Feature Selection and Optimization on Naive Bayes modeling using Genetic Algorithm

Gonçalo Faria Abreu
Instituto Superior Técnico / IT
University of Lisbon
Lisbon, Portugal
goncalo.faria.abreu@tecnico.ulisboa.pt

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 Optimization, Foreign Exchange Market, t-Distributed Stochastic Neighbour Embedding.

I. INTRODUCTION

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 analyze 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, a novel architecture to optimize is provided.

The aspect of system validation will be addressed using multiple ML Concepts, such as, Cross Validation (CV).

The main concepts used for this work, to build the proposed architecture, are Naive Bayes and Genetic Algorithms (GA). TA will be used to generate the input features for the Model.

The main contributions of this work are:

- Problem formulation through the binarization of market time series
- A framework for estimating the performance of a binary classifier given the proposed problem formulation
- An architecture for feature searching and selecting using a modified GA
- Model visualization using the novel t-Distributed Stochastic Neighbour embedding (t-SNE).

A. Motivations

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 work 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.
B. 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, which change over time at practically the speed of light, it is an interesting engineering problem to analyse the signal which results from these interactions. In this work, 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 in which euros and dollars will be exchanged at the current market rate, i.e., if the current rate of EUR/USD is 1.18, it means that in this moment if an individual wants to use the market to change currencies from euros to dollars, they will get 1.18 dollars per 1 euro. In Figure 1, the EUR/USD time series used for this work is presented.

![EUR/USD time series](Figure 1)

The 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).

C. 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 work. 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, 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 work, the following Technical Indicators will be used:

- Relative Strength Index (RSI) [4]
- Volume [5]
- Commodity Channel Index (CCI) [4][5]
- Moving Average Convergence Divergence (MACD) [6]
- Rate of Change (ROC) [6]
- Stochastic Oscillator [7]
- Average True Range (ATR) [5]

![EUR/USD rate with MACD indicator](Figure 2)

In Figure 2 there is an example of the EUR/USD rate with the MACD indicator plotted at the bottom. The references of each TA above, contains the formulas used for their calculation respectively.

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.

II. STATE-OF-THE-ART

In this chapter, there will be a literature review about the current works done 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 work to be compared to.

A. 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.

Potvin, Soriano and Vallée [8] 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%. Panda and Narasimhan [9] used Neural Networks (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 term approach with several other methodologies they conclude that a GA is a strategy for common indexes such as the NASDAQ, S&P 500, FTSE 100, DAX 30 and the NIKKEI 225. Bhau and Lin [14] 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 [15] 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. Gorgulho, Neves and Horta [16] 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 [17] 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

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. Gorgulho, Neves and Horta [16] 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 [17] 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

<table>
<thead>
<tr>
<th>Reference Number</th>
<th>Publication Year</th>
<th>Number of Citations</th>
<th>Methodologies</th>
<th>Evaluation Metric</th>
<th>Type of Data</th>
<th>Dataset Period</th>
<th>Average Return (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9]</td>
<td>2006</td>
<td>90</td>
<td>NN</td>
<td>RMSE</td>
<td>Indian Rupee/USD</td>
<td>January 1994 to July 2003</td>
<td>-</td>
</tr>
<tr>
<td>[15]</td>
<td>2004</td>
<td>596</td>
<td>Hybrid GA</td>
<td>Pure Atomic Operations</td>
<td>10 datasets from UCI</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[18]</td>
<td>1993</td>
<td>441</td>
<td>Dynamic GA</td>
<td>Average Population Performance</td>
<td>Stationary Function (Toy dataset)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[19]</td>
<td>1989</td>
<td>936</td>
<td>GA, k-NN (k-Nearest Neighbours)</td>
<td>Error Rate</td>
<td>Digitized Infrared Imagery of Real Scenes</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>This Work</td>
<td>2017</td>
<td>-</td>
<td>GA, Naive Bayes</td>
<td>Accuracy, ROI</td>
<td>EUR/USD Exchange Data</td>
<td>01.01.2013 to 09.03.2017</td>
<td>4.24%</td>
</tr>
</tbody>
</table>

**TABLE I. STATE OF THE ART IMPORTANT PAPER SUMMARY**

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 [13] original formulation.

B. 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 [13] original formulation.
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 [18] compared three GA methods: 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 [13] work. These ideas will be used in the proposed system architecture in this work.

Siedlecki and Sklansky [19] 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.

C. 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 [20], 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 [21], 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 dependencies are distributed evenly between classes. It is also mathematically shown that if interdependence between features exist it might be cancelled out, and in this way not making an impact on the algorithm classification performance.

In Table I, a summary of the most important papers considered for this work are shown.

III. PROPOSED ARCHITECTURE

The objective of this work is to build a classifier to predict whether the EUR/USD time series signal variations are random or not, using TA as the input features.

A. General Perspective

The diagram representing a general perspective of the architecture is shown in Figure 3.

From the Foreign Exchange time series data, in this case the series representing the EUR/USD rate, a set of Technical Indicators can be calculated through their respective formulas (in Figure 2 there is an example of MACD). 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.

B. Naive Bayes Binary Classifier

The Naive Bayes model is a supervised learning method for classification [20]. 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 work, 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.

The Naive Bayes model is considered a good classifier and a bad estimator [21] 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. A reject variant was also implemented which rejects samples if the model is uncertain about which class they belong to.

