSOLUTE:** Percentage of returns that triggers the Take Profit of the abosulte value. (e.g. 0.9, 1, 2, ...)
- **TAKEPROFIT_PERC:** Percentage of returns that triggers the Take Profit which takes only part of the position. (e.g. 0.2, 0.5, 0.8, ...)
- **TAKEPROFIT_PERC_TAKE:** Percentage of the position that is taken on the partial Take Profit. (e.g. 0.1, 0.2, 0.5)
- **NR_SPLITS:** Number of Splits in the Cross-Validation period. (e.g. 3, 5, ...)
- **TEST_SIZE:** Percentage of data correspondent to the test set on the 2011-2015 period. (e.g. 0.1, 0.2, 0.3, ...)
- **periods:** List of periods to be used while computing the TIs
- **ALGO_GRID_SEARCH:** Boolean to activate Grid Search for the algorithm and preprocessing steps hyper-parmeters. (e.g. True or False)
- **N_JOBS:** Number of virtual processors to use in parallel while running the code.
- **SMOTE_RSEED:** Random seed for SMOTE algorithm.
- **PCA_VARIANCE:** Minimum variance to be maintained while computing the PCA.
- **PCA_N_COMPONENTS:** Number of Components left by the PCA.
- **RSEED:** Random Seed for the RaF
- **NR_ESTIMATORS:** Number of estimators, in this case DTs to constitue a RaF
- **CRITERION:** Measure of statistical dispersion used to select which feature will used to be added to the DT (e.g. 'gini', 'entropy')
- **MAX_FEATURES:** Maximum number of features to be consider in each DT split. It may also be a string that represent a calculation based on the number of features available. (e.g. 'auto', 'sqrt', 'log2', 10)
- **MAX_DEPTH:** Maximum depth of DT
- **MIN_SAMPLES_SPLIT:** Minimum number of samples to occur a split inside the DT
- **MIN_SAMPLES_LEAF:** Minimum number of samples to consider a lead node in the DT
- **MAX_LEAF_NODES:** Maximum number of leaf nodes in the DT.
- **NR_RFS:** Number of RaF experts to be used in the MoE
- **GATING_ARGS:** Name of the gating function to be used (e.g. 'Equal' or 'BinSplit')
- **SPLIT_COLUMN:** Index of column to be used in the gating function BinSplit for spliting the data.

B

Technical Indicators Specification

B.1 Trend Following

Simple Moving Average (SMA): This is one of the most simple TIs as illustrated in Figure B.1. It is generally applied as it smooths the price's movement [3]. The SMA is calculated by the arithmetic average of stocks' past closing price over a defined number of time periods. In function B.1, the variable t represents the current time period, while n represents the number of time periods to be considered. The higher n is, the farther in the past will the sum of closing prices go.

$$SMA(n)_t = \frac{\sum_{i=t-n}^{t} \text{close}_i}{n} \tag{B.1}$$

If the n is low, it means that the recent prices will have more influence. Therefore those averages will respond quickly to changes in price. On the other hand, long-term averages, with higher values of n, are slow to respond to changes in price. This relation reflects in a smoother line for higher values of n.

