Pontifícia Universidade Católica
do Rio de Janeiro

**Rodrigo Canto Corbelli**

# Investigating Optimal Regimes for Prediction in the Stock Market

**Dissertação de Mestrado**

Dissertation presented to the Programa de Pós-graduação em Engenharia Elétrica of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica.

| | |
|---|---|
| Advisor | : Prof. Marley Maria Bernardes Rebuzzi Vellasco |
| Co-advisor: | Prof. Álvaro de Lima Veiga Filho |

Rio de Janeiro
October 2019

**Rodrigo Canto Corbelli**

# Investigating Optimal Regimes for Prediction in the Stock Market

Dissertation presented to the Programa de Pós-graduação em Engenharia Elétrica of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica. Approved by the Examination Committee.

Rio de Janeiro, October 25th, 2019

**Rodrigo Canto Corbelli**

Graduated in Electronic Engineering by the Instituto Militar de Engenharia (IME).

Bibliographic data

Corbelli, Rodrigo Canto

Investigating Optimal Regimes for Prediction in the Stock Market / Rodrigo Canto Corbelli; advisor: Marley Maria Bernardes Rebuzzi Vellasco; co-advisor: Álvaro de Lima Veiga Filho. – 2019.

73 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2019.

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Mercado de Ações;. 3. Previsibilidade;. 4. Algorítimos Genéticos.. 5. PSO.. 6. Séries Temporais.. I. Vellasco, Marley Maria Bernardes Rebuzzi. II. Veiga Filho, Álvaro de Lima. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 621.3

# Acknowledgments

To my advisers Marley Vellasco and Álvaro Veiga for supporting, advising, and also understanding all the challenges during the research of this material. Without their kind guidance, this study would not have been possible.

To my father Mario, who has been always a pillar of strength and kindness in my life. To my mother, Marilea, who taught me to believe in myself and be kind to others. To my grandmother Yara, who showed me how selfless someone can be, and the impact this has in the happiness of people around that person.

To all my friends that cheered me during the process, specially those who read the first versions and contributed to the quality of the text. To all of them whom I talked about this, and all the insights and motivation they provided.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

## Abstract

Corbelli, Rodrigo Canto; Vellasco, Marley Maria Bernardes Rebuzzi (Advisor); Veiga Filho, Álvaro de Lima (Co-Advisor). **Investigating Optimal Regimes for Prediction in the Stock Market**. Rio de Janeiro, 2019. 73p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Predicting stock movements in the market its known to be an extremely difficult task. More than that, the predictability of the series itself is a controversial matter. The present study investigates if this difficulty could be alleviated by choosing specific windows of time where a more structured dynamic prevails, and whether the identification of those moments could be learned from past data. In order to do that, a novel framework is proposed.
This framework is called the Predictability Crawler (P-Craw). It uses optimizations routines such as the Particle Swarm Optimization (PSO) or Genetic Algorithms (GA) to select subsets of historical data where statistical learning algorithms can be more efficiently trained.
To access the accuracy of the method, it is tested against two different datasets. First, simulated data with varying percentage of noise is generated and used. In the simulations, The P-Craw is able to reliably identify the optimal subsets in scenarios ranging from 20% to 100% of predictable samples in the data. Second, intraday data from the Brazilian stocks exchange (BOVESPA) is collected and aggregated into feature and target matrices. When benchmarked against training with the whole samples in the BOVESPA data, the framework is able to significantly raise the correct directional changes of the trained models while reducing the Mean Absolute Error in up to 19%.

## Keywords

Stock Market; Predictability; Genetic Algorithms. Particle Swarm Optimization. Time Series.

