Feature Selection and Optimization on Naive Bayes Modeling using Genetic Algorithm

Gonçalo Faria Abreu

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. Rui Fuentecilla Maia Ferreira Neves

Examination Committee

Chairperson: Prof. António Manuel Raminhos Cordeiro Grilo

Supervisor: Prof. Rui Fuentecilla Maia Ferreira Neves

Members of Committee: Prof. Mário Alexandre Teles de Figueiredo

May 2017

To my family…

Acknowledgements

I would like to thank my supervisor Rui Neves for all the support and opportunity to discuss everything about his knowledge regarding financial markets and evolutionary computation. This enabled me to have the strength needed to tackle every problem that surfaced.

My family was the pillar of my academic path. For their patience and support I treasure the best memories.

Abstract

The starting point in every Machine Learning model application is the input features. When it is unfeasible to search the input feature space with an Exhaustive Algorithm, an Evolutionary Strategy might be an alternative. In this work, an architecture for automatic feature selection and searching is proposed, using a Genetic Algorithm to optimize the Naive Bayes Cross Validated model output estimation. Capital Markets, specifically the Foreign Exchange Market, provides the case study, since when using Technical Analysis as the input features, the problem becomes a combinatorial explosion. The proposed architecture improves the accuracy of the unoptimized system from 51,39% to 53,95% in the test set. An attempt of model visualization is made using the algorithm *t*-Distributed Stochastic Neighbour Embedding.

Keywords

Machine Learning, Evolutionary Computation, Genetic Algorithms, Naive Bayes, Feature Selection, Feature Optimization, Foreign Exchange Market, *t*-Distributed Stochastic Neighbour Embedding.

Resumo

O ponto inicial de qualquer implementação de Aprendizagem Automática são as features de entrada do modelo. Quando o espaço que define as features de entrada torna a sua seleção proibitiva através do uso de um algoritmo de pesquisa exaustiva, uma estratégia evolutiva pode ser uma alternativa viável. Neste trabalho, uma arquitetura automática para seleção e otimização de features de entrada é proposta, otimizando à sua saída, a taxa de acerto do modelo Naive Bayes estimada com recurso a um esquema de rotação. O Foreign Exchange Market serve como caso de estudo, pois ao usar análise técnica como entrada do modelo, o problema torna-se numa explosão combinatória. A arquitetura proposta melhora a taxa de acerto, em comparação com um modelo não otimizado, de 51,39% para 53,95% no período de teste. Através do algoritmo *t*-Distributed Stochastic Neighbour Embedding é feita uma tentativa de visualização das regras alcançadas pelo modelo.

Palavras-chave

Aprendizagem Automática, Computação Evolutiva, Algoritmos Genéticos, Naive Bayes, Seleção de Features, Otimização de Features, Foreign Exchange Market, *t*-Distributed Stochastic Neighbour Embedding.

Table of Contents

List of Figures

List of Tables

List of Acronyms

%D	Fast D Oscillator component
%K	Fast K Oscillator component
ATNN	Adaptive Time Delay Neural Networks
ATR	Average True Range
CCI	Commodity Channel Index
CV	Cross Validation
DJI	Dow Jones Industrial
EMA	Exponential Moving Averages
EMH	Efficient Market Hypothesis
GA	Genetic Algorithms
HMM	Hidden Markov Model
k-NN	k-Nearest Neighbours algorithm
KOSPI	Korea Composite Stock Price Index
KOSPI	Korea Composite Stock Price Index
LDA	Linear Discriminant Analysis
MACD	Moving Average Convergence Divergence
MAPE	Mean Absolute Percentage Error
MCMC	Markov Chain Monte Carlo
ML	Machine Learning
MSE	Mean Squared Error
NMSE	Normalized Mean Squared Error
NN	Neural Networks
Pdf	Probabilistic Distribution Function
QDA	Quadratic Discriminant Analysis
RMSE	Root mean squared error
ROC	Rate of Change
ROI	Return on Investment
RSI	Relative Strength Index
RW	Random Walk
SAX	Symbolic Aggregate Approximation
SVM	Support Vector Machine
TA	Technical Analysis
TDNN	Time Delaying Neural Networks
t-SNE	*t*-Distributed Stochastic Neighbour Embedding

VIX	Volatility Index

List of Symbols

$Close_t$	Close Price at period t
σ	Standard Deviation
μ	Expected Value
y	Categorical Observation Response
Y	Vector with $y_1, \dots, y_n$ target responses
x	Observable Sample Input Feature Vector
X	Matrix of Observable $x_1, \dots, x_n$ samples
$\hat{y}_i$	Estimated i Categorical Response
y_i	i Categorical Response
$\mathfrak{D}$	Dataset Matrix containing Samples (Observations x) and with its corresponding Responses (y_i)

Chapter 1

Introduction

1.1 Overview

The statement that the market moves in a chaotic and unpredictable way is generally accepted. However, the existence of traders is a fact (people whose profession is to analyse the market time series using Technical Analysis (TA), making a profit of its predictability and/or inefficiencies). Therefore, if people with access to this information can consistently beat the market, Machine Learning (ML) methods should be able to capture some patterns (or lack of). This question can be formulated to allow an engineering methodology approach, i.e., can this problem be reduced to a binary classification between an overvalued or undervalued market signal?

In the Financial Markets, there is an important theory to be considered, the Efficient Market Hypothesis (EMH) [1] which states that the market price of an asset is always the fair one, invalidating the possibility of a trader/investor buying or selling an undervalued or overvalued asset, respectively. This hypothesis implies that a binary classifier ML algorithm, trained to identify if an asset is overvalued or undervalued, would report that the market signal was always fair, i.e., the prediction probability of the future signal variation being positive, or negative is always 50%. This would mean that the market is truly a chaotic and unpredictable event, thus the existence of profitable trader (which wouldn't access confidential information by colluding with others) would be questionable. The objective of this thesis is to answer the question: is it possible to build a binary classifier which, using TA as an input, predicts if an asset is overvalued or undervalued, better than random guessing? And if such a system is possible, provide a novel architecture to optimize it.

The aspect of system validation will be addressed using multiple ML Concepts, such as, cross validation (CV) and by evaluating if the system can discard randomly generated signals.

The main concepts used for this thesis to build the proposed architecture are Naive Bayes and Genetic Algorithms (GA). Markov Chain Monte Carlo (MCMC) is used to optimize the Randomly Signal Generator. TA will be used to generate the input features for the Model.

The building blocks used for the construction of the proposed architecture are all documented through various research paper articles presented in Chapter 2.3. The problem formulation is novel, hence the architecture presented in Chapter 3.6.

The main contributions of this thesis are:

1. Problem formulation through the binarization of market time series
2. A framework for estimating the performance of a binary classifier given the proposed problem formulation
3. An architecture for feature searching and selecting using a modified GA
4. Model visualization using the novel *t*-distributed stochastic neighbour embedding (*t*-SNE).

1.2 Motivation and Contents

Financial markets and specially ML have taken central stage in society. With the creation of high speed global communications, Financial markets started altering their behaviour by becoming a complex inter-connected system with no centralization. New methodologies need to be developed to better understand and visualize this new behaviour. This poses an interesting problem, which electrical engineering is well suited to tackle, given its close relation with ML and signal processing, not to mention the communication backbone in which the Financial Markets exists (the internet) was also envisioned by this area.

The study of the Financial Markets may seem to diverge from the scope of Electrical and Computer Engineering but many ML concepts are close to computation, signal processing and Information Theory [2]. This thesis is a study with emphasis on ML and GA. The topic of Financial Markets is the object of study since the approach chosen, as it will be shown, poses a combinatorial explosion, i.e., the selection of which features should be used as inputs to the ML model.

The structure for the thesis is the following:

- Chapter 1 – In this chapter a brief introduction on the subject is made, such as, the most important topics that will be discussed during the work, the main objectives and methodologies used to develop the proposed architecture
- Chapter 2 - An introduction to the baseline knowledge needed to conceptually understand the work developed. Reviewed academic papers, are discussed here, since they were the starting point for this thesis
    - 2.2 – Fundamental concepts to understand this thesis will be presented and formalized, such as, Technical Analysis, the Foreign Exchange Market, Genetic Algorithms, and Naive Bayes binary classifier
    - 2.3 – Collection of papers and its corresponding review. At the end of the topic, a table summarizing the most important documents is presented
- Chapter 3 – Proposed architecture delineation with detailed explanation of all the tools developed
    - 3.2 – The general perspective of the proposed architecture is examined. All the building blocks are presented as well as their interconnections
    - 3.3 – The target is formally defined, i.e., how the signal binarization is performed
    - 3.4 – Matrix X generation using the Technical Indicator Generator. This matrix will be the input for the binary classifier
    - 3.5 – Naive Bayes Binary Classifier is described and the context of his usage
    - 3.6 – The core of the architecture is presented and thoroughly examined. Here, the binary classifier is joined together with the GA to enhance the classification algorithm
    - 3.7 – The Market Simulator is described simulating how the proposed model would perform in real market conditions
    - 3.8 – A Random Signal Generator is proposed. Signals generated by this method, will be estimated by the Naive Bayes binary classifier, to evaluate its ability to discard

signals known a priori to be random

- Chapter 4 - In this chapter all five the case studies are presented
  - 4.2 - How the time series is split into train and test sets
  - 4.3 – Case study A, the performance of the binary classifier is thoroughly described (CV, train and test performance, rejection Naive Bayes classifier and Market Simulator results)
  - 4.4 - Case Study B, the proposed architecture is examined with the same experiments done in Case Study A to allow a comparison between methodologies.
  - 4.5 – Case Study C, the proposed GA is compared to its original implementation
  - 4.6 – Case Study D, signals generated using the Random Signal Generator are tested using the Naive Bayes Classifier to evaluate its ability to discard a signal with no causality patterns
  - 4.7 – Case Study E, proposed architecture model visualization using *t*-SNE algorithm

.

Chapter 2

Background and State-of-the-Art

2.1 Overview

The purpose of this chapter is to define the fundamental concepts that will be used throughout the rest of this thesis regarding the Foreign Exchange Market, Technical Analysis (TA), Genetic Algorithms (GA), Naive Bayes, feature selection and optimization. First, there will be an introduction to familiarize the reader with what the foreign exchange market and TA are. Subsequently, an overview of the working principles of GA and Naive Bayes binary classifiers. Finally, academic papers are reviewed to support some of the choices made throughout the thesis.

2.2 Background

2.2.1 Foreign Exchange Market

The Foreign Exchange Market is the definition used for the globally decentralized market in which currency pairs are exchanged. This market is structured through a hierarchical number of levels since there are many financial institutions involved. With the rise in capacity of communication technologies, and since these institutions are interconnected at practically the speed of light, it is an interesting engineering problem to analyse the signal which results from these interactions. In this thesis, the signal being analysed is the relationship between currency trading pair representing the Euro-Dollar ratio (EUR/USD). A currency trading pair defines the ratio at which euros and dollars are exchanged at the current market rate; i.e., if the current EUR/USD rate is 1.18, it means that in this moment if an individual wants to use the market to exchange currencies from euros to dollars, they will get 1.18 dollars per 1 euro.

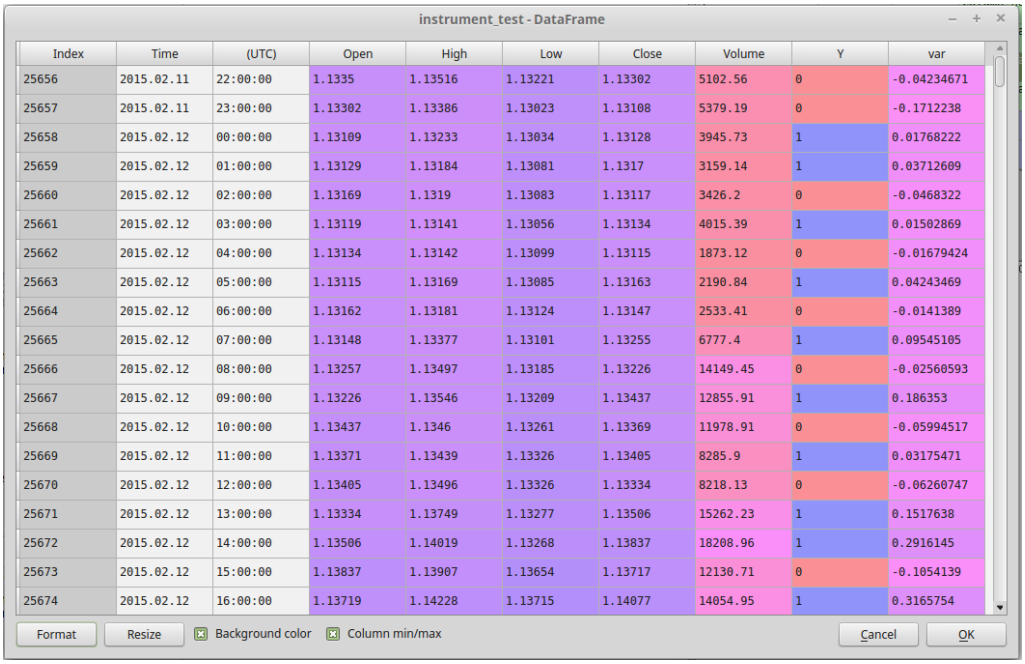
To study the change in the EUR/USD rate, a time series is built from sampling the market at an hourly rate since 2013. Although there are usually two different prices, the Ask and the Bid in which the signal can be built, to formulate the problem an average of the two was created (which will be referred to as mid-point). Nevertheless, in Chapter 4 this price difference will be accounted for, while performing market simulations.

In Figure 2-1 an example of how the EUR/USD signal looks like in matrix form is presented.

Each sample has the information of how the rate EUR/USD behaved between a specific hour period. Figure 2-1 provides an example of the time series and the columns of the data table mean the following:

- *Time* and *(UTC)* columns represent the date and time at which the sample (market observation) was collected
- *Open* symbolizes the value, of the rate, at the beginning of the respective period

- *High*, is the highest value the rate achieved at the respective hour period
- *Low* column, is the lowest value the rate achieved at the respective hour period
- *Close*, is the closing rate value at the respective hour period
- *Volume*, is the number of contracts executed in that hour
- *Y*, is the target to predict and will be explained in a later topic
- *var*, is the percentage change between the last and current closing hour values



Index	Time	(UTC)	Open	High	Low	Close	Volume	Y	var
25656	2015.02.11	22:00:00	1.1335	1.13516	1.13221	1.13302	5102.56	0	-0.04234671
25657	2015.02.11	23:00:00	1.13302	1.13386	1.13023	1.13108	5379.19	0	-0.1712238
25658	2015.02.12	00:00:00	1.13109	1.13233	1.13034	1.13128	3945.73	1	0.01768222
25659	2015.02.12	01:00:00	1.13129	1.13184	1.13081	1.1317	3159.14	1	0.03712609
25660	2015.02.12	02:00:00	1.13169	1.1319	1.13083	1.13117	3426.2	0	-0.0468322
25661	2015.02.12	03:00:00	1.13119	1.13141	1.13056	1.13134	4015.39	1	0.01502869
25662	2015.02.12	04:00:00	1.13134	1.13142	1.13099	1.13115	1873.12	0	-0.01679424
25663	2015.02.12	05:00:00	1.13115	1.13169	1.13085	1.13163	2190.84	1	0.04243469
25664	2015.02.12	06:00:00	1.13162	1.13181	1.13124	1.13147	2533.41	0	-0.0141389
25665	2015.02.12	07:00:00	1.13148	1.13377	1.13101	1.13255	6777.4	1	0.09545105
25666	2015.02.12	08:00:00	1.13257	1.13497	1.13185	1.13226	14149.45	0	-0.02560593
25667	2015.02.12	09:00:00	1.13226	1.13546	1.13209	1.13437	12855.91	1	0.186353
25668	2015.02.12	10:00:00	1.13437	1.1346	1.13261	1.13369	11978.91	0	-0.05994517
25669	2015.02.12	11:00:00	1.13371	1.13439	1.13326	1.13405	8285.9	1	0.03175471
25670	2015.02.12	12:00:00	1.13405	1.13496	1.13326	1.13334	8218.13	0	-0.06260747
25671	2015.02.12	13:00:00	1.13334	1.13749	1.13277	1.13506	15262.23	1	0.1517638
25672	2015.02.12	14:00:00	1.13506	1.14019	1.13268	1.13837	18208.96	1	0.2916145
25673	2015.02.12	15:00:00	1.13837	1.13907	1.13654	1.13717	12130.71	0	-0.1054139
25674	2015.02.12	16:00:00	1.13719	1.14228	1.13715	1.14077	14054.95	1	0.3165754

Figure 2-1 Data table containing a subset of EUR/USD rate

In Figure 2-2, closing values across the time series are plotted.

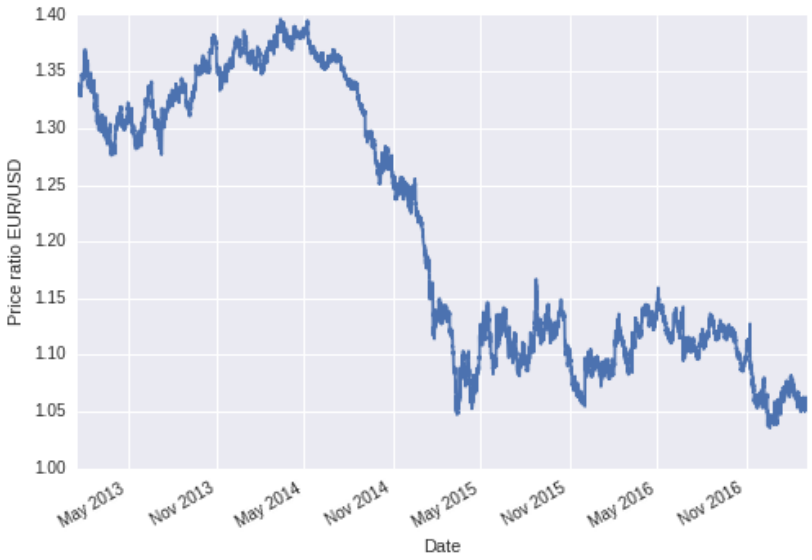


Figure 2-2 EUR/USD rate evolution regarding the period between 01.01.2013 to 09.03.2017

2.2.2 Technical Analysis (TA)

Although there are two types of analysis used in financial markets, fundamental and technical, only the posterior one will be used in this thesis. By reviewing the balance sheets of a company, prospect the future of a market and/or product, and by looking at macro and micro indicators, fundamental analysis tries to elaborate if a financial instrument will be a good investment in the long-term. On the other hand, TA, through the elaboration mostly of transformations of the price and volume, it tries to predict if a financial instrument will increase or decrease its rate in the short-term, potentially enabling a profit. Financial markets widely use technical indicators to exploit existing trends [3] and this is an active research topic.

In this thesis, the following Technical Indicators will be used:

- Relative Strength Index (RSI)
- Volume
- Commodity Channel Index (CCI)
- Moving Average Convergence Divergence (MACD)
- Rate of Change (ROC)
- Stochastic Oscillator
- Average True Range (ATR)

The rest of this section will be dedicated to explaining the motivations of each technical indicator.

2.2.2.A RSI

The Relative Strength Index (RSI) is a momentum oscillator which measures the magnitude and direction of the price [4]. It depends only on the changes of closing prices. RSI tries to make a graphic representation to evaluate if an instrument is overbought or oversold. The calculation of this technical indicator involves the use of Exponential Moving Averages (EMA) to have a smoother representation of the price. The formula used to calculate RSI is

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

RS is the ratio of the exponentially smoothed moving average over *n*-period gains, divided by the absolute value of the exponentially smoothed moving average over *n*-period losses [4]. Although there is a standard value for *n*, its optimal value might change between financial instruments.

The resulting RSI plot with standard *n* period for the EUR/USD signal database referred in Figure 2-2 is shown in Figure 2-3. This plot was drawn in a smaller hour subset for a better visualization on what the different values of RSI might mean.

The horizontal lines at values 30 and 70 symbolize, respectively, where the RSI starts to indicate whether an asset is oversold or overbought [4].

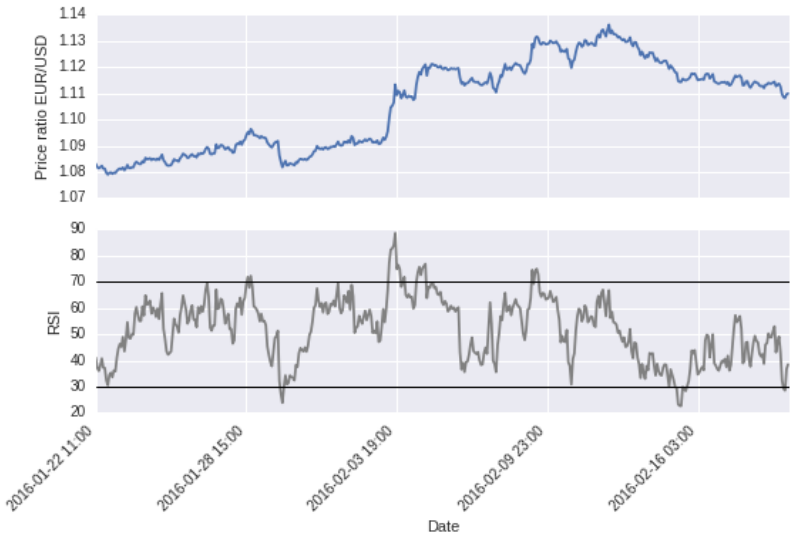


Figure 2-3 EUR/USD rate between a subset of hours with the respective RSI plotted at the bottom

2.2.2.B Volume

Volume is defined as the number of transactions in a specific time period. Usually, increased market volatility, is followed by a drastic change in volume so it is important to use it as an input feature. Volume is used as a secondary confirmation of a trend [5]. Figure 2-4 shows the Volume plot with the respective EUR/USD signal on the same hourly time subset as the RSI.