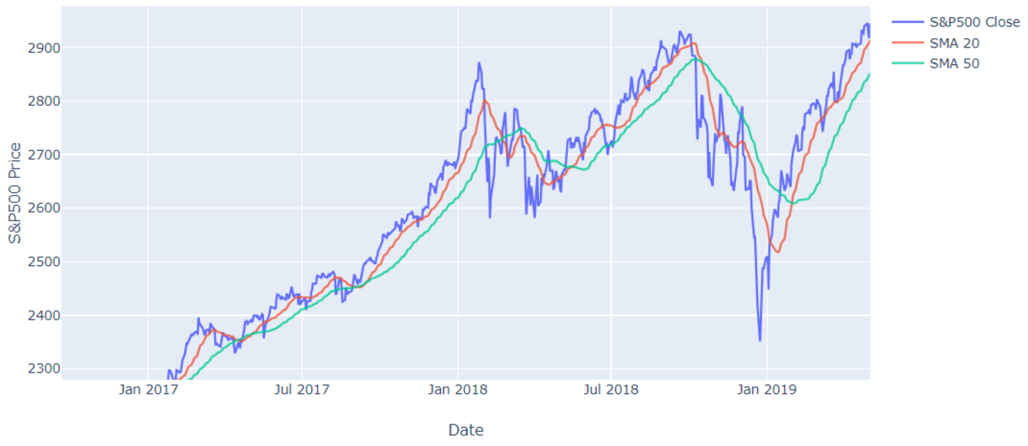


Figure B.1: SMA(20) and SMA(50) over the S&P500

Exponential Moving Average (EMA): While the SMA gives equal weights for any time period it considers, the EMA computes the average exponentially assigning larger weights to newer data, as illustrated in Figure B.2. In equation B.2, n represents the number of time periods to be considered, while α represents the degree of weight to be attributed to the time periods.

$$\begin{gathered} EMA(n)_t = [EMA(n)_{t-1} \times (1-\alpha)] \times (\text{close}_t \times \alpha) \\ \text{with } \alpha = \frac{2}{n+1} \end{gathered} \tag{B.2}$$

It is possible to see that α is dependent on n. Recent prices will have more weight than old prices due to the recursion of the EMA function.

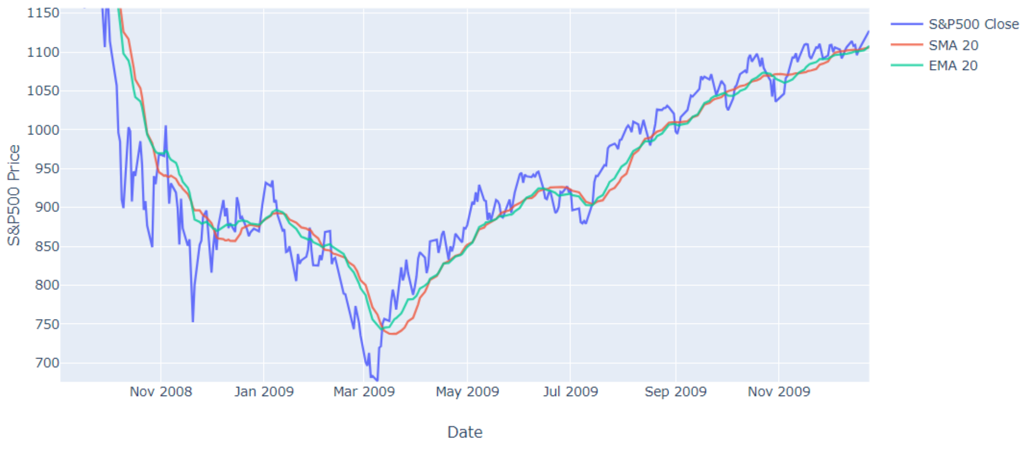


Figure B.2: SMA(20) and EMA(20) over the S&P500

B.2 Momentum Oscilators

Momentum (MOM): This indicator (see Figure B.3) makes the comparison between where the current price is in relation to where the price was. The variable n controls how far in the past is this comparison. If the current price is above the previous price, the indicator is positive. In opposition, if the current price is below the previous price, the indicator is negative. This simple indicator can be calculated by the equation B.3.

$$Momentum_t(n) = \text{close}_t - \text{close}_{t-n} \tag{B.3}$$

Higher values of n allow making a long-term analysis, while lower values of n permit short-term evaluation. This indicator can be a simple solution to detect trends. The market will be in an uptrend, downtrend or sideways if the indicator is positive, negative or zero, respectively.