## Resumo

Corbelli, Rodrigo Canto; Vellasco, Marley Maria Bernardes Rebuzzi; Veiga Filho, Álvaro de Lima. **Investigando Regimes Ótimos para Previsão no Mercado de Ações**. Rio de Janeiro, 2019. 73p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

A previsão de movimentos futuros para o mercado de ações é conhecidamente uma tarefa difícil de ser satisfatoriamente realizada. Além disso, a própria possibilidade desta previsão é constantemente questionada na literatura. O estudo presente investiga se essa dificuldade poderia ser amenizada escolhendo janelas específicas de tempo, onde uma dinâmica mais evidente prevaleça, e se a identificação desses períodos pode ser aprendida através de dados passados. Um framework é proposto para tratar desses problemas.
Esse framework é nomeado de Predictability Crawler (P-Craw). A proposta usa rotinas de otimização como o Particle Swarm Optimization (PSO) e Algorítimos Genéticos (GA) para selecionar sub-conjuntos de dados históricos onde modelos de aprendizado estatístico possam ser treinados de forma mais eficiente.
Para validar a acurácia do método, este é testado em dois diferentes conjuntos de dados. Primeiro, simulações com diferentes níveis de ruído são geradas. Nelas, o P-Craw é capaz de identificar os subconjuntos ótimos em cenários com 20% a 100% de amostras previsíveis. Por fim, dados de transações intradiárias da bolsa de valores brasileira (BOVESPA) são agregados e processados uma matrix de variáveis de entrada e um vetor de previsões. Quando o P-Craw é testado contra o método usual de treinar os modelos em todo conjunto histórico disponível nos dados da BOVESPA, o framework é capaz de aumentar significativamente o número de vezes que o modelo acerta a direção do movimento do preço das ações, enquanto consegue chegar a reduzir em até 19% o erro médio absoluto da tarefa.

## Palavras-chave

Mercado de Ações; Previsibilidade; Algorítimos Genéticos. PSO. Séries Temporais.

# Table of contents

# List of figures

# List of tables

# Definitions

**KNN-GA** Proposed modification to the GA algorithm where the the score of each individual is averaged by the scores of the k nearests individuals before the selection at each stage. 29, 68

**Perturbed PSO** Proposed modification to the PSO algorithm where the particles are perturbed after each iteration according to the sampled subset of that particle. 68

**purity** The percentage of predicable samples in the selected subset. 34, 35, 49, 50

**relative size** The ratio of the number of samples in the selected subset over the size of the whole database. 35, 38–46, 49, 50

**total purity** The percentage of predicable samples in the whole dataset. 34, 35, 37, 38, 42–48, 51

## Acronyms

| | |
|---|---|
| **AMH** | Adaptive Market Hypothesis. 15, 68 |
| **CDC** | Correct Directional Changes. 60, 62, 63, 68 |
| **EMH** | Efficient Market Hypothesis. 15 |
| **GA** | Genetic Algorithm. 25, 29, 50, 51, 69 |
| **HMM** | Hidden Markov Model. 50 |
| **MAE** | Mean Absolute Error. 60–63, 68 |
| **MASE** | Mean Absolute Scaled Error. 24, 29, 37, 38, 60, 62–64, 68 |
| **PSO** | Particle Swarm Optimization. 25, 26, 29–31, 39, 68, 69 |

# 1
# Introduction

In the financial market, forecasting of price series means financial profit, so the matter of how to accurately compute those forecasts is always on the spotlight. In order to tackle this challenge, two schools of thought are generally invoked: Technical and Fundamentalist [1, 2, 3]. To the Fundamental school of analysis, the intrinsic value of a stock paper is the crucial factor, therefore this school pays attention to indicators that go from the financial health of the company to government regulation policies on the sector and macroeconomic data. The Technical branch, on the other hand, looks only at previous movements in price charts and graphics to guide their beliefs concerning future behavior.