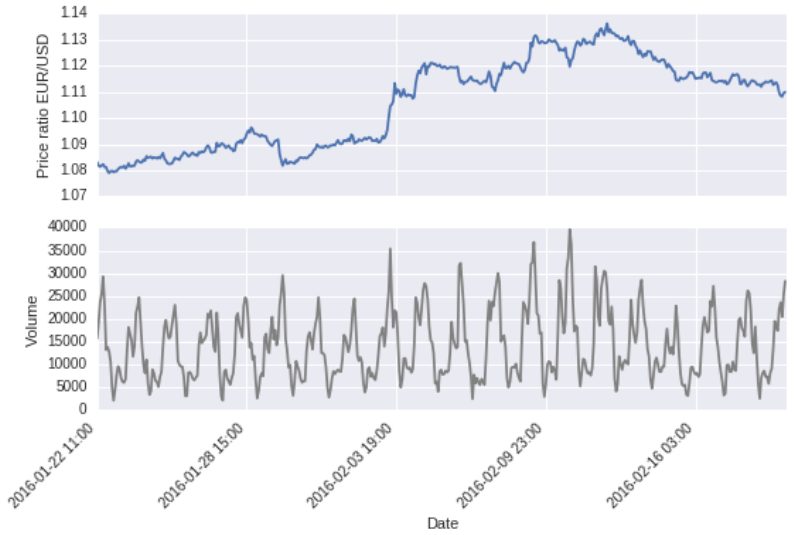


Figure 2-4 EUR/USD rate between a subset of hours with the respective Volume of contracts plotted at the bottom

2.2.2.C CCI

The Commodity Channel Index (CCI) is a price momentum indicator which is used to measure whether a financial instrument is overbought or oversold. The usual interpretation is, if the CCI is above 100 then

the instrument is overbought, if the CCI is below -100 the instrument is oversold [4] [5]. In Figure 2-5 there is an illustration of this concept.

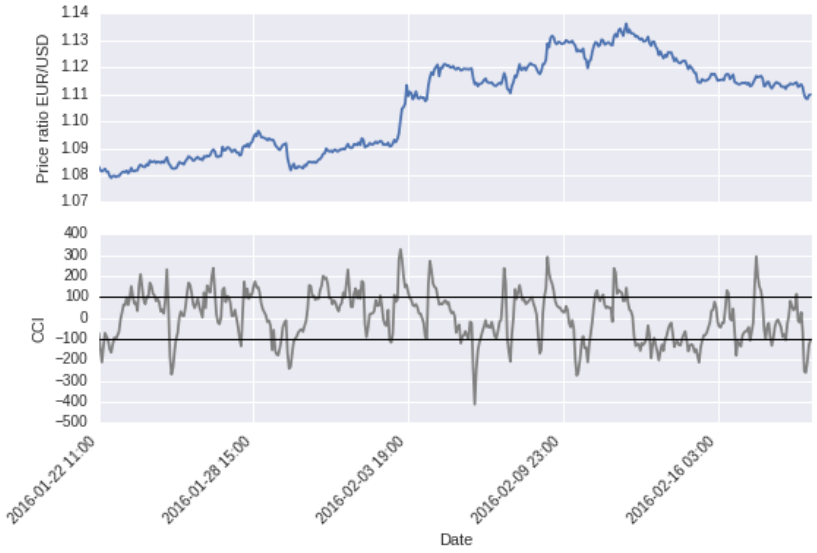


Figure 2-5 EUR/USD rate between a subset of hours with the respective CCI plotted at the bottom

The following equations represent the way this Technical indicator is calculated. The final formula for the calculation of the CCI is

$$CCI = \frac{1}{0.015}\frac{p_t - SMA(p_t)}{\sigma(p_t)}, \quad (2)$$

In,

$$p_t = \frac{High_t + Low_t + Close_t}{3}, \quad (3)$$

p_t is known as the typical price and $High_t$, Low_t and $Close_t$ symbolize the Highest, Lowest, and Closing price of a given instrument at period *t*.

The standard way of calculating a simple *n* period moving average is,

$$SMA_n(p_t) = \frac{1}{n}\sum_{i=0}^{n-1} p_{t-i}, \quad (4)$$

CCI, like RSI, also has an *n* parameter which should be tuned between different financial instruments. In Figure 2-5 the standard *n* period [4] [5] was used.

2.2.2.D MACD

The Movement Average Convergence-Divergence (MACD) is a type of price oscillator, constructed by subtracting a 26-period EMA of the price to a 12-period one [6]. After this step, a 9-period EMA of the MACD is constructed which serves as a signal line. The interpretation is the following: when the signal line crosses the MACD there is an indication that the trend on the market is changing. From the feature

optimization point of view, there are 3 parameters that can be freely chosen; the periods from the two EMA's which are subtracted to make the MACD line, and the period in which the signal line is calculated.

In Figure 2-6 it is possible to see the example referred above. There might be some patterns to search for, when the signal line crosses the MACD. The proposed architecture will try to evaluate if there is such a pattern.

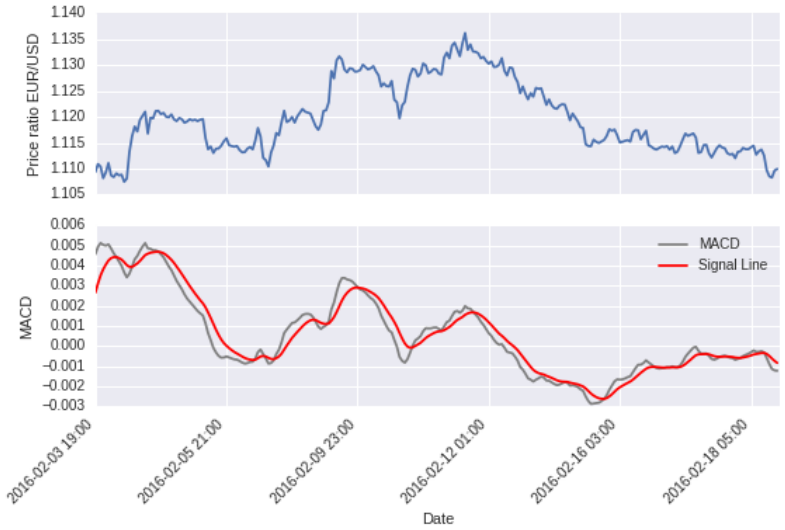


Figure 2-6 EUR/USD rate between a subset of hours with the respective MACD plotted at the bottom

2.2.2.E ROC

Rate of change is a price oscillator which measures the difference between the current price and the price from *n* periods ago [6]. The difference is shown in percentage. The interpretation is the same as the previous oscillators already described, trying to confirm the presence of a trend in the market. In Figure 2-7 a plot of ROC with its respective EUR/USD signal is shown.

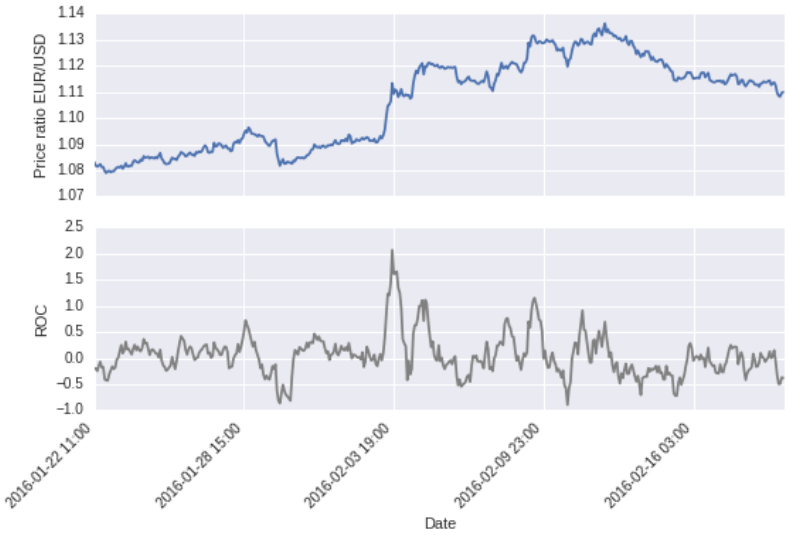


Figure 2-7 EUR/USD rate between a subset of hours with the respective ROC plotted at the bottom

The formula used for calculating ROC is,

$$ROC_n = \frac{Close_t - Close_{t-n}}{Close_{t-n}} \times 100 \quad (5)$$

There are a lot of ROC indicators to choose from, since there is a parameter *n* to pick.

2.2.2.F Stochastic Oscillator

Stochastic Oscillator is a momentum indicator that measures the present strength in a trend [7]. It consists of two lines called %K (also known as FastK, which reacts to faster price movements) and %D (known as FastD which reacts slower to the same price movements). The interpretation is the following, when %K is above the %D line, an upwards trend is probably present in the market. When FastK is lower than FastD, a downwards trend is probably present. If the lines cross, it means that the current trend is losing momentum and there might be a reversal soon. Figure 2-8 serves as an example in which the concepts above can be visualized.

In

$$\%K_n = \frac{Close_t - \min(Low)_n}{\max(High)_n - \min(Low)_n}, \quad (6)$$

$Close_t$ is the current closing price; $\min(Low)_n$ is the minimum price the instrument achieved in the last *n* periods; $\max(High)_n$ is the maximum price achieved by the instrument in the last *n* periods.

The way %D is calculated is

$$\%D_{n1} = SMA_{n1}(\%K_n), \quad (7)$$

as seen in the formula, this technical indicator has two parameters to tune, *n* and n_1. In the proposed architecture chapter, a solution will be presented to tackle this problem.

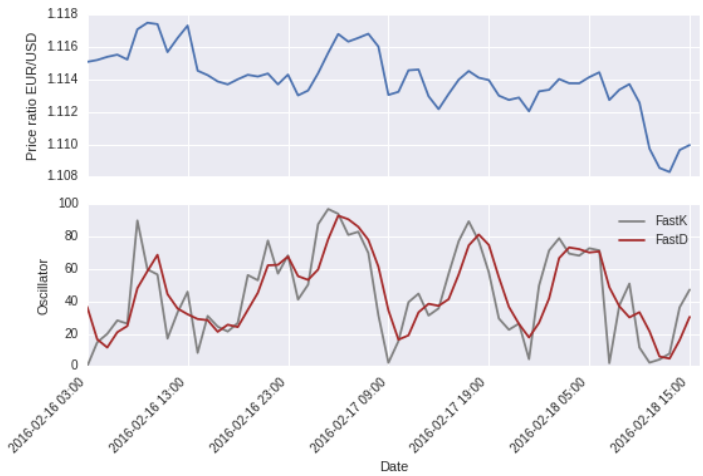


Figure 2-8 EUR/USD rate between a subset of hours with the respective FastK and FastD are plotted at the bottom

2.2.2.G ATR

Average True Range measures the volatility present in the market. ATR doesn't have information about the direction of the price, it measures how strong present volatility on the market is. The first step of calculating this indicator starts with the calculation of the *true range* at period *t*,

$$TR_t = \max[(\text{High}_t - \text{Low}_t), |\text{High}_t - \text{Close}_{t-1}|, |\text{Low}_t - \text{Close}_{t-1}|] \quad (8)$$

After this step, it is necessary to calculate the ATR at point t_0,

$$ATR_{t0} = \frac{1}{n}\sum_{i=1}^{n} TR_i \quad (9)$$

Now the only step left is to calculate the ATR through all the periods on the dataset,

$$ATR_t = \frac{ATR_{t-1}.(n-1) + TR_t}{n} \quad (10)$$

In Figure 2-9 there is an example of the ATR with its corresponding EUR/USD rate. It is possible to see that high market volatility, is usually accompanied by a rise in value of the ATR at that time period.

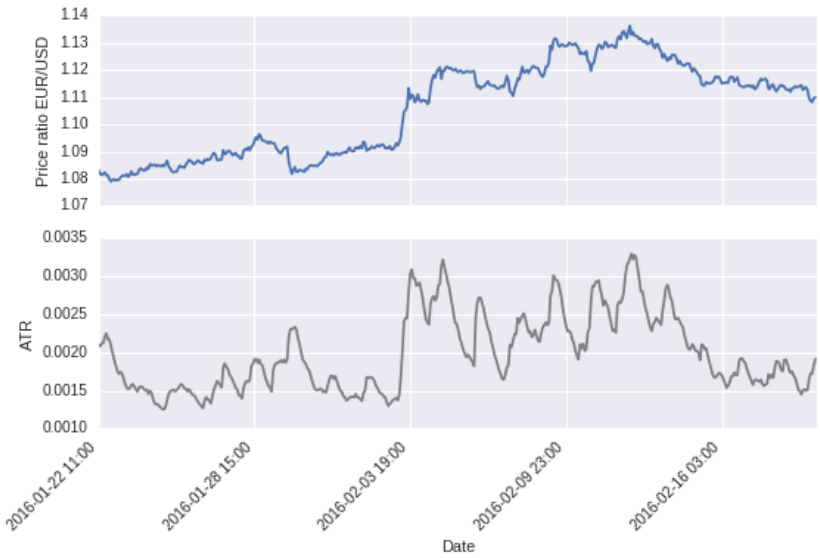


Figure 2-9 EUR/USD rate between a subset of hours with the respective ATR plotted at the bottom

2.2.3 Genetic Algorithm (GA)

To introduce GA [8], it is important to first explain its context inside the area of Evolutionary Computation. This area of computation is inspired on the way life evolves, more specifically using the Darwinian concepts, such as mutation, reproduction, recombination, and selection. The GA is a metaheuristic algorithm; hence it tries to find the global optimum of an optimization problem. It is particularly useful when computation power constrains are present, and the solution space is sparse.

To explain a GA, it is necessary to introduce some terminology. The most basic element is an individual, which is also called a chromosome. A GA is composed of several individuals, which are evaluated through the means of a fitness function to have their corresponding fitness attributed. Originally the chromosome was defined as a bitstring, but it can have different representations such as a float vector. Each bit/float, in the string/vector, can be referred to, as a gene. The first step of the algorithm is to create an initial population of individuals (chromosomes), and this is usually done in a random way, i.e., suppose that each individual is composed of a fixed length vector of 4 floats, each float can have a value between [0,1]. Then, sampling 4 values from a continuous uniform probabilistic distribution function (*pdf*) inside the interval [0,1] can create an individual. This is repeated, as often as necessary, to achieve the desired initial population size. After the population is created the fitness of each chromosome is evaluated and assigned. Subsequently, some of the individuals are selected for mating and copied to the mating buffer. In Holland's [9] original GA formulation, individuals were selected for mating probabilistically regarding its fitness value, i.e., higher values of fitness translate to higher probability of reproduction. The next concept, the genetic operator (usually crossover and mutation) is applied to the population of selected individuals. The mutation operator mutates the gene's inside each chromosome and this is also done in a probabilistic way, for instance, each gene has a probability of being mutated and the mutation follows a rule, usually a continuous *pdf*. The probability of mutation and the way its performed are implementation decisions. The crossover is the operation which executes the mating of one or more individuals, i.e., it combines the bitstring of different individuals. After the offspring is created it is necessary to fill the rest of the population with individuals, since most GA architectures maintain a fixed size population. There are many strategies for this, one of them is to randomly create new individuals. All these transformations, except the initial creation of the population, are encapsulated in cycles, called generations. The GA should have as many generations as it needs to converge toward a stable solution. The Table 2-1 contains the pseudo code demonstrating what was explained above:

```
generation n=0
initialize population P(n) with individuals
evaluate P(n) fitness
while fitness criterion not met:
      n=n+1
      select to reproduce a subset S(n) of P(n-1)
      cross-over and mutate S(n)
      form S'(n)
      evaluate fitness in S'(n)
      replace P(n) from S'(n) and P(n-1)
```

Table 2-1 GA pseudo code

2.2.3.A Chromosome representation

Each individual of a population is a chromosome composed of genes. One way of representing the chromosome is by using a vector of real numbers inside the interval [0,1]. Figure 2-10 shows a possible representation for an individual in a GA.

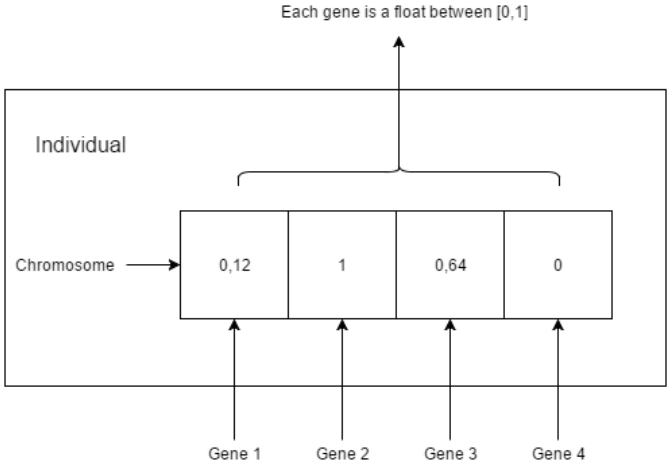


Figure 2-10 Example of an individual in a GA

In this case the chromosome is composed of 4 genes, and each one, is a real number between the interval [0,1]. The meaning of each gene is assigned depending on the optimization problem, for instance, if a GA was used to solve a linear regression problem using a cost function (as fitness function), each of the genes could be a weight which was trying to be determined.

2.2.3.B Fitness Function

The Fitness Function evaluates the performance of an individual inside the population. This way, individuals can be ranked against each other, and the selection for mating, or mutation can be based on the best performers. In Figure 2-11 a chromosome is mapped into a linear regression problem. The Fitness Function outputs the Root Mean Squared Error (RMSE) of the regression which would be used as the scoring metric of this problem. The objective of the GA, in this case, would be to achieve the lowest RMSE possible.

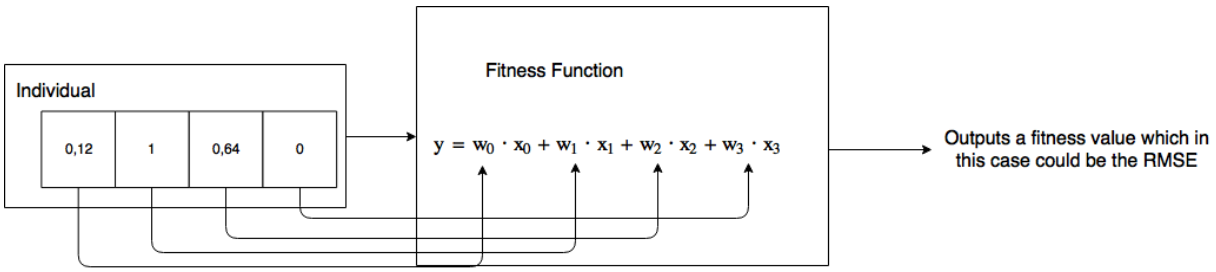


Figure 2-11 Fitness Function evaluation of an individual

2.2.3.C Selection

The selection module on the GA selects which individuals will be submitted to crossover and mutation operations. The method picked for the proposed architecture relies on the compromise between the search diversity and search depth. The most common methods for selecting are the Tournament Selection and Roulette selection.

Tournament selection works as follows: consider there are *N* individuals in a population, *N* tournaments will be performed, and in each tournament *k* individuals will participate. Individual tournaments consist of: after the fitness of each individual is calculated (follow Table 2-1 pseudo code) *k* individuals are randomly sampled from the population. From the *k* individuals, the one with the best fitness is selected for mutation and/or crossover. The tournament is repeated *N* times.

In Roulette selection, each individual is attributed a probability of being selected in relation to its fitness value, i.e., higher fitness individuals have a higher probability of being selected. Then, there are *k* spins of the roulette, in which *k* individuals are sampled according to their probability of being selected.

2.2.3.D Mutation operator

The mutation operator is used to mutate one or a set of genes in a probabilistic way. For example, after a gene is selected for mutation a continuous *pdf* is applied on it. In Figure 2-12 there is an example of a Gene being mutated. There are many choices for the function which performs the mutation, but a common one, is the Gaussian curve. Through the setting of the standard deviation (σ), it is possible to control if the mutation is going to consist of a wider, or narrower interval around the Expected Value (μ).

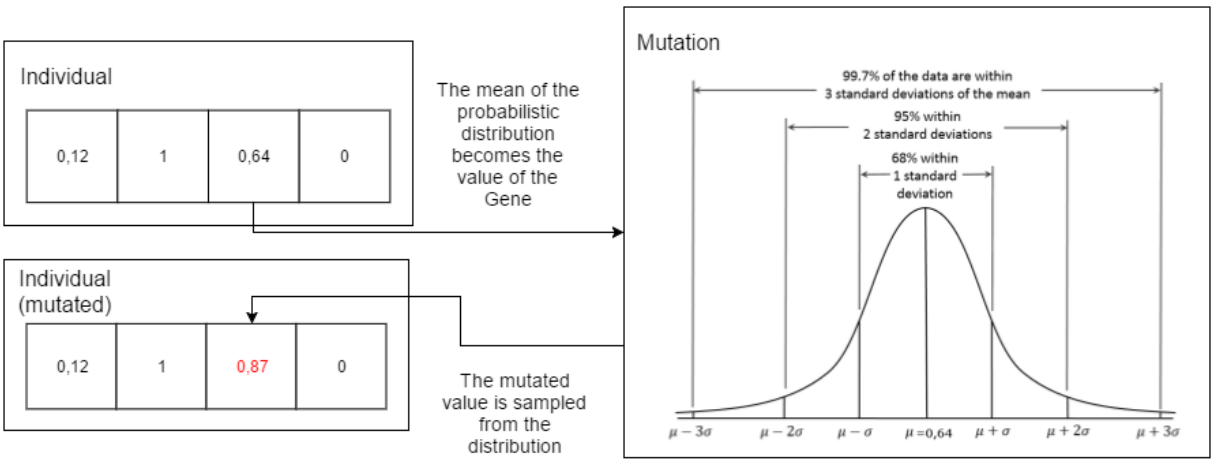


Figure 2-12 Mutation operation on a Gene

2.2.3.E Crossover Operator

The combination of two or more individuals is called Crossover. In this step, parts of the chromosome of each parent are combined to generate a child. Single-point Crossover and Two-point Crossover are examples of this operation. In Figure 2-13 a graphical example of the two operators is provided. Crossover is applied after selection has occurred. Every pair of sequential parents has a probability of being mated, and if they mate, their children substitute their parents' place in the population. Only after this step the mutation operation is applied.