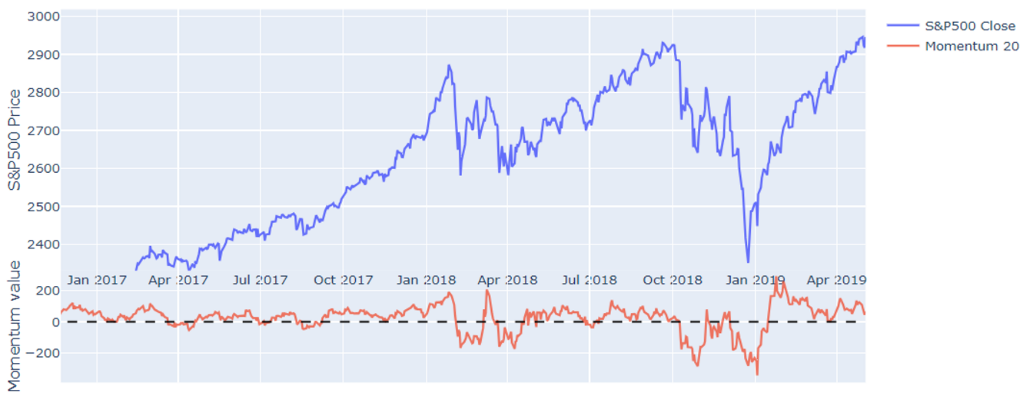


Figure B.3: MOM(20) over the S&P500

Relative Strength Index (RSI): The RSI, illustrated in Figure B.4, aims to measure the speed and magnitude of an asset's direction price movement, to determine overbought or oversold areas. It determines those areas by comparing the importance of the asset's recent gains and losses [3].

Its values range from 0 to 100. Values below 30 are considered oversold levels, while a value above 70 indicates overbought levels. The indicator is calculated by the equations B.4

$$Upward = \begin{cases} Close_t - Open_t & \text{if } Close_t - Close_{t-1} > 0 \\ 0 & otherwise \end{cases} \tag{B.4a}$$

$$Downward = \begin{cases} Open_t - Close_t & \text{if } Close_t - Close_{t-1} < 0 \\ 0 & otherwise \end{cases} \tag{B.4b}$$

$$RS = \frac{EMA_n(Upward)}{EMA_n(Downward)} \tag{B.4c}$$

$$RSI = \begin{cases} 100 - \frac{100}{1+RS} & \text{if } EMA_n(Downward) \neq 0 \\ 100 & \text{if } EMA_n(Downward) = 0 \end{cases} \tag{B.4d}$$

$Upward$ and $Downward$ represent the changes which are calculated every trading period. When the market is on an uptrend, the value of $Downward$ will be 0. But, if the market is on a downtrend, that variable will be calculated as mentioned in equation B.4b.

Since the RSI helps to determine overbought and oversold conditions, it will be more useful on Options with high liquidity.

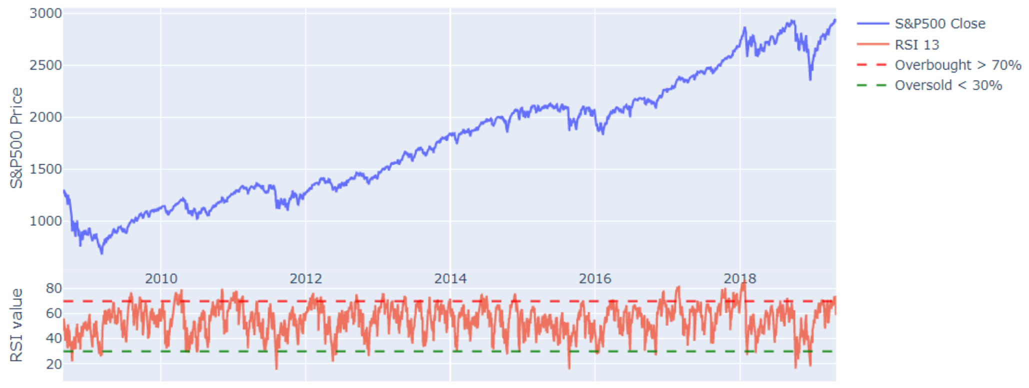


Figure B.4: RSI(13) over the S&P500

Intraday Momentum Index (IMI): This indicator is a TI suited for high-frequency trades. It employs the notions of intraday candlesticks and RSI, providing a suitable range for high-frequency trading by showing overbought and oversold levels. The difference between IMI and RSI while calculating the $Upward$ and $Downward$, is that RSI uses the difference between the prevailing $Close$ price and the $Close$ price of the previous time period, while IMI applies the difference between the $Close$ and the $Open$ price of the current time period.

