Hybrid System Combining Artificial Neural Networks and Support Vector Machines for Trading the Forex Market Intraday

Vasco Amorim do Amaral

Thesis to obtain the Master of Science Degree in

Eletrical and Computer Engineering

Supervisor(s): Prof. Rui Fuentecilla Maia Ferreira Neves

January 2021

Acknowledgments

It is the greatest pleasure of all to be writing this message. My time at *Instituto Superior Técnico* has been beyond amazing, I could have never guessed I would have the opportunity to learn so much. Not only the technical subjects, but specially everything that is required to actually perform a great job whatever, whenever and wherever it is. At the institute I learned that behind every great job there must be an even greater knowledge, I learned how to, independently, research for and gather that knowledge. I improved a lot my capability of working as a team and make use of several points of view, and for all of this I can only be grateful to everyone at Instituto Superior Técnico that either directly or indirectly made it possible for me to have this amazing journey.

To my supervisor, Professor Rui Neves, I have to thank for giving me this amazing opportunity to join great interests of mine in a single project, that was with no doubt the most interesting and rewarding work I have developed so far.

To all my amazing friends that gave me the pleasure to share this journey with me, a big thank you, it would never have been so fun without you guys. A special word to Mathilda, who supported me all the way, being the most reliable friend I could have.

Lastly, but most importantly, to my sister, my mother and my father who gave me every opportunity in life anyone could ever wish for, I could not be more grateful.

Thank you to everyone that made sure they were a part of this,

Sincerely,

Vasco Amorim do Amaral

Abstract

This thesis proposes an intradaily Forex trading system combining a Support Vector Machine capable of identifying and classifying different types of markets, namely bullish, bearish and sideways and several Artificial Neural Networks, one for each market type, capable of finding intradaily price patterns and predict price movements from *12:00pm* till *4:30pm*. An incremental window moving average is applied on a price transformation of logarithmic return rates, creating the transformed data that is fed to the ANN as the features and target value, each sample representing a trading day. The SVM uses price sequence windows, of approximately three months, as features. Several strategies are proposed based on the forecasting done by the ANNs, in order to optimize the final performance of the trading system. The strategies proposed focus on the trading rules to enter the market, and in which direction, long or short, and also the level of confidence, which is represented in the trading size applied for each trade. The training was done with data from 2004 until 2018 and tested for the year of 2019. The final optimized system achieved a return on investment of 87.5% over the testing year and a maximum drawdown of 13%, largely overperforming the comparison methods of *Buy & Hold* and *Sell & Hold*.

Keywords: Artificial Neural Networks, Data Processing, Forex Market, Intraday Trading, Support Vector Machine

Resumo

Esta tese propõe um sistema de negociação de Forex intra-diário composto por uma máquina de vetores de suporte capaz de identificar e classificar diferentes tipos de mercado, nomeadamente *bullish*, *bearish* e *sideways* e várias redes neuronais artificiais, uma para cada tipo de mercado, capazes de encontrar padrões de preço intradiários e prever os movimentos do mesmo desde as *12:00* horas até às *16:30*. Uma média móvel de janela incremental é aplicada numa transformação de preço de taxas de retorno logaritmicas, criando assim a transformação de dados que é fornecida à ANN como a entrada do sistema e os valores a prever, cada amostra representa um dia. A SVM recorre a sequências de preço em janelas de aproximadamente três meses, para usar como dados de entrada. Várias estratégias, baseadas na previsão das redes neuronais, são propostas de modo a optimizar o desempenho final do sistema de negociação. As estratégias propostas são focadas nas regras para entrar ou não no mercado, e em que direção, comprar ou vender, e também no nível de confiança para cada previsão, representado pelo volume usado em cada transação. O treino foi feito com dados desde 2004 até 2018 e testado no ano de 2019. O sistema final optimizado alcançou um retorno no investimento de 87.5% ao longo do ano de teste e um *drawdown* de 13%, resultados largamente melhores que os métodos de comparação, *Buy & Hold* and *Sell & Hold*.

Palavras-chave: Máquina de Vetores de Suporte, Mercado Forex, Processamento de Dados, Redes Neuronais Artificiais, Trading Intradiário

Contents

List of Tables

List of Figures

List of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
B&H	Buy and Hold
BB	Bollinger Bands
EANN	Evolutionary Artificial Neural Network
EMA	Exponential Moving Average
EUR/GBP	Currency Pair - Euro/Pound Sterling
EUR/USD	Currency Pair - Euro/United States Dollar
FNN	Fuzzy Neural Network
Forex/FX	Foreign Exchange Market
GA	Genetic Algorithm
GBP/USD	Currency Pair - Pound Sterling/United States Dollar
IWMA	Incremental Window Moving Average
LSTM	Long short-term Memory
MACD	Moving Average Convergence Divergence
MFI	Money Flow Index
ML	Machine Learning
MLP	Multi-Layer Perceptron
MOO	Multi-objective Optimization
OBV	On-Balance Volume
ReLU	Rectified Linear Units
RMSE	Root Mean Squared Error

RNN	Recurrent Neural Network
ROC	Rate of Change
ROI	Return on Investment
RSI	Relative Strength Index
S&H	Sell and Hold
SMA	Simple Moving Average
SVM	Support Vector Machine
TSS	Total Sum of Squares
USD/JPY	Currency Pair - United States Dollar/Japanese Yen

Chapter 1

Introduction

This first chapter explains the motivation behind this thesis and its main objectives, as well as the contributions made by this study. Lastly, the structure of this document is given alongside a brief description of each chapter's content.

1.1 Motivation

The foreign exchange market is a global market to trade currencies, shared by governments, banks, hedge funds and all kind of traders. It is, by far, the most traded market, as it's volume goes up to 6 trillion dollars per day [1], and respects everyone that has some kind of money in the world. One does not need to be in the financial sector to make use of this market. Either for working purposes, receiving the salary in a different country, for international holidays purpose, to trade cash for the countries currency or even to invest in stocks from a different country, one will need to trade its current money for a different currency, and therefore access the Forex market. The only rule in any financial market is that if the search is bigger than the offer the price will rise and vice-versa, so every time someone accesses a market to trade in it, is actually influencing the course of it.

The other subject on this thesis is Artificial Intelligence, particularly Artificial Neural Networks, which are algorithms that resemble a computational network inspired in biological neural systems, as Walczak and Cerpa [2] said, these models simulate the electrical activity of the brain and nervous system. This are very powerful tools, capable of recognizing and learning patterns from big amounts of data, that a human being is simply not capable of, as we have limited memory and processing power, when compared to a super computer.

So, in my opinion, it is rather amazing to be making use of computers that simulate the human brain, in order to make predictions on a market shared and influenced by every single person and entity on this planet.

1.2 Objectives

The main objective of this work is to implement a complex trading system for the Forex pair *EUR/USD*, capable of forecasting price movements on a specific time horizon and intradaily trade the market in a profitable way.

The system will be based on Artificial Intelligence. A Support Vector Machine will be trained to identify different market conditions and an Artificial Neural Network will be trained, for each set of market conditions, in order to identify price movement patterns and forecast these.

Data regarding the Forex pair *EUR/USD* from 2004 until 2018 will be used for this purpose and the final system will be implemented as if it was built and validated before 2019, and tested during this year on a real time simulation, so the results presented would actually be the ones obtained if one trusted the system to invest its money in the beginning of 2019.

1.3 Contributions

Following are the main contributions presented by this work:

- A system combining ANNs and an SVM, empowered by a data pre-processing module and a final strategy layer, capable of optimizing each other in order to obtain profit in the Forex market.
- The identification of the best time windows to trade the Forex pair *EUR/USD*, intradaily.
- A Support Vector Machine capable of identifying and classifying different types of markets, regarding the behaviour of these.
- The implementation of three Artificial Neural Networks capable of finding intradaily patterns and predict the price movement based on these patterns, until the time of the prediction, and the current type of market.

1.4 Document Structure

This work is presented over five chapters, as described below:

1. **Introduction** - In this chapter the main motivations and goals of this work are described, along the study's main contributions.
2. **Background & State-of-the-Art** - In this chapter the fundamental concepts behind Forex and Machine Learning, required to understand this work, are given. On a second moment, the research done over existing solutions for relevant and similar problems is resumed and presented.
3. **Proposed Architecture** - In this chapter the overall system architecture is described, along all the steps required for the implementation and validation of this.

4. **System Evaluation** - In this chapter the methodologies used to evaluate the system are described, along with three study cases that focus on different features of this system. The results achieved by the final system are described and commented

5. **Conclusion & Future Works** - In this chapter a final overview of the work presented is done, alongside the conclusions taken about the results obtained. Future improvements that can be done on this work are also suggested.

Chapter 2

Background & State-of-the-Art

This chapter presents fundamental concepts about financial markets, as well as artificial intelligence, required to fully understand the topics later discussed. First, a description of the foreign exchange market, commonly used terms, technical analysis and investing strategies is given. Followed by a review of some machine learning algorithms and concepts, such as, *ANNs* and *SVMs*. Lastly, a review of the state-of-the-art on academic studies, on relevant areas for this work, is presented.

2.1 Financial Concepts

2.1.1 Foreign Exchange Market

The foreign exchange market, commonly known as *Forex*, or FX, is a global market to trade currencies, shared by governments, banks, hedge funds and all kind of traders. It is, by far, the most traded market, as it's volume goes up to 6 trillion dollars per day [1]. Unlike stock markets, which are based in a specific place, therefore open and close everyday, the *Forex* market is distributed throughout several trading centers, London, New York, Tokyo and Sydney, being the biggest ones. This makes it possible for the market to work 24 hours a day from Monday until Friday.

Currencies are always traded in the form of a pair, called Cross Currency Pairs, as one always has to sell one currency to buy another, so the rate of the pair is the cost of one currency in the form of the other. Lets take EUR/USD as an example, if the rate is 1.1, this means it will cost 1 unit of the base currency, in this case the Euro, to buy 1.1 units of the counter currency, the US Dollar.

The smallest amount by which a currency quote can change is called a pip, short for percentage in point, it is usually equal to 0.0001, except for Japanese Yen related currency pairs, in which it is equal to 0.01.

There are two types of positions in the market, *long* and *short*. The first one corresponds to buying the first currency, when it is believed that the quote of the pair is going to rise. A short position is the opposite, in this case, one sells the base currency to buy the counter currency, when it is believed that the market is going to fall.

It is important to note that the price to buy or sell the base currency, at a given point in time, is actually different, this is the *ask* and *bid* price, respectively. This values vary by a few pips and this difference is called *spread*, which in most cases is considered the cost of a transaction, assuming there are no other fees. The *spread* also varies according to the liquidity of the market. The bigger it is, which means that a bigger volume of the asset, in this case a currency, is being traded, the smaller the difference, between the *ask* and *bid* price, will be.

2.1.2 Market Analysis

There are two possible ways to approach any kind of financial market, when it comes to research and forecast future trends and price movements, fundamental analysis and technical analysis. The first one is a method grounded on valuing a security or an asset by measuring its intrinsic value. Fundamental analysis studies everything that can affect the market, from the overall economy and industry conditions to the synergy between supply and demand, even political events can be taken into account by a fundamental analyst. On the other hand, technical analysis' core assumption is that any factor that truly influences the market will be factored into the volume of transactions and the securities prices [3]. Grounded on this assumption, technical analysts only look at this two inputs, focusing on its history in order to identify patterns and trends that suggest what will happen in the future. As Hirabayashi et al. [3] stated, ”technical indexes are tools to expect and analyze the change of the price in the future, using the change of pricing generated in the past”. The advocates of this approach defend that investors, as a whole, tend toward patterned behaviors, so price action tends to repeat itself. ”Therefore, technical analysis is a sound alternative to forecast short term forex market movements” [4].

Technical Indicators

Technical analysts have the need to obtain a different perspective, on price evolution, than what the price and volume charts, solely, can give. In order to do that, over the years, several technical indicators have been invented. A technical indicator consists of a formula applied to one or both of the 2 inputs mentioned before. The result can, than, be plotted alongside the existing price and volume charts [5]. The goal is to capture the behavior and feeling of the investors, in order to determine trends, volatility or if an asset is overbought or oversold.

There is no golden rule in the financial world, and technical indicators are no exception, all have their flaws and fail from time to time. In order to mitigate the risk of trusting your money in a mathematical transformation of price quotes, alone, technicians, generally, use a combination of technical indicators [5, 6]. When the right set of conditions lines up, than they can enter the market with less risk, targeting for bigger profits.

Indicators can be separated into groups according to what they try to measure - i.e., trend, momentum, volatility or volume. In table 2.1 it is presented a brief description and some examples for each one.

In table 2.2 is presented some of the most popular technical indicators, alongside its definition and

Table 2.1: Technical Indicator Types with Examples [7, 8]

	Description	Examples
Trend	Measure the direction and intensity of a trend	SMA, EMA, MACD, IWMA
Momentum	Measure the speed at which the price changes, instead of the actual price levels	RSI, ROC, Stochastic Oscillator
Volatility	Measure the stability at which a price moves, useful to identify direction changing movements	BB
Volume	Based on transactions volume data, can help confirm the strength of new trends or fake movements	OBV, MFI

usage for trading strategies.

Table 2.2: Technical Indicators Definition and Usage [7, 8]

Indicator	Formula	Definition and Usage
SMA - *Simple Moving Average*	$\frac{\sum_{i=1}^{n} X(i)}{n}$	Average value of the security's price X over a period of time n. Aims to eliminate the noise of random short-term fluctuations in price to identify trends. Buy signals are generated when the price rises above it's moving average or a faster moving average (shorter period) crosses above a slower one. Sell signals are the opposite.
EMA - *Exponential Moving Average*	$\mathrm{EMA}_t(n) = X_t * \frac{2}{n+1} +$ $\text{EMA}_{t-1}(n) * (1 - \frac{2}{n+1})$	Similar to the SMA, the difference of the EMA is that it weights recent data points with more significance, exponentially, therefore reacting quicker to recent price changes. The use of the EMA is based on the theory that new data better reflects the current trend of a security's price. In order to use it, the same technical rules of the SMA are applied.
IWMA - *Incremental Window Moving Average*	$\frac{\sum_{i=1}^{k} X(i)}{k} \forall k = 2, 3, ..., n$	Similar to the SMA, as it can be seen in formula. Difference is that it is applied on a specific time window, where instead of counting every last n values, starts at the beginning of the window and each time step averages one more value X.
RSI - *Relative Strength Index*	$100 - \frac{100}{1 + \frac{Averagegain}{Averageloss}}$	As an oscillator, RSI computes a value from 0 to 100, indicating if an asset is overbought (over 70) or oversold (below 30). It is known as a momentum indicator that measures the magnitude of recent price changes, typically over the last 14 days.
ROC - *Rate of Change*	$\frac{X_t - X_{t-n}}{X_{t-n}} * 100$	As the name implies, it is a ratio that measures the speed of the price change over a period of time n. It is great to identify momentum and trends, but also overbought or oversold conditions, depending on how fast the price of an asset is moving.

2.1.3 Leverage

Price variations in the market, specially in the foreign exchange market, are usually very small, not crossing 1% during intraday trading [6]. In order to profit bigger amounts of money on small investments, one will need to apply leverage on the said investment. Leverage is the use of borrowed capital for an investment to amplify its returns, whether they are positive or negative.

In the *Forex* context, leverage is provided to the investors by their brokers, companies that are specialized in financial trading [6]. The amount of leverage varies, usually, between 100:1 and 10:1, according to each broker and the risk associated to the investment itself. A leverage of 100:1 means that the investment is multiplied by 100, so if the investor gives 1,000€ in advance, the investment will be 100,000€.

As said before, leverage amplifies both profits and losses, making it a very risky tool which needs to be used with caution. Let's take the investment example given before, 1,000€ with 100:1 leverage. At this point the total amount invested is 100,000€, being that 99,000€ are owed to the broker. In case a valorization of 2% happens, the asset bought is now valued at 102,000€, so the investor now owns 3,000€, which represents a valorization of 200%, tripling the original investment. So instead of making a 20€ profit, without the use of leverage, the profit was 2,000€. The opposite scenario is also possible, if the asset falls 2% and is now valued at 98,000€, not only the investor loses all his initial money, but now owes 1,000€ to the broker, instead of losing just 20€, which represents a loss of 200%. This cannot happen on a "normal" investment since an asset cannot be valued under 0, so the risk is never greater than 100%. The way brokers found to mitigate the risk of investors losing more money than what they can afford, is the requirement that they have a specific margin, a percentage of the total amount of money invested, available in their account, also known as *margin trading*. This percentage varies according to the current value of the security bought, and margin calls can be made by the broker itself, in case it stops being respected - i.e., brokers can close negative positions when the investor is already losing money that is borrowed.

2.2 Machine Learning Concepts

2.2.1 Supervised and Unsupervised Learning

Every machine learning algorithm, as the names indicates, needs to learn its job, before executing it. There are two different methods for training an *AI* algorithm, called supervised and unsupervised learning. The first, and the only one used throughout this work, is the most popular one, used mostly for prediction and classification problems. Its definition is that the algorithm, during the training stage, has access to the correct answers it is supposed to classify or predict. It usually uses them in a system of try and error so that it gets closer to the optimal solution and ready to face real time problems where the correct answer is not know yet. Unsupervised learning is mostly used to find patterns where the humans cannot find them, or to group big amounts of data, referred to as clustering, in a meaningful way, the difference is that this is done without having a "correct answer" for the problem.

2.2.2 Features, Samples and Labels

The common thing about every single *ML* algorithm, both supervised and unsupervised, is that it works on top of some kind of input data, that carries the necessary information for the algorithm to make its prediction. Let's take as an example, a system to correctly diagnose diseases in patients. This system would be fed with different types of information, maybe the presence or absence of some specific symptoms and other medical measurements, such as blood pressure, cholesterol and so on. Each input, in this case each type of information about a patient, is called a *feature* and all the information regarding a single patient, the value of each *feature*, is called a *sample*. Finally, for supervised learning problems, the correct output to be predicted, in this example, the disease present in each patient, is called a *label*.

2.2.3 Overfitting and Underfitting

When training an algorithm with a limited amount of training data, there are two common issues, known as *overfitting* and *underfitting*, a representation of both is shown in figure 2.1.

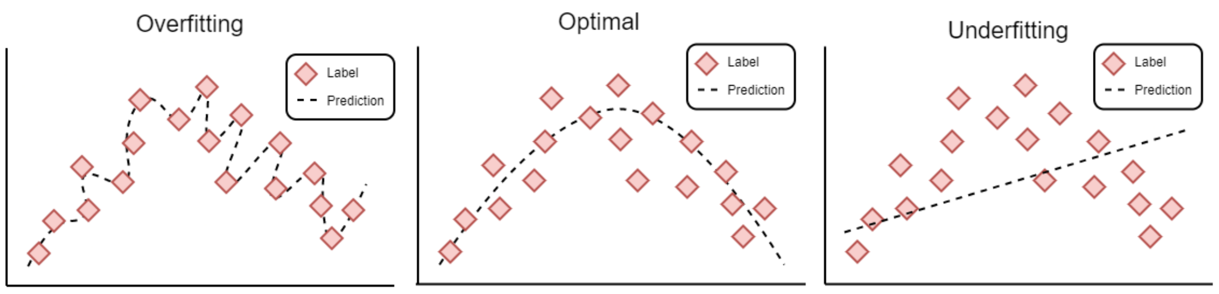


Figure 2.1: Overfitting and Underfitting

The first happens when the algorithm fits the training data so perfectly well that it cannot be generalized for the rest of the problem, so it will perform poorly in real time situations. The second is the exact opposite, the algorithm generalizes the solution so much, so that the average error is low, that in the end, it will also perform poorly, since no sample can be actually predicted with confidence. Usually happens when there has not been enough training cycles yet.

2.2.4 Generalization, Stability and Convergence

When training a *ML* algorithm, there are three important concepts that are crucial to respect or achieve, are they, generalization, stability and convergence.

Generalization is the ability of the algorithm to maintain a good performance when being fed with out-of sample data, which is data that has never been seen before, during training.

Stability is the ability of the algorithm to work with a relatively satisfactory performance for similar problems but different datasets. For example, an *ANN* that would be trained to predict earthquakes in Iceland, would have a good stability if it could maintain a good performance simulating the same detection but in another place, without additional training, which logically would not be the optimal solution for the new problem.

Convergence is the ability to shrink the error simultaneously for every input pattern being learned, the perfect convergence happens when the errors for every sample are minimum and equal to each other.

2.3 Artificial Neural Networks

Artificial Neural Networks are algorithms that resemble a computational network inspired in biological neural systems, as Walczak and Cerpa [2] said, these models simulate the electrical activity of the brain and nervous system. Each node, called perceptron, emulates a single neuron and works as a processing unit. Several perceptrons arranged in layers and connected to each other form the network, as shown in figure 2.2, this type of *ANNs* are usually known as Multilayer Perceptron networks *(MLP)*.