On the antagonist side to both approaches lies the Efficient Market Hypothesis [4]. The hypothesis states that all the information regarding an asset is already incorporated in it's price at any given time, therefore it is impossible to consistently beat the market. Despite of this conjecture, in recent literature machine learning techniques are constantly evoked to tackle the challenge of market forecasting using both technical and fundamental data [5, 6, 7, 8, 9, 10]. There is no clear winner is this task, and the most difficult benchmark to beat remains the random walk model [11]. To add to the discussion, a number of studies shows that the mere publication of market forecast solutions degrades their performance [12, 13, 14]. The resulting scenario is then one where trained algorithms can hardly beat the most simple model, and even when they do, can lose their predictive power over time. To reconcile these facts a more recent conjecture, the Adaptive Market Hypothesis, speculates a dynamic evolution of efficiency [15].

With this evolution it is possible that a specific strategy or model work at some time-frames, but can perceive only noise at others. This reality can be a problem for the regular supervised learning procedures [16] used in machine learning. Namely, the use of all available data in training can incorporate noisy samples that don't offer any structure to be learned. Those samples can worsen the final performance instead of helping. Noisy observations can degrade not only the training stage, but also mask the performance when used in out-of-sample evaluation. To study this issue, different proxies for market efficiency have been proposed [17, 18, 19, 20, 21] and enhancement has been uncovered when training and evaluation happen in respect to a selected "inefficient" subset of data [17, 19]. The different proxies proposed included Hurst's Exponents, entropy measures, different hypothesis test's p-values and linear regression coefficients values.

Instead of focusing on a specific proxy, this work aims at proposing a general framework to address the selection of predictable samples. This framework is called the Predictability Crawler (P-Craw). The P-Craw is meant to be an additional step to be performed when training a statistical learning model. This step filters which data points to be used in the training and evaluation of the model, removing noisy observations that can degrade performance.

## 1.1 Objective

The objective of the study is to propose the P-Craw framework for forecasting in regimes where not all observations are predictable. The proposed framework is able to use past data to learn how to identify predictable moments. It also uses the learned structure to select when to make a prediction or not in a new out-of-sample observation. In order for the P-Craw to be valid, it is also a necessary goal to assert it's capability of generating significant enhancements when used.

## 1.2 Methodology

To assert that the objective was fulfilled, the proposed methodology is composed of two parts:

1. A simulated series is created with both predictable and unpredictable samples, and it is used as a reference for the model. The first part of the study tests the P-Craw in those scenarios to assert if it can correctly discover the real dynamics of the simulation without any privileged information.

2. A real dataset is used: The intraday trade records for the brazilian stock market data. A feature representation is proposed to the price series. The data is split into training and test sets, and a statistical learning model is used to predict future behavior in two ways. In the first way the model is trained using the whole training set, and the error metrics are computed with respect to the whole test set as well. In the second way, the proposed framework is used and the model is trained only in the selected samples in the training set, and is evaluated only at the selected samples at the test set. The second part should then confirm if the P-Crawl brings a significant improvement in the error metric when compared to the usual approach.

## 1.3 Contribution

The main contribution of the study is the P-Crawl, acting as a new method that can be used both to train and evaluate machine learning and statistical models in regimes of changing predictability.

The work also contributes to the discussion regarding market efficiency. The existence of regimes with varying performance when trained in different time-frames corroborates with the evolution of efficiency proposed by the Adaptive Market Hypothesis. Therefore, the results discussed help to unveil evidence in favor of this conjecture.

Another contribution relates to the comparison between Technical Indicators. The relative importance of those in the prediction task is discussed in both the whole data available and the regimes selected for prediction.

## 1.4 Organization

The present document is organized in the following manner:

- Chapter 2 discusses the Efficient and Adaptive Market Hypothesis, as well as their consequences. It covers previous studies concerning regime switches in the market, exploring both the contributions given and where the ideas presented at them could be improved.
- Chapter 3 presents the P-Craw framework, describing each building block and the implementations used in the present context.
- Chapter 4 presents the results for both parts of the described methodology, discussing the conclusions in both in the simulation study and in the real-life case.
- Chapter 5 summarizes the findings and conclusions of the study and enumerates the challenges left open for future research.