$$Upward = \begin{cases} Close_t - Open_t & \text{if } Close_t - Open_t > 0 \\ 0 & otherwise \end{cases} \tag{B.5a}$$

$$Downward = \begin{cases} Open_t - Close_t & \text{if } Close_t - Open_t < 0 \\ 0 & otherwise \end{cases} \tag{B.5b}$$

$$IMI(n) = \frac{\sum_{i=1}^{n} \text{Upward}}{(\sum_{i=1}^{n} Upward + \sum_{i=1}^{n} \text{Downward})} \times 100 \tag{B.5c}$$

The IMI can be calculated by function B.5c. It is the sum of the $Upwards$ divided by the sum of $Upwards$ and $Downwards$. After this, the value is multiplied by 100, so it ranges from 0 to 100. The variable n, which is the number of days to look in the past, is generally set to 14. Such as the RSI, if the final value is above 70, the asset is estimated overbought. On the other hand, the asset is considered oversold, if the resulting number is less than 30.

B.3 Volatily Indicators

Average True Range (ATR): The ATR [22], illustrated in Figure B.5, is an indicator which measures volatility, by rotting the complete range of the asset's price for that period of time. It is the EMA of the $TrueRange_t$ as described in equation B.6b. The value of n, which represents the number of time periods that are going to be taken into consideration, is generally set to 14 days.

$$\text{True Range}_t = \max\left(High_t, Close_{t-1}\right) - \min\left(Low_t, Close_{t-1}\right) \tag{B.6a}$$

$$ATR(n)_t = EMA_n\left(\text{True Range}_t\right) \tag{B.6b}$$

This $TrueRange$ can be calculated using the equation B.6a. It represents the maximum variation since the close of the last time period, so it will be the difference between the $High$ price and the Low price of this time period if the $Close$ price of the last period is in that range. If the $Close_{t-1}$ is lower than Low_t it will be the difference between $High_t$ and $Close_{t-1}$. However, if the $Close_{t-1}$ is higher than $High_t$ it will be the difference between Low_t and $Close_{t-1}$.

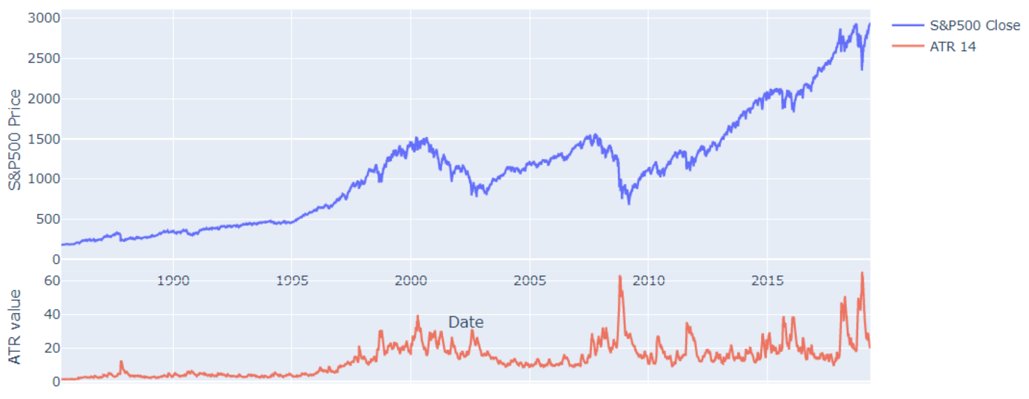


Figure B.5: ATRs(14) over the S&P500

Bollinger Bands (BBANDs): The BBANDs are considered the most famous TI to measure volatility. This indicator is composed of three lines or bands, as illustrated in Figure B.6. The middle band, which goes along the price, is an SMA (equation B.1). The two other bands are placed two standard deviations (σ) above and below the middle band. The $Upper$ and $Lower$ bands expand and shrink as the volatility of the asset rises or falls, respectively since σ describes how prices are disbanded around an average [3].