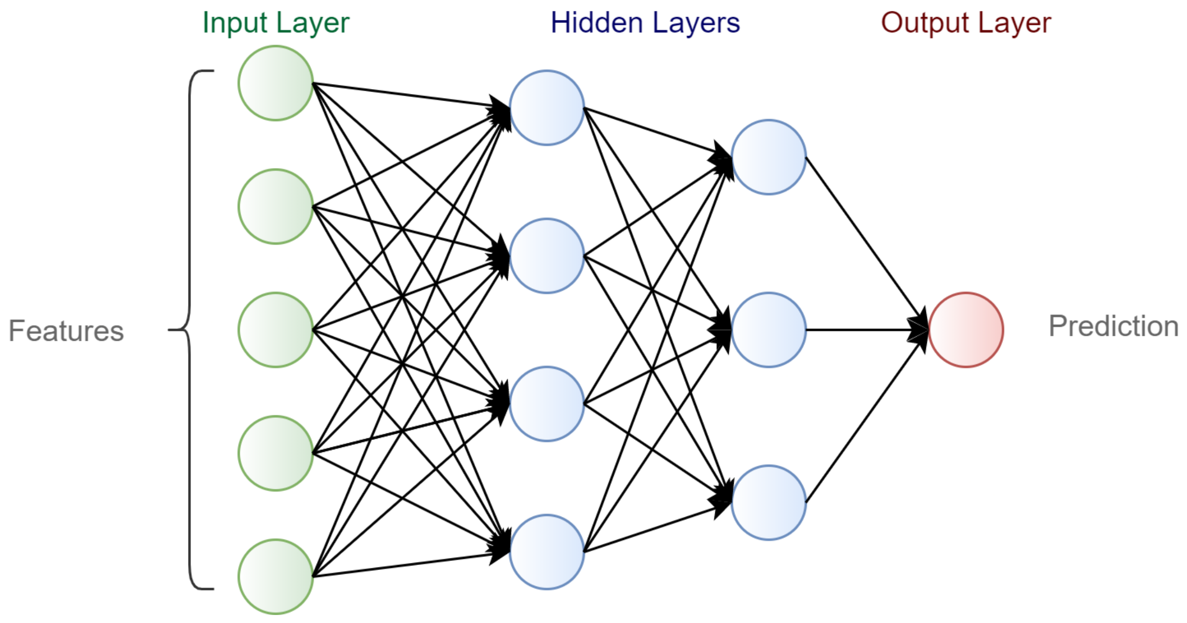


Figure 2.2: Structure of an Artificial Neural Network

2.3.1 Layers and Nodes

The first layer of the network, called input layer, represents the features of the sampling data, having one node per feature, which will serve as input for the next layer. As seen in figure 2.2, typically each perceptron is connected to every node in the next layer, but some connections can be erased to avoid overfitting. Every connection has its own weight associated, that will be multiplied by the signal entering the connection and the next perceptron will sum every signal that receives to then produce an output for the next layer. The weights associated to a connection simulate ”the strengthening of neural pathways in the brain” [2] and the neural network learns by adjusting these, during the training process, to better match the labeled samples at the last layer. This layer, known as output layer, can either be a single neuron for numeric predictions, such as time series, or several nodes, each corresponding to a different category, in a classification problem, where the output will be given as a probability of the sample belonging to the class represented by each node. It is also possible to use a network with multiple outputs to predict different aspects. For example, having several medical symptoms and measures of a patient as features, each node of the output layer would represent the presence of a different disease, where unlike the last scenario it is possible to have more than one of the outputs activated.

2.3.2 Single Perceptron

In order to mathematically describe a *MLP*, let's take a supervised learning example where the sampling data is given as $(x^{(i)}, y^{(i)})$. The vector x contains the features which will serve as input for the network and y contains the labels for the training data, which is the goal of the network to predict and will be used in the learning stage. For this, a "complex non-linear form of hypotheses", $h_{W,b}(x)$, will be defined [9], with parameters W and b that will be fitted to the data during the learning process.

In figure 2.3 is represented one of the network's processing units, a single perceptron. It takes as input 3 features, represented by x_1, x_2, x_3, plus an intercept term $(+1)$, called a bias unit, so that the output of a neuron doesn't depend solely on its inputs but can have a term of its own. The output is given by $h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$, where $f()$ is the activation function.

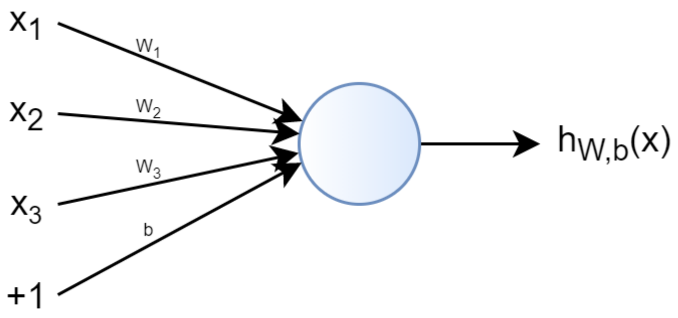


Figure 2.3: Structure of a single perceptron

2.3.3 Activation Function

There are several activation functions, the most commonly used are the Sigmoid, Hyperbolic Tangent and Rectified Linear Unit *(ReLU)*, as shown in table 2.3, with its respective ranges and derivatives. In this study, the activation function used is the hyperbolic tangent, as it is later explained in SECTION 3.2, and will be used on this chapter from now on.

Table 2.3: Activation Functions

	Function	Range	Derivative
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$[0;1]$	$f'(x) = f(x)(1-f(x))$
Hyperbolic Tangent	$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$[-1;1]$	$f'(x) = 1-(f(x))^2$
Rectified Linear Unit *(ReLU)*	$f(x) = max(0,x)$	$[0;+\infty]$	$f'(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$

2.3.4 Forward Propagation

Forward propagation, as the name says, is the step where the network takes the inputs and runs them through its neurons until it gets an output. On an already trained network, doing real time predictions, this is the only step that happens, since no parameter needs to be updated, so the network just makes predictions based on how it was trained. During the training phase, forward propagation works exactly the same, the difference is that a second step is than applied. This step is called *backpropagation*, is how a *MLP* learns and it is later described in detail.

As shown in figure 2.3 a perceptron takes several inputs and produces an output, based on an activation function. Forward propagation happens when every neuron in the network is producing its output, which is being used as an input for the next layer, and so on, until the last output of the network is produced. This dynamics are shown on a simple network in figure 2.4, alongside the notation used to describe it mathematically.

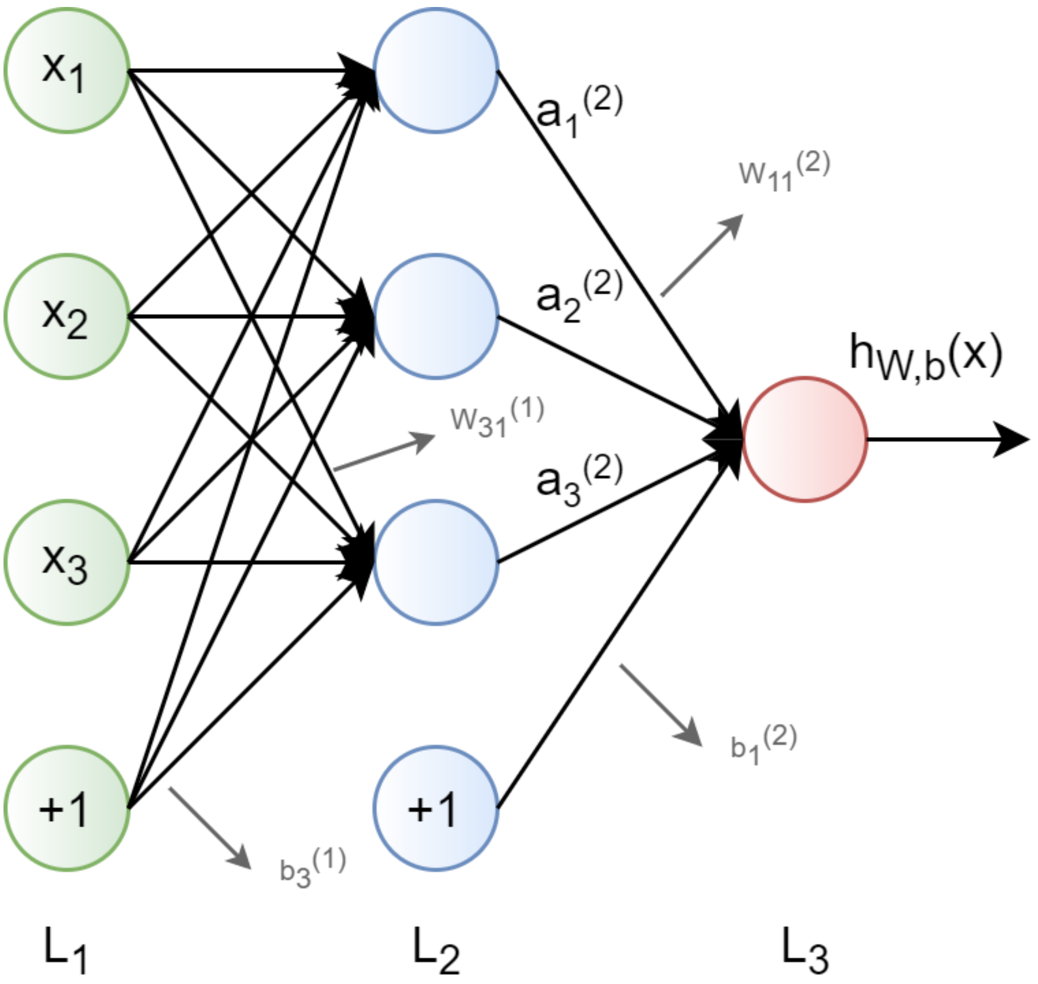


Figure 2.4: Behaviour of an Artificial Neural Network

Each layer l is labeled as L_l, from L_1 until L_{n_l}, which is the output layer, s_l is used to indicate the number of neurons in layer l besides the intercept term. In this example $n_l = 3$, so the network has parameters $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$, with $W_{ij}^{(l)}$ being the weight of the connection between node

j in layer l and node i in layer $l+1$. In this case, $W^{(1)}$ and $W^{(2)}$ belong to a $3X3$ and $1X3$ dimensional space, respectively. Following the same notation, $b_i^{(l)}$ is the bias of perceptron i in layer $l+1$ and $a_i^{(l)}$ is the output of neuron i in layer l, also called activation, from the activation function which is the last step performed by a processing unit. The forward propagation of this example can now be described by the equations 2.1.

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \tag{2.1a}$$

$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \tag{2.1b}$$

$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \tag{2.1c}$$

$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)}) \tag{2.1d}$$

If this expressions can be written in vector notation, it is possible to arrange parameters in matrices and use "fast linear algebra routines to quickly perform calculations in the network" [9], which is an important aspect of any *ML* algorithm. In order to do so, $z_i^{(l)}$ will be used to represent the total weighted sum, which the activation function of unit i receives - i.e., $a_i^{(l)} = f(z_i^{(l)})$, so $z_i^{(l+1)} = \sum_{j=1}^{n} W_{ij}^{(l)}x_j + b_i^{(l)}$. Following this notation, equations 2.1 can now be written like this:

$$z^{(2)} = W^{(1)}x + b^{(1)} \tag{2.2a}$$

$$a^{(2)} = f(z^{(2)}) \tag{2.2b}$$

$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)} \tag{2.2c}$$

$$h_{W,b}(x) = a^{(3)} = f(z^{(3)}) \tag{2.2d}$$

Finally, as x represents the inputs of the network, which is the first layer of neurons, one can assume that $a^{(1)} = x$, so equations 2.2 can be written in a general form for any number of layers and neurons, to describe forward propagation in a *MLP* network as follows in equations 2.3.

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)} \tag{2.3a}$$

$$a^{(l+1)} = f(z^{(l+1)}) \tag{2.3b}$$

2.3.5 Backpropagation

Backpropagation is how a neural network learns, by adjusting the parameters W and b so that predictions get closer to target values, or labels. After forward propagation, based on the network's prediction along with the sample's label, which is the actual value the network should have guessed, it is possible to go through the network's connections backwards and adjust the weights, the goal is that the difference between prediction and label becomes smaller every time *backpropagation* is applied.

For a given training set of m samples, or *batch*, given as $(x^{(1)}, y^{(1)}, ..., (x^{(m)}, y^{(m)})$, it is possible to apply a batch gradient descent, which aims at reducing the cost function, or error, of the entire *batch* at once.

A *batch* is a group of any number of training samples, can either be composed of a single sample or the entire training set, as well as any size in between. The bigger the *batch* the lower the computational cost of the *backpropagation* algorithm, but can be prone to *underfitting*, as the network is being adjusted to fit the entire training set at once.

There are several cost functions possible to apply, in this study the total sum of squares was chosen, as it is later explained in chapter 3, and will be used on this chapter from now on. For a single sample (x, y), or a batch of 1 element, a squared-error cost function is given by 2.4.

$$J(W, b; x, y) = \|h_{W,b}(x) - y\|^2 \tag{2.4}$$

For a *batch* of m samples, the overall cost function is given by $J(W, b)$ as shown in equation 2.5.

$$\begin{aligned} J(W, b) &= \left[\sum_{i=1}^{m} J(W, b; x^{(i)}, y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ji}^{(l)})^2 \\ &= \left[\sum_{i=1}^{m}\left(\|h_{W,b}(x^{(i)}) - y^{(i)}\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ji}^{(l)})^2 \end{aligned} \tag{2.5}$$

The first term, it's intuitive to understand that represents the total sum of squares. The second term of the overall cost function is a regularization term that aims to prevent overfitting, by reducing the magnitude of the weights. Knowing that the bigger the weights the bigger the cost function will be, these will decay to zero if no other error were to be found. It is also known as weight decay term and it is not applied to the bias units, $b_i^{(l)}$. Finally, the weight decay parameter λ, controls the relative importance between the two terms.

Now that the network has an overall cost function defined as in 2.5, the goal is to minimize said cost function, $J(W, b)$ as a function of W and b. Each iteration of the gradient descent will update the weights and bias terms as follows in equations 2.6. However, before the first iteration of the neural network's training, all the weights and bias terms need to be initialized randomly. The random factor is crucial, as it is the only way to break symmetry - i.e., if all the network's parameters were to be initialized with the same values, than all would ”learn” the same function and change in the same way, the network would never be able to train itself for non linear approaches.

$$\begin{aligned} W_{ij}^{(l)} &= W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W, b) \\ b_i^{(l)} &= b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}}J(W, b) \end{aligned} \tag{2.6}$$

The learning rate, α is the variable that controls the speed at which the network learns - i.e., how big will the magnitude of the change in parameters W and b be, each time these are updated. The smaller α is, the longer it will take to find the optimal solution, if it is too big, however, it can "jump" between solutions so far apart that it never finds better solutions. It is now understandable that in order to update said weights, one only needs to compute the partial derivatives of the cost function $J(W,b)$ which are given by:

$$\begin{gathered}\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) = \left[\sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b,x^{(i)},y^{(i)}\right] + \lambda W_{ij}^{(l)} \\ \frac{\partial}{\partial b_{i}^{(l)}} J(W,b) = \left[\sum_{i=1}^{m} \frac{\partial}{\partial b_{i}^{(l)}} J(W,b,x^{(i)},y^{(i)}\right]\end{gathered} \tag{2.7}$$

As seen in equations 2.7, in order to compute the said partial derivatives regarding the cost function $J(W,b)$, first, the ones regarding the cost function of a single sample, $J(W,b;x,y)$, need to be computed, and that's where the *backpropagation* algorithm will be useful. A step by step procedure on how to apply it to compute partial derivatives and train a network for a *batch* of samples is now provided. For each equation there are two different notations, the second represents a matrix-vectorial notation which is crucial to perform fast computations as described before.

1st Step: For a training sample (x,y), apply forward propagation as in 2.3 in order to compute the activation at the output of every neuron, as well as the output value of the hypothesis $h_{W,b}(x)$.

For every perceptron of the network, an "error term" $\delta_i^{(l)}$ needs to be computed, this measures the error relative to each neuron - i.e., how much was each node responsible for the error at the end of the network. The error at the output node can be measured directly from the target value. For hidden units, this term will be calculated from the error at the next layer that uses its activations as inputs.

2nd Step: For every perceptron i at the output layer L_{n_l} compute the error term as follows:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \|y - h_{W,b}(x)\|^2 = -2(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \tag{2.8a}$$

$$\delta^{(n_l)} = -2(y - a^{(n_l)}) \bullet f'(z^{(n_l)}) \tag{2.8b}$$

3rd Step: For every hidden layer l, from the last to the first, compute the error term as follows:

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)}\right) \cdot f'(z_i^{(l)}) \tag{2.9a}$$

$$\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)}\right) \bullet f'(z^{(l)}) \tag{2.9b}$$

In the last two steps, $f'(z_i^{(l)})$ needs to be computed, considering that $f(z)$ is the hyperbolic tangent function, than $f'(z_i^{(l)}) = 1 - (a_i^{(l)})^2$ and $a_i^{(l)}$ is already known from the first step of forward propagation.

4th Step: Finally, the partial derivatives regarding a single sample can be computed as follows:

$$\begin{aligned}\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) &= a_j^{(l)} \delta_i^{(l+1)} \\ \frac{\partial}{\partial b_i^{(l)}} J(W,b;x,y) &= \delta_i^{(l+1)}\end{aligned} \tag{2.10a}$$

$$\begin{aligned}\nabla_{W^{(l)}} J(W,b;x,y) &= \delta^{(l+1)} (a^{(l)})^T \\ \nabla_{b^{(l)}} J(W,b;x,y) &= \delta^{(l+1)}\end{aligned} \tag{2.10b}$$

Now that the partial derivatives for a single sample are computed, the procedure is the same for every sample in that batch. While these are being computed, are being summed together and stored in a matrix the same dimension as $W^{(l)}$ and $b^{(l)}$, represented by $\Delta W^{(l)}$ and $\Delta b^{(l)}$, in order to to compute the partial derivatives of the total cost function as shown in equations 2.7.

5th Step: Finally, parameter W and b can be updated after a batch of training as follows:

$$\begin{aligned}W^{(l)} &= W^{(l)} - \alpha \left[\Delta W^{(l)} + \lambda W^{(l)} \right] \\ b^{(l)} &= b^{(l)} - \alpha \left[\Delta b^{(l)} \right]\end{aligned} \tag{2.11}$$

These steps will be repeated for every *batch* in the training set, called an *epoch*, and then for as many *epochs* as one needs to reduce the cost function $J(W,b)$ to the desired values.

2.4 Support Vector Machines

Support Vector Machines are supervised learning algorithms based on decision boundaries, developed in the late 90s by Vladimir Vapnick [10]. It is currently, the most commonly used Machine Learning algorithm for pattern classification [10], mostly because of its simplicity, when compared to other ML algorithms, and applicability in a broad spectrum of classification problems, even when the training samples are limited [11].

2.4.1 Support Vectors

The fundamental goal of an *SVM* is to find an hyperplane in an N-dimensional space that separates the data points according to their classification, being N the number of features that define a sample, so each one is represented by a single point. There will always be several hyperplanes that can do the job - i.e., possible solutions, specially for problems with a small number of training samples. So the best solution is the hyperplane that maximizes the distance between the support vectors of both classes (for a binary classification problem), known as margin. Support vectors are the data points that are closer to the decision boundary, being the ones used to compute the solution. Different support vectors will generate different hyperplanes and therefore a solution with a different margin, as shown in figure 2.5. The bigger the margin, the bigger the confidence when classifying future samples using the trained SVM, this approach is called ”Widest Street Approach”.

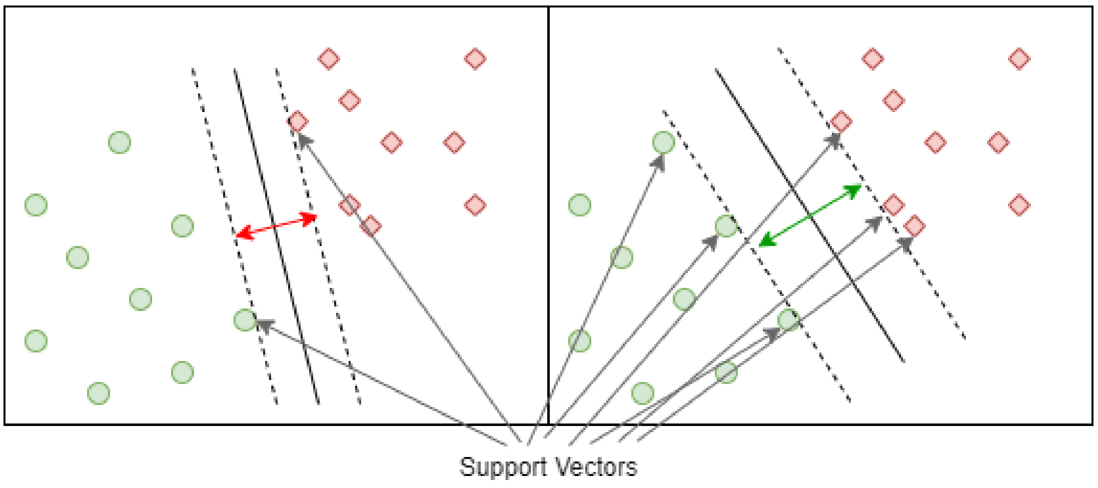


Figure 2.5: Influence of support vectors

2.4.2 Widest Street Approach

Let's take a simple example on a 2 dimensional plane, so it's easier to visualize. Each sample is classified between A and B based on 2 features, x_1 and x_2. The hyperplane is, in this case, a single line, represented by γ, as shown in figure 2.6.