# 2
# Previous Work

## 2.1
## Market Efficiency

In his seminal paper [4] Fama introduced the Efficient Market Hypothesis (EMH). The hypothesis states that price series innovations on the market follow a random walk pattern, where each new price adjustment is independent of previous information. In the argument, the fierce competition for profits assures that any new knowledge is immediately incorporated. With the information already reflected in price, is then impossible for a trader to create a strategy that consistently beats the market. This line of thought is put in a nutshell in the sentence "Prices fully reflect all available information".

Not all studies agree with this statement though. As a matter of fact, in [22] it is argued that a perfectly efficient market would not be plausible at all. The reason is that there are costs inherent to information gathering and arbitrage, and if there is no excess return to be made in exchange for this cost there would be little reason for traders to trade, causing the market to eventually collapse. In this context, the payoff of information gathering would be directly linked to the inefficiency, and a market equilibrium would arise from players paying less attention to saturated markets, lowering those markets efficiency levels, and focusing on new assets, raising those assets efficiency.

More Recently, Lo [15] proposed a new paradigm, the Adaptive Market Hypothesis (AMH). In the AMH, each agent is viewed as a specimen with constrained knowledge acquired through past experiences. Those agents would then not optimize their utility function, but rather find the best solution constrained to their current beliefs. In this context, the equilibrium is part of an evolutionary game between those agents fighting for resources (economic gains), learning and shifting their preferences towards different financial assets and arbitrage strategies.

This evolution would entail different efficiencies in different markets at different times in respect to different strategies, and thus allows for profitable opportunities to exist from time to time. The theory has gained force in light of evidence such as changing correlation coefficients over time in price series and trading techniques that showed vanishing performance once they were published in academic studies [13].

## 2.2
## Hamilton and Engel Markov Switching Model

The first line of studies worth mentioning is not directly related to market inefficiency, but with the more general idea of regime-switching. This idea has a milestone at the work of Hamilton and Engel [23], whose objective was to correctly identify regimes of price trends. A simple Markov Chain model was proposed to address the identification of those trends in the dollar exchange rate series, as depicted in Fig. 2.1.

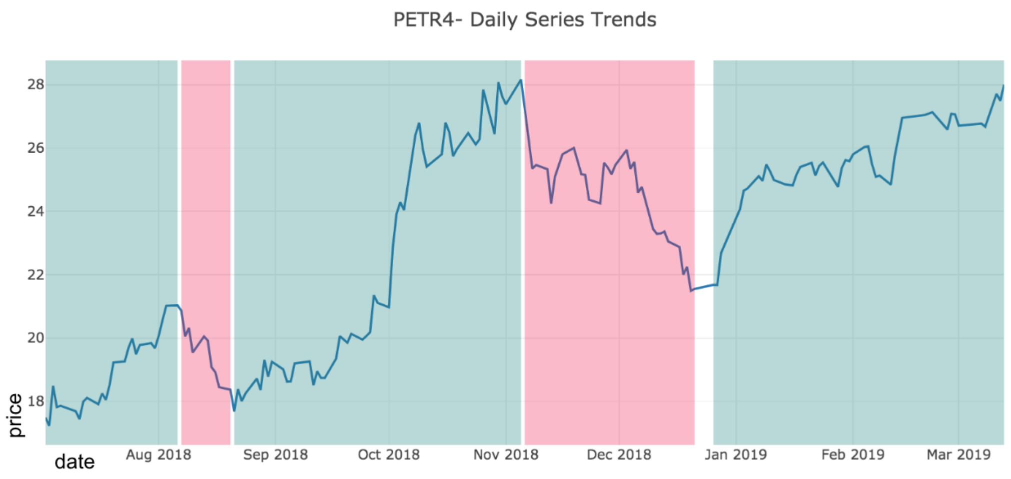


Figure 2.1: Example of trend identification in price series

In their proposal, it is postulated the existence of a latent variable $S$, which can take the values 1 or 2. The conditional distribution of returns is then $N(\mu_1, \sigma_1)$ or $N(\mu_2, \sigma_2)$ depending on the current value of $S$. The random variable S follows a first-order Markov process with transition matrix as in 2-1.