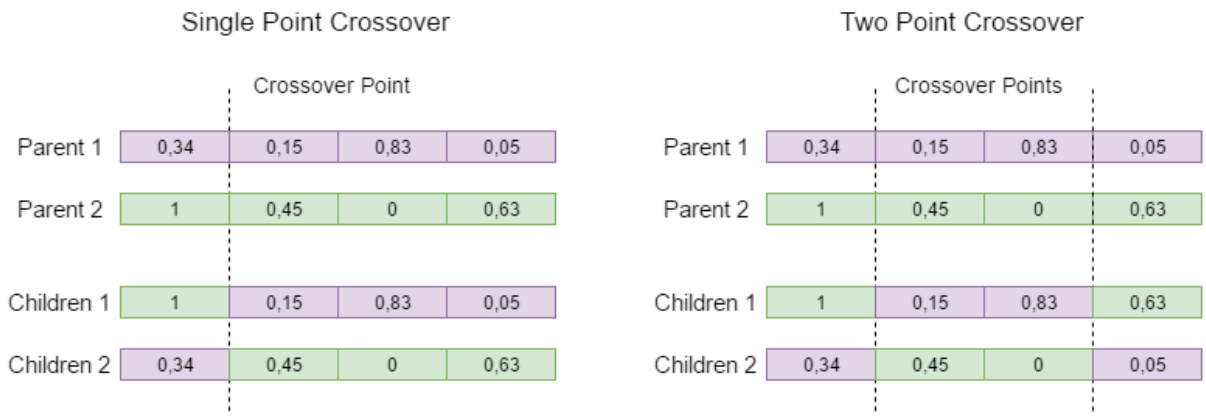


Figure 2-13 Single Point Crossover and Two Point Crossover illustration

2.2.4 Naive Bayes

The Naive Bayes model is a supervised learning method for classification [10]. In this model, a naive assumption that all the features are independent, combined with Bayes theorem, provides a fast and simple computational approach for binary classification. This is an important remark for this thesis, as it will be shown in a later chapter, since this model will serve as the endpoint to a fitness function for the proposed GA.

Naive Bayes is the simplest case of a Bayes network and its diagram is shown in Figure 2-14.

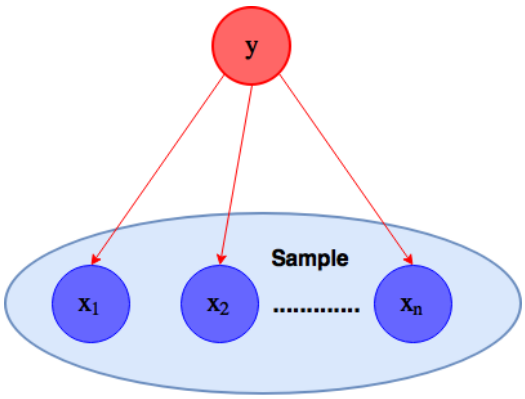


Figure 2-14 Naive Bayes model diagram

Consider the Categorical Response y to a dependent vector $\mathrm{x} = \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix}$, applying the Bayes theorem to $P(y|x_1 \dots x_n)$,

$$P(y|x_1 \dots x_n) = \frac{P(y)P(x_1 \dots x_n|y)}{P(x_1 \dots x_n)} \tag{11}$$

If a naive assumption that all the features (vector $\mathbf{x}$) are independent is made, the following simplification is possible,

$$P(x_i|y, x_1 \dots, x_{i-1}, x_{i+1} \dots x_n) = P(x_i|y) \tag{12}$$

With this assumption, it is possible to rewrite Equation (11) into

$$P(y|x_1 \dots x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1 \dots x_n)} \tag{13}$$

If $P(x_1 \dots x_{1n})$ is constant the following rule can be used to do classification.

$$P(y|x_1 \dots x_{1n}) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \Longrightarrow \hat{y} = \arg\underbrace{max}_{y}\ P(y)\prod_{i=1}^{n} P(x_i|y) \tag{14}$$

To estimate $P(y)$ and $P(x_i|y)$, maximum a posteriori (MAP) can be used.

If the input data is continuous, a Gaussian assumption on the input features can be made. The parameters that need to be calculated for each feature x_i are: σ_y and μ_y using MAP on each of the classes that y represents. The Gaussian curve can be written as

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}} \tag{15}$$

2.2.4.A Naive Bayes with rejection

The Naive Bayes model is considered a good classifier and a bad estimator [11] in part because of the naive assumption that all the input features are independent among themselves. As such, the probability of a given sample being classified as a specific class, should not be taken as properly calibrated probability, however it can be used as an indicator of the estimation confidence, if a study is made to verify this level of confidence.

Equation (16) shows how the definition of a rejection level $N_{rejection}$ is made, and its application on the Naive Bayes model. If the assigned probability of a class is inferior to the $N_{rejection}$ the model discards that sample by not making the classification, effectively rejecting that classification opportunity due to the lack of confidence on the estimation.

$$\hat{y} = \arg\underbrace{max}_{y}\ P(y)\prod_{i=1}^{n} P(x_i|y)\,, \qquad if\ \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1 \dots x_n)} > N_{rejection} \tag{16}$$

2.2.5 Machine Learning (ML) Concepts used for Model Evaluation

In this chapter, the ML concepts used to evaluate the performance of the model will be described. These concepts consist on metrics to evaluate classification methods and ways to estimate the models' performance solely based on the training dataset.

2.2.5.A Accuracy

Accuracy is widely used as an empirical classification measure [12] [13]

$$accuracy = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1\,(\hat{y}_i = y_i) \tag{17}$$

If the predicted class $\hat{y}_i$ and the real occurrence y_i are equivalent, that sample was correctly classified and counts as 1 on the summation, otherwise it counts as 0. The arithmetic mean of this summation is called $accuracy$.

2.2.5.B Precision and Recall

Although $accuracy$ is an important metric, it is insufficient to understand and evaluate a model. Precision and recall are important and essential measures to analyse any classification model because they represent information which is not contained within the $accuracy$ measure. Figure 2-15 provides a graphical representation which will be used to better illustrate these concepts.

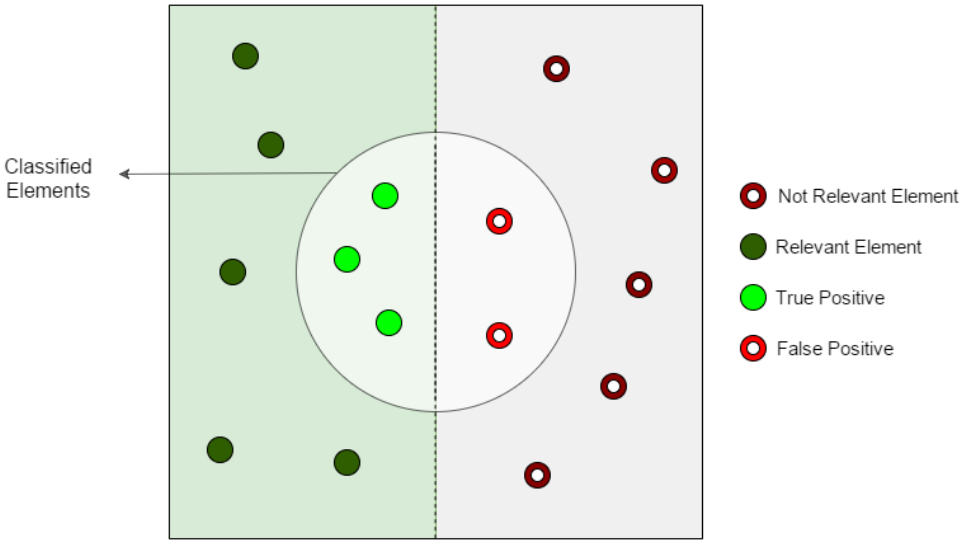


Figure 2-15 Diagram for explaining Precision and Recall

Precision measure is the fraction of positive elements which were correctly classified. In Figure 2-15 we can assign the following legend, $tp = True\ Positive$, $fp = False\ Positive$, to build

$$precision = \frac{\sum tp}{\sum tp + \sum fp} \tag{18}$$

If a binary classifier model is being evaluated, it can have different precision measures for each class. A system can have a very good performance on one of the classes, and a very poor one on the other. This is one of the types of information precision can reveal about a system.

Recall, also known as sensitivity, is the portion of correctly classified elements divided by the total

relevant elements, in other words, the portion of elements which are correctly classified within the total occurrences inside a specific class. If $fn = False\ Negative$ then

$$recall = \frac{\sum tp}{\sum tp + \sum fn} \tag{19}$$

2.2.5.C Cross Validation (CV)

CV is a technique used to estimate the performance [14] of an estimator given a dataset. Although there are many types of CV, for this thesis *k*-fold CV scheme will be used and so it is the only type that will be reviewed.

K-fold CV is often known as rotation estimation and its process consists of splitting a given dataset $\mathfrak{D}$ into *k* mutually exclusive subsets $\mathfrak{D}_1, \mathfrak{D}_2, \ldots, \mathfrak{D}_k$ of equal sizes. Then, the following operation will be repeated *k* times: the chosen ML model will be trained on all mutually exclusive sets, except one, which will be used as a test subset. In this test subset, accuracy will be computed. The loop stops when all mutually exclusive subsets $\mathfrak{D}_1, \mathfrak{D}_2, \ldots, \mathfrak{D}_k$ are used as test sets and their corresponded accuracy is computed. The estimated accuracy of the model in question, is the mean of all accuracies computed for each mutually exclusive set serving as a test set. The variance of the computed accuracies can be used to set confidence intervals around the mean accuracy level, a result derived from the central limit theorem [14] regarding the use of Binomial and Bernoulli distributions.

In Figure 2-16 there is an example of a 3-fold CV scheme where the accuracy of a binary classification model, which evaluates if a circle is red or green, is estimated. A dataset, which contains samples of green and red circles, is divided into 3 mutually exclusive subsets blue, purple and orange. To estimate the accuracy of the model, using 3-fold CV, there are three iterations in which each mutually exclusive subset serves as a test set, for a model which will be trained on the remaining subsets, in this case, the other two subsets. After these iterations are made, the mean of the computed accuracies from blue, purple and orange subsets will be the estimated accuracy of this model. The accuracy variance can also be calculated and is used to build confidence intervals, using standard deviation, where the real accuracy of the model should probabilistically fall into.

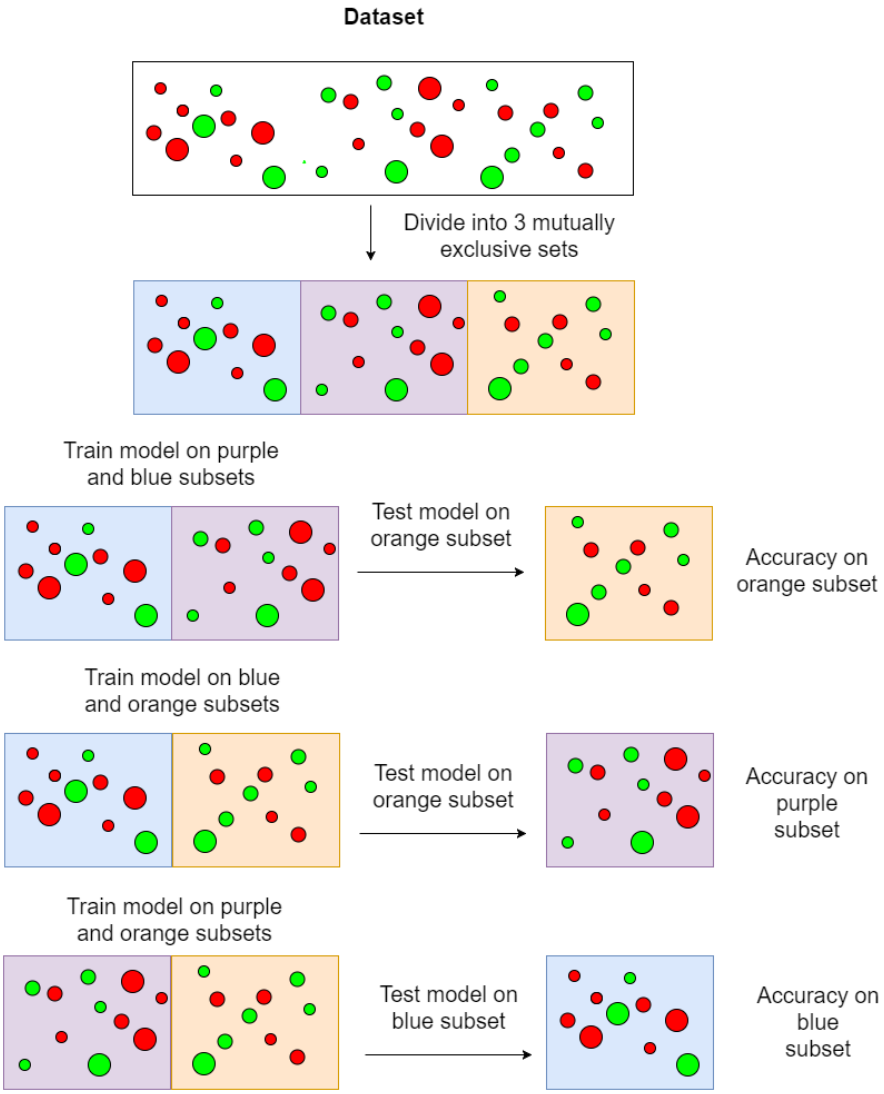


Figure 2-16 Diagram which represents a 3-Fold CV scheme

2.3 State-of-the-Art

2.3.1 Introduction

This chapter contains a literature review about current work on Foreign Exchange Markets, Financial Markets, GA, and Naive Bayes. This will provide some insight about what is being done in academia, regarding the financial world, and at the same time serving as a baseline for this thesis to be compared to.

2.3.2 Works on Financial Markets

In this Chapter, papers describing research on Financial Markets modelling will be overviewed. Various

methodologies and different architectures will be shown with mixed results.

Chan and Teong [15] used Neural Networks (NN) to enhance the use of technical indicators in Foreign Exchange Markets and published the paper in 1995. The frequency of data used was daily and the inputs of the NN are the Highs, Lows and Closes of the last five days. The quantity being predicted by the model is the value of the Highs, Lows and Closes of the instrument three days into the future thus calculating future values of technical indicators. Trades were then decided with pre-defined technical indicator rules calculated over the predicted value output by the NN. Potvin, Soriano and Vallée [16] used genetic programing to generate trading rules, buy or sell, based on technical indicators applied to the Canadian stock market. They made different rules for long and short strategies, considering that the short-term strategies should only be trained in a short span historical data. Both strategies were tested during a 256 days' historical period. The conclusion they arrive at is that the rules are only better than the buy and hold strategy when the market is relatively stable or with an average growth of 0%. Yao and Tan [17], in the year 2000, published a paper where they used NN to predict Foreign Exchange Market currency pairs. The frequency in which they analysed the market was on a weekly basis. For the inputs of the NN they used two methodologies, one using delayed time series and the other using technical indicators. They claim that the approach where they only used time series delayed data did not yield acceptable results, although, in their words the TA approach had impressive results. The metric used to evaluate the performance of the regression was Normalized Mean Squared Error (NMSE). Panda and Narasimhan [18] used NN to make a "one step ahead" regression model for the currency pair Indian Rupee/USD. The sample frequency was weekly and the period analysed between January 6 of 1994 and July 10 of 2010. They claim their approach had a significant improvement over a linear autoregressive and Random Walk (RW) methods and inputs for the models were the corresponding delayed time series. Hassan and Nath [19] used Hidden Markov Model (HMM) to predict the price change of airliners' stocks. The samples used were the daily prices of the stocks between December 18, 2002 and September 29, 2004. The performance criteria used was the Mean Absolute Percentage Error (MAPE). In their experimentation, they compared the proposed architecture with state of the art NN concluding that they had equivalent performances for the problem. The reasons the authors used HMM were that the predictions were "explainable" whereas the NN were not, also the computation time was much faster. Nowadays there have been some developments on the NN structure visualization (2015) [20] using t-SNE (2008) [21]. Huang, Nakamori and Wang [22] used Support Vector Machine (SVM) to predict the evolution of NIKKEI 225 index. They compare this methodology with RW, linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) also proposing an ensemble model of all these methodologies. The data used is from the period between January 1 of 1990 and December 31 of 2002. They claim SVM had a 73% prediction rate and the ensemble 75%. An interesting personal note is that only 36 observations are used as test samples which is not statistically significant. Kim [23] used SVM to predict daily variation direction of the Korea composite stock price index (KOSPI). For this analysis, he used technical indicators as the inputs of the model. The author also compared his approach with other state of the art methods such as the NN and case-based reasoning. The SVM had the best hit rate (57%) predicting the daily movement of the index.

2.3.3 Works on Genetic Algorithms (GA)

In this section, papers using GA will be overviewed, some of them will be specific applications, while other, novel methodologies on how to improve Holland's [9] original formulation.

Pinto, Neves, and Horta [3] used a Multi-Objective GA to optimize a set of trading strategies or rules. Technical indicators were used as inputs of the model with emphasis on the Volatility Index (VIX) and Pareto front to optimize the best tradeoff between risk and financial return. GA searched the best technical indicators to use and its weight on building the trading rule. The achieved results, between 2006 and 2014, were 10% higher annualized returns than the performance on buy and hold strategies for common indexes such as the NASDAQ, S&P 500, FTSE 100, DAX 30 and the NIKKEI 225. Bhanu and Lin [24] used a GA towards selecting features to discriminate the targets from the natural clutter false alarms in SAR images. They conclude that a GA is a successful method to select features when comparing with other state of the art procedures. Oh, Lee, and Moon [25] used several GA implementations to address feature selection. They proposed a hybrid GA to achieve better convergence properties in local search operations. After performing analysis on several UCI datasets and comparing their approach with several other methodologies they conclude that GA's are a methodology to take into account when dealing with the feature selection problem. Canelas, Neves, and Horta [26] used Symbolic Aggregate approximation (SAX) and a GA to analyse the presence of underlying patterns in stocks financial time series belonging to the S&P500 index. They managed to find several patterns and back testing showed it could perform better than buy and hold strategies while avoiding stock market crashes. Huang and Wang [27] used a GA to perform feature selection and parameters optimization on an SVM based model and compared it with standard parameter optimization algorithm, Grid search. They used several UCI datasets to compare the performance of both algorithms and concluded that by using a GA to perform this search, the model used less features and achieved higher hit rates than by using Grid search on all the datasets. Gorgulho, Neves and Horta [28] used a GA approach to manage a stock portfolio using technical indicators as model inputs. They conclude, throughout the testing period, that their approach beats buy and hold strategy and effectively avoids losing money on the market crash of 2008. Grefenstette [29] proposed a slight modification to the original GA, where some individuals, usually the worst evaluated on the current generation of the algorithm, are replaced by randomly generated ones tuning this effect by a hyperparameter called replacement rate. With this simple modification, the GA can maintain a continuous exploration of the search space throughout each generation, i.e., gaining some of the Monte Carlo algorithm properties while maintaining its ability for local searching of maximums or minimums. Later this modification to GA's was formalized as Random Immigrants' approach. Cobb and Grefenstette [30] compared three GA approaches: standard GA, Random Immigrants and a novel approach called Hyper Mutation, which consists of dynamically changing an individual's mutation probability to increase their local search capacity. Their work shows that both approaches have significant advantages over the standard GA approach originally proposed by Holland's [9] work. These ideas will be used in the proposed system architecture in this thesis. Oh, Joo and Kim [31] proposed a GA portfolio optimization scheme for index fund management. The paper objective is to prove that an index fund can greatly increase its performance, by using a GA

such as the one proposed. They compare their approach with the industry standard algorithm for picking stocks to follow a specific index. The chosen period of training and testing is between January of 1999 and December of 2001 and the index being followed is the KOSPI (Korea Composite Stock Price Index). They conclude that their GA approach follows the index with less variance and error than the industry approach. Kim and Shin [32] proposed a hybrid time series forecasting architecture relying on the GA, Adaptive Time Delay Neural Networks (ATNN) and Time Delaying Neural Networks (TDNN) to detect temporal patterns. The GA was used to tune important parameters of the NN such as the time delaying period. The index being forecasted was the KOSPI for a period of 833 days between January of 1997 and December 1999. The scoring metric used was Mean Squared Error (MSE). They concluded that their architecture achieved better scoring metrics (MSE of 3.38) than those systems without the GA (MSE of 4.41). Shin and Lee [33] proposed a GA architecture which produces models to predict corporate bankruptcy. They state the advantage of using such a system against state of the art NN is that it can produce an easily human readable rule that experts can understand. The objective of this research was a model which would serve as a basis for credit rating analysis. The output of the system produced a tree based rule which can be understood opposed to the NN approach. The dataset used was balanced and consisted of 528 externally audited mid-size manufacturing firms (264 filed for bankruptcy while the other 264 did not). The average hit-rate reported, on train and validation, is 80%. Siedlecki and Sklansky [34] use a GA for selecting features to design an automatic pattern classifier. The objective of their research was to prove if a GA was a reliable method to make feature selection over a large input space (more than 20 features), so that they can reduce the input space mitigating the curse of dimensionality while raising the credibility of the classifier. The conclusion arrived at is that a GA search, for this type of problem, significantly outperforms other exhaustive search methods for the same amount of computational time, such as the branch and bound method and sequential search.

2.3.4 Works on Naive Bayes

Although the Naive Bayes model is usually used on text classification tasks, the algorithm is very robust and can be used for every binary classification task. It has some unique properties, as it is a model created using probability theory while benefiting computational efficiency, since it is very lightweight. Lewis [10], in 1998, made a white paper reviewing various Naive Bayes methodologies since at the time this algorithm was already well known and studied. He explains why this model was so important when Natural Language Processing was still using Bag of Words representation to classify which subject was a document. The last topic of the paper focused on a very important question, the violation of independence (which is naively assumed by the model) which naturally occurs on almost every dataset in existence. He points out that there are many scientific papers which mathematically prove, that even if features present a high degree of dependence they don't affect the performance of the Naive Bayes classifier (even when an independence assumption was made). Zhang [11], in 2004, publishes a paper with a novel explanation on how the dependence distribution between input features plays a role on the performance of the Naive Bayes algorithm. A mathematical proof is presented, which proves that Naive Bayes can still be an optimal classifier if the dependences are distributed evenly between classes. It is also mathematically shown that if inter-dependence between features exist it might be cancelled out,

and in this way not making an impact on the algorithm classification performance. Huang, Lu, and Ling [35] made a comparison between Naive Bayes, SVM and Decision Tree classifier on various UCI datasets. Their conclusion is that they all have a similar level of performance in terms of accuracy. Smith and Bull [36] used Genetic Programing and a GA to pre-process data before feeding it into a Decision Tree, with the objective of enhancing the classifier performance. With this architecture, they manage to explore the automatic creation of new features from those already available on the data, thus exploring hidden relationships between features. Afterwards, they compared their architecture with the standard Decision tree model over ten different and publicly available datasets showing significant improvements on 8 of them. Subsequently, they substitute the Decision Tree classifier with other classifiers, such as Naive Bayes, and show that their approach maintains the results, proving its robustness.