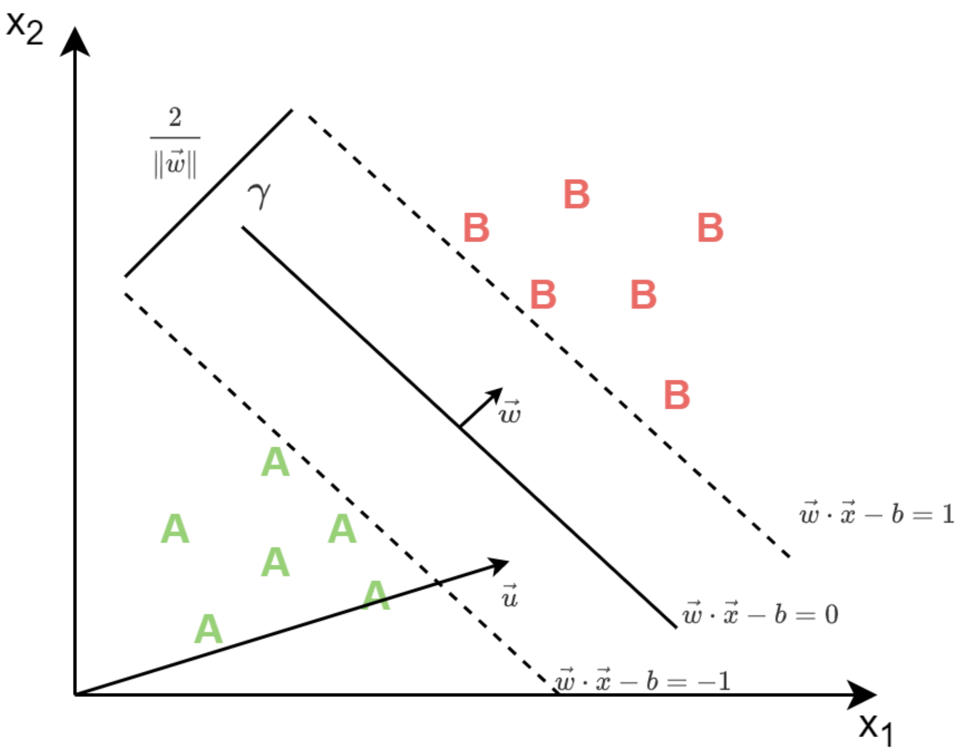

x_2
$\frac{2}{\|\vec{w}\|}$
γ
B
B
B
B
B
B
$\vec{w}$
A
A
A
A
A
A
$\vec{u}$
$\vec{w} \cdot \vec{x} - b = 1$
$\vec{w} \cdot \vec{x} - b = 0$
$\vec{w} \cdot \vec{x} - b = -1$
x_1

Figure 2.6: Behaviour of a Support Vector Machine

In order to define a decision rule based on the separating hyperplane, let's start by assuming a vector $\vec{w}$ of any length, perpendicular to γ, and an unknown vector $\vec{u}$, representing a new sample. Next step is to classify the said vector, as belonging to the A or B side of the decision boundary. For this, vector

$\vec{u}$ needs to be projected on vector $\vec{w}$, as $\vec{w} \cdot \vec{u}$, this internal product will give the distance of $\vec{u}$ in the same direction as $\vec{w}$ and if it is bigger than a constant, C, then it means that it has passed the decision boundary and will therefore be classified as a B sample. So it is valid to say that the decision rule for a B sample is given by $\vec{w} \cdot \vec{u} + b \geq 0, C = -b$. The next step is now to decide which b and $\vec{w}$ to use, as there are no constrain rules for it yet. With this in mind, Vapnick introduced two constraints, one for each side of the boundary, $\vec{w} \cdot \vec{x_M} + b \geq 1$ and $\vec{w} \cdot \vec{x_M} + b \leq -1$, to describe B and A samples, respectively. For "Mathematical Convenience", this two equations can be written as one, $y_i(\vec{w} \cdot \vec{x_l} + b) - 1 \geq 0$, where y_i is 1 for B samples and -1 for A samples.

Given that, to define the hyperplane γ, one needs to calculate the widest distance between the support vectors. From the constraint rules it is known that $\vec{w} \cdot \vec{x_M} = 1 - b$ and $\vec{w} \cdot \vec{x_F} = 1 + b$, so $\vec{w} \cdot (\vec{x_M} - \vec{x_F}) = 2$. Dividing the result by the length of $\vec{w}$ one gets the distance between samples as given in 2.12.

$$\frac{\vec{w}}{\|\vec{w}\|} \cdot (\vec{x_M} - \vec{x_F}) = \frac{2}{\|\vec{w}\|} \tag{2.12}$$

Maximizing the distance function, it is possible to find the widest possible distance, so from 2.12 one can conclude that $\|\vec{w}\|$ needs to be minimized. For "mathematical convenience" for the next steps, $\frac{1}{2}\|\vec{w}\|^2$ will be minimized instead. Lagrange multipliers will be used for this purpose as described in the following steps.

1st Step: Find where L does not have derivatives:

$$L = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{l} a_i(y_i(\vec{w} \cdot \vec{x_l} + b) - 1) \tag{2.13}$$

2nd Step: Compute derivatives of L:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{l} a_i y_i x_i = 0 \implies w = \sum_{i=1}^{l} a_i y_i x_i \tag{2.14}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{l} a_i y_i = 0 \tag{2.15}$$

3rd Step: Replace derivatives (2.14 and 2.15) in equation 2.13:

$$L = \sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i,j=1}^{l} a_i a_j y_i y_j (x_i \cdot x_j) \tag{2.16}$$

From 2.16 one can conclude that the optimization of the decision boundary depends solely on pairs of samples (x_i, x_j) and the decision rule is given by 2.17 and 2.18, for A and B samples, respectively.

$$\sum_{i=1}^{l} a_i y_i x_i \cdot \vec{x} + b \leq 0 \tag{2.17}$$

$$\sum_{i=1}^{l} a_i y_i x_i \cdot \vec{x} + b \geq 0 \tag{2.18}$$

2.4.3 Higher Dimensional Spaces and Kernels

From this example, it seems simple to separate samples with a straight line, even for higher dimensions which can not be visualized. Looking at the equations 2.17 and 2.18 it is concludable that the classification depends solely on the inner product of the sample vectors, so it is independent from the number of dimensions the vectors have. The problem now is to classify samples which can not be separated by a straight line. With that in mind and the fact that a *SVM* works in any dimensional space, Vapnick decided to bring the samples to higher dimensional spaces until these can be separated by an hyperplane, which would correspond to a straight line in a 2 dimensional space. In order to do so, a transformation $\phi(\vec{x})$ is applied to vector $\vec{x}$ which represents the sample. Now to maximize the distance between the support vectors, one needs to compute the inner product $\phi(\vec{x_M}) \cdot \phi(\vec{x_F})$. The next steps will be the same as before, so in the end the classification depends solely on the inner product of an unknown vector in a high dimensional space as a function of vectors in the original space [6], as represented in equation 2.19.

$$\phi(\vec{x_l}) \cdot \phi(\vec{u}) = K(x_i, \vec{u}) \tag{2.19}$$

The transformation $\phi(\vec{x_l}) \cdot \phi(\vec{u})$ does not need to be known, as it is represented by a function K, referred to as a *kernel* in *Machine Learning*, which in the end is the only thing needed to maximize the distance between support vectors. The three most common kernels for support vector classification are shown in table 2.4.

Table 2.4: SVM different kernels

Linear Kernel	$K(x, y) = x^T y + c$
Polynomial Kernel	$K(x, y) = (x^T y + c)^n$, n is the kernel's order
Radial Basis Function Kernel	$K(x, y) = e^{-\frac{\|x-y\|}{\sigma}}$

The three kernels will be explored deeply, later in chapter 3.

2.5 State-of-the-Art

This section summarizes the extensive research done over the topics of Artificial Intelligence applied to financial markets. Some of the solutions presented in the scientific community, so far, are presented, as well as older models which would became an inspiration for the following works. This section is divided between relevant works on the Forex market, systems built over Artificial Neural Networks and the usage of Support Vector Machines in financial forecasting models. A table summarizing the most relevant works is presented at the end of this section.

2.5.1 Research on Forex

The literature review on AI solutions applied to Forex, revealed that a technical approach is usually the preferred choice, either analyzing price sequences or technical indicators. Fundamentals, which are very useful for dealing with other financial markets, such as stocks, are usually left aside when dealing with currency rates.

On a recent study from 2019, Chihab et al. [12] proposed a system to trade the Forex market intraweekly, based on a combination of some of the most common technical indicators. The system was implemented combining a Probit Regression model and a Random Forest algorithm, and achieved a ROI of 78% over 17 weeks, on the *GBP/USD* currency pair. A year before, Carapuço et al. [13] proposed a model of reinforcement learning, based on a neural network that uses the *Q-learning* algorithm to train three layers of ReLU neurons, this network trains itself over the testing period with past test data, making it possible to always be in the market and up to date. The proposed system achieved a ROI of 114% from 2010 till 2017 in the *EUR/USD* market, making an average yearly ROI of 16.3%.

Ealier in 2007 Yao et al. [4] had suggested a different approach than the traditional forex trading. Supported on the assumption that ”the key in Forex trading is to pick the right currency to trade at the right time”, this study proposes a system capable of managing a Forex portfolio of different currencies, instead of the usual trading systems based on single currency movements. The objective is achieved by implementing a fuzzy neural network (FNN) capable of forecasting price movements over a range of currencies. Yao et al. [4] claims that the ”experimental results on real world Forex market data show that the proposed mechanism yields significantly higher profits against various popular benchmarks”.

The research done over this topic indicates that a lot of different ML algorithms can be used to make accurate predictions on Forex time series, however, Petropoulos et al. [14] states that ”in real-world trading settings, no single machine learning model can consistently outperform the alternatives” and proposes a voting system, in 2017, that combines the predictions of several algorithms, namely SVMs, random forests, ANNs and naive Bayes classifiers.

It is suggested on different studies that hybrid ML models combining different algorithms, usually outperform single algorithm systems. The most common case is the use of an algorithm to optimize the other, which is actually doing the forecasting. For example, Yu and Wang [15] proposed an online learning algorithm in 2007, capable of optimizing the learning stage of a neural network, outperforming the classical approaches of batch learning and the Levenberg-Marquard optimizer.

Later in 2012 Maknickiene and Maknickas [16] suggested an alternate version of a recurrent neural network (RNN) based on a genetic algorithm (GA), which uses a Long short-term memory network (LSTM), called *EVOLINO*. In order to use this model to trade the *EUR/USD* FX pair, Maknickiene and Maknickas [16] creates a system that uses several of this networks and a Delfi method, which works as a voting system depending on the individual outputs of each network, to ultimately generate the trading signal.

Özorhan et al. [17] implemented an hybrid system in 2017, based on a SVM to predict the ”direction and magnitude of movement of currency pairs in the foreign exchange market”. The SVM used several technical indicators as features adapted for each trading day by a genetic algorithm. The results obtained

”suggest that using trend deterministic technical indicator signals mixed with raw data improves overall performance and dynamically adapting the input data to each trading period results in increased profits” [17].

2.5.2 Research on Artificial Neural Networks applied to Financial Markets

Artificial Neural Networks showed to be a widely used solution for forecasting time series in financial markets, became very popular in the 90s when more studies on the area started to arise, usually out-performing traditional trading strategies.

”Technical analysis is not designed to deal with non-uniform periodic, and discontinuous functions”, such as forex prices time series, was claimed by Chan and Foo Kean Teong [18], and in 1995 they proposed a model composed of a single neural network capable of optimizing new technical indicators in order to open trades before the common, and widely used, technical indicators generate trading signals, followed by most of the crowd. Later in 1997, Yao and Tan [19] reported ”empirical evidence that a neural network model is applicable to the prediction of foreign exchange rates”. Their work consisted in the use of technical indicators and price sequences as features for the model proposed, composed by five neural networks, each forecasting a different major currency in relation to the USD. The study showed that ”significant paper profits can be achieved for out-of-sample data with simple technical indicators”, without an extensive training dataset.

As the evolution in the field of computational intelligence progressed, more complex solutions for the financial area, regarding newly optimized Artificial Neural Networks, started to be study.

In 2009 Butler and Daniyal [20] attempt to predict the movement of the stock market using an evolutionary artificial neural network (EANN). The results showed that the optimal solution was achieved by updating the EANN’s weights through genetic operations, such as crossover and mutation, and that the traditional backpropagation method tended to overfit the data. A second conclusion taken from this study is that a training approach with multi-objective optimization (MOO) produces better results, represented in a bigger return on investment (ROI), than a traditional single objective optimization. A MOO consists of optimizing the neural network at each iteration, considering more metrics than just the prediction error in the cost function, for example the final ROI or the accuracy.

Evans et al. [21], in 2013, proposed an intradaily trading system for 3 related FX pairs, *EUR/USD*, *GBP/USD* and *EUR/GBP*, which trades the market that provides the bigger level of confidence, each day, according to the other two, for example if the 3 forecasted movements are in accordance. The forecasting is implemented by an Artificial Neural Network, which has its parameters and topology tuned by a Genetic Algorithm, in order to achieve optimal performance. The optimal proposed model achieved a prediction accuracy of 72.5% and an annualized net return of 23.3%. Furthermore, an important statistical test, of this study, ”confirmed with a significance of more than 95% that the daily FX currency rates time series are not randomly distributed”. Later in 2016 Galeshchuk [22] used a model based on ANNs to predict exchange rates in 3 different markets, *EUR/USD*, *GBP/USD* and *USD/JPY*, comparing the results and predictability of each. The results obtained suggested that *EUR/USD* was the most predictable market in daily steps, with an average relative prediction error of 0.2%, while *USD/JPY*, with a value of 0.3% for the same error in monthly steps, was the market to perform better under such circumstances. Lastly, *GBP/USD*, with an average error of 1.9% was the best performance for quarterly steps, with the *USD/JPY* performing at 3.5% relative error, showing a huge rise in unpredictability for bigger time ranges.

Neural networks are still widely used nowadays, because of their great adaptability to the features provided and also a low computational cost when compared to new ML algorithms that were invented over the years. On a very recent study from 2020, Zafeiriou and Kalles [23] implement an intraday forecasting system based on an ANN which is fed data at the *tick* of the currency pair *EUR/USD*. This system ”aims to simulate the judgment and decision making of the human expert, responding in a timely manner to changes in market conditions, thus facilitating the optimization of ultra-short-term transactions” [23]. The neural network generates a ”trend forecasting signal” and positions are opened in the market when a good opportunity arises, for example the price is shorter the the predicted trend. The final system was tested for more than 2 million data points, corresponding to October of 2018 and reached an accuracy of 78%.

2.5.3 Research on Support Vector Machines applied to Financial Markets

Support Vector Machines are also widely used in the financial time series prediction area, some prefer it over ANNs for they reduced computational cost and easier understandability of what is happening during the training stage. As Kyoung-jae Kim [24] claimed, ”SVMs are promising methods for the prediction of financial time-series because they use a risk function consisting of the empirical error and a regularized term which is derived from the structural risk minimization principle”. In 2003, he applied a simple support vector machine to predict the stock market prices and compared the performance with a backpropagation neural network and a case-based reasoning, achieving an accuracy of 57.8% against 54.7% and 51.9% by the NN and CBR, respectively.

”The SVM has been applied to the problem of bankruptcy prediction, and proved to be superior to competing methods such as the neural network, the linear multiple discriminant approaches and logistic regression” [25]. In 2007 Hua et al. [25] proposed a system combining support vector machines with a logistic regression, in order to improve accuracy, capable of predicting financial distresses in companies, based on fundamental data, showing promising results since it ”outperformed the conventional SVM”.

More recent studies keep proving the reliability of the SVM, both in forecasting, classification and optimization of more complex ML systems.

”The trend of currency rates can be predicted with supporting from supervised machine learning in the transaction systems such as support vector machine” [26]. In 2018 Thu and Xuan [26] achieved a return on investment, trading the Forex pair *EUR/USD* over 2016, of 33.8% using a single SVM to predict the direction of the market. A simple strategy was implemented to trade accordingly with the direction forecasted. In the same year, Jubert de Almeida et al. [6] implemented an hybrid system between a Support Vector Machine and a Genetic Algorithm. The second implements a classic approach of optimizing trading rules based on different technical indicators, the first (SVM), however, presents a unique approach of pre-classifying the type of market at the current moment and consequently using one of three distinctly trained GAs, according to said type of market. The best proposed system achieved a ROI, over a period slightly bigger than a year, of 83%.

2.6 Chapter Conclusions

This chapter is crucial to understand the solutions proposed on this work and the themes discussed. A background on financial markets, particularly, the foreign exchange market, is given, as well as some important Machine Learning concepts used throughout this thesis. Followed by a detailed description of the theory behind Artificial Neural Networks and Support Vector Machines. The literature review included some promising solutions for trading the Forex market based on ML algorithms, which will be used as a starting point, in order to implement the proposed trading system. In table 2.5 the most relevant scientific works from the State-of-the-Art are summarized.

Table 2.5: Most relevant scientific works from State-of-the-Art

Work	Year	ML Approach	Financial Market	Evaluation Method	Period	Return
[21]	2013	ANN optimized by GA	EUR/USD, GBP/USD, EUR/GBP	Accuracy & ROI	2010 - 2012	72.5% & 23.3%
[6]	2018	SVM & GA	EUR/USD	ROI	2015 - 2016	83%
[13]	2018	RNN with Q-learning optimizer	EUR/USD	ROI	2010 - 2017	114%
[27]	2003	SVM	Stock Market	Accuracy	1989 - 1998	57.8%
[4]	2007	Fuzzy Neural Networks	Several Currencies	Accumulated Return	2004 - 2006	33.95%
[26]	2018	SVM	EUR/USD	ROI	2016	33.8%
[12]	2019	Probit Regression & Random Forest	GBP/USD	ROI	2017	78%
[23]	2020	ANNs with Technical Indicators	EUR/USD(ticks)	Accuracy	Oct 2018	78.1%

Chapter 3

Proposed Architecture

This chapter describes the final trading system proposed and all of its components, as well as how it was implemented and the validations made at each step.

First, an overall description of the trading system functionability is given, as well as a detailed explanation of each layer's job and how do these interact with each other. Secondly, each layer is explored deeply, in terms of the algorithms incorporated, the implementation steps and the required validations in search of the optimal solutions.

The entire system is implemented and tested in Python [28], given the broad spectrum of ML and data science libraries available.

3.1 Overall Structure

This work's architecture is composed of four main layers. Each one of those has its own specific goal, and functionability in order to accomplish it. These layers are connected to each other as shown in figure 3.1 and together form a complex system for trading, intradaily, the *Forex* pair *EUR/USD*, in a profitable way.

This system works on a daily basis, it analyzes price data from midnight till noon, and makes a prediction on a 4 and a half hour horizon - i.e., tries to predict, with a certain level of confidence, if the price of this pair will rise or fall until 4:30pm. Based on that prediction, and the level of confidence, the system will eventually take one of 3 decisions, either go long, short or skip that trade. If the decision is one of the first two and decides to open a trade, it will also compute the volume of that trade, based on the amount of money available to invest and the said level of confidence. As the prediction has a specific time horizon, the trade will be closed at 4:30pm no matter the price, it is also important to note that the volume traded is only possible with *margin trading*, which consists in the use of leverage. Another perk of intraday trades is that no *Swap* needs to be payed, since it is charged overnight on open positions.

Following is a description of each layer's function and how these interact with each other in order to ultimately take said decision.