$$\begin{bmatrix} p_{11} & 1 - p_{22} \\ 1 - p_{11} & p_{22} \end{bmatrix} S_t = S_{t+1} \tag{2-1}$$

The system is entirely described by the vector $\Theta = (\mu_1, \mu_2, \sigma_1, \sigma_2, p_{11}, p_{22})$. This way, the model could capture up-trends in a regime with a positive mean, while downtrends would signal a negative tendency. The model also allows for asymmetry in trend probabilities, being able to model brief but sharp regimes for price appreciation while depicting losses in a subtle but more consistent manner. Inference on these parameters can be made using either the Hamilton Filter as originally proposed [23], or the Baum-Welch algorithm [24]. However, as pointed out later by Engel [25], this model fails to beat the random-walk benchmark in out-of-sample observations, despite showing some evidence in superior prediction of price change direction.

## 2.3 Regime Switching for Efficiency Identification

Regime switching models have also been used to model market efficiency. One important example is [26], where the dynamic of price changes at the dollar exchange rate series after the reveal of new information at instant $\tau$ is studied. In the author formulation, the exchange rate $e_t$ is proportional to a variable $z_t$ and the expected value at instant $t$ of the future variation $\Delta e_{t+1}$. The variable $z_t$ is composed of a random noise $\vartheta_t$ plus a drift term $\tilde{\mu}$ that changes after the information is revealed. Eq. 2-2 summarizes the process.

$$e_t = \alpha z_t + \gamma E_t[\Delta e_{t+1}]$$

$$z_t = \tilde{\mu} + \vartheta_t, \text{ with } \begin{cases} \tilde{\mu} = \mu_0, \text{ for } t < \tau \\ \tilde{\mu} = \mu_1, \text{ for } t \geq \tau \end{cases} \tag{2-2}$$

$$\vartheta_t \sim N(0, \sigma^2)$$

The agents can be skeptical about the information. For example, if it is an announcement of a new governmental policy, there can be doubt on whether this policy will be indeed effective or not. The question is then how they proceed in their believes. Once the information is made public, the participants know the market can be in either one of two scenarios: $P_0$, where $\tilde{\mu} = \mu_0$ or $P_1$, where $\tilde{\mu} = \mu_1$. It is proposed that they update their belief at each new observation according to the Bayes equation [16]. If the information consolidates, the series is expected to reach a new equilibrium point. If the market refutes the effectiveness of the announcement, it will return to the previous state and the exchange-rate series will exhibit a bubble pattern. The goal is then, as it usually is in technical analysis, to derive a sign prediction of future market movements. At any given $t > \tau$, is easy to see that

$$E_t[\Delta e_{t+1}] > 0, \text{ iff } \frac{1}{t - \tau + 1} \sum_{j=\tau}^{t} z_t > \frac{\mu_0 + \mu_1}{2} \tag{2-3}$$

Which is just saying that $P_1$ is more likely than $P_0$ if the mean of $z_t$ following the new information is closer to $\mu_1$ than it is than $\mu_0$. In terms of the observable exchange-rate $e_t$, Eq. 2-3 can be written as

$$E_{\tau+\kappa}[\Delta e_{\tau+\kappa+1}] > 0, \text{ iff } \frac{1}{\kappa + 1} \sum_{j=\tau}^{\tau+\kappa} e_t > \frac{E_t[e_t|\tilde{\mu} = \mu_0] + E_t[e_t|\tilde{\mu} = \mu_1]}{2} \tag{2-4}$$