Typically the value n, which represents the number of time periods t to be considered, is 20 days.

$$\text{Middle band } = SMA_t(n) \tag{B.7a}$$

$$\text{Upper band } = SMA_t(n) + 2 \times \sigma_t(n) \tag{B.7b}$$

$$\text{Lower band } = SMA_t(n) - 2 \times \sigma_t(n) \tag{B.7c}$$

Prices above the $Upperband$ can be considered too high, and prices below the $Lowerband$ would be too inexpensive. If this happens, as the value of $2 * \sigma$ covers 95% of the price data, it can be a signal that the price is going to reverse back inside the limits of the BBANDs. In options trading, if the price goes below the $Lowerband$ can be a signal to take a Long position on a Call Option or a Short position on a Put Option. On the other hand, if the price goes above the $Upperband$ should be approached by short-selling a Call Option, or long-selling a Put Option.

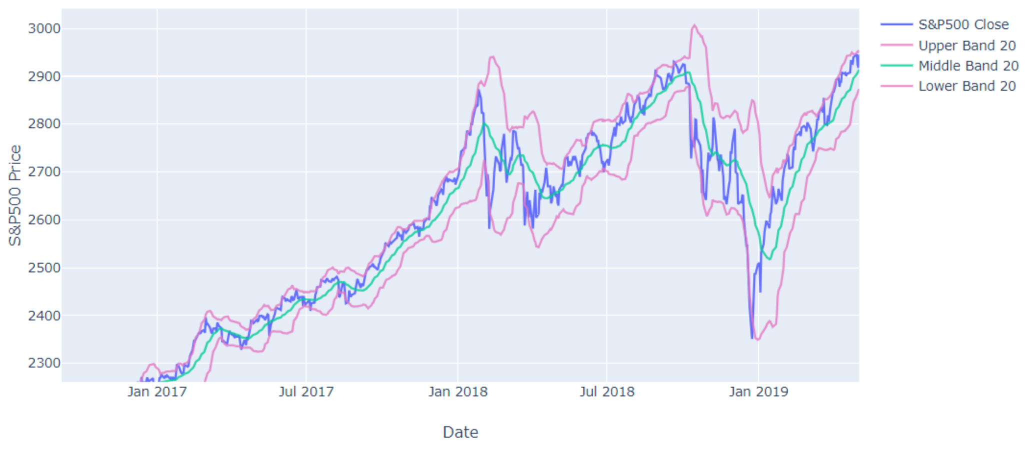


Figure B.6: BBANDs(20) over the S&P500

B.4 Volume Indicators

Put-Call Ratio (PCR): The PCR compares the trading volume between Put Options and Call Options. From this ratio, it is possible to make conclusions from the overall market sentiment.

$$\text{Put/Call Ratio } = \frac{\text{Put Volume}}{\text{Call Volume}} \tag{B.8}$$

When the indicator's value is above 1, it means that are more volume in Put Options. If people are trading more Puts, it represents a downtrend in the market, as people try to buy insurance for their Short position on the market. Contrarily, if the PCR is lower than 1, it means there is more Call volume, indicating an upcoming uptrend.

This indicator can also be seen as a contrarian indicator. A ”contrarian indicator” helps investors to contradict the market, taking the exact opposite action of the majority of people. Traders who use this investment strategy expect to predict market turns in early stages, due to loss of purchasing or selling power from the market.

Open Interest (OI): Represents the entire number of open contracts that exist in a determined period of time. These contracts can be options or futures. If a new contract is held between two parties, the open interest is incremented. If a contract expires and both parties exit their position, open interest decreases.

OI and Volume are confused a lot. The difference is when exists a transaction between holders of an existing contracted. In this case, the Volume value will be changed, while the OI value will not be affected.

It is possible to draw conclusions from the comparison between the price of the asset and the open interest as described in Table B.1.

Table B.1: Interpretations for Price and Open Interest directions

Price	Open Interest	Interpretation
Rising	Rising	Market is strong
Rising	Falling	Market is turning weak
Falling	Rising	Market is weak
Falling	Falling	Market is getting strong

Some Option traders use the Call Open Interest and Put Open Interest instead of volume when calculating the PCR [3], as in equation B.9.

$$\text{Put/Call OI Ratio} = \frac{\text{Put Open Interest}}{\text{Call Open Interest}} \tag{B.9}$$