1. The **Data Preparation Layer** is responsible for preparing the data to feed as an input to every other

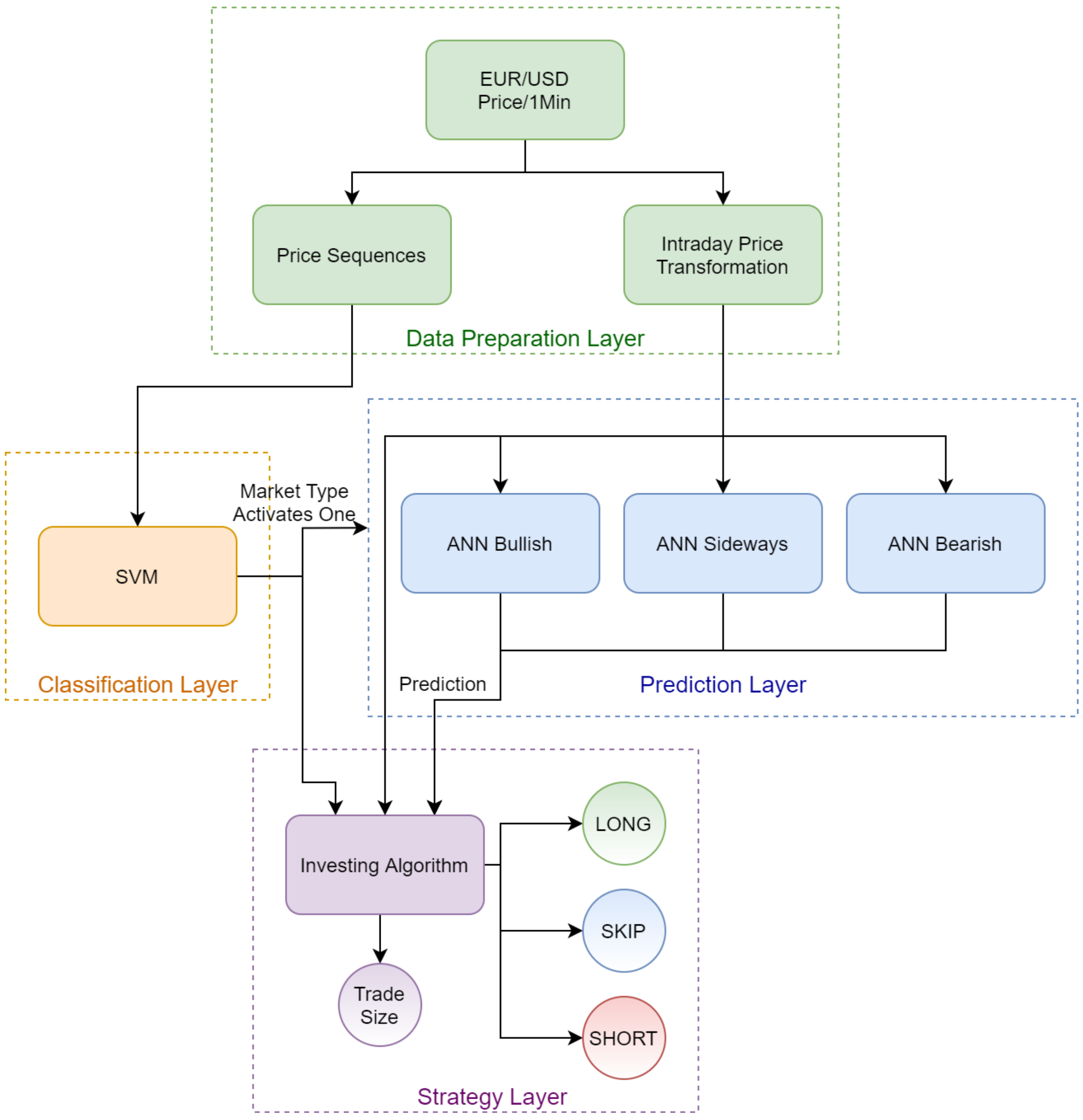


Figure 3.1: System overall architecture

layer. From the raw prices, ticked at the minute, of the *FX* pair *EUR/USD*, on one side arranges a three month window of price sequences to feed the classification layer while on the other prepares daily samples of a scaled transformation, explored deeply during implementation in section 3.2.3, to feed the prediction and strategy layer, respectively, as seen in figure 3.1. All the raw data comes from *.csv* files, covering prices at the minute from 2004 to 2019, downloaded from *dukascopy.com*.

2. The **Classification Layer** is based on Jubert de Almeida et al. [6] work and consists of a single *Support Vector Machine* to accomplish its role in the system. The goal is to add a layer, before the actual prediction, that has the ability to classify the market in three different groups, either a *bullish* market, which has an upgoing trend, a *sideways* market or a *bearish* market, which has a downgoing trend. For this, it will have access to a completely different timeframe than the next

layer, while the prediction layer is intradaily based - i.e., each sample only has information about one specific day, this layer has access to time windows of three months. Any *ML* algorithm is subject to *overfitting*, or *underfitting* logically, but *Artificial Neural Networks*, specifically, are very prone to overfit a solution. The idea behind this classification layer, in order to mitigate this issue, is based on the popular saying ”if you cannot beat them, join them”. If it is hard to avoid *overfitting* without generalizing the problem too much, one can use it to its own advantage. With the data clustered in three different groups, based on the type of market at the time, one can train three different neural networks that will only see data belonging to a specific type of market, each, and therefore be *overfitted* to that kind of data, which is not a problem as long as the classification layer accomplishes its goal, accurately, during real time predictions. To summarize, data on the last three months will be given to the *SVM*, which will classify it into one of three types and activate one, and one only, of the neural networks on the next layer, accordingly.

3. The **Prediction Layer** is the core of this work, its goal is to predict price movements on a four and a half hours horizon, fundamented in neural networks. It can be looked as if this is the main layer and the others work for this one, each doing a different optimization, but if the prediction is inaccurate, the entire system will fail. This layer will receive a sample per day, which consists of a transformation of price sequences from midnight until noon, and output its prediction, on that transformation, at *4:30pm*. As it is a supervised learning system, it will also receive the accurate value of this transformation at *4:30pm*, during the training stage - i.e., the sample’s *label*. It is composed of three, distinctly trained, *Artificial Neural Networks* activated one a time depending on the type of market being traded at the moment, previously classified by the last layer. Its output will be fed to the strategy layer, alongside the initial data transformation and other meaningful variables, so that it can decide what to do at *12:00pm* based on what it predicts will happen until *4:30pm*.

4. The **Strategy Layer**, similarly to the data preparation one, has no machine learning behind. It is a simple algorithm that joins the outputs from all other layers to ultimately take a decision between skipping a trade or take a long or short position, and, for the last two, also the size, or volume, of the said trade. Different strategies will be tried and placed against each other in the results chapter 4, in order to evaluate how these behave in different conditions and which one brings more profit. All the different approaches, regarding the investment strategy, will be discussed in detail, later in section 3.5.

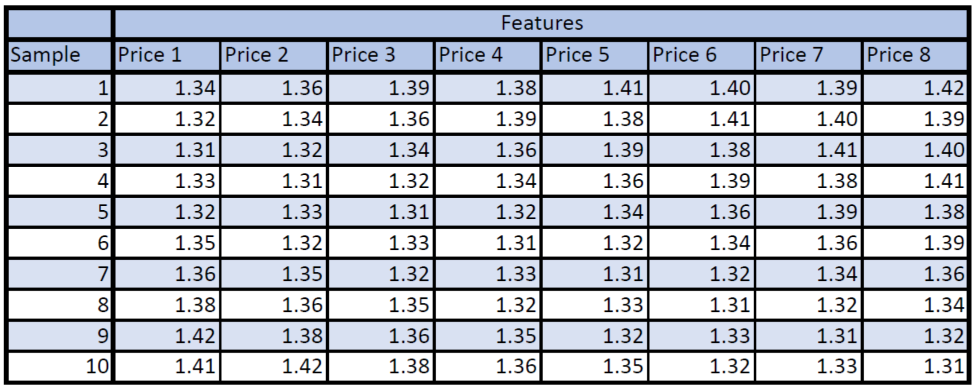
3.2 Data Preparation Layer

The data preparation layer has to prepare the features to feed both the Support Vector Machine and the Artificial Neural Networks in the classification and prediction layer, respectively. For this, it will make use of *.csv* files, downloaded from *dukascopy.com*, containing the raw data of *EUR/USD* price values per minute, from 2004 until 2019. All the data will be treated using a *Python* library named *Pandas* [29].

This library was build with efficiency as its main goal, it is a reliable and fast way to write, analyze and manipulate big amounts of data, specially when comparing to *python* native tools, such as arrays and lists.

3.2.1 Features for Classification

For the classification of the type of market, price sequences of approximately 3 months were used as the features in the *SVM*. For this purpose, a matrix of n samples for 66 inputs was created. Each vector has the information about the last 66 days, approximately 3 months, as one value per day, and for this value, the price at *12:00pm* was chosen, as it is the time that the next layer will do its prediction and possibly enter the market. A reduced example of how this matrix would look like is shown in figure 3.2.



	Features							
Sample	Price 1	Price 2	Price 3	Price 4	Price 5	Price 6	Price 7	Price 8
1	1.34	1.36	1.39	1.38	1.41	1.40	1.39	1.42
2	1.32	1.34	1.36	1.39	1.38	1.41	1.40	1.39
3	1.31	1.32	1.34	1.36	1.39	1.38	1.41	1.40
4	1.33	1.31	1.32	1.34	1.36	1.39	1.38	1.41
5	1.32	1.33	1.31	1.32	1.34	1.36	1.39	1.38
6	1.35	1.32	1.33	1.31	1.32	1.34	1.36	1.39
7	1.36	1.35	1.32	1.33	1.31	1.32	1.34	1.36
8	1.38	1.36	1.35	1.32	1.33	1.31	1.32	1.34
9	1.42	1.38	1.36	1.35	1.32	1.33	1.31	1.32
10	1.41	1.42	1.38	1.36	1.35	1.32	1.33	1.31

Figure 3.2: Example of price sequence windows

It is important to note that a sliding window is in use, consequently, for each new sample all the prices will be one day sooner and a new price will be entered at the beginning of the vector, as can be seen, in figure 3.2.

3.2.2 Labels for Classification

As explained in chapter 2, Support Vector Machines are supervised learning algorithms, so a label is needed for the learning stage of the algorithm. This label, which is the output the *SVM* will be responsible for producing, takes one of three values, represented by integers, the market is either *bullish*, sideways or *bearish*. To create this label, the first step was to classify "by hand" the market zones of the entire data set, as shown in figure 3.3.

Once the market zones are classified, the label for each sample will be the classification corresponding to the last day of the price window, which corresponds to the day the classification would actually be taken in production - i.e., when the system is actually doing real time predictions.

3.2.3 Features for Prediction

For the prediction, a system of intradaily samples was chosen, with one sample corresponding to the trading prices of a single day from midnight until noon, with the target value being that same price

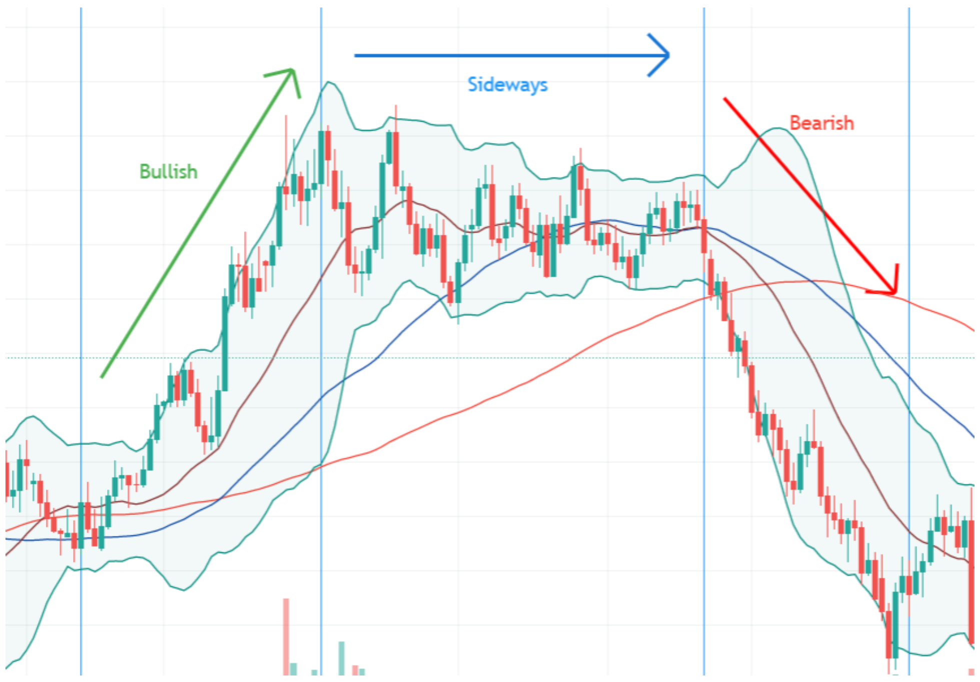


Figure 3.3: Example of market zones classification

at *4:30pm*. According to Evans et al. [21], Forex intraday price rates are noisy, chaotic and present non-stationary behaviour, so to make a prediction on those there is the need to apply some kind of transformation to deal with these issues.

The first issue that needs to be dealt with, is the sampling frequency used to represent an entire day. On one side, high frequency samples means too much information, ”useless and sometimes disorienting” [21], while low frequency might mean that crucial information to identify patterns can be missing. In his book, Refenes [30] stated that a sampling period between 5 and 60 minutes is ideal for the forex market depending on the currency pair. For this work a sampling period of 30 minutes was chosen, which corresponds to a vector with a size of 25 features, to be fed as an input to the artificial neural networks. However, this layer has to treat the data until *4:30pm*, as this is the time corresponding to the target value, used for training. Lastly, in order to mitigate the noise present in price variations during the day, an average of 30 minute windows was taken for each point. An example of a few different samples, after this smoothness transformation is applied, representing price variations during the day, is represented in figure 3.4. This prices correspond to the year 2019 of the forex pair *EUR/USD* and each line represents a different day, as shown in the figure's label.

The next step is to have a meaningful data representation for the *ANNs*, Vanstone and Finnie [31] claimed that it is crucial that *ANNs* do not have visibility of the raw market prices, otherwise identical patterns happening at different price zones will be treated as distinct ones, making the generalization close to impossible. So this data needs to be scaled to a specific range where identical patterns will be treated as so. Evans et al. [21] suggests an approach based on return rates between the current point and the last one, accomplished by a log difference transformation as shown in equation 3.1, where P_t represents the price at time t and P_{t-1} the one before, at $t-1$.

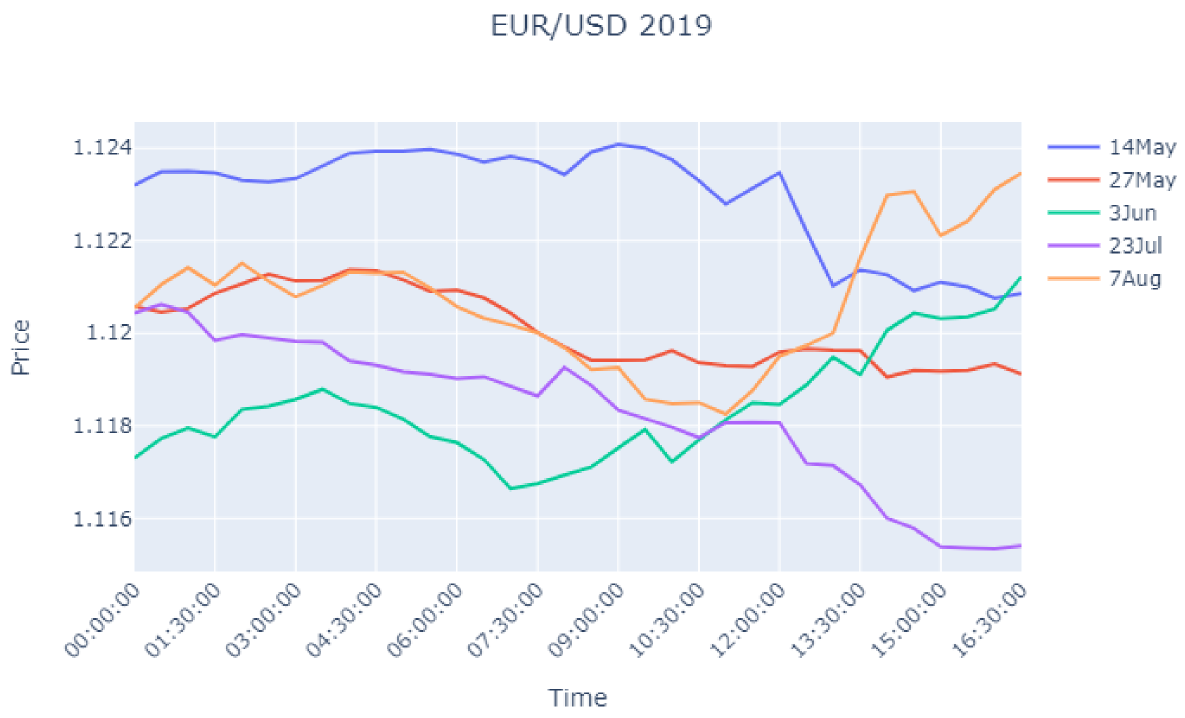


Figure 3.4: Price during the day of different *EUR/USD* samples

$$r = ln(\frac{P_t}{P_{t-1}}) \tag{3.1}$$

In figure 3.5 this transformation is shown on the same examples used before in figure 3.4.

Now that the generalization issue has been resolved, a new one rises, which is convergence, as this transformation makes each point depend solely on itself and the last one. In this conditions it is hard for an *ANN* to make a prediction, as the data would be considered random. A possible way to overcome said issue, is with the use of a technical indicator that takes all this return rates in consideration, which actually represent price movements. For example, if one can average every return rate, this would give information about how much the price moved, in average, during the day. As each sample starts at midnight and has no information behind that point, a moving average with a window sized n is not possible. The workaround is the use of the Incremental Window Moving Average, introduced in section 2.1.2. This indicator takes in consideration the entire window that is behind a certain point and for each new point that window will grow, as can be seen in equation 3.2, which is ideal for the intradaily samples used in this work.

$$y_{(i,j)} = \frac{\sum_{l=1}^{k} x_{(l,j)}}{k} \forall k = 2, 3, 4, ...n \tag{3.2}$$

The outcome of the IWMA transformation is the actual data that will be fed to the prediction layer, containing the input features needed for the *ANNs* and also the target value to be predicted. In figure 3.6 is shown this transformation on the same examples presented before in both figures 3.4 and 3.5.

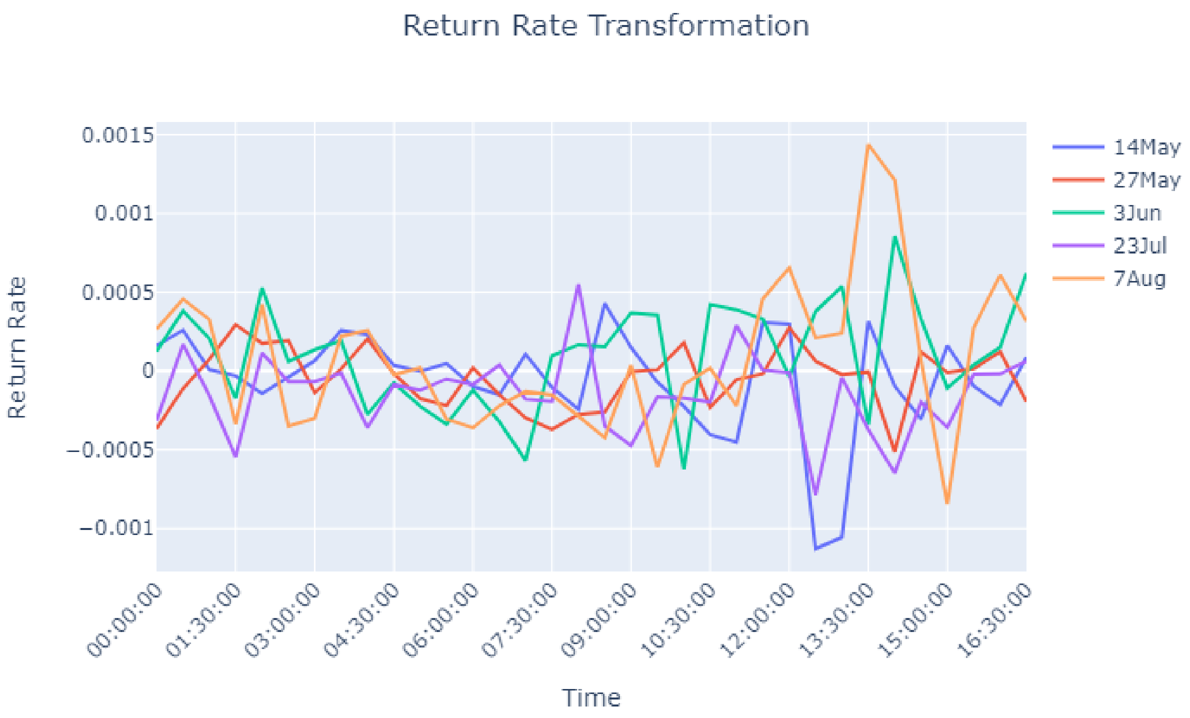


Figure 3.5: Return rate transformation of different samples

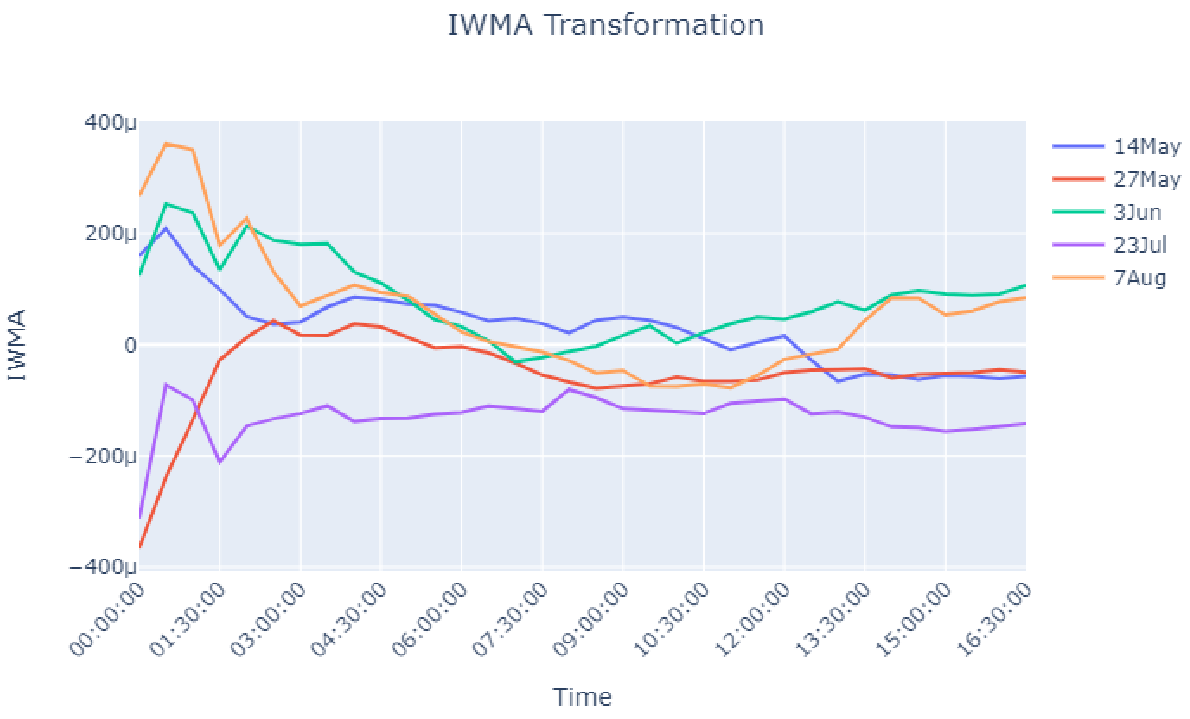


Figure 3.6: Incremental window moving average transformation of different samples

3.3 Classification Layer

This layer's function is very simple but crucial on the entire system functionability, its job is to classify the current market as belonging to one of 3 types, bearish, bullish or sideways, relying on a single *SVM*. This classification will be used on the next layer, the prediction one, as explained before, to decide which neural network to use. It will also be useful for the strategy layer (strategy D) to adjust decision rules of when to enter the market.