The problem is that in practice the values $\mu_0$ and $\mu_1$ are not known. A possible workaround can be used if one notices that, should the expected value of observations increase after instant $\tau$, the mean of the series computed only after this period would be greater than a mean estimator using previous information as well. If the probability assigned to the regime shift is close to one and $\delta$ time steps have elapsed since the announcement, the average of the last $\rho = 2\delta$ observations compared to the last $\delta$ would be an unbiased estimator [26]. If the market skepticism about the information makes $P_1$ small though it would be necessary to use a smaller $\rho$ in order to have possibly more samples of $P_1$ and account for the effects of the prior beliefs. Either way, this can be expressed as

$$E_{\tau+\kappa}[\Delta e_{\tau+\kappa+1}] > 0, \text{ iff } \frac{1}{\kappa + 1} \sum_{j=\tau}^{\tau+\kappa} e_t > \frac{1}{\rho} \sum_{j=\tau+\kappa-\rho}^{\tau+\kappa} e_t \tag{2-5}$$

Equation 2-5 can be recognized by many as the Moving Average rule of technical analysis [27]. In practice, the time $\tau$ at which new information might have been incorporated is usually not known, what makes it impossible to compute accurately values for $\kappa$ and $\rho$ (respectively the windows for the short-term and long term averages). Although this may lead to a biased estimator, as long as $\rho$ is greater than $\kappa$ it will be consistent, and the signal can be used to alarm if a regime switch has occurred.

The importance of this study to the present context is not only that it derives a technical indicator from a regime switch perspective, but that this indicator is only meaningful in specific situations. The derivation above was reasoned in light of an actual incorporation of information leading to a change in market equilibrium, and so would be unwise to be used at every arbitrary moment.

To address this specification, the modelling of Hamilton and Engel is evoked and the exchange rate is once again assumed to be directly influenced by a latent variable $S_t$, which evolves according to 2-1. At state $S_0$, identified as having higher variance, the market is assumed to be in a more information efficient regime where technical analysis would not be appropriate, and a signal based on fundamental ideas is used instead. In state $S_1$, a moment of inefficiency is detected as the market supposedly tries to incorporate new information and the technical indicator derived in 2-5 is used. In their experimental setup, it is found that the technical indicator influence is statistically significant in the assigned regimes.

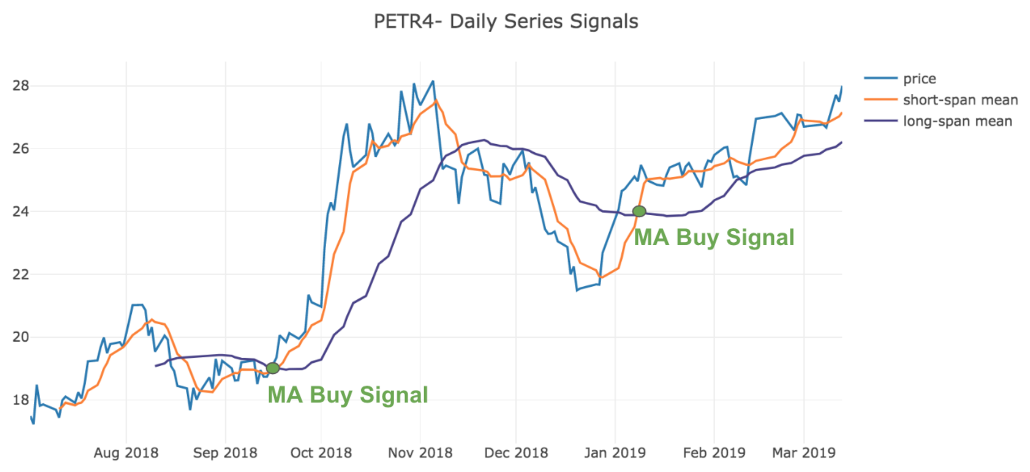


Figure 2.2: Example usage of the Moving Average indicator

## 2.4 Discussion

In the previous session, all the models incorporated the idea of regime-switching. Some with a more simplistic modelling, and some with more complex approaches. A possible pitfall in this line of research is the necessity of having to incorporate the non predictable time frames in the model. To the extent of this discussion, it is fruitful to dive into the concept of what would be a non-predictable window. As stated in [4], in a practical definition for the market trader, a series has independent increments as long as he/she cannot use past information to increase expected gains. Daily Futures price series, for example, show strong evidence of conditional heteroskedasticity and rejects the BDS test for the independence hypothesis [28], but it is not trivial how to use this information to create monetary gains.