In Table 2-2, a summary of the most important papers considered for this Thesis are shown.

Table 2-2 State of the Art Important Paper Summary

Reference Number	Publication Date	Number of Citations	Methodologies	Evaluation Metric	Type of Data	Dataset Period	Average Return (%)
[3]	2015	3	Multi-Objective Evolutionary System and VIX	Risk Exposure, ROI	NASDAQ, S&P 500. FTSE 100, DAX 30, NIKKEI 225	January 2004 to May 2014	8,06
[16]	2004	220	Genetic Programing	ROI	TSE 300	June 1992 to June 2000	-4,22
[18]	2006	90	NN	RMSE	Indian rupee/USD	January 1994 to July 2003	-
[25]	2004	596	Hybrid GA	Pure Atomic Operations	10 datasets from UCI	-	-
[28]	2011	49	GA	ROI	DJI	January 2003 to June 2009	20,99
[30]	1993	441	GA, Random Immigrants GA, Triggered Hypermutation GA	Average Population Performance	Stationary Function (Toy dataset)	-	-
[34]	1989	936	GA, k-NN (k-Nearest Neighbours)	Error Rate	Digitized Infrared Imagery of Real Scenes	-	-
[36]	2005	87	GA, Decision Tree, and Naive Bayes	Accuracy	10 datasets from UCI	-	-
This Work	2017	-	GA, Naive Bayes	Accuracy, ROI	EUR/USD Foreign Exchange Data	01.01.2013 to 09.03.2017	4,24%

2.4 Chapter Conclusions

In this Chapter, all fundamental concepts and research used for this thesis were described. All the building blocks were presented and they will serve as basis to build the optimization architecture for the binary classifier.

The GA, which will feature search and select was introduced and explained, and papers where similar methodologies proved to work, were reviewed. The proposed architecture will feature concepts acquired while studying these documents, such as, Random Immigrants and Hyper-Mutation.

The study of Naive Bayes' independence assumption is important for this thesis, since the input features for the model originate from the same source signal, i.e., input features might be correlated and dependent. It's possible to conclude from the papers reviewed, that the independence assumption of Naive Bayes can be violated since those dependencies might be cancelled out or evenly distributed between features without hurting the optimality of the classifier.

Chapter 3

Proposed Architecture

3.1 Overview

In the first part of this chapter the problem analysed is going to be formally defined and a general diagram showing the architecture is shown. Afterwards, each component of the diagram is thoroughly examined starting with a simple binary classifier, with no optimization, which will serve as a baseline in which the proposed architecture will be compared to. Then, the architecture developed for this thesis will be properly introduced followed by the description of the Market Simulator module. Finally, a random signal generator modelled using the time series *pdf* is described (this generator will be used to check if the architecture hallucinates and finds patterns on signals known a priori to be random). The theory used to understand this architecture was described in detail in the previous chapter. The GA presented here will contain ideas taken from paper reviews, such as Random Immigrants and Triggered Hypermutation [30].

3.2 General Perspective

The objective of this thesis is to build a classifier to predict whether the EUR/USD time series signal variations are chaotic or not, using TA as the input features. The diagram representing a general perspective of the architecture is shown in Figure 3-1.

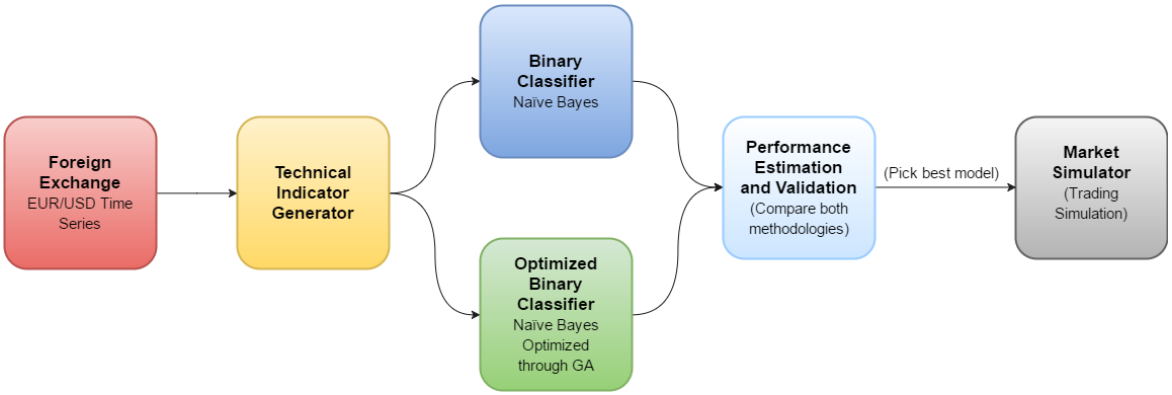


Figure 3-1 Diagram Representing a general perspective of the proposed architecture

From the Foreign Exchange time series data, in this case the series representing the EUR/USD rate, a set of Technical Indicators are obtained (in Chapter 2.2.2 the detailed description of how these indicators are calculated is discussed). This set of indicators are used to train two classifiers, a Naive Bayes binary classifier, and a GA optimized Naive Bayes binary classifier. After the training process, both performances of each methodology are estimated, validated, and compared. The best methodology is then used on a Market Simulator, which evaluates whether this approach could be used on the real

market to make a profit. This however is not the main objective of this work. The main objective is to check whether a GA can boost the performance of a simple binary classifier, when multiple combinations of the input features pose a combinatorial explosion.

For the rest of this Chapter, each building block will be described in detail, starting with the target formulation, the first step in building a binary classifier.

3.3 Target Formulation – Vector Y

When using methods of supervised learning, it is necessary to formulate which quantity is going to be predicted. As above mentioned, for this thesis a binary classifier is proposed, i.e., it is going to be predicted whether the signal which represents the EUR/USD rate is going to have a positive variation, or a negative one. The quantity predicted is going to be represented as y, as seen in Figure 2-1 (Y, in Figure 3-2, is a vector which contains all the targets available for the analysed period). Y follows a binomial probability distribution, i.e., $y \in \{0,1\}$, where 1 symbolizes if the signal had a positive variation and 0 a negative one. This process can also be called signal binarization through thresholding. If $Close_t$ is the closing price of the EUR/USD rate at the current hour, and $Close_{t-1}$ the closing price in the previous hour, then y_t can be defined as in the Equation (20).

$$y_t = \begin{cases} 1, & \frac{Close_t - Close_{t-1}}{Close_{t-1}} \geq 0 \\ 0, & \frac{Close_t - Close_{t-1}}{Close_{t-1}} < 0 \end{cases} \tag{20}$$

Now that y is defined, the problem can be written in a probabilistic way to use Naive Bayes: if x_t are the observable features at current time (Technical indicators), and y_{t+1} the direction of the signal in the future, then $P(y_{t+1}|x_t)$ can be computed by making an independence assumption as in Chapter 2.2.4.

3.4 Technical Indicator Generator - Matrix X

The objective of the Technical Indicator Generator is to generate the input features and samples used to train the binary classifiers in conjunction with vector Y. Foreign Exchange time series enters this generator, which contains all the equations described in Chapter 2.2.2, and outputs the matrix X.

Matrix X will be representing the input features (Technical indicators) of the Naive Bayes model. In Figure 3-2 there is a visual representation of X and Y across the time series. Each row of the X matrix represents a set of Technical Indicators observed at a specific hour, i.e., x_1 – set of technical indicators values observed at hour 1, x_2 – set of technical indicators values observed at hour 2 ... x_t – set of

technical indicators values observed at hour t. On the right of Figure 3-2 several target observations are shown.

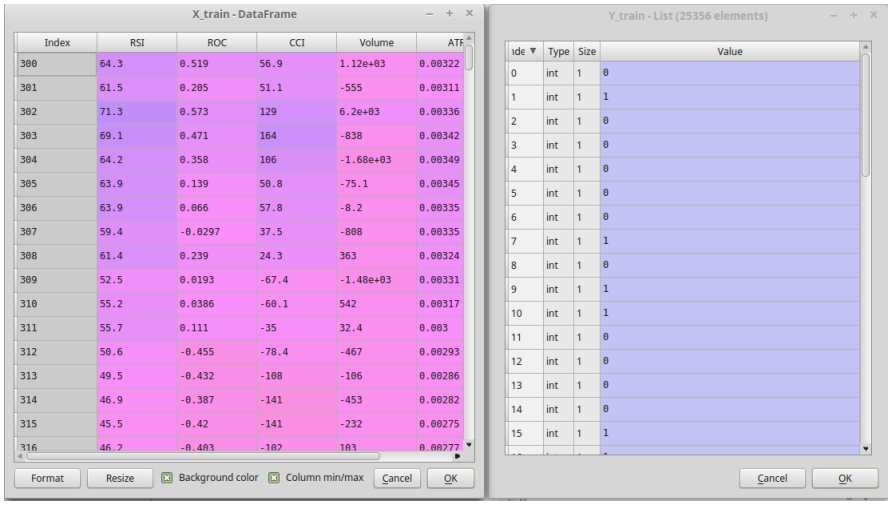


Figure 3-2 Matrix X and vector Y

3.5 Naive Bayes Binary Classifier

Now that the quantity to predict y (also known as target) is defined, and the sample matrix X is explained, it is time to introduce a diagram which joins the two, using the Naive Bayes methodology already presented before. In this model, all the features described in Chapter 2.2.2 are used with their parameters set to default values, and that's why it will be called unoptimized model. All the Technical Indicators have a default value which its creator chose as a starting point. For instance, the standard parameter *n* used to calculate the necessary EMA of the RSI is 14, but there is no guarantee that this is the optimal value for the EUR/USD instrument at the sampled hour frequency. To sum up, these are the standard values for each Technical Indicator:

- RSI - parameter *n*=14
- CCI - parameter *n*=14
- MACD - signal EMA *n*=9, fast EMA *n*=12, slow EMA *n*=26
- ROC - parameter *n*=10
- Stochastic Oscillator - parameter %K=5, parameter %D=3
- ATR - parameter *n*=14
- Volume - has no parameter to tune since it is a metric which represents the number of contracts occurred.

With these default values defined (and the time series information of the EUR/USD) a matrix like the one presented in Figure 3-2 can be constructed. In conjunction with the Y, a model like the one in Figure 3-3 can be formulated to make a binary classifier. $\hat{y}$ is the predicted value given the current observation

of the market x_t, i.e., $p(\hat{y})$ is the probability estimation of the market going up given x_t and $1 - p(\hat{y})$ the probability estimation of the market going down. The way Naive Bayes is trained is described in Chapter 2.2.4. In the System Evaluation chapter, this model will also include the rejection variant described in Chapter 2.2.4.A.

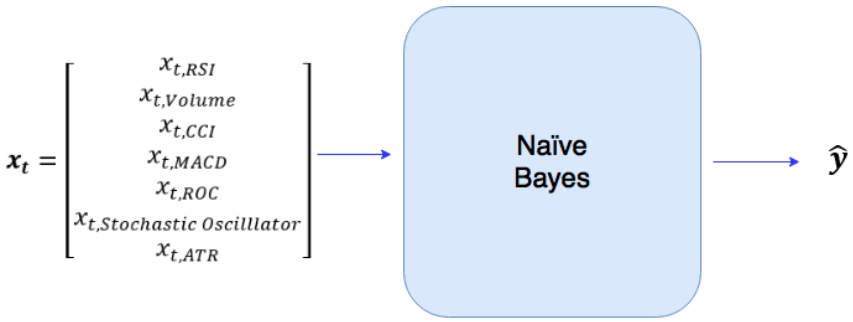


Figure 3-3 Naive Bayes unoptimized model

In Figure 3-4 a general diagram of the unoptimized model is shown. The time series will be split into a train set and test set. After estimating the performance of the binary classifier using a 7-fold CV scheme, the test set will be used to further validate this result. If the test set has enough samples it is statistically significant and can confirm that this CV scheme makes good performance estimations on the overall accuracy of the classifier.

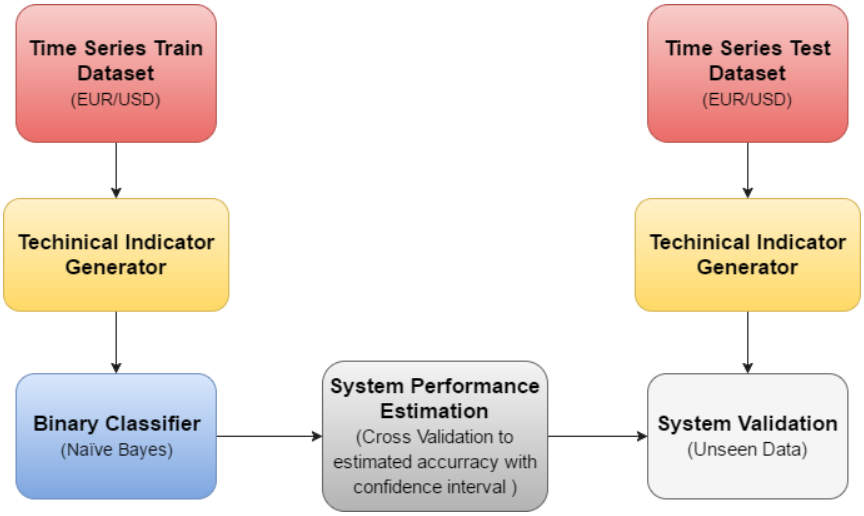


Figure 3-4 Unoptimized architecture

3.6 Optimized Naive Bayes through GA search

Like it was implied in the previous chapter there is no guarantee a specific parameter chosen for a Technical indicator is the one which translates to the best accuracy, given the EUR/USD financial instrument. Likewise, since almost every Technical Indicator involves the calculation of a mean, some of them might overlap and be redundant (for instance RSI and CCI have the same default parameter

which they use to calculate the mean of the given time series). It is clear, that there might be a combination of features (and feature parameters) which will result in the best overall accuracy of the system, but since there are more possible combinations than the computational power available to try a purely Monte Carlo search (or another exhaustive search), a GA approach is proposed to search the feature space efficiently.

First, a general diagram of the system is going to be thoroughly described, afterwards specifics about the GA used will be discussed, such as the alterations made to the original approach suggested by Holland's work [9]. Both the implementation of the GA (with the resort of the DEAP framework [37]) and Naive Bayes were original and written in Python 3 programing language. The Technical Indicator Generator was also coded in Python 3 encapsulating all those formulae described in Chapter 2.2.2 to correctly transform the time series input. For graphical plotting Matplotlib [38] was used. Scikit [39] framework served as a rapid ML prototyping tool.

In Figure 3-5 a simplified view of the proposed architecture is shown. Given a binary classifier, and a time series dataset, the GA is going to search and optimize the set of technical indicators which maximizes the accuracy of the given classifier. The output of the system, not only gives the best subset of Technical Indicators, but also makes an estimation, with confidence intervals, on what the performance of the system on unseen data will be. CV is used to prevent overfitting and estimate the accuracy confidence intervals. For further confirmation, the system will be validated using a test set, unseen in the optimization process.

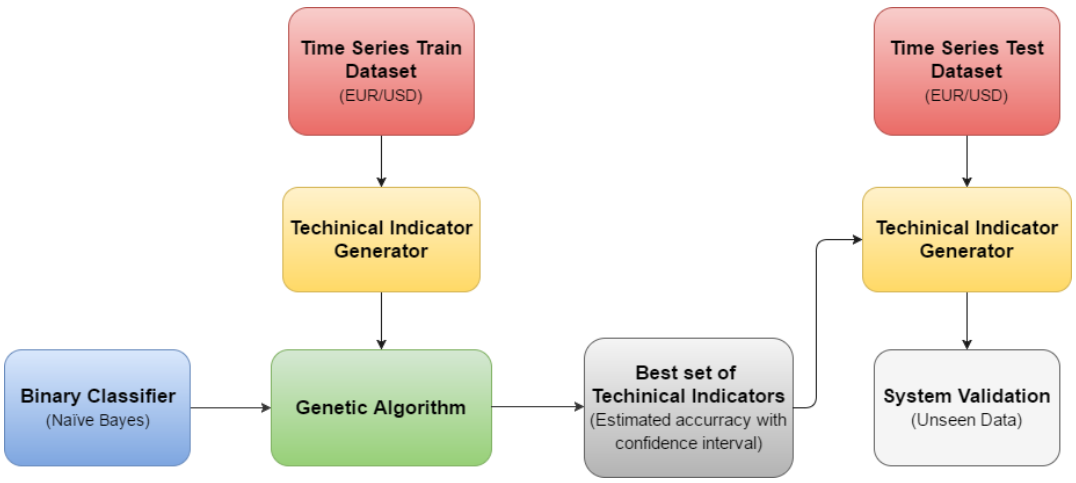


Figure 3-5 Simplified overview of the proposed architecture

To comprehend what the GA is doing exactly it is necessary to present a more detailed diagram of that block. In Figure 3-6 a diagram which represents the functioning of the GA across generations is shown, and below, the pipeline explanation. This approach searches multiple combinations of Technical Indicators and estimates what is the combination that can provide the best overall CV accuracy estimation of the system.

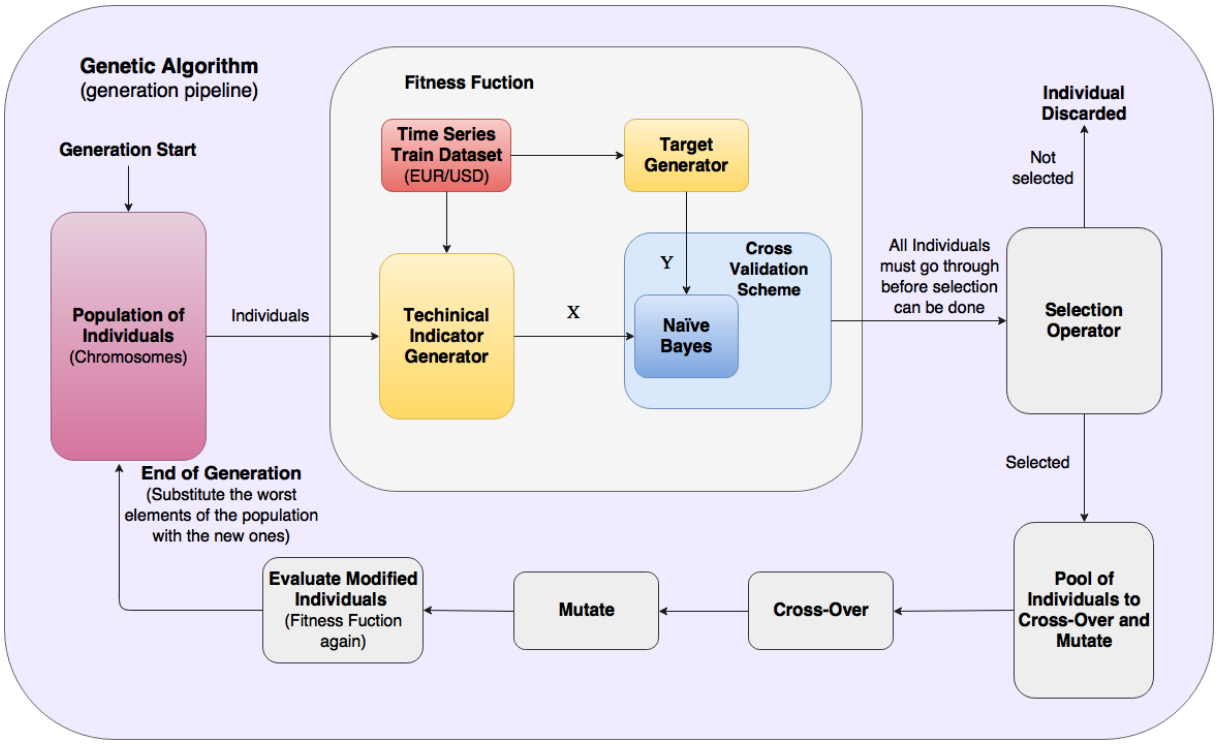


Figure 3-6 Genetic Algorithm Overview

Each generation starts with a population of chromosomes which need their fitness evaluated hence they are fed into the fitness function. Each chromosome has encoded the information that the Technical Indicator Generator needs to generate its unique set of Technical Indicators (X matrix), using the information obtained from the time series dataset. Before the X matrix is used to train the Naive Bayes model, the target vector Y is needed. Once again, this target is generated from the time series dataset. Now that X and Y are calculated, the Naive Bayes model can be trained, and its performance estimated (accuracy metric described in Chapter 2.2.5.A) using a 7-fold CV Scheme (this is the fitness value of the individual, i.e., the accuracy of the Naive Bayes model given a combination of Technical Indicators). After all the individuals have their fitness calculated they go through the Selection operator which decides, through a tournament, which are selected and which are discarded. For the resulting individuals, Cross-over and Mutation operators are applied. The newly formed chromosomes (individuals) need to have their fitness evaluated before they can be compared to the old ones in the population. The individuals from the population that are worse than the newly formed ones, are replaced, finishing this generation cycle.

The Selection operator used makes tournaments between three individuals, and the Crossover operator chosen is Single Point Crossover.

Since individuals don't share interdependencies among themselves, the fitness function can be completely and safely parallelized across all computational units available. A linear speed-up is achievable which represents a major advantage of using a GA. The Naive Bayes Classifier is chosen, since it is one of the most computational efficient ML models, hence it can be re-calculated more often in the same amount of time. When the chromosome is defined, it will be easy to understand that there are more than 10^{80} combinations to search for.

3.6.1.A Chromosome Representation

The Chromosome encodes the Gene information necessary for the Technical Indicator Generator to generate a set of Technical Indicators. Every information necessary to construct each Technical Indicator presented in Chapter 2.2.2 is mapped into a vector which constitutes the Chromosome. Each Technical indicator contains a Selector, i.e., the value being 1 means it will be produced by the Technical Indicator Generator and included in X matrix, while a value of 0 means, that in this chromosome, this technical indicator will be discarded and non-present in the X matrix. All the parameters to construct the Technical Indicators means are bounded between $[2,300]$. A value lower than 2, makes it impossible to construct a discrete mean. The upper bound value (300) was chosen to limit the search space of the algorithm, since these parameter limitations already make it unfeasible to search the feature space with an exhaustive search ($> 10^{21}$ combinations). Furthermore, from the Financial perspective, a signal value from 300 hours ago, should have no contribution to next hour variation.