The implementation of this layer was done using the *scikit-learn* [32] framework, which ”assures reliability in terms of computation and stability” [6]. It is a very simple and easy to use python library incorporating a wide range of functions and features to work with Support Vector Machines, including data preprocessing and dimensionality reduction [32]. This framework not only facilitates the implementation of the *SVM* in terms of coding, as the depth is very reduced when working with ”high-level” libraries, also improves computational costs as every function is already optimized.

The input features chosen are 3 month price sequence windows and the labels were obtained by classifying the market manually, all the steps for the data processing are explained in detail in section 3.2. This sections explains how this layer is implemented, how the best hyperparameters were searched for and the results obtained.

3.3.1 Performance Metrics

Before deciding which parameters work best for the SVM, first the evaluation metrics need to be chosen. For a multi-class classification problem, relying only on accuracy is not enough. The 3 metrics used are accuracy, precision and recall. Before explaining each, the difference between true and false, positives or negatives, needs to defined. A *true positive* is a sample correctly classified, while a *false positive*, relatively to class ”X”, is a sample classified with the label ”X” when in fact belongs to another class. A *true negative*, still relatively to class ”X”, is a sample that does not belong to it and therefore is not classified as ”X”. Lastly, a *false negative* is a sample classified not belonging to class ”X” when it should in fact be classified as ”X”. The three different metrics used and its' definitions can be seen in table 3.1.

Table 3.1: Definition of different metrics [33]

Metrics	Definition
Accuracy	$\frac{\text{True Positives}+\text{True Negatives}}{\text{Total Samples}}$
Precision	$\frac{\text{True Positives}}{\text{True Positives}+\text{False Positives}}$
Recall	$\frac{\text{True Positives}}{\text{True Positives}+\text{False Negatives}}$

Accuracy represents the percentage of correctly classified samples on the total sample space. **Precision** represents the level of confidence of a classification for a given class, a high level of precision doesn't necessarily mean that samples belonging to the given class are being well classified, but it means that when attributed to that given class there is a high probability of being correctly classified, as there are only a few false positives. **Recall** is the opposite of precision, resembles how likely is a sample to be rightly classified if it belongs to a given class, high values mean that samples belonging to this class are being rightly classified, however for overfitted models the recall can be perfect but the precision very bad, when all samples are being classified as belonging to a given class for example, so the objective of a good performing system is to score high in all of three metrics.

Cross-Validation

A method of cross-validation, the *K-fold* validation, is also applied. The goal here is to make sure the solution is not biased - i.e., overfitted to the data on which it was trained. This method is applied on the training data to validate the chosen parameters and before the final evaluation on the out-of-sample testing data, as described in figure 3.7.

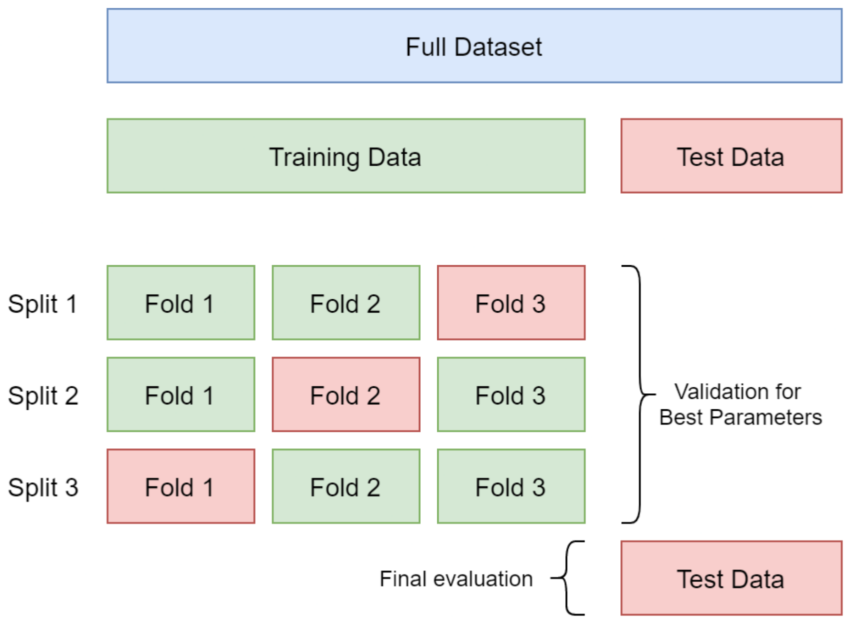


Figure 3.7: Example of *K-Fold* Cross-Validation (K=3)

The *K-fold* strategy is exemplified in figure 3.7 for a K of 3, which corresponds to 3 different folders and 3 iterations. The method is simple, it splits the available testing data in K folders of the same size, and then for each iteration it will train the entire model with the data from every folder except one, which will be used for testing. In the end, instead of a single accuracy value, this method will return K values, one for each folder used as testing, the closer to each other they are, the more stable the system is trained.

3.3.2 Hyperparameters

As described in section 2.4, the first hyperparameter to tune is the type of kernel used, with 3 options available. The simplest one is the linear kernel, it is very fast to train, however is limited for problems where data is linearly separable. Both the radial basis function and the polynomial kernel can approach non linear problems, with the polynomial one being more versatile but taking much longer times during training phase, specially for bigger dimensions - i.e., higher n values.

The *C* parameter is the one regarding the cost function, the higher it is, the bigger the influence of each individual support vector, getting bigger penalizations for wrongly classified samples, which in the limit is extremely dangerous regarding overfitting, specially for less representative training datasets.

Lastly, the *Gamma* parameter, non-existent for the linear kernel, controls the curvature of the decision boundary, with higher values meaning a higher curvature. Again, it depends on the data to find the perfect values for this parameter, however, bigger values are usually prone to overfit as decision boundaries will shrink around samples.

3.3.3 Grid search Validation

In order to evaluate the best parameters, according to the metrics described, a grid search algorithm was applied. The grid search algorithm will train and test the *SVM* multiple times with every combination of *C* and *Gamma* values for the 3 kernels, according to table 3.2. The best solutions will be evaluated and ultimately one set of hyperparameters will be chosen.

Table 3.2: Hyperparameters to be applied on grid search for *SVM*

Kernel		C	Gamma
Linear		$[1; 10; 100; 1000]$	Non-Applicable
Polynomial,	n=2,3,4	$[1; 10; 100; 1000]$	$[10^{-1}; 10^{-2}; 10^{-3}]$
RBF		$[1; 10; 100; 1000]$	$[10^{-1}; 10^{-2}; 10^{-3}; 10^{-4}]$

In addition to the metrics described (accuracy, recall and precision) a cross validation of 5-Fold is also used to discard the possibility of overfitting for the best solutions found. The value 5 was chosen has it is big enough to avoid biased solutions, and keeps the folders with a meaningful size for the training data available. Below in table 3.3 the best parameters found for each kernel, as well as the corresponding performances and cross validations, can be seen.

Table 3.3: Best parameters and results for the SVM

Parameters		**Precision** [%]	**Recall** [%]	**Accuracy**	**Cross-Validation** [%]
		Average			
Linear	C = 1000	[81 ; 86 ; 60] 76%	[60 ; 89 ; 80] 76%	82.35%	[80.9, 49.5, 75.5, 63.1, 78.8]
Polynomial (n=3)	**C = 100 Gamma = 0.01**	**[83 ; 89 ; 68] 80%**	**[73 ; 91 ; 78] 81%**	**85.93%**	**[87.8, 78.1, 84.3, 83.6, 86.3]**
RBF	C = 100 Gamma = 0.01	[99 ; 79 ; 83] 87%	[51 ; 100 ; 4] 51%	80.97%	[60.3, 49.1, 77.1, 70.4, 76.2]

This table provides the values for precision and recall per classification group, from left to right, being the first the bearish markets, then sideways and then the bullish ones. Looking at table 3.3 it is obvious that the best overall solution is the polynomial kernel, even though its computational time is considerably higher, it is the most balanced solution. The linear solution appears to be overfitted just by looking at the parameter *C* which takes the biggest value of all, 1000. Also has a 60% value both for precision and recall, even though it is in different classification groups. The last ”red-flag” is a 49.5% value for one of the 5-Fold cross validations, which is very distinct from all others, once again showing a biased solution. The radial basis function kernel solution also presents signs of overfitting, again one of the cross validation values is less than 50% which is very bad. It is also obvious from looking at precision and recall values, as the precision for the bearish and bullish markets is slightly better, and then the recall for the sideways markets is way bigger, it means that the system classifies most of the markets as sideways, therefore having the perfect recall but not so good precision and vice-versa for the other classes.

The solution chosen to apply on this layer is the polynomial one presented with a degree of 3, a C value of 100 and a Gamma of 0.01, as all values of precision and recall look very stable. The accuracy is also very good at almost 86% and the lowest cross validation value is at 78.1% with a maximum deviation of 9.7%.

3.4 Prediction Layer

This layer is composed by three distinctly trained neural networks and its job is to predict intradaily price movements on a four and a half hours horizon. The final system solution uses the previous layer to classify the market type and then only the correspondent ANN is activated. Each ANN is trained with the corresponding data - i.e., the bullish one will only be trained with samples classified as belonging to a bullish market. The transformation being predicted is explained in section 3.2 and the output of this layer, the prediction of said transformation at *4:30pm*, is passed to the next layer to apply a trading strategy. The implementation of this layer was done using the *TensorFlow* framework, which ”enables fast prototyping, state-of-the-art research, and production” of ANN models [34].

Several tests and interpretations had to be made on this layer regarding the choice of hyperparameters to train the ANN, its topology (hidden layers and nodes) and the performance metrics. For every test presented below, to validate the decisions made, the result of an average of three distinct iterations is presented, given the randomness factor of ANNs one test cannot supply enough confidence to make assumptions on what works better.

3.4.1 Training and Validation

In order to train an ANN, there must be a stopping criteria, otherwise the network will train itself to eternity until it is perfectly fitted to the training data, which, logically, would result in an overfitted network. There are two possible approaches to define the stopping criteria, the simplest one, is to manually define the number of epochs that the training stage will last, which is not the best approach as the optimal solution can still be far from reached or the solution reached is already overfitted. The second approach, used in this work, is to randomly separate part of the training samples to use as validation at each epoch, and stop the training when the error on the validation set starts increasing, whereas the loss on the training set is always decreasing, unless the learning rate is extremely high. The data samples used in the validation set are not presented to the network in the training stage, therefore reducing the size of training samples available. The training dataset is composed of 2340 samples, which represent the *EUR/USD* price rates from 2010 till 2018, while the testing dataset is composed of 260 samples, corresponding to the same price rates on the year of 2019. From the training set, 15% of the samples are chosen randomly to be used as the validation set.

3.4.2 Activation Function

As the values of the IWMA, which represent both features and target values for the ANN, are in the range [-1;1], as explained in section 3.2.3, the activation function chosen is the hyperbolic tangent, shown in equation 3.3, one the of the most common activation functions for MLP networks and which range is the same as the IWMA, [-1;1], as mentioned before in section 2.3.3.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.3}$$

3.4.3 Loss Function and Performance Metrics

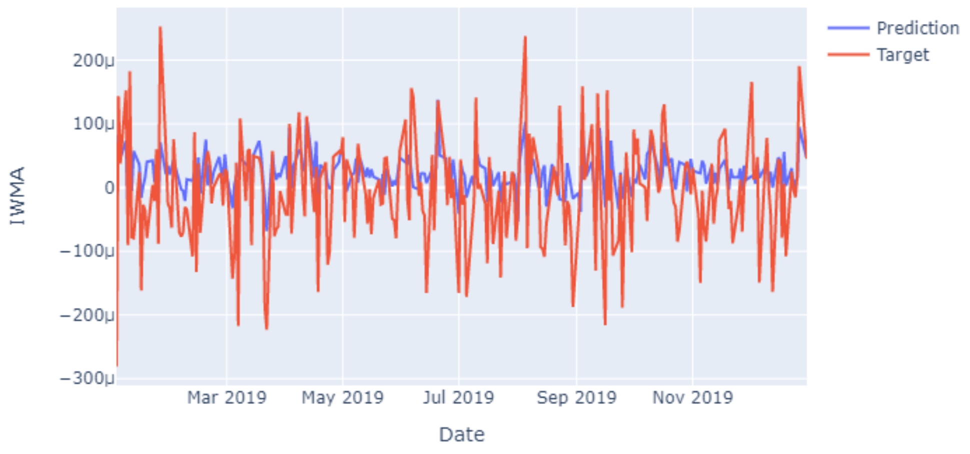
The most common metric when referring to prediction neural networks is the Mean Squared Error, in this work a variation of it was used, the Root Mean Squared Error (RMSE) as described in equation 3.4, where Y_i corresponds to the target value of a sample i and $\hat{Y}_i$ the correspondent prediction. This function includes a square root to eliminate the fact that errors are squared, which happens to keep errors always positive, so that one error does not ”eliminate” another.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2} \tag{3.4}$$

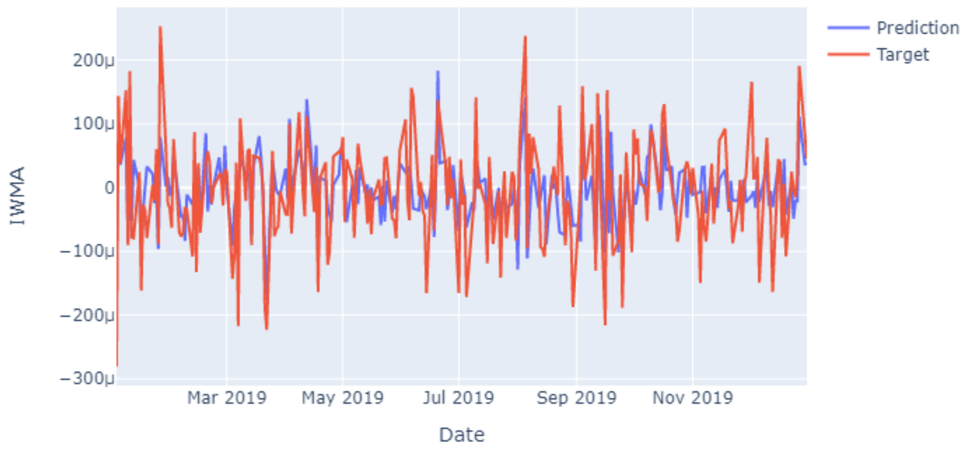
The RMSE is usually a fairly choice for using as the loss function of a neural network, the loss or cost function’s job is explained in detail in section 2.3.5. For this work, however, a different loss function was chosen, the Total Sum of Squares (TSS), which is defined in equation 3.5, even though the RMSE was kept as a performance metric.

$$TSS = \sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{3.5}$$

The choice for this loss function comes from the necessity to improve the convergence of the neural network. It was noticed that the first loss function, RMSE, during the training stage per batches generalizes the solution so much, in order to lower the mean error of the entire batch, that in the end, none of the predictions are close enough to the target values. Instead fluctuate in a smaller range between those. Only for very small learning rates does this function start to provide better solutions, at the cost of a greatly increased computational cost. On the other hand, similar solutions can be achieved, with acceptable values for the learning rate, using the TSS which will give more importance to each prediction error individually. Both the phenomenons described can be seen in figures 3.8 for a similar network topology and training parameters, where the only difference is the cost function.



(a) RMSE as loss function



(b) TSS as loss function

Figure 3.8: Comparison between RMSE and TSS as loss functions

Another performance metric used in the next evaluations is the accuracy, which at this point is considered to be the relation between prediction and target value in reference to the point at *12:00pm*, instead of the actual trade accuracy, as a trade strategy is not in place yet, and is represented in table 3.4 has *Accuracy**. This relation will be proven later, in the strategy implementation 3.5, to be one of the most important things to predict accurately. A sample is considered rightly predicted, for this specific accuracy computation, when it is in the same direction as the target value in relation to the initial value, at midday - i.e., if both (target and prediction) are higher or both lower than the initial value, even if distant from each other, than it is an accurate prediction. This will be proven not to be always true when comparing the transformation to the actual price movement, so the total loss of the network is still very important to keep the predictions and the target values as close as possible to each other.

3.4.4 Network's Topology

One of the most important things in a neural network is its topology - i.e., the number of layers and nodes per layer. The input and output layer is given by the problem itself, in this case, the input layer has 25 neurons, the number of features, and the output layer has 1, the single prediction the network is supposed to output. So the margin to optimize an ANN to a specific problem is in the hidden layers and nodes. There is not a rule on how to build an ANN, any topology is valid as long as it suits the problem. There are, however, a few guidelines that fit the majority of problems. For example, the number of neurons on a given layer should be in between the number of nodes from the last and the following layer [35], having more neurons than inputs would mean redundant neurons learning the same functions as each other.

The computational time to train an ANN increases with the number of layers and perceptrons, also having more nodes in the entire network increases the risk of overfitting the training dataset. So an ANN, usually, will be grown in either depth or width until it fits the problem. Deep networks, meaning a considerable amount of layers and less neurons per layer, can be better to find specific patterns on complex problems, as they will shrink the solution by passing every sample through a lot of layers, so the network can extract more features from the inputs provided, which comes at the expense of possibly overfitting, depending on how large and representative is the training dataset. On the other hand, wide networks, meaning less layers and more neurons per layer, can have a better generalization, since it is possible to find more patterns on data, as there are more neurons on the final prediction and not a shrinked solution.

In order to find the optimal solution, for the topology of this work's ANNs, a series of tests were made on different topology configurations, with more or less layers and neurons. After having a validation of which kind of configuration has better performance, a test on a narrower range of similar topologies is applied. On table 3.4 the results of the tests applied on different kinds of topologies can be analyzed.

Table 3.4: Evaluation metrics for different ANN topologies (Average of 3 tests per configuration)

Topology	Total Loss (TSS)	RMSE	Accuracy*
[12]	4.024$\times10^{-9}$	4.742$\times10^{-5}$	62.1%
[20]	3.926$\times10^{-9}$	4.695$\times10^{-5}$	**62.7%**
[12 - 8]	4.17$\times10^{-9}$	4.904$\times10^{-5}$	56.2%
[20 - 16]	4.278$\times10^{-9}$	4.969$\times10^{-5}$	57.6%
[12 - 10 - 8 - 6]	4.223$\times10^{-9}$	4.86$\times10^{-5}$	58.6%

The tests provided the expected results, given the unpredictability and noise of the forex market it was expected that wider networks would behave better. As stated, fewer layers prevent overfitting, and even though the error metrics are not as different, the accuracy regarding the direction of the movement on the IWMA proves that a single layer with more neurons is the better approach. The next search is done on a narrower range of similar configurations, changing only the number of neurons. An additional grid search, according to table 3.5, for each of the topologies was done, in order to discover the best learning rate and batch size for the network to use. The results of this entire validation set for the best parameters are provided in table 3.6.