Financial gains are related to knowledge regarding the direction and value of price movements. If the error of the predicted values are smaller than the unconditional uncertainty, or if the direction of price movements can be inferred with greater precision, the data can be considered to have some degree of predictability. In the present study, a predictable series is one where a model can be trained on past data to decrease uncertainty about future movements. This definition describes predictability of a series in regard to a specific statistical model. It can be the case that the series is considered predictable in respect to one model but unpredictable to another one.

In this context, unpredictability is not necessarily encompassed by one regime, but rather in every regime not explainable by the current model. In [26], unpredictable regimes (or in the context, regimes only predictable through fundamental analysis) were supposed to be inferred through the variance of normal innovations. Notwithstanding, it can be the case that periods of unpredictability encompasses not only different volatilities, but different shock distributions, and even a variety of other high-orders non-linear dependencies. Overall, detecting unpredictable regimes by assuming inherent characteristics of them might be problematic.

On the subject of Tree-based methods, one could argue that an algorithm such as Random Forests is capable of dealing with uncertainty in a natural way. The tress in the forest could just output the mean target value whenever a specific pattern could not be assigned to a sample point. Notwithstanding, it would be not trivial to assert when a mean prediction was assigned because of this uncertainty or because a predictable regime actually forecasted it. Even if this problem is solved, the model still has complications. In the trees, the top nodes would be responsible for the regime identification, while the lower hierarchy would do the predictions for each regime. As previously discussed, growing deep trees might lead to over-fitted models, so in practice very few splits (or none at all) would be in charge of each task. The problems of regime identification and regime forecasting are complex enough on their own, and so it would be more promising to address them in a separate way.

Models such as Neural-Networks have a high potential of adaptability and learning of difficult multi-step problems, and have been used before in price series prediction [29, 30, 31, 6] . But as already pointed, efficiency in the market can happen in more than one configuration, and a flexible algorithm could be tempted to "learn the noise" instead of discovering truly predictable samples. The signal-to-noise ratio in the training sample could worsen the bias-variance trade-off present in statistical learning problems [32], making a high variance algorithm even more prone to over-fitting. It would be ideal then if those models could be trained only at the inefficient periods of the market.

The question then is how to create a procedure that:

1. Does not make strong assumptions on the structure of efficient time-frames.

2. Separate the tasks of regime classification and regime forecasting into distinct ones

3. Train the final model only at the predictable subset of the data.

An approach in this direction can be found at [17]. In this study, the authors use the Hurst Exponent as a proxy for predictability by computing a rolling window estimate and monitoring the evolution of the estimator. A Hurst Exponent far from 0.5 indicates a dependence structure on the time-series. An exponent near 0 represents a mean-reverting pattern, and value near 1 indicates a trend-following behaviour. In [17] the initial data was broken into two groups: one with exponents greater than .65 and one with exponent values between .54 and .55. For each one of those two groups, a split in training, validation and test set was made. Then, a Neural Network was trained on the training set, with the validation set used to early-stop the back-propagation algorithm, and the performance is evaluated on the test set. After this, an unpaired Student's t test was performed on the null hypothesis that there was no difference in the performance, which was rejected on a p-value of $7.0290 * 10^{-10}$, showing very strong evidence of enhancement in performance when training on the group with the greater exponent.

The example in [17] was able to conform to items 2 and 3 of the list of conditions, but still made an assumption on the structure of the efficient time frames. Namely, that those periods would have a specific range of Hurst Exponents. It is, of course, a reasonable assumption, but one might be tempted to explore what others dependency indicators might have been used to classify those intervals. As a matter of fact, in other studies the predictability of financial series is analyzed through different metrics, ranging from information theory measures to regression coefficients [18, 19, 21].