This vector in the algorithm, is composed by 16 real numbers of values bounded between $[0,1]$. These values are then mapped, using Table 3-1, to their respective intervals. The reason the values are bounded between $[0,1]$ is to provide a universal mutation operation function. This way, the mutation function can always apply the same mutation operation without having hard-coded edge cases, i.e., some genes can't be lower than 2 (parameters) while others can't be lower than 0 (selectors). Also, the Gaussian Mutation operation in the value scale $[0,1]$ is always non-parametric, which is an algorithmic advantage, and makes sense with the original formulation of Holland's work [9].

Chromosome Gene Mapping							
RSI		Volume	CCI		Stochastic Oscillator		
Selector	*n* parameter	Selector	Selector	*n* parameter	Selector	%K parameter	%D parameter
0 or 1	2 to 300	0 or 1	0 or 1	2 to 300	0 or 1	2 to 300	2 to 300
MACD				ROC		ATR	
Selector	Signal EMA	Slow EMA	Fast EMA	Selector	*n* parameter	Selector	*n* parameter
0 or 1	2 to 300	2 to 300	2 to 300	0 or 1	2 to 300	0 or 1	2 to 300

Table 3-1 Gene information contained in each Chromosome

3.6.1.B Random Immigrants

This modification to the original GA implementation, called random Immigrants, was proposed by Grefenstette [29]. As explained in Chapter 2.3.3 this alteration replaces the worst elements of the population, at the end of each generation, with randomly generated ones. This procedure, is controlled by a hyper-parameter called replacement rate (which is a percentage of the population). By doing this,

the GA gains the ability of searching different space regions even when it has almost converged, i.e., it gains some of the Monte Carlo's exhaustive search properties. This can be beneficial to escape a converging to a local maxima/minima situation. Because the worst elements of the population are the ones being replaced, it doesn't affect the converge rate of the algorithm negatively, considering that the replacement rate is chosen in a conservative way. In Table 3-2 there is the pseudo code which encapsulates the original GA approach with the Random Immigrants modification.

In Chapter 4 both the standard GA approach and Random Immigrants will be compared.

```
generation n=0
initialize population P(n) with individuals
evaluate P(n) fitness
while fitness critireon not met:
      n=n+1
      select to reproduce a subset S(n) of P(n-1)
      cross-over and mutate S(n)
      form S'(n)
      evaluate fitness in S'(n)
      generate random individuals R'(n)
      replace P(n) from S'(n), P(n-1) and R'(n)
```

Table 3-2 Original GA pseudo code with Random Immigrants modification

3.6.1.C Hyper-Mutation

Hyper-Mutation was an alteration to the original GA code proposed in [30] where the mutation rate changed to a higher level when the trigger is fired. In this thesis, not having an improvement over three subsequent generations, is the trigger chosen. The mutation operation is the responsible GA operator for local search, therefore, when fitness hasn't improved over three subsequent generations it considers that it is very close to the maximum fitness, consequently searching around that solution is the best course of action. After the trigger is fired, the mutation rate is doubled, and stays this way till the end of the algorithm run. This is different from what the original article suggested (where they de-activate hyper-mutation), but testing has shown that forcing local search in this specific problem is the best solution. The pseudo code for this alteration is in Table 3-3 and it includes the Random Immigrants' modification presented in Table 3-2.

```
generation n=0
initialize population P(n) with individuals
evaluate P(n) fitness
while fitness critireon not met:
      n=n+1
      select to reproduce a subset S(n) of P(n-1)
      if fitness is not higher than 3 generations ago:
            cross-over and hyper-mutate S(n)
      else:
            cross-over and mutate S(n)
      form S'(n)
      evaluate fitness in S'(n)
      generate random individuals R'(n)
      replace P(n) from S'(n), P(n-1) and R'(n)
```

Table 3-3 Original GA pseudo code with Random Immigrants and Hyper-Mutation modification

3.6.1.D Elitism

The final alteration to the standard GA implementation is Elitism. With the use of Crossover and Mutation there is the possibility that the fittest individual of a population gets discarded during one of these operations. This is a clear fault while optimizing an objective function (since the best solution might be lost across the search). Elitism provides a solution to this problem; the fittest individuals of a population (across all generations) are always propagated to the next iteration, unchanged.

For this thesis, only the best individual is propagated, ensuring that the best solution yet is never discarded.

3.7 Market Simulator

The Market Simulator is a software developed to estimate the performance of a hypothetical model, simulating market orders, on a specific dataset. The inputs of the simulator are a binary classifier and a time series dataset. The results are detailed graphics and metrics of its performance. In Figure 3-7 a diagram of the simulator is shown.

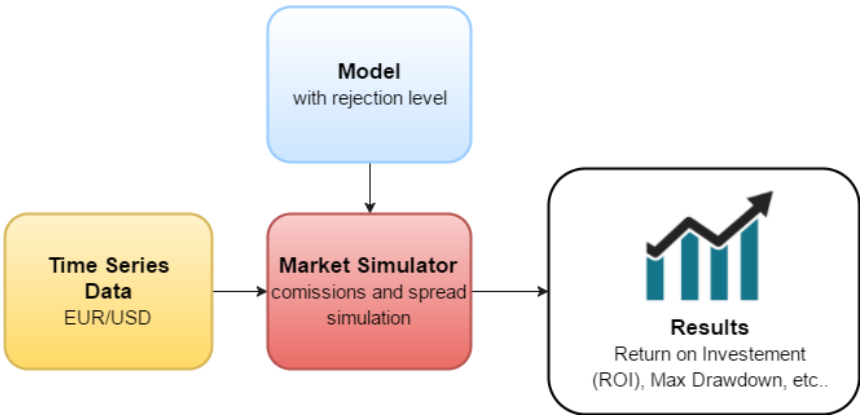


Figure 3-7 Diagram of the Market Simulator

The simulator was designed like a state machine, Long and Short being its principal states. In Financial Markets a Long position is when an Investor makes a contract with a broker, in which it gets more value if a positive variation of a given asset, such as the EUR/USD, happens in the future. On the other hand, if the EUR/USD evolves in a negative way, such as the USD becomes more valuable in relation to the EUR, that same contract loses value. The Short position is the inverse of the Long position, the investor contract with the broker gains value, if the USD becomes more valuable in relation to the EUR, or loses its value if the opposite occurs. To make a contract with a broker a fee applies; in the case of the EUR/USD, the commission for each transaction is encapsulated on the simulator, 10 USD dollars per 1 million USD traded.

Transaction costs are not the only fee associated in making a market trade. The Spread is the difference between the Bid and Ask prices. For this thesis, all the signal analysis was made with the mean price between the Ask and Bid, called midpoint. When an investor wants to make a Long contract with the broker, he “bids” the market with an offer, hence the Bid price. On the other hand, when an investor wants to make a Short contract, he “asks” the market, hence the Ask price. The Bid and Ask price are always separated by a dynamic spread which must be accounted for while simulating trades (the Bid is always lower than the Ask).

Each simulated contract is always using a fixed amount of investment, i.e., if the model has a positive or negative return with an operation, the next trade will still be the fixed predefined value. In this way the normalized ROI can be computed.

The simulator has 4 states in which he operates:

- LONG – A Long contract is opened with all the fees associated, such as the spread and transactional costs. The variation of the asset is registered and saved for performance analysis
- SHORT- A Short contract is opened with all the fees associated with the transaction, in the same way as Long. The variation of the asset is registered and saved for performance analysis
- HOLD – The contract currently opened stays this way, hence no transaction fees are applied and only the variation of the asset is accounted for the models performance
- OUT – The simulator stays out of the market or the currently opened contract is closed, with all the transaction fees associated.

In Figure 3-8, a diagram illustrating the Market Simulator State Machine is shown.

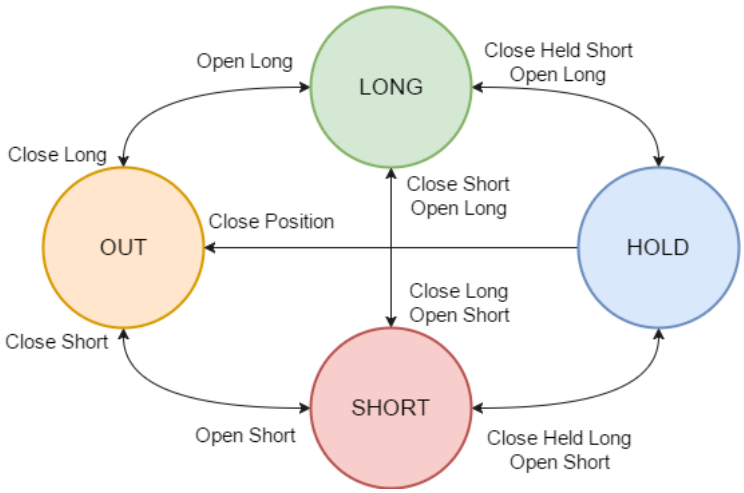


Figure 3-8 State Machine Diagram of the Market Simulator

The Market Simulator queries the model for a signal, which is used to indicate what the next action will be, resulting in a new state (Long, Short, Hold or Out position) for the next time period (hour). The Market Simulator provides the model with the data to generate the signal necessary for the action. Given the received signal, the state machine diagram in Figure 3-8 is followed for each of the hour samples present in the time series. The metrics regarding the market performance of the Model are saved for analysis.

Two important Metrics are shown in a plot, the ROI, and Maximum Drawdown.

The ROI formula between hour period t and $t-1$, is given by the Equation (21).

$$ROI_{t-1\to t} = \frac{Close_t - Close_{t-1}}{Close_{t-1}} \tag{21}$$

Maximum Drawdown is the maximum decline, from a peak, in some stochastic financial variable [40]. Equation (22) has the formal definition used for this simulator.

$$MDD(T) = \max_{\tau \in (0,T)} \left[\max_{t \in (0,\tau)} ROI_t - ROI_\tau \right] \tag{22}$$

3.8 Random Signal Generator

In the architecture described above, the original EUR/USD dataset is split into train and test so there can be a validation of the system. An interesting question to ask is, can this type of architecture identify a signal that is known a priori to be random? For this purpose, a Random Signal Generator has to be created to further validate the performance of the architecture. If the architecture is well built, it should be able to discard signals that are random.

The Random Signal Generator can be modelled to look like the original EUR/USD dataset, but

completely random. Consider for a moment the signal that describes the EUR/USD behaviour has no patterns, i.e., it is completely random and there is no correlation between prices at different hours. If this is true, the *pdf* observed can be used to construct an equivalent signal (or many of them). The first step to generate the Random Signal Generator is to observe the histograms of the variations (Figure 2-1), at each hour, between the Open price/Low prices presented below in Figure 3-9; the Open price/High price presented below in Figure 3-10; the Open price/Close prices presented below inn Figure 3-11; the Volume presented below in Figure 3-12. By drawing samples from these source distributions, a signal like the EUR/USD is generated, but with no correlation, hence there are no patterns to be found. A binary classifier, should have 50% accuracy on a signal generated by this method.

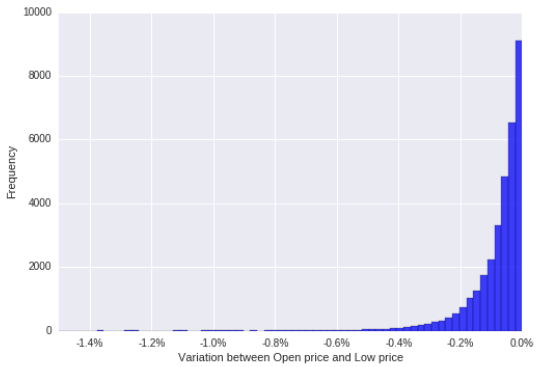


Figure 3-9 Percentage Variation between Open and Low prices histogram

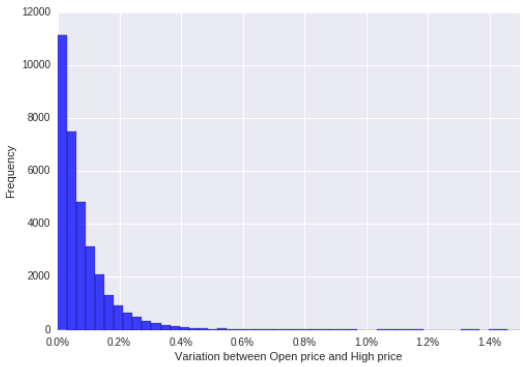


Figure 3-10 Percentage Variation between Open and High prices histogram

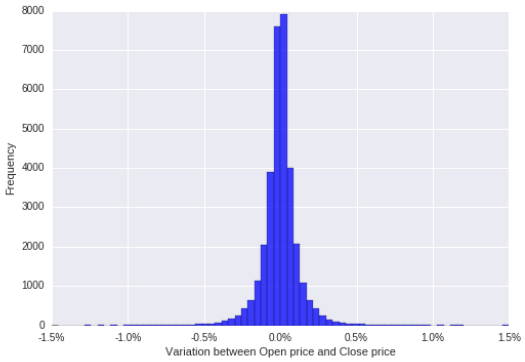


Figure 3-11 Percentage Variation between Open and Close prices

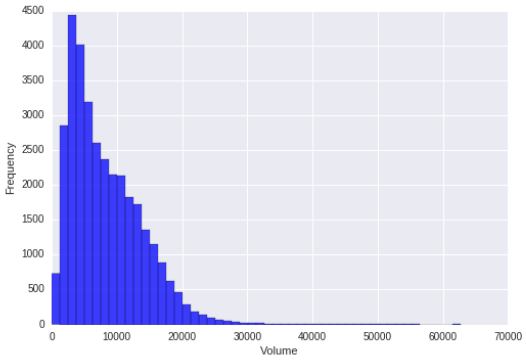


Figure 3-12 Volume histogram

To sample from these histograms and approximation to a probability density function was made using MCMC. This class of algorithms is widely used in statistics, econometrics, physics, and computer science [41] for optimization and integration solving. Instead of “guessing” which parameters are optimal for approximating Figure 3-9, Figure 3-10, Figure 3-11 and Figure 3-12 to the Cauchy and Half-Cauchy *pdf*, MCMC can be used to this purpose. In this way, the uncertainty of these parameters can be measured and seen in a trace plot.

The specific algorithm used was the Metropolis–Hastings and the framework in which the sampling was done is called PyMC3 [42]. In Figure 3-13, Figure 3-14, Figure 3-15 and Figure 3-16 are the trace plots regarding the MCMC approximation without the burn-in period.

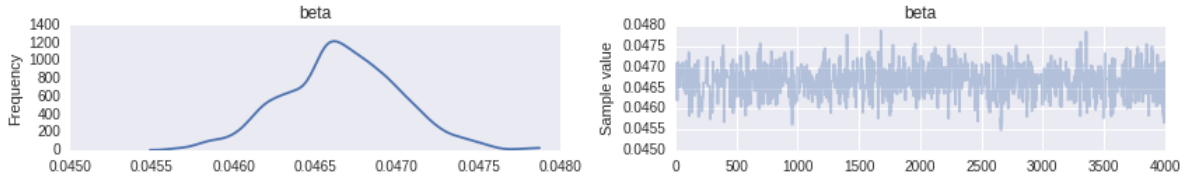


Figure 3-13 Trace-plot for Half-Cauchy MCMC approximation to the Percentage Variation between Open and Low prices

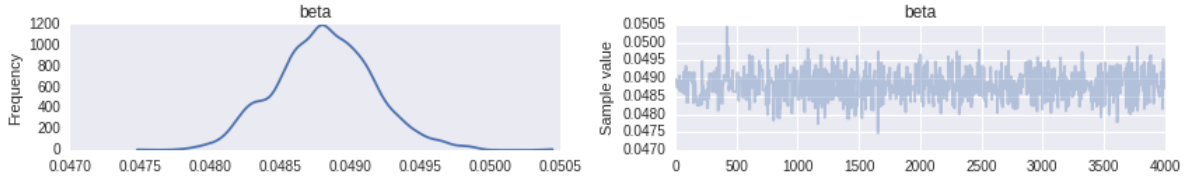


Figure 3-14 Trace-plot for Half-Cauchy distribution MCMC approximation to the Percentage Variation between Open and High prices

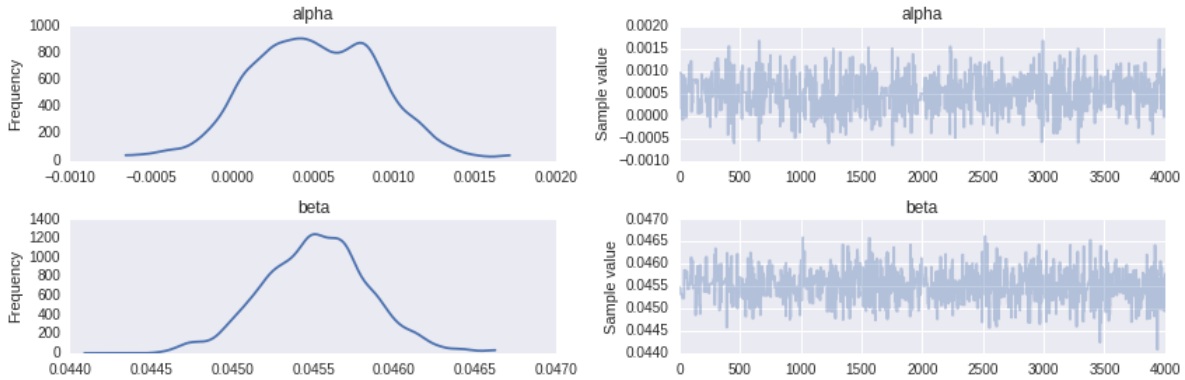


Figure 3-15 Trace-plot for Cauchy distribution MCMC approximation to the Percentage Variation between Open and Close prices

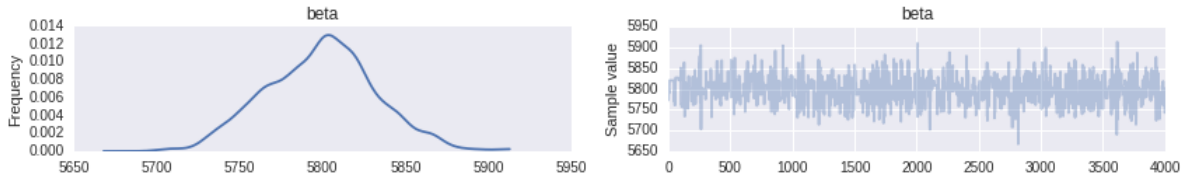


Figure 3-16 Trace-plot for Half-Cauchy distribution MCMC approximation to the Volume

In Figure 3-17, Figure 3-18, Figure 3-19 and Figure 3-20 is the posterior predictive check ,i.e., samples drawn from the distributions approximated with MCMC over the original set of market observations. For every case, except the Volume, the Cauchy and Half-Cauchy assumptions turn out to be good simulators of the market.

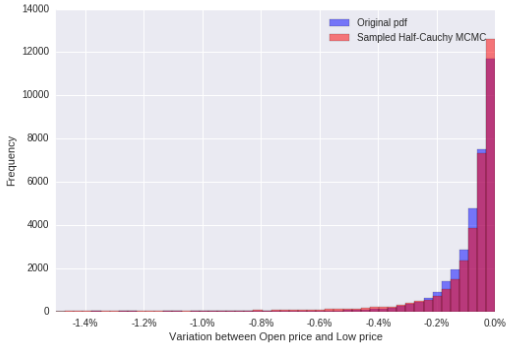


Figure 3-17 Sampled Variation between Open and Low prices histogram

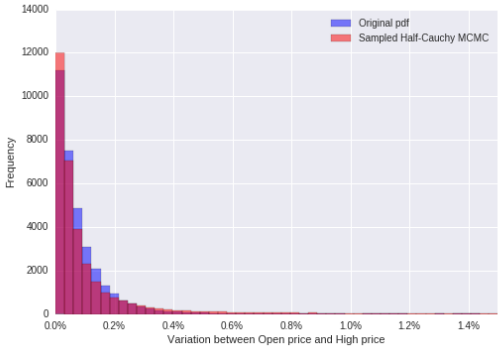


Figure 3-18 Sampled Variation between Open and High prices histogram

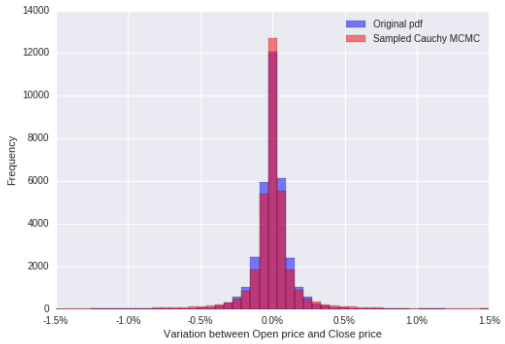


Figure 3-19 Sampled Variation between Open and Close prices histogram

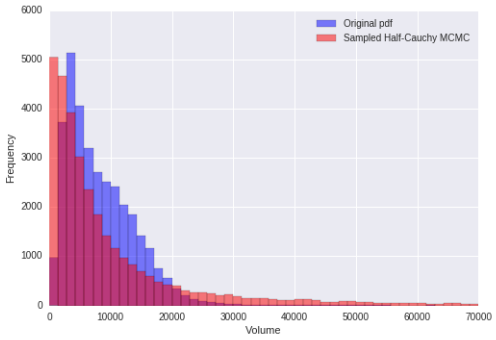


Figure 3-20 Sampled Volume histogram

The idea of the proposed generator, is to generate a random signal. Since each sample drawn from the approximated *pdfs* is an independent event, there is no correlation with the sample previously drawn. A binary classifier shouldn't be able to predict whether the next value of the series has a positive or negative variation. The reason the Cauchy and Half-Cauchy distributions were chosen were to make the signal more "similar" to the market from the point of view of the system, but the objective of the generator is not to fully emulate the signal. For that purpose, a Generative model [42] should be built and is out of scope of this thesis.

Now that the all *pdfs* are estimated, multiple random signals can be generated. The initial price in which the random signal starts is the same price as the original EUR/USD dataset. This way, it is expected that the random signals will be bounded on the same price scale. From the original starting price, each subsequent variation will be drawn from the estimated *pdf* described above and multiplied be the last price. Signals randomly generated this way are present in Figure 3-21 and Figure 3-22. The original EUR/USD signal is in Figure 2-2.