Table 3.5: Hyperparameters to be applied on grid search for *ANN*

Batch Size	$[4; \quad 8; \quad 16; \quad 32]$
Learning Rate	$[10^{-3}; 10^{-4}; 10^{-5}; 10^{-6}]$

Table 3.6: Evaluation metrics for optimal topologies (Average of 3 tests per configuration)

Topology		Total Loss (TSS)	RMSE	Accuracy*

[18]	BS = 4
	LR = 10^{-5}
		3.937$\times 10^{-9}$	4.719$\times 10^{-5}$	60.25%

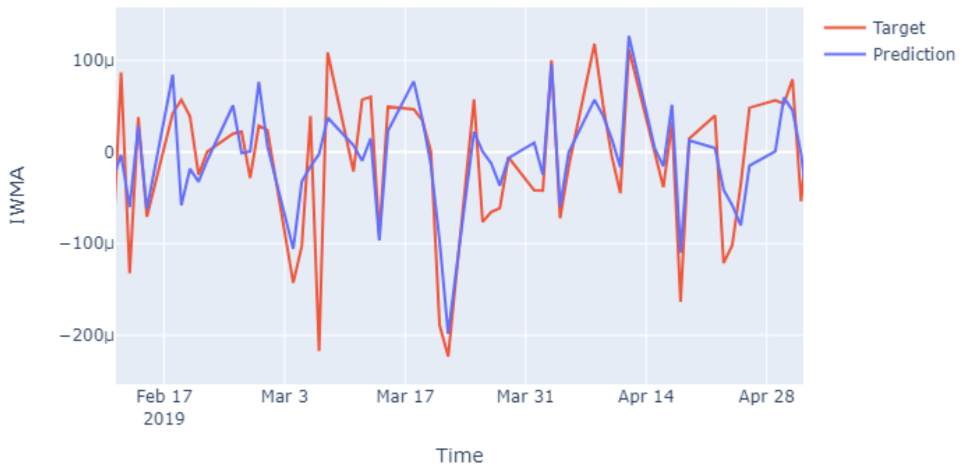
[20]	BS = 8
	LR = 10^{-5}
		3.926$\times 10^{-9}$	4.695$\times 10^{-5}$	62.7%

[22]	**BS = 8** **LR = 10^{-5}**	**3.803$\times 10^{-9}$**	**4.597$\times 10^{-5}$**	**63.5%**

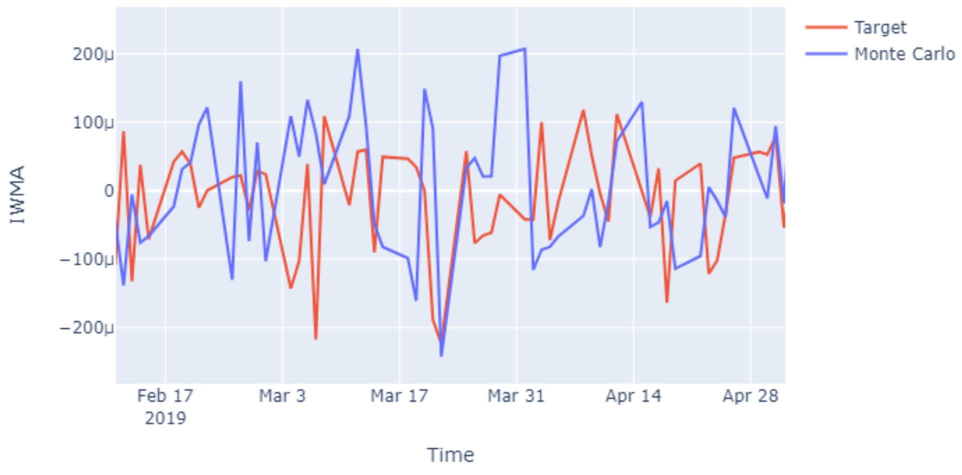
[24]	BS = 4
	LR = 10^{-6}
		4.362$\times 10^{-9}$	5.025$\times 10^{-5}$	57.44%

The network which performed better was the one with 22 perceptrons in one hidden layer, with an accuracy of 63.5% (over IWMA direction as mentioned) and a RMSE of 4.596×10^{-5}. The best learning rate was 10^{-5} and a batch size of 8 was used. This is the topology used during chapter 4 to evaluate the results obtained by the final system. An example of the comparison between target values and prediction, of this network, on a partial test range, is shown in figure 3.9, alongside a Monte Carlo simulation with equal mean value and standard deviation to the target data, which clearly validates the network against a random distributed sample.

To conclude this section, it is important to mention that only one generalized neural network with the entire training dataset was submitted to this test, which is not the optimal solution regarding the final system, as each network should be tuned to its optimal state, but it is required for the study cases to be coherent, for example, if the SVM actually improves the system or is irrelevant.



(a) Target values and Prediction for optimal topology



(b) Target values and Monte Carlo simulation

Figure 3.9: Validation of chosen network against Monte Carlo simulation

3.5 Strategy Layer

This layer is responsible for ultimately find a way to make money in the forex market based on the *ANNs* predictions of the IWMA transformation, on a 4 and a half hours horizon. For this purpose, a detailed study on the relation between the IWMA and the actual price during the day was made, to better comprehend the patterns and possibilities on what can happen to the price for a given prediction, which is described later in this section.

For this purpose, 3 different strategies, regarding the logic behind the decision to make, are implemented, and tested against each other in chapter 4, to evaluate the pros and cons of each. A fourth strategy is also implemented and tested, this one focus on a dynamic trading size based on the logic decisions of the third, and most complex, strategy. The decision tree of the strategy layer can be seen in figure 3.10 and each strategy and the theory behind it is explained below in detail.

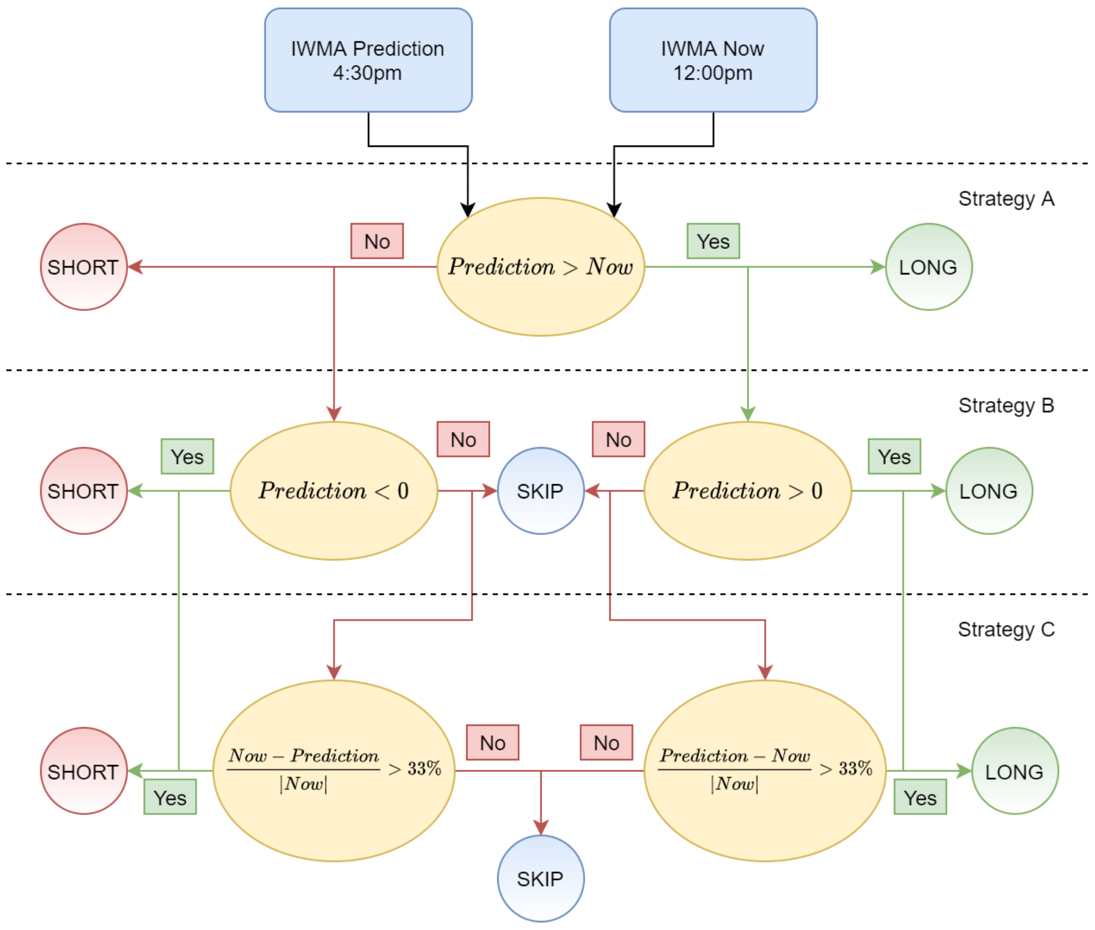
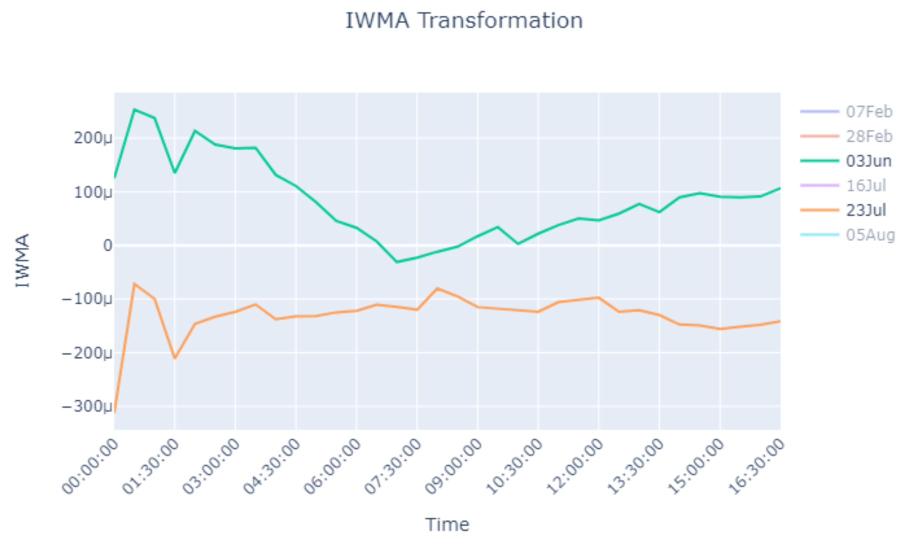


Figure 3.10: Diagram of the strategy layer decision tree

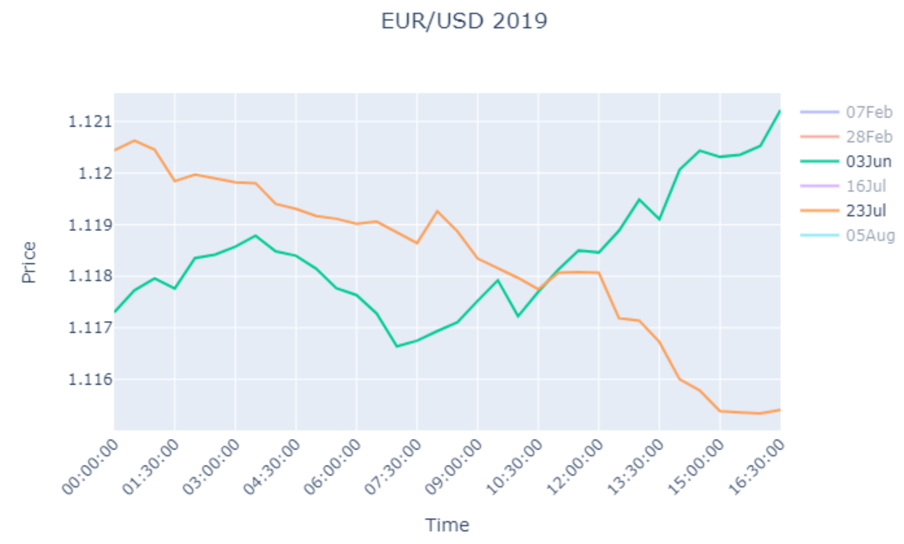
3.5.1 Strategy A

First of all, it is important to understand the meaning behind the transformation that is actually being predicted. As explained in section 3.2, the final transformation, the IWMA on the return rate from one price point to another, is an accumulated average of how much the price moved during the day. The basic concept behind this prediction, which will be proved to not always be true, is that if this average is bigger at *4:30pm* than at *12:00pm*, than the price had bigger moves up from *12:00pm* to *4:30pm* than down, and is expected to be higher in the horizon predicted. The same happens in the other direction and both these examples, the most common ones, can be seen in figures 3.11, where this relation between the transformation being predicted and the actual price movement is evident.

From this assumption, *strategy A* was created as seen in figure 3.10, it is the simplest strategy of all and consists of simply taking a long position, buying the first currency (EUR), when the prediction is bigger than the IWMA at the moment, or taking a short position otherwise, buying the second currency (USD), with no possibility of skipping the trade.



(a) IWMA example A



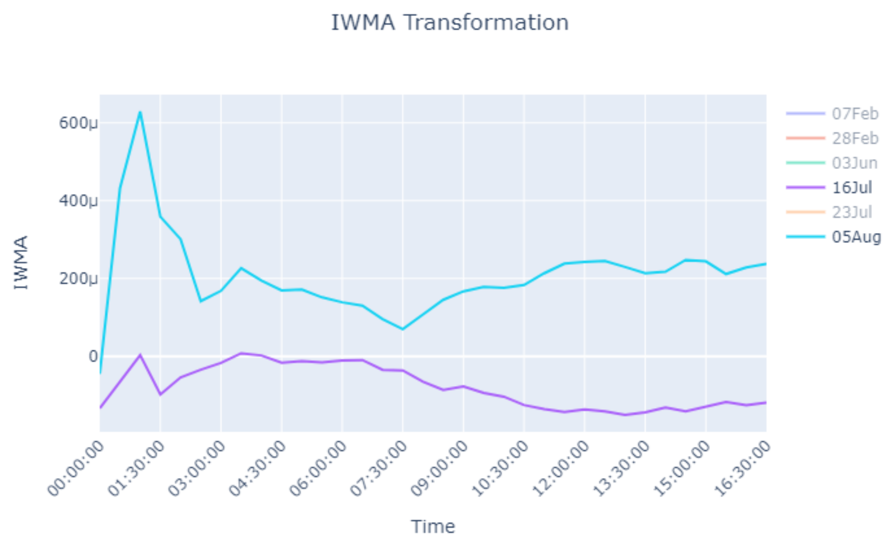
(b) Price example A

Figure 3.11: Relation between IWMA and Price for *strategy A* scnerario

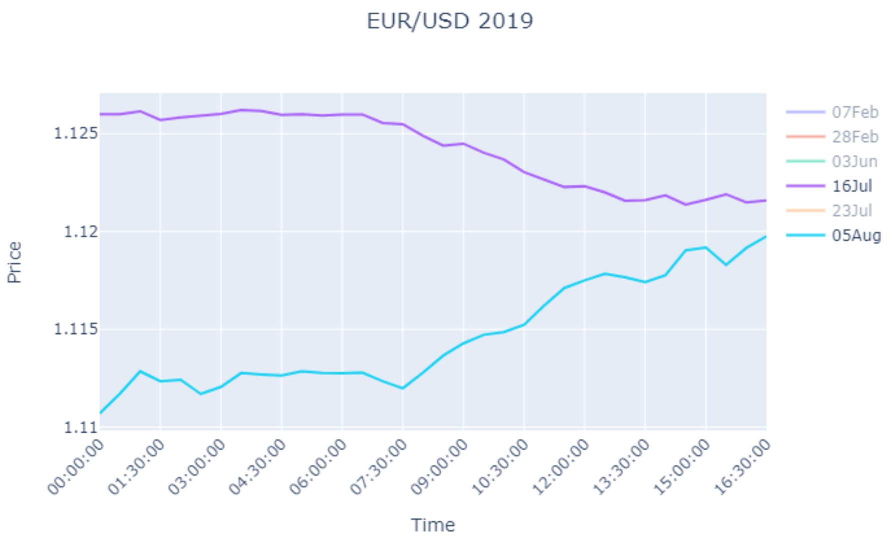
3.5.2 Strategy B

The second pattern spotted on the data is that it is possible that the IWMA decreases but the price increases anyway, the opposite is also valid. This cases were searched for and found to happen for positive IWMA values when these are decreasing and the price increasing and negative for the opposite case. Both this examples can be seen in figures 3.12, the explanation for this event is that the price increases abruptly during the day, for example, making the IWMA on the return rates having high positive values, and then from *12:00pm* to *4:30pm*, even though it keeps increasing, it slows the rate at which it is doing so, making the new values for the return rate lower than the previous ones and, consequently, the final IWMA lower. The opposite example is also shown, which works the same way.

From this new assumption, *strategy B* was created as seen in figure 3.10, from the confirmation



(a) IWMA example B



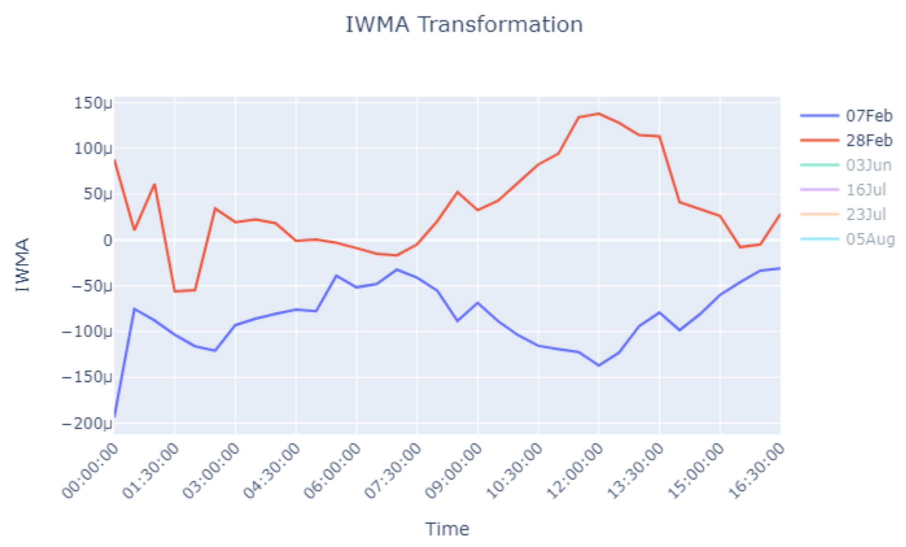
(b) Price example B

Figure 3.12: Relation between IWMA and Price for *strategy B* scnerario

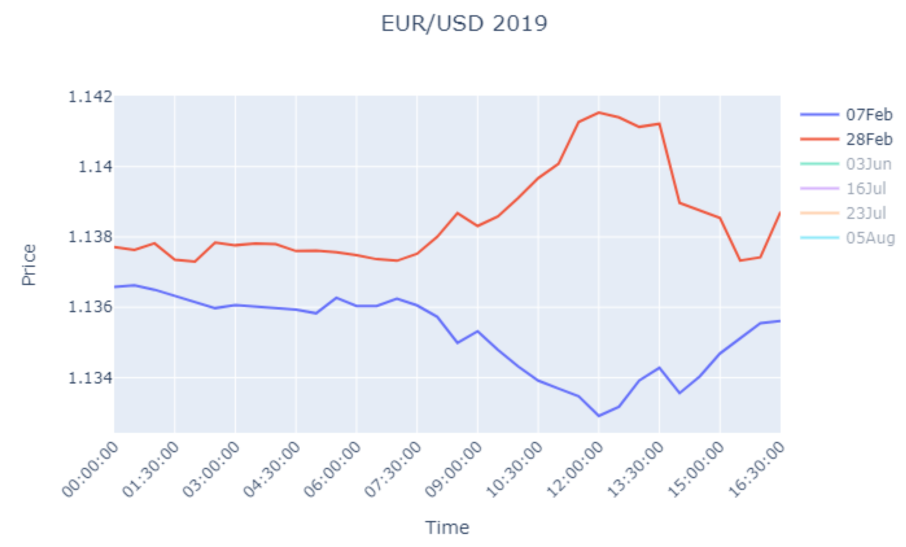
strategy A makes, if the prediction is bigger than the current value or not, makes a new confirmation. If it is in fact bigger, checks if this happens in the positive range of values, and only so will go long, otherwise it will skip the trade, which was not a possibility for *strategy A*. Again, the opposite is also valid, if the IWMA decreases, the short trade will only be activated on the negative range of values. This is the safest pattern, where the price direction is guaranteed as long as the prediction is accurate, which is not this layer's function. When the IWMA predicted is bigger than the current value in the positive range, it means that the price is rising from midnight until noon, and then until *4:30pm* rises even more, this pattern was found to be true for the entire dataset, even though it does not happen that often. So the problem for *strategy B* is that it skips too many trades.

3.5.3 Strategy C

The third pattern found on data is that it is possible for the IWMA to increase in the negative range of values and the price to increase as well, which is the opposite of what *strategy B* is based on. This is frequent for price correction during the prediction horizon. For example, the price would decrease abruptly during the day, making the IWMA have very high magnitude negative values, and then from noon till *4:30pm* would correct with a considerable move up, making the IWMA rise as well, while staying in the negative range. The opposite remains valid, and both this examples can be seen in figures 3.13.



(a) IWMA example C



(b) Price example C

Figure 3.13: Relation between IWMA and Price for *strategy C* scnerario

From this final assumption, *strategy C* was created as can be seen in figure 3.10, in order to respect the risks avoided by *strategy B* but at the same time benefit from trades as shown in figures 3.13. This strategy consists of a new confirmation before skipping a trade that *strategy B* would skip, while a long or short trade from *strategy B* happens the same way. This confirmation is based on how big is the movement predicted in the IWMA, if it is bigger than 33% from the current value, either up or down, then it is assumed as a big correction move and the trade is executed anyway, as it would if happening in the "right" range of values. Otherwise, it is considered a "strategy *B* case" and the trade is skipped.