C. Target Formulation

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 \( y_t \) (time series variation at period \( t \)). \( Y \) (vector which contains all the \( y \) variables across \( t \) calculated from the time series) 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 (1).

\[
y_t = \begin{cases} 
1, & \frac{Close_t - Close_{t-1}}{Close_{t-1}} \geq 0 \\
0, & \frac{Close_t - Close_{t-1}}{Close_{t-1}} < 0
\end{cases}
\] (1)

D. Optimized Naive Bayes through GA search

Each Technical indicator has one or many free parameters which affect the way they filter the time series, hence it is impossible to know a priori which parameters translate to the best performance accuracy of the classifier, 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 [13]. Both the implementation of the GA (with the resort of the DEAP
In Figure 4 a simplified view of the proposed architecture is shown. 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 of the system. For further confirmation, the system will be validated using a test set, unseen in the optimization process.

To comprehend what the GA is doing exactly it is necessary to present a more detailed diagram of that block. In Figure 5 a diagram which represents the functioning of the GA across generations is shown. 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.

E. Random Immigrants

This modification to the original GA implementation, called random Immigrants, was proposed by Grefenstette [17]. 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.

F. Hyper-Mutation

Hyper-Mutation was an alteration to the original GA code proposed in [18] 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 II and contains both Hyper-Mutation and Random Immigrants changes.

IV. SYSTEM EVALUATION

In this 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 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).

A. Train and Test Split

To have reliable performance estimation a split in the 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 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 till 09.03.2017 sampled at an hourly rate. The train set will be the first 80% of

Table II. Proposed GA Pseudo-Code

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>generation n=0</td>
<td>initialize population P(n) with individuals</td>
</tr>
<tr>
<td>evaluate P(n) fitness</td>
<td>while fitness criterion not met:</td>
</tr>
<tr>
<td>n=n+1</td>
<td>select to reproduce a subset S(n) of P(n-1)</td>
</tr>
<tr>
<td>if fitness is not higher than 3 generations ago:</td>
<td>cross-over and hyper-mutate S(n)</td>
</tr>
<tr>
<td>else:</td>
<td>cross-over and mutate S(n)</td>
</tr>
<tr>
<td>form S'(n)</td>
<td>evaluate fitness in S'(n)</td>
</tr>
<tr>
<td>generate random individuals R'(n)</td>
<td>replace P(n) from S'(n), P(n-1) and R'(n)</td>
</tr>
</tbody>
</table>

Figure 6. EUR/USD time series signal divided into train and test sets
the time series, leaving the test set with the final 20%. In Figure 6 there is a visual representation of this split. By making this split, the train 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.

B. Naive Bayes Classifier with rejection

The first step is to estimate the performance using k-fold CV scheme. The estimated accuracy for a 7-fold CV scheme is 53.4% (±/− 2.2%). To enhance the accuracy of the simple binary model rejecting was implemented, so that some 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. The rejection was chosen so that half of the samples are rejected, this way, the train and test set are still statistically significant. In Table III the Naive Bayes classifier metrics on train and test sets are shown.

<table>
<thead>
<tr>
<th>Label</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>0</td>
<td>54.58%</td>
<td>28.71%</td>
</tr>
<tr>
<td>1</td>
<td>54.92%</td>
<td>25.35%</td>
</tr>
<tr>
<td>Average/Total</td>
<td>54.75%</td>
<td>27.03%</td>
</tr>
</tbody>
</table>

The market simulation will be performed with the rejection model (the transactional fees would prohibit a model which was always making trades). In Figure 7 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%.

The performance of the model in the test set is presented on Figure 8. 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%.

C. Proposed optimization of the Naive Bayes Classifier with rejection

The architecture developed for this thesis will optimize the simple Naive Bays model, by searching the optimal combination of input features. 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); If maximum fitness does not improve over 3 subsequent generation, Hyper-Mutation is triggered.

The convergence analysis of the GA was made using 10 different runs of the algorithm. On Figure 9 the Mean Maximum, Mean Average, and Mean Minimum GA fitness across all the runs is shown (solid lines represent the mean of all runs, while the shaded, each individual run).

To make a fair comparison with the previous case study the rejection level chosen for the proposed optimized classifier also rejected half the samples. 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 IV.

**TABLE IV. TRAIN AND TEST RESULTS FOR THE OPTIMIZED NAIVE BAYES**

<table>
<thead>
<tr>
<th>Label</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>0</td>
<td>55.77%</td>
<td>26.15%</td>
</tr>
<tr>
<td>1</td>
<td>56.15%</td>
<td>29.05%</td>
</tr>
<tr>
<td>Average/Total</td>
<td>55.96%</td>
<td>27.60%</td>
</tr>
</tbody>
</table>

It is possible to conclude, through the analysis of Table III and Table IV that the proposed optimization significantly achieves better results.

The best individual optimized through the GA with Naive Bayes Rejection algorithm, without spread or transactional costs this model would yield 44.76% returns in Market Simulation (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%. The detailed results are show in Figure 10.

In Figure 11, the performance of the Market Simulation of the optimized model in the test set is presented. 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%.

**D. Model Visualization with T-SNE**

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) [12]

**Figure 10. Best individual rejection model, with GA optimization, market simulation in train set with predictions generated through CV**

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 favors 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” [11] uses t-SNE to show state values of what the maximum expected reward will be, according to the action performed.

**Figure 11. Best individual rejection model, with GA optimization, market simulation in test set**
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 higher 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 12 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 color gradient shows there is an important correlation between what the model is capturing and the local structure present in the dataset.

V. 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.

Although the Naive Bayes 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%).

Model visualization is a hot topic on ML. 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.

REFERENCES


Figure 12. t-SNE embedding where each sample (point in space) represents a market observation. The color 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.