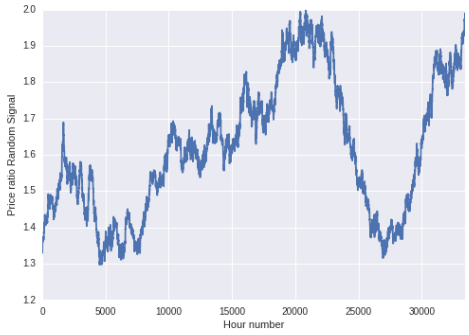


Figure 3-21 First randomly generated Signal

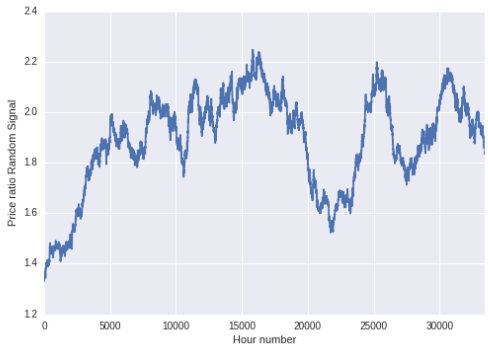


Figure 3-22 Second randomly generated Signal

3.9 Chapter Conclusions

The simple binary classifier role is to prove whether the pursuit of an optimization procedure is worth it, i.e., how random is the binarization of a given time series.

The proposed optimization for the binary classifier will try to find more meaningful features, than those generated by the default parameters documented by technical indicator creators.

Market Simulator will provide the metrics necessary to evaluate whether a model like the ones described, would perform in real market conditions.

Finally, generated signals known a priori to be random, produced with the Random Signal Generator, will be classified by the simple binary classifier to evaluate its ability to discard them.

All components being evaluated in Chapter 4 are presented and described in detail. The proposed architectural main decisions are reasoned and properly based on the literature reviewed in Chapter 2.

Chapter 4

System Evaluation

4.1 Overview

In System Evaluation Chapter, the proposed architecture is benchmarked in a diverse number of tests. The data originates from the EUR/USD time series dataset, as shown in Figure 2-1. First, a split on the dataset is defined, dividing the EUR/USD time series in train and test set. All the performance estimation will be made in the train set (the test set will be used to evaluate if the performance estimation holds on unseen data).

Afterwards six case studies are presented:

- Case Study A – The Naive Bayes binary classifier is benchmarked in train and test set without any feature optimization. Improved Rejection classifier and Market Simulation results are presented
- Case Study B – The optimization proposed in Chapter 3 is benchmarked and compared to the original Case Study A approach. The optimized Rejection variant of the Naive Bayes is also examined and compared. Finally, the model performance on Market Simulation is reviewed.
- Case Study C – The Classical GA approach is compared to the proposed modifications described on Chapter 3.6 (convergence rate and best individual)
- Case Study D – Random Signals generated using the model presented on Chapter 3.8 are fed to the Naive Bayes model to evaluate its ability to discard a signal known a priori to be random
- Case Study E - An attempt to visualize the model decisions is made using a Manifold method called *t*-SNE [21]

4.2 Train and Test Split

To test the proposed architecture, the time series representing the EUR/USD currency pair is chosen (Figure 2-1). To have reliable performance estimation a split in this time series is necessary, dividing it in a train and test set. The train set will be used to generate all the models in this this chapter and to estimate its future performance. The test will always be held out, until the training process is finished and the performance confidence intervals estimated. After this step, the system performance is validated on the unseen test set. Although CV estimation is considered reliable, as a real performance estimator, having a test set held out is an extra layer of system validation.

The EUR/USD dataset contains all the time series information between the period from 01.01.2013 until 09.03.2017 sampled at an hourly rate. The training set will be the first 80% of the time series, leaving the test set with the final 20%. Figure 4-1 is a visual representation of this spliting.

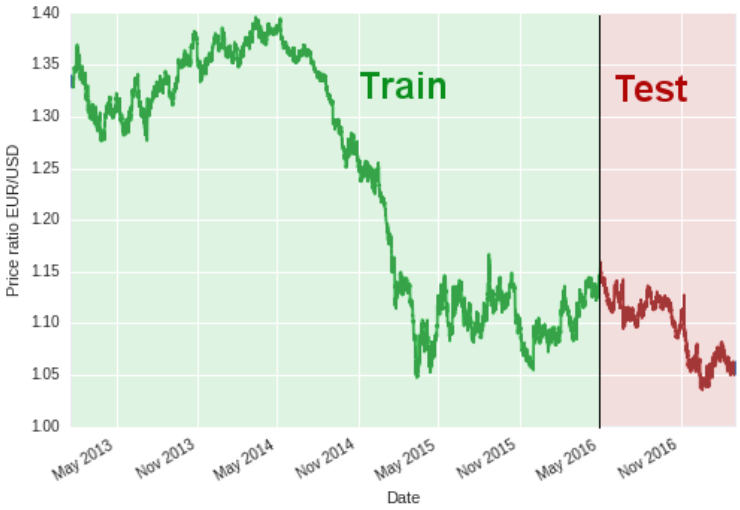


Figure 4-1 EUR/USD time series signal divided into train and test sets

By making this split, the training set contains 20726 training samples leaving the test set with 5182 samples. This dataset split is valid, since the number of samples contained in each set is statistically significant.

4.3 Case Study A – Simple Binary Classifier

In first case study, the Naive Bayes binary classifier is used to predict whether the EUR/USD time series signal is going to have a positive variation, i.e., $Close_{t+1} > Close_t$, or a negative one, in the next hour period. The training and test sets are the ones presented above on Chapter 4.2. The input features are the Technical Indicators (RSI, Volume, CCI, MACD, ROC, Stochastic Oscillator, ATR) with their parameters initialized by their default setting, as described in Chapter 3.5. This model has no optimization and will serve as a baseline which the proposed architecture will be compared to. The rejection variant of Naive Bayes, Chapter 2.2.4.A, will be also be analysed in an attempt to improve the binary classifier, since the proposed architecture will also make use of the rejection variant model.

4.3.1 Cross-Validation Results

The first step is to estimate the performance using *k*-fold CV scheme as shown in Chapter 2.2.5.C. In Figure 4-2 and Figure 4-3 are plots, showing what is the estimated accuracy of the model and its respective 95% confidence interval while rising the number of folds used in the CV scheme.

The number of samples in the test set, regarding the CV estimation, decreases with the rising number of folds. With a higher number of folds, the CV training set has more samples, so it is expected that the absolute value of the accuracy estimation is better. After examining the plots below, a 7-fold CV scheme

was chosen, since it has an acceptable computational to performance estimation ratio. Note that increasing the number of folds causes the computational time to rise proportionally. The estimated accuracy for a 7-fold CV scheme is $53{,}4\ \%\ (+/-\ 2{,}2\%)$.

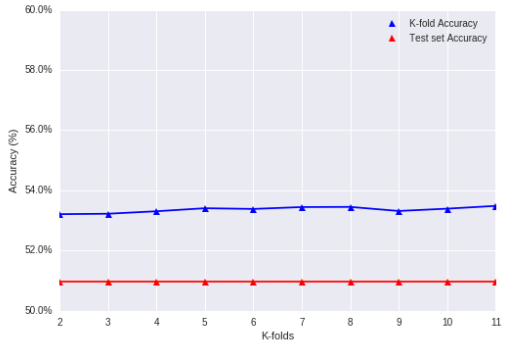


Figure 4-2 Estimated accuracy through *k*-fold CV scheme

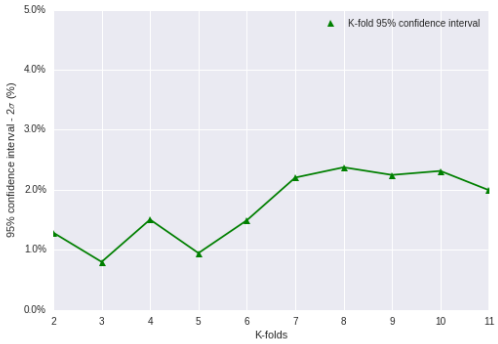


Figure 4-3 Estimated 95% confidence interval of the accuracy

4.3.2 Train and Test Results

After having the model performance estimated, the metrics accuracy, recall and precision have been measured in both the train and test set. The training accuracy is $53{,}4\%$. On Table 4-1 the precision and recall measures are presented regarding the train set.

Class	Precision	Recall	Number of Occurrences
0	53,01%	54,91%	10292
1	53,89%	51,99%	10434
Average/Total	53,46%	53,44%	20726

Table 4-1 Metrics of the binary classifier in train set

The accuracy in the test set is $50{,}96\%$. On Table 4-2 the precision and recall measures are presented regarding the test set. Since the test set is a held out one (never seen on the training process) it serves to validate the performance estimation made by the CV scheme on the previous topic.

Class	Precision	Recall	Number of Occurrences
0	51,13%	49,77%	2598
1	50,81%	52,17%	2584
Average/Total	50,97%	50,96%	5182

Table 4-2 Metrics of the binary classifier in test set

4.3.3 Rejection Naive Bayes

As described in Chapter 2.2.4.A it is possible to enhance the accuracy of a simple binary model by rejecting some of the samples. These samples will be rejected if the model is too uncertain about what class they belong to. Although there are some areas where rejecting samples is not the optimal course of action, in a financial trading model, it is beneficial to reject classifying some samples, to mitigate the high risk of misclassifying.

To introduce sample rejection a $N_{rejection}$ as to be chosen, as shown in Equation (16). To understand how different $N_{rejection}$ values might impact the model performance it is necessary to make a plot over all the possible values $N_{rejection}$ can take. In Figure 4-4 and Figure 4-5 the accuracy progression on train and test set with different $N_{rejection}$ values are plotted. The test set reflects the tendency shown on the train set, i.e., a higher $N_{rejection}$ value translates to a better overall accuracy of the system.

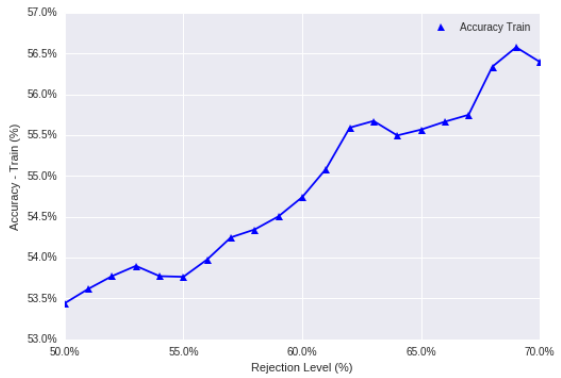


Figure 4-4 Train set accuracy improvement with the $N_{rejection}$ modification

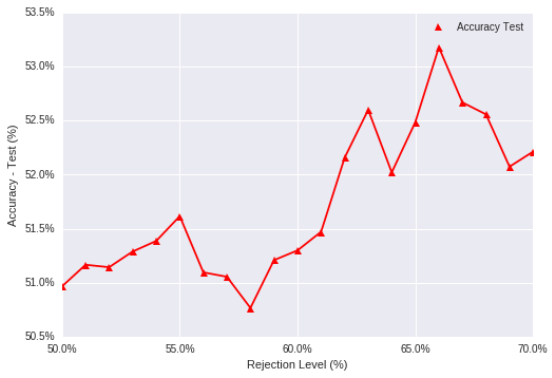


Figure 4-5 Test accuracy improvement with the $N_{rejection}$ modification

With the results of Figure 4-4 and Figure 4-5 it is important to study how many samples are rejected, and classified, with the increase of $N_{rejection}$. In Figure 4-6 and Figure 4-7 the number of samples classified over the rejection level, in train and test set are plotted. When the rejection level is >70% only less than 10% of the samples are being classified, i.e., ≈ 90% of samples are being discarded. If the test set has a total of 5182 samples and 90% are being discarded, only 518 are being classified, this is

an important fact to take in consideration when analysing statistically significant test sets.

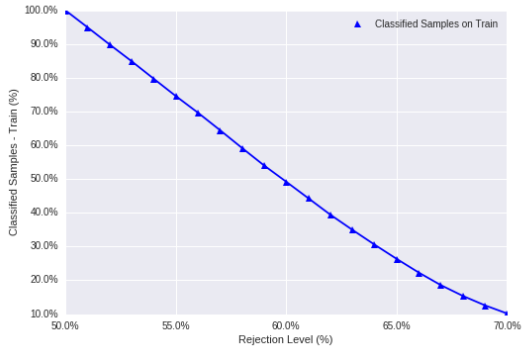


Figure 4-6 Number of samples classified in train set over rejection level

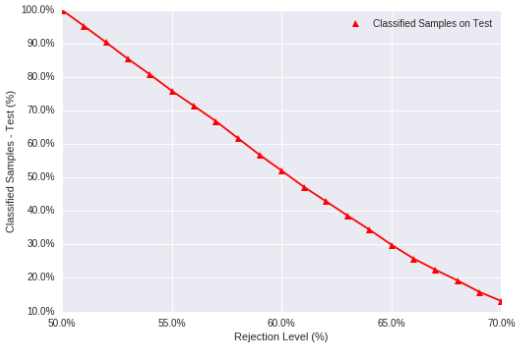


Figure 4-7 Number of samples classified in test set over rejection level

To have a statistically significant train and test set an $N_{rejection} = 60\%$ is chosen. This ensures at least 50% of samples are being classified. The performance of the Naive Bayes rejection model variant on train and test set is shown on Table 4-3 and Table 4-4. The accuracy on train set is 54,74% and in the test set, 51,30%.

Class	Precision	Recall	Percentage of Samples Classified
0	54,58%	28,71%	52,6%
1	54,92%	25,35%	46,17%
Average/Total	54,75%	27,03%	49,36%

Table 4-3 Metrics of the rejection model variant in train set with $N_{rejection} = 60\%$

It can be seen, by the examination of the results, that the rejection model improved the original one. The major disadvantage of this methodology is the rejection of samples. When building a model to do transactions with the financial market, rejecting samples where there is higher value of uncertainty is of interest to the investor, since each transaction bares transactional fees. Using the Market Simulator, the cost of these fees will be examined.

Class	Precision	Recall	Percentage of Samples Classified
0	52,75%	25,48%	48,31%
1	50,03%	27,90%	55,77%
Average/Total	51,39%	26,69%	52,03%

Table 4-4 Metrics of the rejection model variant in test set with $N_{rejection} = 60\%$

4.3.4 Market Simulator

The market simulation will be performed with the rejection model, since this one has better accuracy in relation to the non-rejection one (the transactional fees would prohibit a model which was always making trades) . In Figure 4-8 the performance of Market Simulation in train set can be seen; the predictions used to make this plot were generated through *k*-fold CV to prevent train overfit. It is possible to perceive that without transactional costs and spread, the model would have a very positive result yielding approximately 28,96% returns over the course of 3 years. Considering transactional and spread costs, the performance of the model lowers to approximately 12,19%. The performance of Long and Short contracts is also individually plotted. The “Long and Short” line accounts for the overall performance of the system. The max drawdown period is between the two red dots (September 2014 and April 2015), and its value is -15,58%.

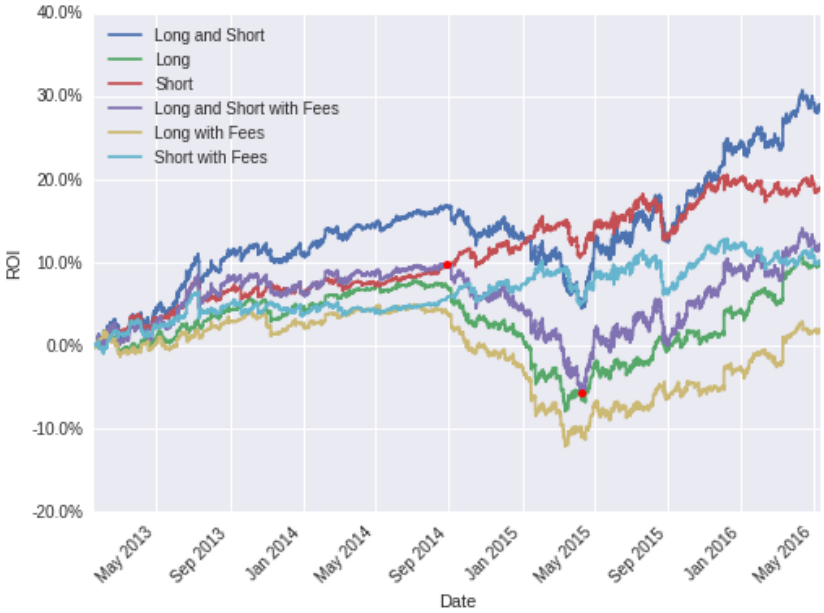


Figure 4-8 Rejection model, with no optimization, market simulation in train set with predictions generated through CV

Table 4-5 summarizes the performance metrics for the market simulation in train set regarding the un-

optimized model. The percentage of correct trades is inferior to the one showed on Table 4-3 due to accounting for the transactional and spread costs.

Position	Trades Executed	Percentage of correct Trades (positive ROI)	Total ROI (ROI with fees)
Long	4186	54,9%	1,96%
Short	5414	54.6%	10,23%
Average/Total	9600	54,75%	12,19%

Table 4-5 Performance metrics of the market simulation using the rejection model variant in train set

The performance of the model in the test set is presented on Figure 4-9. Since the test set was never seen in the training process, it is a good indicator that the problem is well formulated and result estimations work as expected. Without transactional and spread costs, this model would yield approximately 0,43% through the course of 11 months. Considering transactional and spread cost the ROI is -4,27%. The max drawdown is between the two red dots in the plot (June 2016 and February 2017), and its value -7,50%.

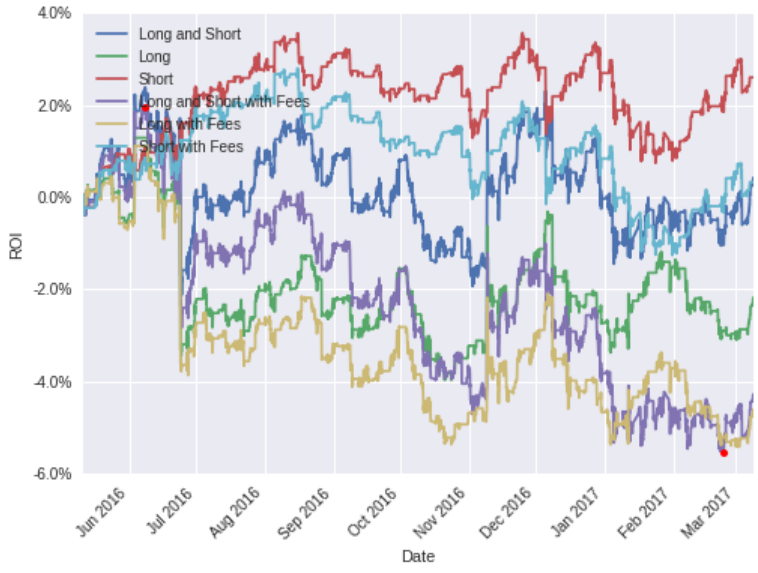


Figure 4-9 Rejection model market simulation in test set

Table 4-6 summarizes the model performance market simulation in the test set.

Position	Trades Executed	Percentage of correct Trades (positive ROI)	Total ROI (ROI with fees)
Long	727	50,0%	-4,60%
Short	662	52,7%	0,33%
Average/Total	1389	51,35%	-4,27%

Table 4-6 Performance metrics of the market simulation using the rejection model variant in test set

4.4 Case Study B – Proposed optimization on the Binary Classifier

In this Chapter the architecture developed for this thesis will optimize the simple Naive Bays model, by searching the optimal combination of input features. The heuristic method, which will perform the search, will be the GA presented throughout Chapter 3.6.

4.4.1 Parameters and Convergence Analysis

Since the GA performs the search based on random sampled individuals, it is necessary to evaluate its convergence through multiple runs. To validate the parameters chosen for the GA, the convergence of multiple runs must be inside an acceptable confidence interval.

The parameters chosen for the GA were:

- 1000 individuals
- 100 generations
- 5% replacement rate (Random Immigrants)
- 50% Probability of Crossover
- Tournament Selection with 3 individuals
- 20% Probability of mutation (40% when Hyper-Mutation triggers)
- When maximum fitness does not improve over 3 subsequent generation, the Hyper-Mutation is triggered

The convergence analysis of the GA was made using 10 different runs of the algorithm. On Figure 4-10 The Mean Maximum, Mean Average, and Mean Minimum GA fitness across all the runs is shown (the shaded areas represent the 95% confidence intervals where the values of those 10 runs lie). Since the GA is maximizing the accuracy of the CV estimation, Figure 4-11 shows only the Best, Mean, and Worst

Maximum GA Fitness across the same 10 runs. These figures prove that the chosen parameters converge in an acceptable manner across all runs.

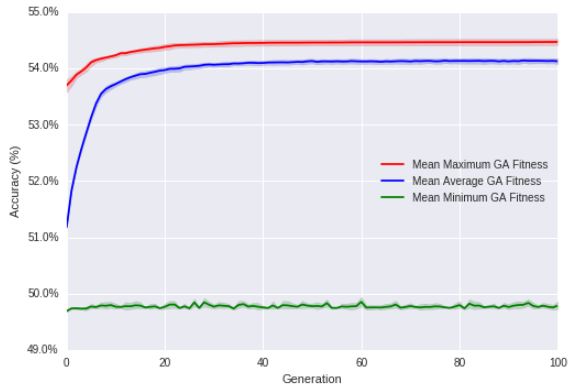


Figure 4-10 Mean fitness values across 10 different GA runs

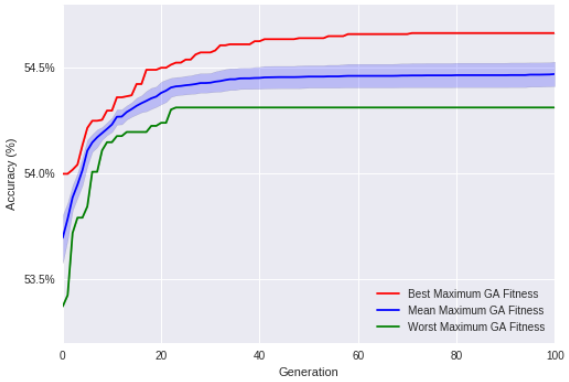


Figure 4-11 Maximum GA fitness across 10 different GA runs

On Figure 4-12 and Figure 4-13, instead of having shaded areas to represent the 95% confidence interval, every different run of the GA is plotted, having their mean value on a solid line.