3.5.4 Strategy D

A fourth strategy is also considered, which consists of adjusting the trading size dynamically, instead of the trading rules as the other 3 strategies' focus. This one, *strategy D*, has the same trading rules as *strategy C*, with a slight variation, but instead of a fixed trading size, which depends solely on the available balance in the other strategies, it will use a dynamic volume which depends on the magnitude of the change of the IWMA, computed by *strategy C*. The goal behind it is to benefit on a larger scale from safer trades, where the correction is expected to be big, and reduce the risk on trades closer to the edge of possibly keeping the price in the same direction even when the IWMA is moving in the other. The only variation from *strategy C* is the 33% value used as threshold, which in *strategy D* varies according to the type of market classified by the *SVM*, as defined in table 3.7

Table 3.7: Threshold variation values for *strategy D*

Market	Prediction	Threshold
Bullish	$Prediction > Now$	20%
	$Prediction < Now$	40%
Sideways	$Prediction > Now$	33%
	$Prediction < Now$	33%
Bearish	$Prediction > Now$	40%
	$Prediction < Now$	20%

3.5.5 Trading Size

The trading size used for the first 3 strategies depends exclusively on the current balance, and has a proportion of 1 lot per 10,000€, which corresponds to a leverage of 10. The trading size for *strategy D* has the same proportion between balance and lot and is then multiplied by 1.5 and the percentage variation in the IWMA, as shown in equation 3.6, making it range from 0.5 lots to 2 lots per 10,000€, the 2 lots maximum is a limited imposed manually, which corresponds to a leverage of 5 till 20.

$$\begin{gathered} s = \frac{|Prediction - Now|}{|Now|} * 1.5, \quad s \leq 2 \\ volume = s * \frac{Balance}{10,000} \end{gathered} \tag{3.6}$$

While this leverage values might seem high, it has to be considered that this system is build to have open positions in the market for 4 and half hours per day, which is a very small timeframe, so big leverages need to be in use for this approach to be profitably worth it. Even though, a certain level of conservatism is maintained as the maximum leverage used is 20 which is only 2/3 of the maximum allowed leverage in Europe, which is 30 for the forex pair *EUR/USD* [36].

It is important to note that the considerations for this layer were made on the real data, and not the prediction that comes from the ANN - i.e., this layer is optimized to work with an ANN that is doing a perfect prediction. Otherwise, everytime the ANN got optimized there was the possibility that this layer was counting on previous errors, and therefore the final system accuracy would get worse instead of improving alongside the ANN improvements, for example, the inclusion of the classification layer and 3 different ANNs.

3.6 Chapter Conclusions

This chapter summarizes the practical work done behind the proposed trading system, all the technologies used and the libraries and frameworks to achieve it. From the data preparation layer, until the final decision being taken at the strategy layer, which depends on the outputs of both the prediction and classifications layers, every single implementation is described in detail, as well as the validation tests taken to make sure the optimal solution was being achieved.

Chapter 4

System Evaluation

This chapter presents all the tests to which the final trading system was subject too, as well as specific case studies in order to evaluate the best possible approach. Firstly the measures used, for the purpose of evaluating the possible solutions, are described. On a second moment, three distinct case studies, each focusing on different possible approaches for a specific area of the final trading system, are described in detail, as well as the results obtained for each and the conclusions taken from the experiments. Lastly, the final solution is compared to the classical approaches of *B&H* and *S&H*.

4.1 Evaluation Metrics

In order to evaluate the performance of the system on the several tests presented in this chapter, as well as compare different possible implementations, as in the study cases analyzed below, rigorous evaluation metrics are needed. Three different metrics were used to test the final system on this chapter, the trade accuracy, return on investment and drawdown. In order to compare networks the "Accuracy* (IWMA)" is also provided.

4.1.1 Trade Accuracy

Different than the accuracy referred to in section 3.4.4, which was relative to the direction of the IWMA, the trade accuracy simply measures how many trades were successful or not, as a percentage of the entire testing range, as shown in equation 4.1.

$$Accuracy[\%] = \frac{Profitable\ Trades}{Total\ Trades} \times 100 \tag{4.1}$$

This is a crucial measure to start with as it gives a general idea if the system is predicting price movements correctly or not, when considerably above 50% randomness can be discarded. It is not ideal, however, to compare different strategies and approaches, as it does not give an idea of how much money is being made in the market, one could have a great trade accuracy but lose more money in the few trades missed than all of the successful ones combined.

4.1.2 Return on Investment

The return on investment (ROI) is by far the most widely used metric in the financial markets, specially when accessing an investment or financial product. In its core, is the actual measure of the investment itself, returns a percentage that indicates how much was the profit or loss of said investment relatively to itself, as shown in 4.2.

$$ROI[\%] = \frac{Return - Investment}{Investment} \times 100 \tag{4.2}$$

This is the best metric to compare how well a given system performed in the market, as in the end, one can know how much would his investment grow if applied on a system with a given ROI. This measure, however, has no information about the risk of said investment, which is also a crucial thing for any investor, as no one wants to have their assets at risk, or at least an unknown risk.

4.1.3 Drawdown

A popular metric to access the risk, or have an idea of how much was the risk to which the system was exposed in the past, is the drawdown. This, measures the magnitude of the falls that happen in the total balance, over the amount of time said investment is active, in this case, while the trading system is running. The drawdown returns a percentage indicating how much the investment lost relatively to the last peak, as shown in 4.3

$$Drawdown[\%] = \frac{Last\ \ Peak - Local\ \ Minimum}{Last\ \ Peak} \times 100 \tag{4.3}$$

4.2 Case Studies

This section describes three different case studies regarding decisions made on distinct layers, for the final trading system presented in this work. In each case study several options are analyzed and tested against each other, in order to optimize the final system.

1. The first case study compares different time ranges regarding the prediction horizon used by the trading system.
2. The second case study analyzes the inclusion of an SVM to classify the type of market and consequently the use of 3 different ANNs by the trading system.
3. The last case study compares the different strategies implemented by the strategy layer, in addition to the strategies of *B&H* and *S&H*, to evaluate the final performance of the system.

For the training of the ANNs, the data used is referent to the years of 2010 till 2018, while the training of the SVM is done with data from 2004 till 2018. All the case studies are tested and simulated on the year of 2019, which is never presented to the AI algorithms during the training stage. For every test presented below, the result of an average of three distinct iterations is presented, given the randomness factor of ANNs one test cannot supply enough confidence to make assumptions on what works better. It is also important to note that the only commission taken into account by the trading system is the *spread* for each trade, as swap and other commission do not apply. The *spread* considered is 2 pips per trade, which is slightly higher than the average *spread* at the time the trade is entered (1.7 pips) [37].

4.2.1 1st Case Study - Prediction Time Window

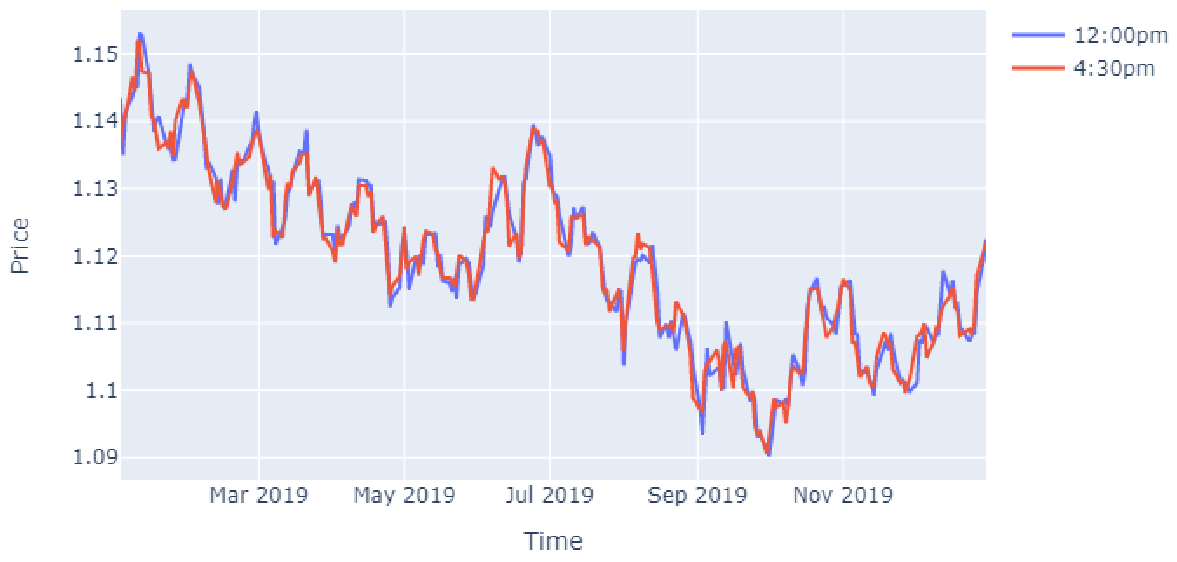
The first case study compares two different time ranges for the prediction horizon which the trading system uses. It is important to note that the system at this point is composed of a single ANN and the strategy in place to calculate the ROI is the first and simplest strategy, presented in 3.5.1. The first time range is of four and a half hours, from *12:00pm*, time at which the system makes a prediction and opens a trade, until *4:30pm*, time of the prediction itself and at which the trade is closed. The second time range is of six hours, from *1:30pm* until *7:30pm*. The price variations on the FX pair *EUR/USD* over the year of 2019, between both this time ranges, can be seen in figure 4.1.

It is possible to see that the red line deviates more from the blue line in the second example, which means the average price variation is bigger, as expected for a bigger time window. Both this values and the performance of the ANN in both situations can be seen in table 4.1, alongside the results of the simulation for both time ranges.

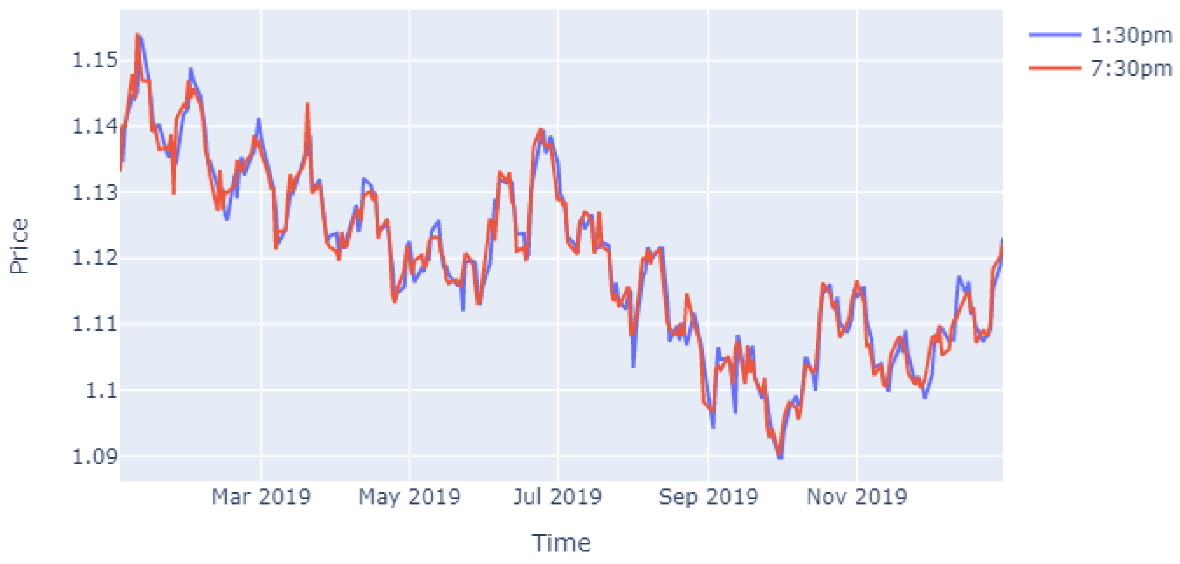
Table 4.1: Evaluation metrics for different market hours strategy (*EUR/USD* 2019)

Measures		Entry at *12:00pm* Close at *4:30pm*	Entry at *1:30pm* Close at *7:30pm*
ROI		27.5%	-23.7%
Max Drawdown		20.6%	27.2%
Accuracy		54.6%	51.5%
Total Trades		260	260
	Longs	87	80
	Shorts	173	180
Max ROI		43.0%	0%
Min ROI		-20.6%	-33.7%
Accuracy* (IWMA)		63.5%	60.8%
Average Price Variation		17.6pips	19.7pips

Even though, in average, there is a smaller price variation between *12:00pm* and *4:30pm* (17.6 pips) than between *1:30pm* and *7:30pm* (19.7 pips), which means less margin to make a profit, the system showed to perform considerably better for the first case scenario, as can be seen in figure 4.2, with a ROI of 27.5% when compared to a negative ROI of -23.7%. The trade accuracy is also slightly better, 54.6% compared to 51.5%, which is too close to 50%, explaining the negative ROI. However, the IWMA accuracies of the networks are very similar, 63.5% and 60.8%, which indicates a good performance of both, so the problem is not in the prediction of the IWMA transformation, but in the relation of this to the



(a) *EUR/USD* 2019 rates between *12:00pm* and *4:30pm*



(b) *EUR/USD* 2019 rates between *1:30pm* and *7:30pm*

Figure 4.1: Comparison of *EUR/USD* rates during both time ranges for 1st case study

actual price movement.

The fact that the relations, explored in detail in section 3.5, between IWMA and price movement are not as coherent for the second time range, as for the first, has several reasons. The first, and most logical, is the fact that the time window is bigger and, consequently, there is more time for unpredictable variations. Beyond, the pattern explained for the second scenario (strategy B) 3.5.2, also happens more often. There is more time for the price to keep growing steady, for example, but with a lower return rate each timestep, and consequently the IWMA will be lower in the end, even though the price is bigger.

In addition to the size of the prediction horizon, the actual hours at which the trades are being opened

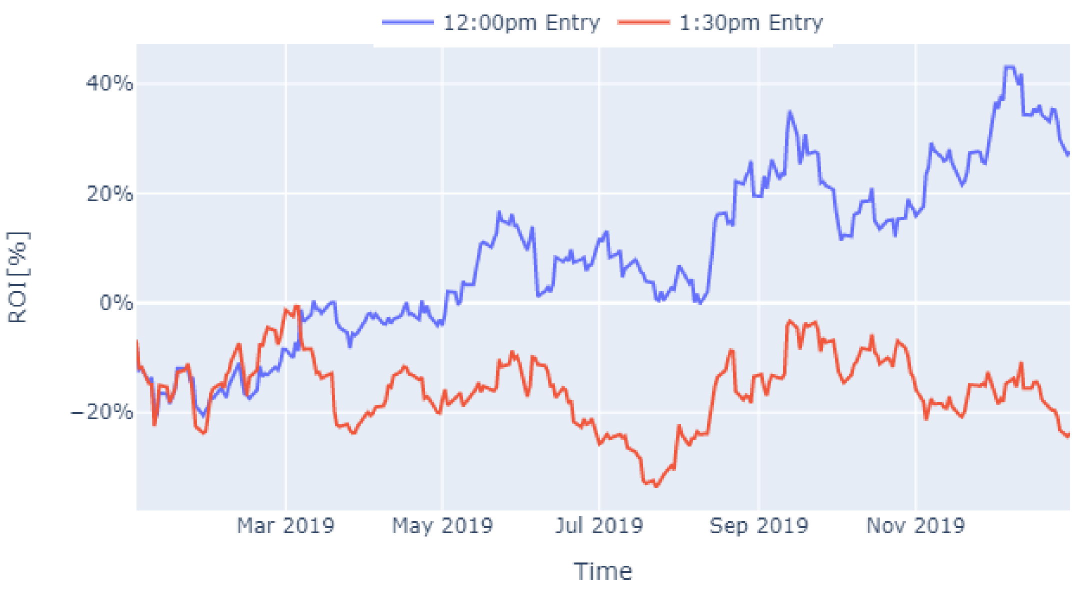


Figure 4.2: ROI over time for different market hours strategy (*EUR/USD* 2019)

and closed is also relevant. The *EUR/USD* market has the biggest trading volumes, usually, between *12pm* and *4pm*, as shown in figure 4.3, consequence of having both the European and American trading sessions running during this time. With more active traders on the market, movements can be more predictive and impulses bigger, corrections and unexpected movements usually happen after hours, as there is less volume in the market, so the price is subject to a bigger volatility and the will of who is actually in the market. A highest trading volume also means a narrower spread [21], which is the only commission this trading system is subject too, so it is crucial that it assumes the minimum possible value. Lastly, the daily exchange rate considered and used by most official sources, such as banks and exchange offices, in Europe, is taken between *4:00pm* and *4:30pm* [38], which means that the ”big players”, as banks and hedge funds are usually referred to, will be in the market making sure the price is in the desired zone by this time, once again, making this the most predictive time window to trade at.

Two conclusions can be taken from this case study, the longer the time window for a market prediction, the bigger the profit margin one can make, however, the unpredictability of the market rises and so does the difficulty of solving the problem as suggested in this work. Secondly, the peak hours of the market - i.e., the time range with the biggest trading volume, represented mostly by banks and hedge funds from both Europe and the USA, presented to have the most predictable, and well defined, price movements.

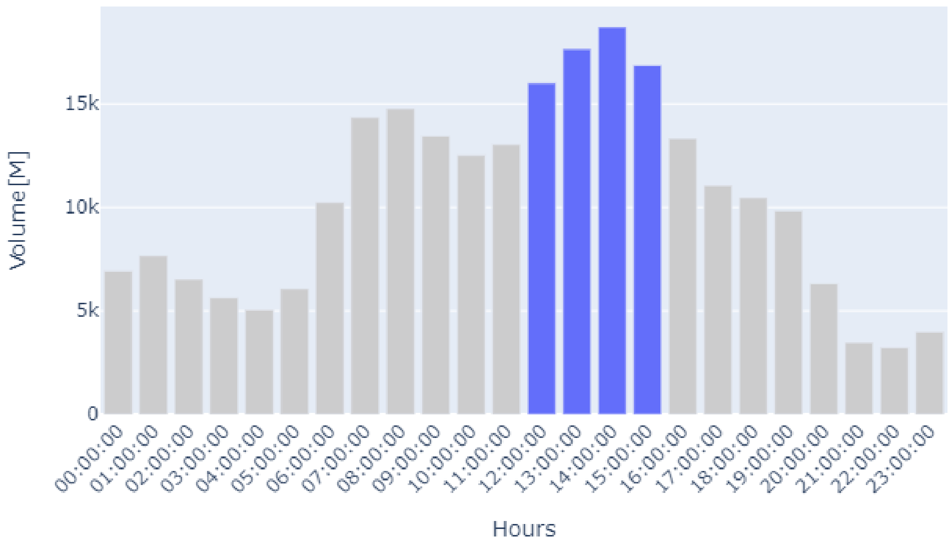


Figure 4.3: Average Traded Volume per hour (*EUR/USD* 2017-2019)

4.2.2 2nd Case Study - Market Classification

The second case study compares the performance of a single ANN to the full system with 3 ANNs, distinctly trained and being activated according to the SVM classification of the type of market, either bullish, bearish or sideways, as explained in detail in section 3.3.

The first step is to classify the testing data referent to year of 2019 with the previously trained and validated SVM. This classification was done with an accuracy of 81.9% referent to the classification done manually, as can be seen in table 4.2, alongside the number of accurate classifications per type of market.

Table 4.2: Classification of actual SVM versus "hand-labeled" data (*EUR/USD* 2019)

Measures	**Manual Labels**	**SVM Classification**	
		Accurate	Missed
Samples	260	260	
Bearish	70	32	5
Sideways	190	181	42
Bullish	0	0	0
Accuracy	-	81.9%	

During the year of 2019, the forex pair *EUR/USD* was never considered to be in a bullish market, which the SVM classified accurately, the missed classifications are mostly in the transition zones between sideways and bearish markets, which even when being labeled manually, are dubious to classify, as there is not a written rule one can follow, as it happens with the ANN forecasting, which ultimately the price either rose or felled during that day, and the prediction is either correct or wrong.

The next step is to use the respective ANN to forecast the price movement for each day, according to the type of market classification for that day, given by the SVM. The results obtained for the complete system, over the testing period, are shown in table 4.3 alongside the performance obtained by a single ANN. At this point, only strategy A is being used and the evolution of the ROI for both this approaches can be seen in figure 4.4.

Table 4.3: Evaluation metrics for single ANN versus SVM complete system (*EUR/USD* 2019)

Measures	Single ANN	SVM System
ROI	27.5%	31.4%
Max Drawdown	20.6%	20.2%
Accuracy	54.6%	53.8%
Total Trades	260	260
Longs	87	95
Shorts	173	165
Max ROI	43.0%	55.5%
Min ROI	-20.6%	-15.0%
Accuracy* (IWMA)	63.5%	61.9%

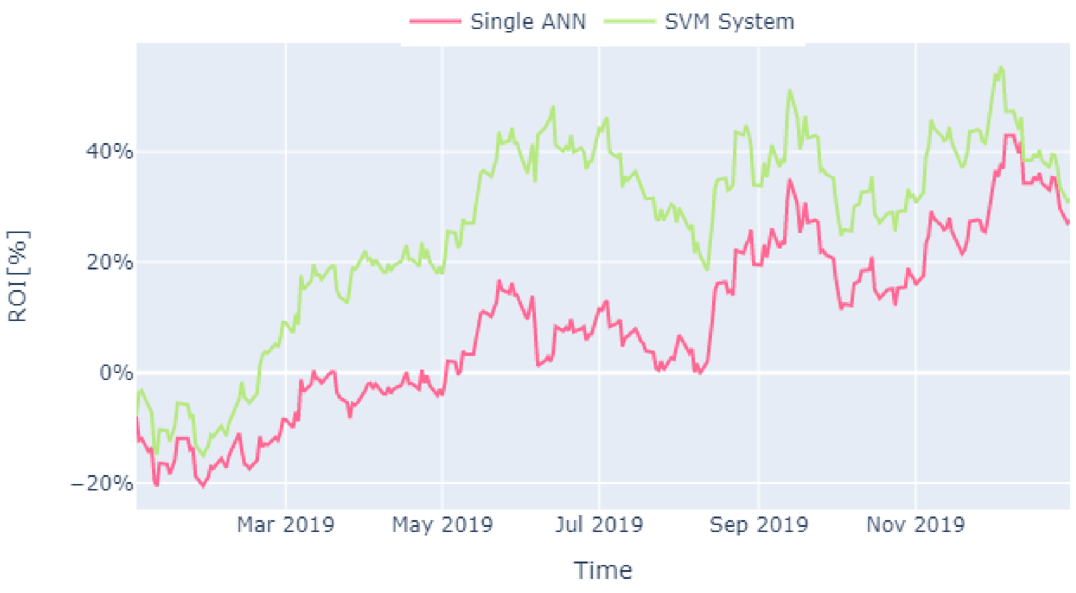


Figure 4.4: ROI evolution for a single ANN and the hybrid SVM-ANNs system (*EUR/USD* 2019)

Looking at the performance measures, it is clear that the SVM system outperformed the single ANN, with a final ROI of 31.4% compared to 27.5%, it also peaked the ROI at 55.5%. The maximum draw-downs are very similar, 20.2% for the complete system versus 20.6%, which indicates that this approach is not yet mitigating the risk. It is interesting, however, that both the accuracies measured are better for the single ANN system, this can be due to the fact that the SVM missed roughly 20% of the classifications. So for wrongly classified markets, an ANN trained with data belonging to a different type of market is being used, proving that a generalized ANN performs better under all circumstances, and this system depends on a correct classification previously done by the SVM.

Analyzing figure 4.4, it is possible to see that at the beginning the ROI for the complete system grows much faster than the one of a single ANN, it is only after July that a fall happens for the SVM system and not for the single ANN, bringing both ROIs much closer. This is where the wrong predictions of the SVM are happening, harming the final system composed of 3 ANNs.

From this case study, can be concluded that the complete system using distinctly trained ANNs, for each type of market, outperforms a single ANN, as long as the prediction done by the SVM is accurate. If this fails, the ”overfitted” networks, can do more harm than good, so the next step is to apply the strategy layer, in order to fully benefit from the classification done by the SVM, while avoiding the missed trades.

4.2.3 3rd Case Study - Trading Strategy

The third case study compares three different trading strategies regarding the decision logic behind the direction of a trade to take in accordance to the prediction of the IWMA, plus a fourth strategy regarding the size, or volume, of each trade. In addition to the strategies implemented by the system and described in detail in section 3.5, the strategies *B&H* and *S&H*, for the same test period, are also implemented.

Buy & Hold **and** ***Sell & Hold***

Buy & Hold and *Sell & Hold* are the two most classical approaches on a financial market, based on the fundamental that markets follow trends and fluctuations are not predictable but a consequence of the trading noise, this strategies simulate the returns of an investment as if one would buy the asset and hold - i.e., keep the position open until the end of the testing period. In the specific case of the forex market, the *B&H* strategy corresponds to keep a long position opened, while the *S&H* strategy represents keeping a short position opened. For this case study, a leverage of 10 was used in both this strategies as it is the leverage used by the trading system and the results would not be comparable otherwise.

The results obtained for each strategy during the year 2019 are presented in table 4.4 and a graphical representation on the evolution of the ROI for each strategy is shown in figure 4.5.

Strategy A is the simplest of all strategies, never skips a trade, totaling 260 trades over the entire test period, making it the riskiest strategy of all, reflected in a maximum drawdown of 20.2% and a minimum ROI of -15%. The final ROI obtained, 31.4%, was enough to beat the *S&H* strategy which only obtained a ROI of 21.5%. However, the maximum drawdowns, 20.2% and 20.5%, are very close, which supports the claim that strategy A is still too dependent on the market, not providing a better risk than a fundamental analysis of staying short all year long. Looking at figure 4.5 is also possible to understand that strategy A movements on balance are very inconsistent, similar to the *S&H* ones.

Table 4.4: Evaluation metrics for strategies proposed (*EUR/USD* 2019)

Measures	Strategy A	Strategy B	Strategy C	Strategy D	B&H	S&H
ROI	31.4%	36.1%	73.4%	87.5%	-21.5%	21.5%
Max Drawdown	20.2%	16.2%	11.9%	13.0%	56.3%	20.5%
Accuracy	53.8%	53.4%	57.4%	58.3%	0%	100%
Total Trades	260	73	176	156	1	1
Longs	95	3	18	17	1	0
Shorts	165	70	158	139	0	1
Max ROI	55.5%	53.4%	87.4%	108.1%	8.4%	52.8%
Min ROI	-15.0%	0%	0%	-2.6%	-52.8%	-8.4

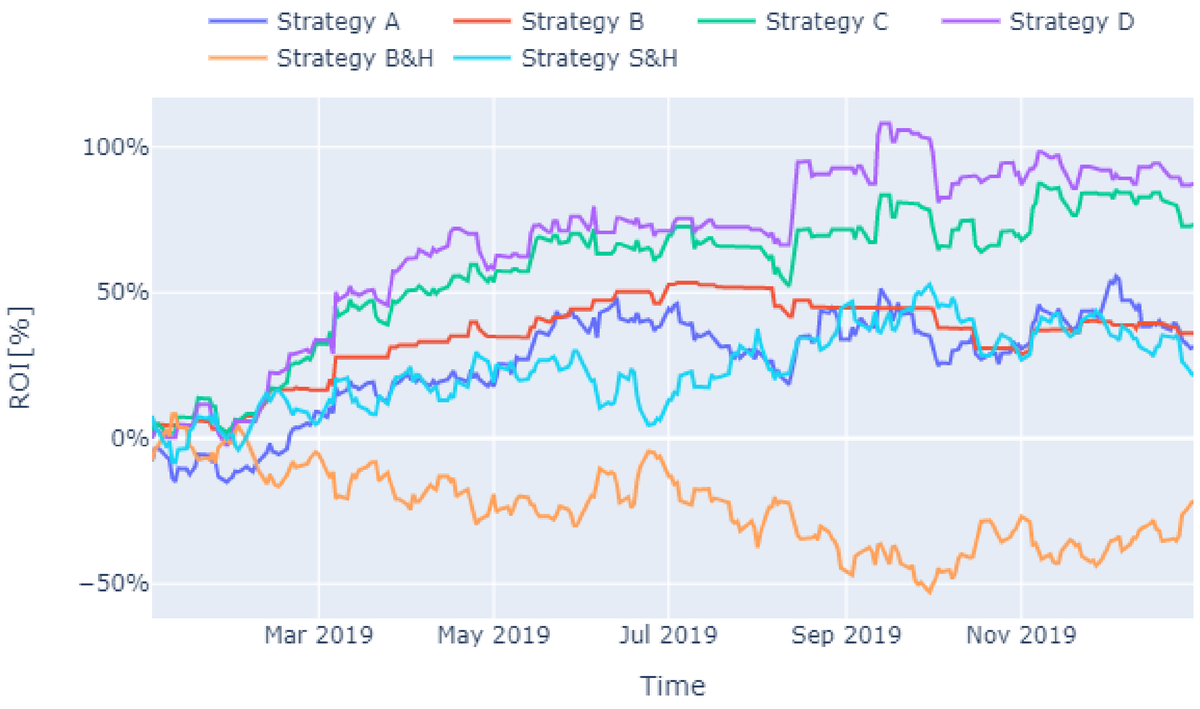


Figure 4.5: ROI evolution for every strategy proposed (*EUR/USD* 2019)

Strategy B is the most conservative of all strategies, the one with less movements, corresponding to a total of 73 trades. Being the safest strategy in terms of price movement confidence, it is expected to present the smallest risk when taking a trade, which should be reflected on the drawdown, besides the fact of never having a negative ROI. Even though, strategy B presented a bigger maximum drawdown, 16.2%, than strategies C and D, by looking at figure 4.5, it is possible to understand that this strategy has the smoothest movements, and smallest falls. The bigger drawdown is a consequence of not availing the big movements up as the other strategies, never regaining the balance lost on a series of missed trades, so even though the drawdown is bigger, it happens over a larger period, making this the strategy with the least abrupt movements on balance. This strategy obtains a final ROI of 36.1%, beating strategy A's 31.4%, even with a slightly worse accuracy, 53.4% compared to 53.8%.

Strategy C is the most complete of the first three strategies, with a relatively high amount of trades entered, 176, and an accuracy of 57.4%. It is the strategy with the smallest maximum drawdown at only 11.9% and a final ROI of 73.4%, largely superior to strategies A and B. Observing figure 4.5 one can see the evolution of the ROI in strategy C, as expected, is less steady than strategy B, but does the same job avoiding the falls of strategy A without losing as many opportunities as B. As the trading size is directly related to the size of the current balance, the more the ROI grows, the bigger the impulses it can make, on accurate trades, and consequently bigger falls.

Strategy D, as expected, follows the evolution of strategy C, since the trading rules are practically the same. The objective was that this strategy would benefit greatlier from movements with a bigger level of confidence, and have smaller falls on uncertain trades, which was accomplished by an accuracy of 58.3% and a final ROI of 87.5%. This pattern can also be seen in figure 4.5, as the falls and impulses up happen at the same time, for both this strategies.

From this case study, can be concluded that strategy D outperforms all other strategies in every aspect except for the maximum drawdown, 13%, slightly higher than strategy C. However, this drawdowns happen at the same time and is a consequence of the event explained before, that the bigger the balance, the bigger the falls, since the trading size depends on the current balance.

4.3 Chapter Conclusions

This chapter evaluated the several steps taken throughout this work in order to build the final intradaily trading system. It presented 3 case studies, where different options were compared in order to see which one fits better the final goal of the trading system, which ultimately is to maximize the ROI without jeopardizing the risk management.

The first case study concluded that the best time to be in the *EUR/USD* forex market is from *12:00pm* till *4:30pm*, where the trading volume is the highest and the price movements the most predictable ones.

The second case study showed that the approach of having 3 distinctly trained ANNs, one for each type of market, clearly outperforms a single ANN, bringing much bigger profits, as long as the SVM is doing an accurate prediction, otherwise the outcome will be worse than having a single ANN, which logically has a better generalization for any type of market, since it was trained with the entire training dataset.

Lastly, the third case study, confirmed that the most complete strategy of all is the fourth one presented, **strategy D**, which not only incorporates more complex trading rules, as strategy C, but also makes use of the market classification and the level of confidence provided by the ANN, to adjust trading sizes dynamically. Making possible to avail bigger and more predictable movements and mitigate the risk on more uncertain predictions.

An annualized ROI of 87.5%, achieved by the final trading system over the year of 2019, is the dream on any investor, however, the use of leverage has to be taken in consideration, which not only brings bigger profits, but also bigger risks. For example, a catastrophic event could happen during the time window at which the trade is opened, making the market change directions completely and possible even bringing a negative ROI from one day to another.

Chapter 5

Conclusion & Future Works

5.1 Conclusion

The final system presented by this work, combines a single SVM with three distinct ANNs, in order to make intraday trades in the Forex market of the currency pair *EUR/USD*. The system was trained with data referent to the years of 2004 till 2018 and tested for the year of 2019, providing a great yearly return on investment of 87.5%.

The SVM receives price sequence windows, of approximately three months, to use as features, in order to classify the different market types, bullish, bearish or sideways. Depending on the classification, one of the ANNs, which was trained with the correspondent type of data, is activated, and performs an intraday forecasting, at *12:00pm*, for the price movement until *4:30pm*. The strategy layer, based on the prediction done by one of the ANNs, the type of market classified by the SVM and the current balance, takes one of 3 options, either to enter a long trade, a short or skip that day, it also decides the size of said trade for the first two options.

Support Vector Machines presented to be a reliable method for classification of financial time series, as long as the labels, for the classification purposes required, are provided and consistent, making it possible to define the patterns wanted to be identified and the SVM does the job of identifying it in new data.

Artificial Neural Networks, as expected from the literature review, presented very good forecasting results, as long as correctly optimized and trained with enough and relevant data. The use of different ANNs for different types of market, reducing the generalization of the ANN itself and therefore slightly overfitting the training data, showed to perform better than a single ANN in terms of forecasting the price movements, as long as the correct network is being used, so if the classification is missed, the probability of an unsuccessful forecast rises.

Lastly, the peak hours of the *EUR/USD* Forex market, from *12:00pm* till *4:30pm* - i.e., the time range with the biggest trading volume, represented mostly by banks and hedge funds from both Europe and the USA, presented to have the most predictable, and well defined, price movements.

Trading the currency pair *EUR/USD* intradaily, can bring huge profits when predictions are accurate and leverage is applied, however, the use of leverage comes with an increased risk. It was not the case during any of the tests applied on this system, but financial markets have proved over and over to be very unpredictable environments, so one needs to be conscious that the possibility of unplanned events during the time positions are open in the market, without any kind of control, can completely change the course of the price movement and have a huge negative impact in the overall ROI, from a single trade, especially when applying high leverages.

5.2 Future Works

Even though this work achieved considerably good results, there is always space for improvement. When analyzing the case studies presented in section 4.2, some issues arise, which can be addressed with a more robust and complete system. Possible improvements to this work should focus on a better risk management and mitigation, as well as improving the performance of both the ML algorithms. The following approaches should be considered:

- A secondary system to control market events and prices during the prediction horizon, when there are open positions, would prevent catastrophic losses from happening and therefore mitigate the risk the system is exposed to. Can be done with something as simple as the inclusion of stop losses, or a more robust system with some ML algorithm to perform anomalies detection in the patterns predicted.
- The ANNs would perform better if a genetic algorithm was used to control the hyperparameters and topologies of the networks, instead of the manual tuning of parameters which was done on this thesis. The SVM could also benefit from the inclusion of a GA to control not only the hyperparameters, but the size of the price sequences given as features.
- Instead of classifying three types of market, which had to be labeled manually in order to train the SVM. It would be interesting to explore some kind of unsupervised learning algorithm, capable of clustering different types of markets, according to the patterns found, which would for sure be different than what the human eye can see. This way, more than 3 types of markets could be explored and the system would be less reliable on the human side.
- Finally, AI should be introduced in the strategy layer in order to find optimal trading strategies, improving profits and reducing the losses. For example, a GA could be used, using the outputs of every other layer, and maybe even the inclusion of some technical indicators, to implement a more robust trading strategy capable of maximizing the ROI and mitigating the risk.

Bibliography

[1] L. Ni, Y. Li, X. Wang, J. Zhang, J. Yu, and C. Qi. Forecasting of forex time series data based on deep learning. *Procedia Computer Science*, 147:647 – 652, 2019.

[2] S. Walczak and N. Cerpa. Artificial neural networks. In R. A. Meyers, editor, *Encyclopedia of Physical Science and Technology (Third Edition)*, pages 631 – 645. Academic Press, New York, third edition edition, 2003. URL `http://www.sciencedirect.com/science/article/pii/B0122274105008371`.

[3] A. Hirabayashi, C. Aranha, and H. Iba. Optimization of the trading rule in foreign exchange using genetic algorithm. pages 1529–1536, 2009.

[4] S. Yao, M. Pasquier, and C. Quek. A foreign exchange portfolio management mechanism based on fuzzy neural networks. In *2007 IEEE Congress on Evolutionary Computation*, pages 2576–2583, 2007.

[5] 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*, 38(11):14072 – 14085, 2011. URL `http://www.sciencedirect.com/science/article/pii/S0957417411007433`.

[6] B. Jubert de Almeida, R. Ferreira Neves, and N. Horta. Combining support vector machine with genetic algorithms to optimize investments in forex markets with high leverage. *Applied Soft Computing*, 64:596 – 613, 2018. URL `http://www.sciencedirect.com/science/article/pii/S1568494618300036`.

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

[8] S. B. Achelis. *Technical Analysis from A to Z, 2nd Edition*. McGraw-Hill, 2000.

[9] C. S. D. Stanford University. Multi-layer neural networks. URL `http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/`.

[10] S. K. Satapathy, S. Dehuri, A. K. Jagadev, and S. Mishra. Chapter 1 - introduction. In S. K. Satapathy, S. Dehuri, A. K. Jagadev, and S. Mishra, editors, *EEG Brain Signal Classification for Epileptic Seizure Disorder Detection*, pages 1 – 25. Academic Press, 2019. URL `http://www.sciencedirect.com/science/article/pii/B9780128174265000016`.

[11] D. A. Pisner and D. M. Schnyer. Chapter 6 - support vector machine. In A. Mechelli and S. Vieira, editors, *Machine Learning*, pages 101 – 121. Academic Press, 2020. URL http://www.sciencedirect.com/science/article/pii/B9780128157398000067.

[12] Y. Chihab, Z. Bousbaa, M. Chihab, O. Bencharef, and S. Ziti. Algo-trading strategy for intraweek foreign exchange speculation based on random forest and probit regression. *Applied Computational Intelligence and Soft Computing*, 2019, 2019.

[13] J. Carapuço, R. Neves, and N. Horta. Reinforcement learning applied to forex trading. *Applied Soft Computing*, 73:783 – 794, 2018. URL http://www.sciencedirect.com/science/article/pii/S1568494618305349.

[14] A. Petropoulos, S. P. Chatzis, V. Siakoulis, and N. Vlachogiannakis. A stacked generalization system for automated forex portfolio trading. *Expert Systems with Applications*, 90:290 – 302, 2017. URL http://www.sciencedirect.com/science/article/pii/S0957417417305493.

[15] L. Yu and S. Wang. An online learning algorithm with adaptive forgetting factors for feedforward neural networks in financial time series forecasting. *Nonlinear Dynamics and Systems Theory*, 1, 03 2007.

[16] N. Maknickiene and A. Maknickas. Application of neural network for forecasting of exchange rates and forex trading. 05 2012.

[17] M. O. Özorhan, İsmail Hakkı Toroslu, and O. T. Şehitoğlu. A strength-biased prediction model for forecasting exchange rates using support vector machines and genetic algorithms. *Soft Computing*, 21:6653 – 6671, 2017.

[18] K. C. C. Chan and Foo Kean Teong. Enhancing technical analysis in the forex market using neural networks. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 2, pages 1023–1027 vol.2, 1995.

[19] J. Yao and C. L. Tan. A case study on using neural networks to perform technical forecasting of forex. *Neurocomputing*, 34(1):79 – 98, 2000. URL http://www.sciencedirect.com/science/article/pii/S0925231200003003.

[20] M. Butler and A. Daniyal. Multi-objective optimization with an evolutionary artificial neural network for financial forecasting. pages 1451–1458, 07 2009.

[21] C. Evans, K. Pappas, and F. Xhafa. Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation. *Mathematical and Computer Modelling*, 58(5):1249 – 1266, 2013. URL http://www.sciencedirect.com/science/article/pii/S0895717713000290.

[22] S. Galeshchuk.

[23] T. Zafeiriou and D. Kalles. Intraday ultra-short-term forecasting of foreign exchange rates using an ensemble of neural networks based on conventional technical indicators. In *11th Hellenic Conference on Artificial Intelligence*, page 224–231, New York, NY, USA, 2020. Association for Computing Machinery.

[24] Kyoung-jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307 – 319, 2003. URL `http://www.sciencedirect.com/science/article/pii/S0925231203003722`.

[25] Z. Hua, Y. Wang, X. Xu, B. Zhang, and L. Liang. Predicting corporate financial distress based on integration of support vector machine and logistic regression. *Expert Systems with Applications*, 33(2):434 – 440, 2007. URL `http://www.sciencedirect.com/science/article/pii/S095741740600159X`.

[26] T. N. T. Thu and V. D. Xuan. Using support vector machine in forex predicting. In *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, pages 1–5, 2018.

[27] K. jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307 – 319, 2003. URL `http://www.sciencedirect.com/science/article/pii/S0925231203003722`.

[28] Python. Python 3.9.1 documentation. URL `https://docs.python.org/3/`.

[29] Pandas. Pandas documentation. URL `https://pandas.pydata.org/docs/index.html`.

[30] A.-P. Refenes. *Neural Networks in the Capital Markets*. John Wiley Sons, Inc., 1994.

[31] B. Vanstone and G. Finnie. An empirical methodology for developing stockmarket trading systems using artificial neural networks. *Information Technology papers*, 36, 2009.

[32] Scikit-Learn. Machine learning in python, . URL `https://scikit-learn.org/stable/`.

[33] Scikit-Learn. Precision-recall, . URL `https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html`.

[34] TensorFlow. Tensorflow core. URL `https://www.tensorflow.org/overview`.

[35] I. Kaastra and M. Boyd. Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 10(3):215 – 236, 1996. URL `http://www.sciencedirect.com/science/article/pii/0925231295000399`.

[36] A. Saks. Finance feeds: European leverage restrictions: How did it affect volumes so far? URL `https://financefeeds.com/european-leverage-restrictions-how-did-it-affect-volumes-so-far-we-investigate/`.

[37] Teletrade. Negotiation conditions. URL `https://www.teletrade.eu/pt/trade/condition-mt4/forex`.

[38] B. de Portugal. Exchange rates. URL `https://www.bportugal.pt/en/taxas-cambio`.