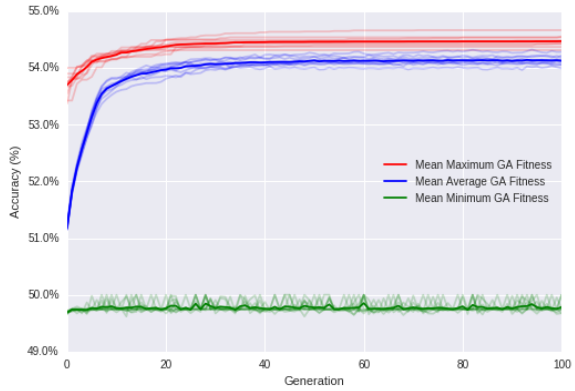


Figure 4-12 Every fitness value across 10 different GA runs

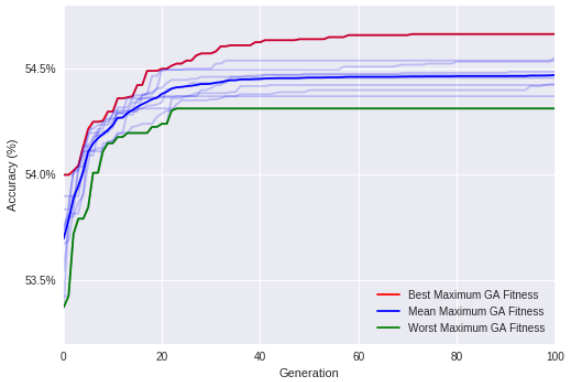


Figure 4-13 Every maximum GA fitness value across 10 different GA runs

4.4.2 CV, Train, and Test Results

For this problem, the GA is being used to maximize the estimated CV accuracy. This way, the most optimized individual, from all the runs, will be picked to make the posterior model study (given more computational power, it is expected that better individuals would result from the optimizing runs).

The best CV estimated accuracy achieved by a GA was $54{,}7\,\%\,(+/-\,2{,}2\%)$. The detailed metrics of the optimized binary classifier in train set are present in Table 4-7.

Class	Precision	Recall	Number of Occurrences
0	54,46%	53,14%	10292
1	54,86%	56,17%	10434
Average/Total	54,66%	54,67%	20726

Table 4-7 Metrics of the GA optimized binary classifier in train set

The test set accuracy was 52%. The detailed test set performance is in Table 4-8.

Class	Precision	Recall	Number of Occurrences
0	52,31%	48,85%	2598
1	51,78%	55,22%	2584
Average/Total	52,04%	52,03%	5182

Table 4-8 Metrics of the GA optimized binary classifier in test set

It's possible to conclude that, in comparison with the un-optimized binary classifier, the GA optimized one improves the test set accuracy from 50,97% to 52,04%.

4.4.3 Rejection Optimized Naive Bayes

In Figure 4-14 and Figure 4-15, the accuracy is plotted against an increasing $N_{rejection}$ level. It is possible to conclude that the classifier is better calibrated to the one shown in Chapter 4.3.3.

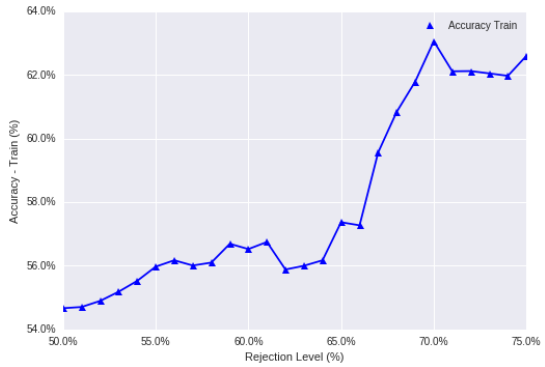


Figure 4-14 Train set accuracy improvement with $N_{rejection}$ modification on GA optimized rejection Naive Bayes

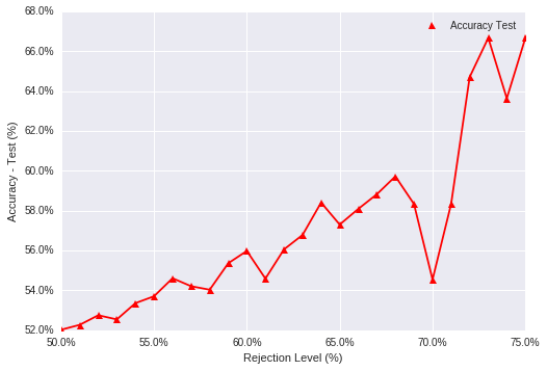


Figure 4-15 Test set accuracy improvement with $N_{rejection}$ modification on GA optimized rejection Naive Bayes

To make an equivalent comparison between the un-optimized rejection Naive Bayes and the GA optimized one, it is necessary to pick a $N_{rejection}$ which classifies $\approx 50\%$ of all samples present in data.

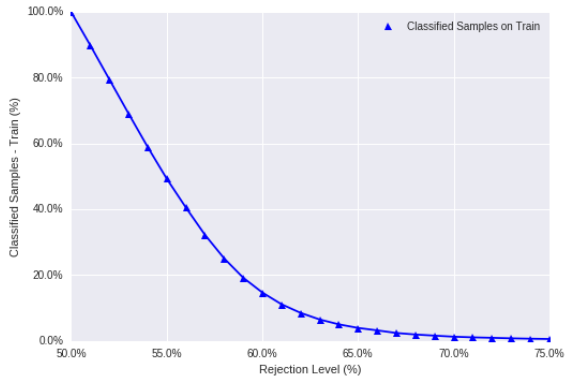


Figure 4-16 Number of samples classified in train set over rejection level for the optimized Naive Bayes

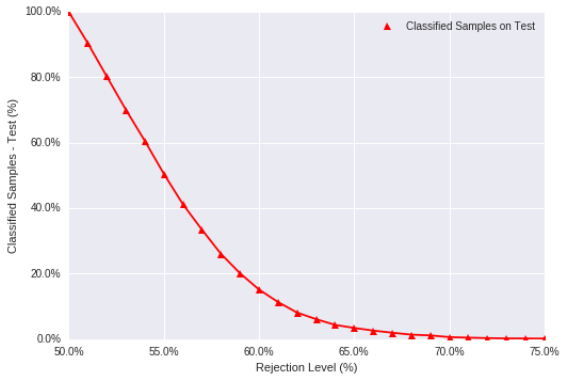


Figure 4-17 Number of samples classified in test set over rejection level for the optimized Naive Bayes

Though the analysis of the Figures it is possible to conclude that the $N_{rejection} = 55\%$ results in a model which rejects 50% of the samples.

On Table 4-9 the detailed metrics of the optimized Rejection model in train set are shown.

Class	Precision	Recall	Percentage of Samples Classified
0	55,77%	26,15%	46,88%
1	56,15%	29,05%	51,73%
Average/Total	55,96%	27,60%	49,32%

Table 4-9 Metrics of the optimized rejection model variant in train set with $N_{rejection} = 55\%$

On Table 4-10 the detailed metrics in the test set are shown. It's possible to conclude that the optimized rejection model substantial improves in comparison with the one presented on Chapter 4.3.3, improving the overall accuracy from 51,39% to 53,95% in test set.

Class	Precision	Recall	Percentage of Samples Classified
0	55,81%	24,79%	44,42%
1	52,09%	29,49%	56,66%
Average/Total	53,95%	27,14%	50,52%

Table 4-10 Metrics of the optimized rejection model variant in test set with $N_{rejection} = 55\%$

4.4.4 Market Simulator

4.4.4.A Best Maximum Fitness Individual from all optimization runs

In this Chapter, the best individual optimized through the GA with Naive Bayes Rejection algorithm will be tested in the Market Simulator module. The $N_{rejection}$ chosen is 55% since with this value the model classifies 50% of all classifiable samples, as in Chapter 4.3.4, thus making a fair comparison between different approaches. Both models have the same number of opportunities to make a profit in market (50% of the available trades). In Figure 4-18 the market simulation performance of the optimized rejection model in train set is shown. Without spread or transactional costs this model would yield 44,76% returns (the un-optimized model had 28,96% on the same conditions). With spread and transactional costs the overall return of Longs and Shorts is 21,65%. The max drawdown period is shown between the two red dots, and its value is -8,73%.

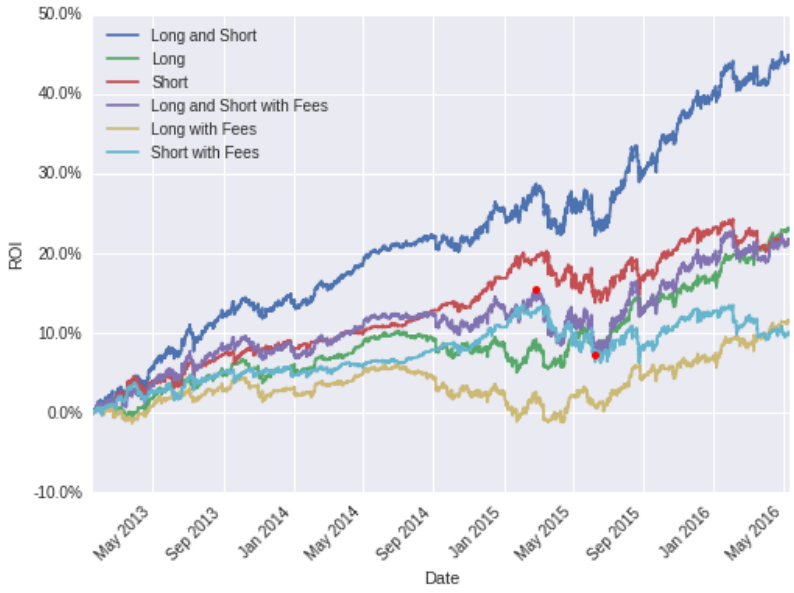


Figure 4-18 Best individual rejection model, with GA optimization, market simulation in train set with predictions generated through CV

In Table 4-11 the market simulation metrics of the optimized rejection model are described in detail.

Position	**Trades Executed**	**Percentage of correct Trades** (positive ROI)	**Total ROI** (ROI with fees)
Long	5398	56,2%	11,51%
Short	4825	55,8%	10,14%
Average/Total	10223	56,0%	21,65%

Table 4-11 Performance metrics of the market simulation using the best individual optimized rejection model variant in train set

The plot in Figure 4-19, shows the performance of the optimized model in the test set. The ROI, with spread and transactional costs is 4,24%, which is a significant improvement when comparing to the ROI achieved by the un optimized model, -4,27%. The max drawdown, represented between the two red dots is -4,48%.

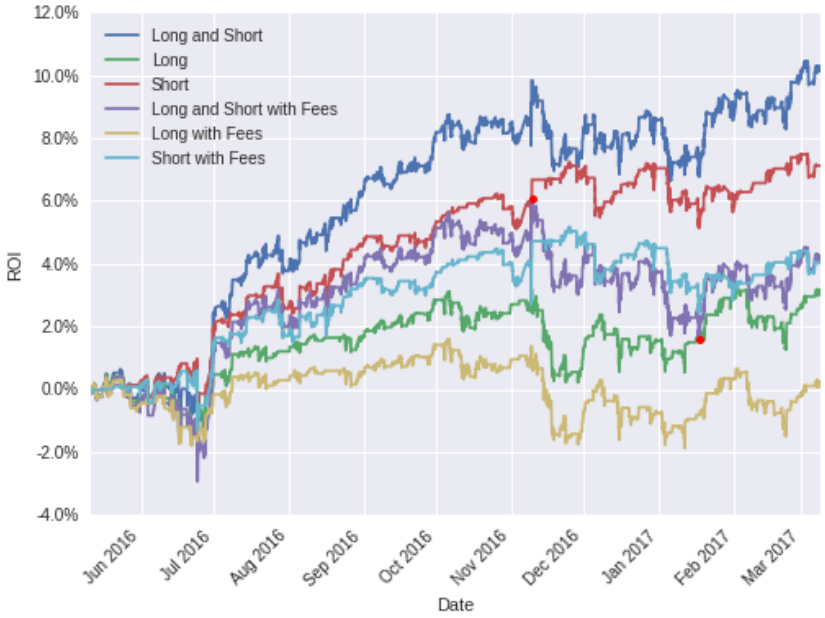


Figure 4-19 Best individual rejection model, with GA optimization, market simulation in test set

In Table 4-12, a summary of the detailed metric performance in the test set using the Market Simulation module, is described.

Position	**Trades Executed**	**Percentage of correct Trades** (positive ROI)	**Total ROI** (ROI with fees)
Long	1464	52,0%	0,22%
Short	1154	55,8%	4,02%
Average/Total	2618	53,9%	4,24%

Table 4-12 Performance metrics of the market simulator module using the best individual optimized rejection model variant in test set

4.4.4.B Worst Maximum Fitness Individual from all optimization runs

Since the GA was performed 10 times, through the inspection of Figure 4-11, the worst algorithm run achieved a CV accuracy of 54,3%. If the same analysis is performed on this individual such as the one on Chapter 4.4.3, the $N_{rejection} = 54{,}5\%$ classifies 50% of the samples. This allows to make a fair comparison with both models already reviewed. In Figure 4-20, the Market Simulation of the worst maximum individual across all the GA runs is presented. It is possible to conclude that the performance of the individual is worse than the one presented on Chapter 4.4.4.A (the best individual across all runs) but better than the unoptimized version of the Naive Bayes (as expected).

Figure 4-20 Worst individual rejection model, with GA optimization, market simulation in train set with predictions generated through CV

In Table 4-13 the detailed market performance metric of this individual is presented.

Position	**Trades Executed**	**Percentage of correct Trades** (positive ROI)	**Total ROI** (ROI with fees)
Long	5850	55,6%	1,74%
Short	4385	55,7%	10,72%
Average/Total	10235	55,65%	12,46%

Table 4-13 Performance metrics of the market simulation using the worst individual optimized rejection model variant in train set

In Figure 4-21 and Table 4-14 the market performance metrics of the worst maximum fitness individual across all GA runs is presented in test set.

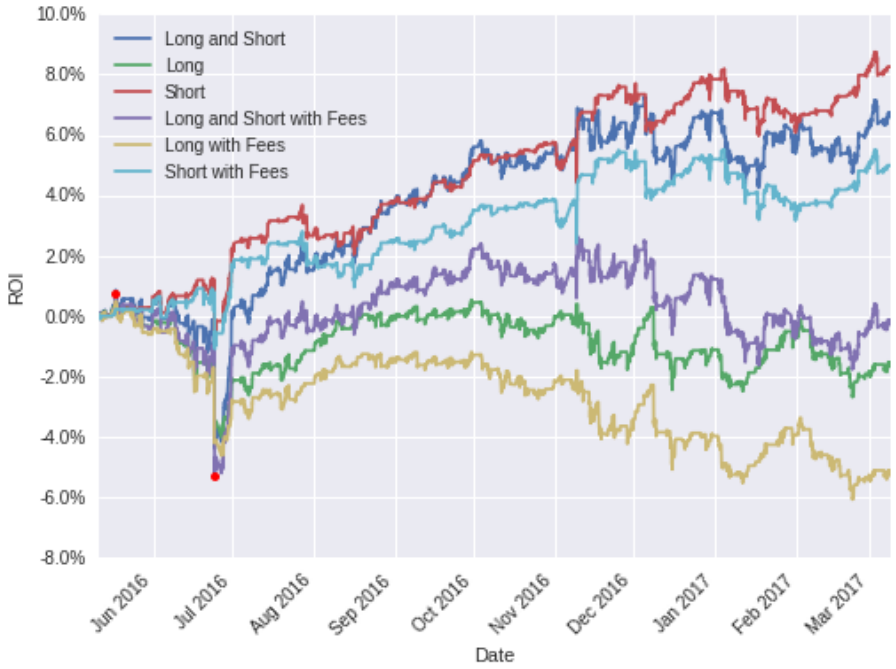


Figure 4-21 Worst individual rejection model, with GA optimization, market simulation in test set

Position	**Trades Executed**	**Percentage of correct Trades** (positive ROI)	**Total ROI** (ROI with fees)
Long	1491	52,4%	-5,14%
Short	1114	56,9%	4,99%
Average/Total	2605	54,65%	-0,15%

Table 4-14 Performance metrics of the market simulation using the worst individual optimized rejection model variant in test set

4.4.4.C Average of Maximum Individuals from all optimization runs

The GA had 10 different runs, each one with its respective maximum fitness value, corresponding to a chromosome unique representation. In Chapter 4.4.4.A and Chapter 4.4.4.B both the Best and Worst maximum individuals across all GA runs were evaluated in terms of market performance. For this topic, the following experiment will be presented:

- For each maximum fitness solution across all 10 GA runs, the $N_{rejection}$ which classifies half the dataset samples is individually chosen for each unique chromosome
- Each individual chromosome is evaluated with Market Simulator module
- An average of all maximum fitness individuals market performance is calculated

In Figure 4-22 and Figure 4-23 the market performance of all maximum fitness individuals across all 10 runs of the GA is presented on both train and test set, respectively.

Figure 4-22 Average of maximum individuals rejection models, with GA optimization, Market Simulation in train set with predictions generated through CV

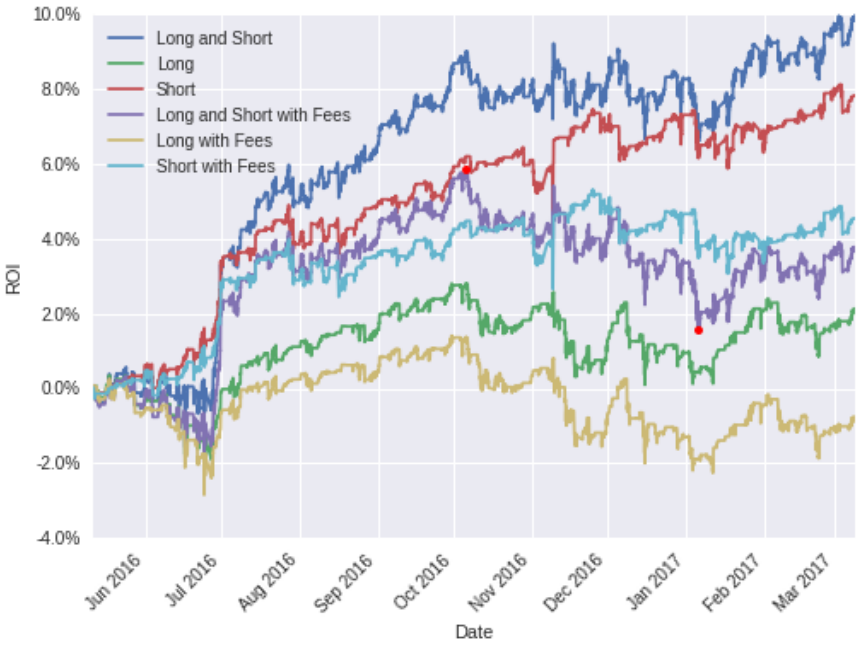


Figure 4-23 Average of maximum individuals rejection model, with GA optimization, market simulation in test set

4.5 Case Study C – Classical GA compared to the proposed Dynamic GA

In this Chapter, the convergence of the proposed GA architecture is related against the standard Hollands [9] implementation. To compare with the previous runs of the proposed GA, 10 runs of the standard GA were executed and its performance metrics recorded. In Table 4-15 the initial parameters of each GA are summarized.

Standard GA	Proposed GA
1000 individuals	1000 individuals
100 generations	100 generations
50% Probability of Crossover	50% Probability of Crossover
Tournament Selection with 3 individuals	Tournament Selection with 3 individuals
20% Probability of Mutation	20% Probability of Mutation - (40% when Hyper-Mutation Triggers)
	5% replacement rate (Random Immigrants)
	When maximum fitness does not improve over 3 subsequent generation, the Hyper-Mutation is triggered
	Elitism of one individual

Table 4-15 Standard GA and proposed dynamic GA starting parameters to compare convergence over multiple runs

To have a better representation of both algorithms two convergence plots are presented. In Figure 4-24 the solid lines are the mean values of Maximum, Average and Minimum fitness of both algorithms, while the shadowed areas are the 95% confidence interval of all the algorithm runs. The biggest difference is the minimum Fitness of each approach. The proposed Dynamic GA minimum fitness is always near 50% due to the replacement rate in each generation. Since the standard approach only makes combinations and mutations of the initial population, its minimum fitness value converges upwards.

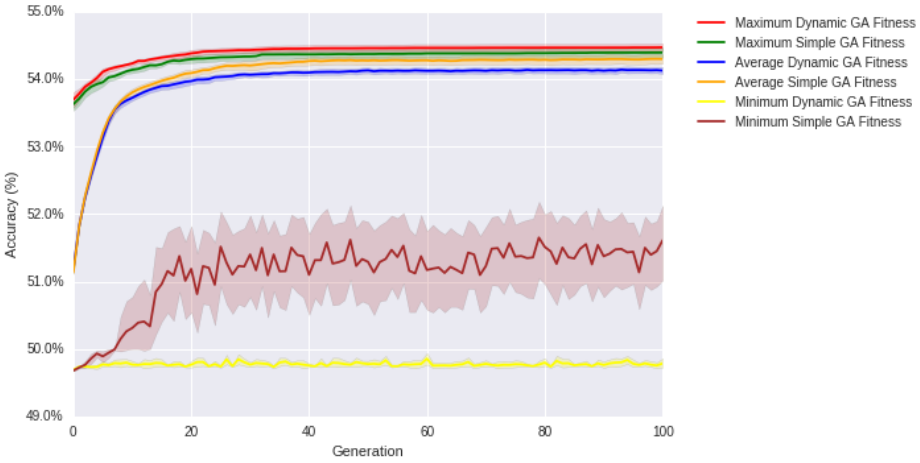


Figure 4-24 Fitness metrics with 95% confidence interval of 10 different runs for standard GA and proposed Dynamic GA

Since the scale of Figure 4-24 does not detail the maximum fitness that each approach can achieve, Figure 4-25 is also presented.

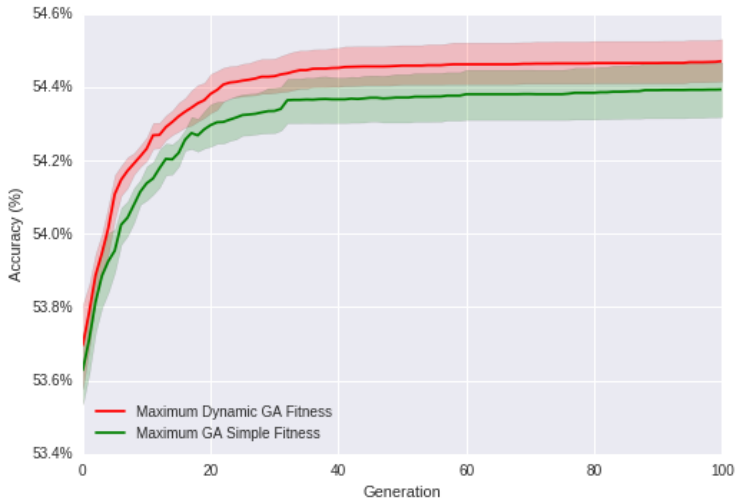


Figure 4-25 Average maximum fitness with 95% confidence interval of 10 different runs for standard GA and proposed Dynamic GA

It is possible to conclude that the proposed GA implementation converges faster, while achieving a better maximum fitness value, when comparing to the original GA implementation.

4.6 Case Study D – Random Signal Analysis

If the assumption that the market is a truly random and unpredictable event, is considered a true statement, the following question arises: “How often a ML classifier miss correctly identifies a random signal as a predictable one?”

To answer this question, the Random Signal Generator proposed on Chapter 3.8 was created. This Generator is used to create random signals, which are then evaluated by the unoptimized Naive Bayes classifier. Each iteration follows this sequence:

- Generate a random signal with 20726 samples (same number of samples the original EUR/USD dataset has)
- Estimate the accuracy using Naive Bayes with *k*-fold CV scheme
- Save the mean estimated accuracy across all fold and its estimated 95% confidence interval

For this experiment 100k random signals were generated and the results are shown in Figure 4-26. Each point represents the estimated accuracy (*x* axis) on a randomly generated signal with its 95% confidence interval (*y* axis).

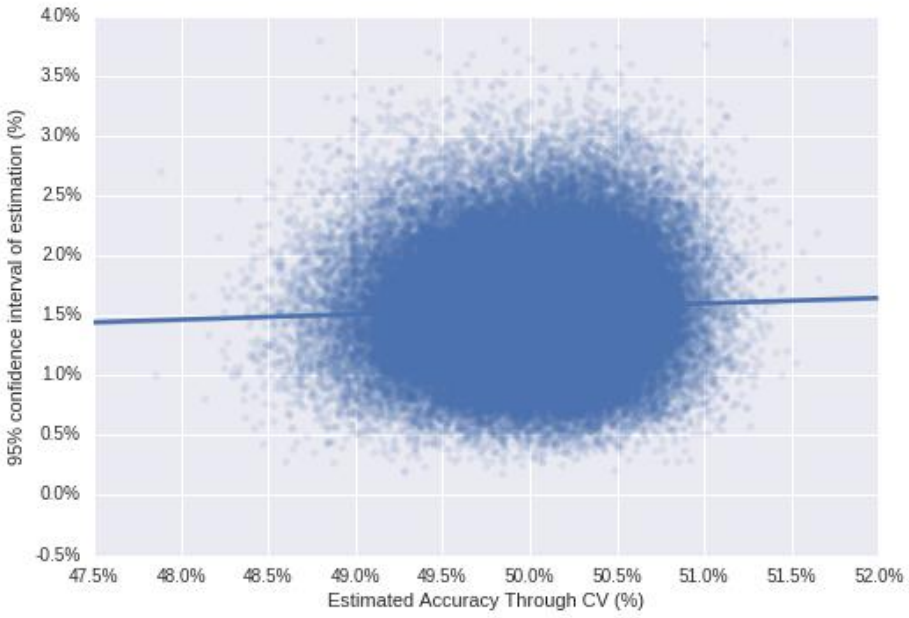


Figure 4-26 Estimated accuracy on randomly generated signals with the 95% confidence interval of the estimation. The solid line represents a linear regression of the data

From the analysis of the Figure 4-26 it is possible to conclude, that a signal known a priori to be random is rarely classified like a non-random signal. Even in the rare event in which a random signal is classified like non-random one, its estimated 95% confidence interval rises the suspicion about the validity of the result. It's also possible to conclude that the 95% confidence interval rises linearly with the estimated accuracy. Nonetheless the maximum estimated accuracy across all random signals is 51,7% which is far from 53,4% estimated on Chapter 4.3.1.

Given enough experiments, the generator proposed on Chapter 3.8 would generate an identical signal to the one like the EUR/USD time series, and the Naive Bayes model would wrongly identify it like a non-random one. As it is proven above, this is an unlikely event, thus, the probability that the EUR/USD time series is a random signal is statistically improbable.

4.7 Case Study E – Model Visualization

One relevant question to answer while developing an ML practical implementation is: "what sort of relationships and patterns are encoded within the model". Recently there have been some advances on model visualization with the development of tools like *t*-SNE (t-Distributed Stochastic Neighbour Embedding) [21].

t-SNE is a visualization technique for laying out a large and high dimensional datasets in 2-d maps while preserving the local structure present in the data. A very crude explanation about the algorithm is the following: it minimizes an objective function, using gradient descent, which measures the discrepancy between similarities in high dimensional data, with the similarities project onto a lower dimensional map (hence the name embedding). To summarize, *t*-SNE is used to build a 2-d map which represents a higher dimensional map local similarities while minimizing the Kullback–Leibler divergence between the two. *t*-SNE favours the local similarity between the high dimensional points, preserving it, i.e., distance between non-similar points on the 2-d map may have a different interpretation.

One recent paper by authors working at Goggle DeepMind, "Human-level control through deep reinforcement learning" [20] uses t-SNE to show state values of what the maximum expected reward will be, according to the action performed.

Figure 4-27 is an attempt to visualize what sort of patterns, the model developed for this thesis is searching for. To build this plot the following steps were taken:

- Pick the best individual from the optimization process described in Chapter 3.6
- Construct a matrix of observations X from the time series defined in Chapter 3.4
- Perform the t-SNE algorithm in an unsupervised way to the matrix X of observations, reducing each sample to two dimensions
- Predict the sample (each observation) probability of being Label 1 (market having a positive variation on the next hour) using Naive Bayes binary classifier with the *k*-fold CV scheme
- Plot each market observation onto two dimensions with a colour gradient symbolizing the predicted probability by Naive Bayes classifier
- Plot the last 100-hour window of similar samples with high confidence that they are a certain Label

Each point in Figure 4-27 is a market observation, where in each cluster, points that are near each other (preserving local structure) are expected to be similar (remember that this is a projection onto a 2-D plane). In some regions of this plot, similar points have equal probability of being Label 1, or Label 0. For example, on the bottom left price charts, each plot represents the beginning of year 2015 and 2016; Naive Bayes identifies this pattern with a higher probability of having a positive variation (even though the model has no time awareness, just pattern awareness). There are some price signals which are similar and have more probability of being of a certain class, and the most certain ones are market

inefficiencies. For instance, both price charts on the bottom right have a lower probability of having a positive market variation, thus having a high probability of a negative market variation, furthermore it is interesting to note that they both show a similar behaviour, an abnormal increase on the instruments price in a short period of time. It is an anomalous market behaviour to have such a price variation in a short amount of time, and the Naive Bayes model predicts that there is a higher probability of a negative variation. If the market was a truly chaotic and unpredictable event there shouldn't be plot areas where the probability of having a positive variation, or not, is much higher, or much lower, than 50%. It appears that the developed model is capturing some memory present in the signal variation represented in time series patterns.

Since Figure 4-27 is built using an unsupervised technique, without knowing what the target is, it would be interesting to do a thorough study on what the local clusters might mean, thus building new and more representative features. Plotting the colour gradient shows there is an important correlation between what the model is capturing and the local structure present in the dataset.

Another interesting thing to note, is that the structure present in Figure 4-27 is enhanced because of the optimization process proposed in Chapter 3.6 (which results in the individual which maximizes the accuracy of the system). In a way, the proposed architecture in conjunction with the Naive Bayes is providing information gain by optimizing the best filtered view of the time series signal.

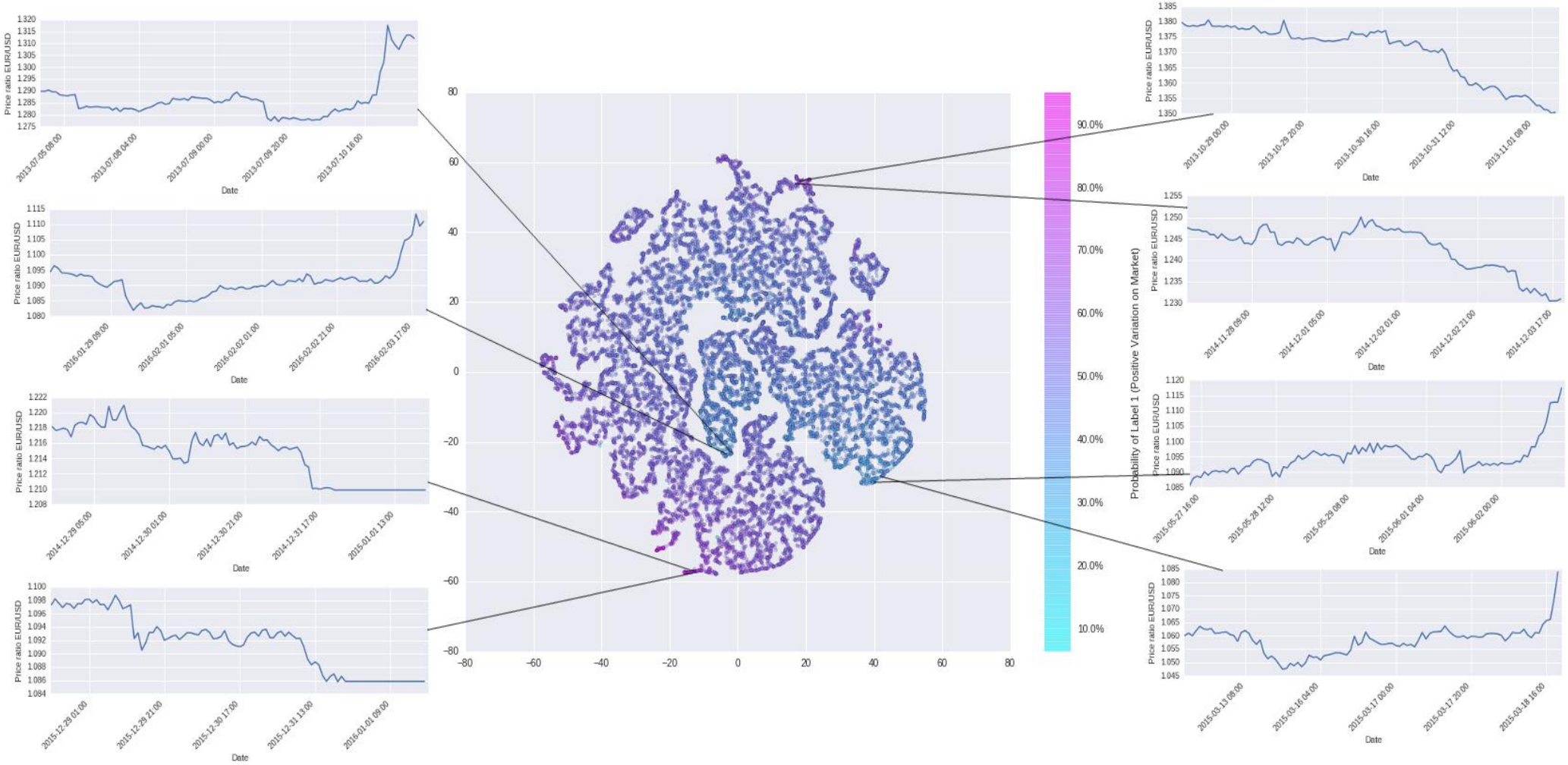


Figure 4-27 t-SNE embedding where each sample (point in space) represents a market observation. The colour gradient is the probability, of the next time period having a positive market variation, calculated by the Naive Bayes binary classifier using a *k*-fold CV scheme

Chapter 5

Conclusions

The main conclusion for this thesis is that Evolutionary Computation mixed with probabilistic classifiers (Naive Bayes) provide a simple and efficient infrastructure to search and optimize features. The Foreign Exchange Market, specifically the EUR/USD, was the dataset in which the proposed architecture was tested. Although the initial objective for this thesis was to prove that there is a benefit of mixing different areas of ML, to achieve results in an efficient manner, the tests developed to test this, provide interesting conclusions on the Foreign Exchange Market.

Technical Indicators, due to their unknown and non-formalized origin, are viewed in general, as a non-scientific way to look at the markets (although they are considered mainstream in financial circuits). Through analysing their mathematical formulation, it is possible to conclude, that momentum indicators are a way to represent the variation of a time series signal, using mostly discrete moving averages. In Control theory, moving averages are a standard way to filter a signal, so it is not un-natural, that financial markets developed their analogy to those concepts, even if those are not formalized. In Chapter 4.3, using market standard Technical Indicators, results in a classifier which is better than random guessing, proving that there is some ground truth in TA. With the use of the Random Signal Generator presented in Chapter 3.8, the results from Chapter 4.6 (analysis of hundred thousand random signals modelled after the *pdf*s of the real market), further validate that the EUR/USD signal is not at times efficient, thus making it somewhat predictable.

Although the Simple Binary Rejection Classifier has already achieved better performance than random guessing, the proposed architecture, which searched and optimized features using a modified GA, has proved that it is possible to boost the Naive Bayes classifier accuracy by a considerable amount (from 51,39%, on test set, to 53,95%). Testing both approaches, with the Market Simulator presented in Chapter 4.3.4, it is possible to conclude, that a profitable trading model is achievable by heuristically searching feature combinations.

In Chapter 4.5, the proposed modifications to the GA are compared to the original algorithm formulation, proving faster convergence while achieving better maximum fitness values consistently.

Model visualization is a hot topic on ML nowadays hence Chapter 4.7. By making use of *t*-SNE algorithm it is possible to visualize what type of patterns the Naive Bayes is learning from the input data. This visualization opens the path to Future Work since it has revealed a lot of local clusters which were not present in the Standard Technical indicator features, only in the GA optimized ones. It is possible to conclude that the points where the algorithm is most certain of the price direction are based on inversions after a market inefficiency, such as an abnormal price variation in a short span of time. It would be interesting to make an analysis of these events while referring to EMH.

5.1 Future Work

This thesis is the starting point for many different initial problem formulations. One unexplored area is the feature extraction through unsupervised learning such as clustering, and data visualization

techniques like the one presented in Chapter 4.7 using *t*-SNE. Clustering could be used to extract information from the local clusters that *t*-SNE revealed.

In this thesis, the time series was binarized to transform the problem into a classification, whereas, it could have been predicted as a regression.

The optimization method chosen was a GA, another promising state of the art option is Bayesian optimization through Gaussian process approximation (optimizes bounded variables in a black box function).

Deep learning provides a versatile infrastructure to capture patterns in time series; 1-D Convolutional layers can be used instead of TA due to their capability of generating high level data representations; Long short-term Memory neural network cells have great capacity to deal with sequences, such as, time series.

To summarize:

- Different optimization method like Bayesian Optimization through Gaussian Process approximation, instead of GA
- Long short-term Memory or 1-D Convolutional Layers to capture temporal dependencies
- Unsupervised learning for feature extraction
- Different supervised methods, such as, Logistic Regression, Decision Tress with boosting and/or bootstrap, Support Vector Machines
- Develop new cost functions or use different ones
- Formulate the problem has a simple regression, or has an ordinal one
- Ensemble Models from different methodologies
- Online Learners

References

[1] Malkiel, Burton G. "The efficient market hypothesis and its critics." *The Journal of Economic Perspectives* 17.1 (2003): 59-82.

[2] MacKay, David JC. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[3] Pinto, J. M., Neves, R. F., & Horta, N. (2015). Boosting Trading Strategies performance using VIX indicator together with a dual-objective Evolutionary Computation optimizer. Expert Systems with Applications, 42(19), 6699-6716.

[4] Colby, Robert W., and Thomas A. Meyers. *The encyclopedia of technical market indicators*. New York: Irwin, 1988.

[5] Kirkpatrick II, Charles D., and Julie A. Dahlquist. *Technical analysis: the complete resource for financial market technicians*. FT press, 2010.

[6] Achelis, Steven B. Technical Analysis from A to Z. New York: McGraw Hill, 2001.

[7] Lin, Xiaowei, Zehong Yang, and Yixu Song. "Intelligent stock trading system based on improved technical analysis and Echo State Network." Expert systems with Applications 38.9 (2011): 11347-11354.

[8] Bäck, Thomas, David B. Fogel, and Zbigniew Michalewicz, eds. Evolutionary computation 1: basic algorithms and operators. Vol. 1. CRC Press, 2000.

[9] Booker, Lashon B., David E. Goldberg, and John H. Holland. "Classifier systems and genetic algorithms." Artificial intelligence 40.1 (1989): 235-282.

[10] Lewis, David D. "Naive (Bayes) at forty: The independence assumption in information retrieval." European conference on machine learning. Springer Berlin Heidelberg, 1998.

[11] Zhang, Harry. "The optimality of naive Bayes." *AA* 1.2 (2004): 3.

[12] Sokolova, Marina, Nathalie Japkowicz, and Stan Szpakowicz. "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation." *Australasian Joint Conference on Artificial Intelligence*. Springer Berlin Heidelberg, 2006.

[13] Powers, David Martin. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." (2011).

[14] Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection." *Ijcai*. Vol. 14. No. 2. 1995.

[15] Chan, Keith CC, and Foo Kean Teong. "Enhancing technical analysis in the Forex market using neural networks." *Neural Networks, 1995. Proceedings., IEEE International Conference on*. Vol. 2. IEEE, 1995.

[16] Potvin, Jean-Yves, Patrick Soriano, and Maxime Vallée. "Generating trading rules on the stock markets with genetic programming." *Computers & Operations Research* 31.7 (2004): 1033-1047.

[17] Yao, Jingtao, and Chew Lim Tan. "A case study on using neural networks to perform technical forecasting of forex." *Neurocomputing* 34.1 (2000): 79-98.

[18] Panda, Chakradhara, and V. Narasimhan. "Forecasting exchange rate better with artificial neural network." *Journal of Policy Modeling* 29.2 (2007): 227-236.

[19] Hassan, Md Rafiul, and Baikunth Nath. "Stock market forecasting using hidden Markov model: a new approach." *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*. IEEE, 2005.

[20] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529-533.

[21] Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." *Journal of Machine Learning Research* 9.Nov (2008): 2579-2605.

[22] Huang, Wei, Yoshiteru Nakamori, and Shou-Yang Wang. "Forecasting stock market movement direction with support vector machine." *Computers & Operations Research* 32.10 (2005): 2513-2522.

[23] Kim, Kyoung-jae. "Financial time series forecasting using support vector machines." *Neurocomputing* 55.1 (2003): 307-319.

[24] Bhanu, Bir, and Yingqiang Lin. "Genetic algorithm based feature selection for target detection in SAR images." *Image and Vision Computing* 21.7 (2003): 591-608.

[25] Oh, Il-Seok, Jin-Seon Lee, and Byung-Ro Moon. "Hybrid genetic algorithms for feature selection." IEEE Transactions on Pattern Analysis and Machine Intelligence 26.11 (2004): 1424-1437.

[26] Canelas, António, Rui Neves, and Nuno Horta. "A SAX-GA approach to evolve investment strategies on financial markets based on pattern discovery techniques." *Expert Systems with Applications* 40.5 (2013): 1579-1590.

[27] Huang, Cheng-Lung, and Chieh-Jen Wang. "A GA-based feature selection and parameters optimization for support vector machines." *Expert Systems with applications* 31.2 (2006): 231-240.

[28] Gorgulho, António, Rui Neves, and Nuno Horta. "Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition." *Expert systems with Applications* 38.11 (2011): 14072-14085.

[29] Grefenstette, John J. "Genetic algorithms for changing environments." *PPSN*. Vol. 2. 1992.

[30] Cobb, Helen G., and John J. Grefenstette. *Genetic algorithms for tracking changing environments*. NAVAL RESEARCH LAB WASHINGTON DC, 1993.

[31] Oh, Kyong Joo, Tae Yoon Kim, and Sungky Min. "Using genetic algorithm to support portfolio optimization for index fund management." *Expert Systems with Applications* 28.2 (2005): 371-379.

[32] Kim, Hyun-jung, and Kyung-shik Shin. "A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets." *Applied Soft Computing* 7.2 (2007): 569-576.

[33] Shin, Kyung-Shik, and Yong-Joo Lee. "A genetic algorithm application in bankruptcy prediction modeling." Expert Systems with Applications 23.3 (2002): 321-328.

[34] Siedlecki, Wojciech, and Jack Sklansky. "A note on genetic algorithms for large-scale feature selection." *Pattern recognition letters* 10.5 (1989): 335-347.

[35] Huang, Jin, Jingjing Lu, and Charles X. Ling. "Comparing naive Bayes, decision trees, and SVM with AUC and accuracy." Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. IEEE, 2003.

[36] Smith, Matthew G., and Larry Bull. "Genetic programming with a genetic algorithm for feature construction and selection." *Genetic Programming and Evolvable Machines* 6.3 (2005): 265-281.

[37] Fortin, Félix-Antoine, et al. "DEAP: Evolutionary algorithms made easy." *Journal of Machine Learning Research* 13.Jul (2012): 2171-2175.

[38] Hunter, John D. "Matplotlib: A 2D graphics environment." Computing In Science & Engineering 9.3 (2007): 90-95.

[39] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of Machine Learning Research 12.Oct (2011): 2825-2830.

[40] Magdon-Ismail, Malik, and Amir F. Atiya. "Maximum drawdown." (2004).

[41] Andrieu, Christophe, et al. "An introduction to MCMC for machine learning." *Machine learning* 50.1-2 (2003): 5-43.

[42] Salvatier, John, Thomas V. Wiecki, and Christopher Fonnesbeck. "Probabilistic programming in Python using PyMC3." *PeerJ Computer Science* 2 (2016): e55.

[43] Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.