# 3 P-Craw

This section presents the P-Craw framework, describing each part of the methodology in detail and presenting the implementation used in the results section. First, a general view is explained, with an in-depth approach being carried for each one of the building blocks afterwards.

## 3.1 General Framework

The P-Craw is defined to adapt the training of a statistical model to a situation where some data points might not be beneficial to the process. The most important part then is a structure capable of selecting a subset of the available data free of those points, representing only the predicate samples in the dataset.

A selector $S$ is the building block responsible for this task. Once the best subset $\mathbb{P}$ is defined, it is used in two different training tasks. The first is to fit a prediction model $M$. This one uses only the samples in the selected subset. The second is to train a classifier $C$, responsible to label out-of-sample observations into predictable or unpredictable ones. This task uses all the available samples, with the target to be learned being if they are present in the selected subset or not. The idea is that by training to reproduce the classification of the selected samples, this classifier will learn the structure that differentiates the data present in it from the rest.

The process so far constitutes the training phase of the framework. To test the performance of the method, the P-Craw turns the out-of-sample observations into features and feed them to the classifier. The vector is then selected as predictable or not. In the positive case, the trained forecaster $M$ is used to perform the actual prediction. If the classification returns negative, the process discards that samples, moving to analyze the next one. Fig. 3.1 depicts the process, and the following sections describe each building block in details.

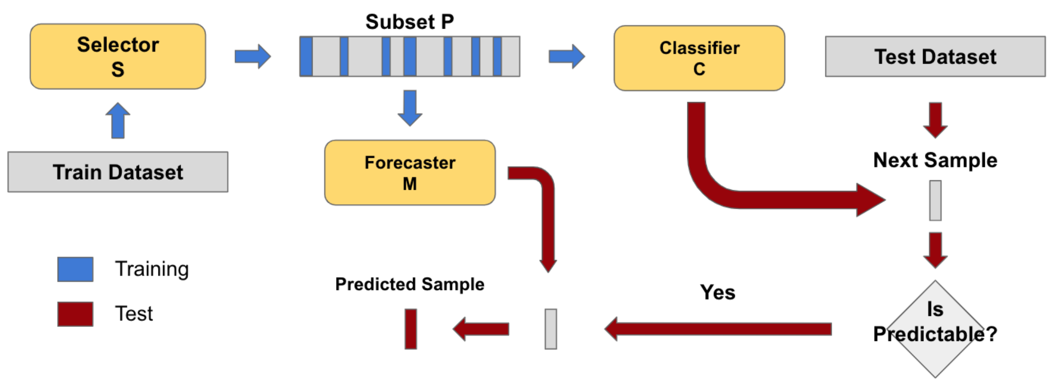


Figure 3.1: Proposed Architecture

## 3.2 Selector

Suppose the trader is given a set of historical data $\mathbb{D}$. The task is to find the subset $\mathbb{P}^\star$ in the set of all possible subsets $\mathbb{S}$ of $\mathbb{D}$ that represents the "most predictable" time frames available. Implicit in this task is the necessity to compare two possible subsets and be able to say which one better represents inefficient periods. This calls for a function $F : \mathbb{S} \in \mathbb{D} \mapsto \mathbb{R}$, which can be used to construct a ranking structure among possible sets. With this function defined, the problem now becomes an optimization procedure, in which one must find:

$$\mathbb{P}^\star = \underset{\mathbb{S}\in\mathbb{D}}{\operatorname{argmax}} F(\mathbb{S}) \tag{3-1}$$

This function needs to score subsets in respect to their ability to "be explained" by a statistical model. In order to carry that, the time series is represented as feature and target matrices. The scoring process must be paired with a specific model to be trained on the given subset, and the outcome of this training is used to compute $F$. Once this scoring mechanism is settled, an optimizer search amongst the space of possible subsets to find the solution of Eq. 3-1. Fig. 3.2 illustrates the process.

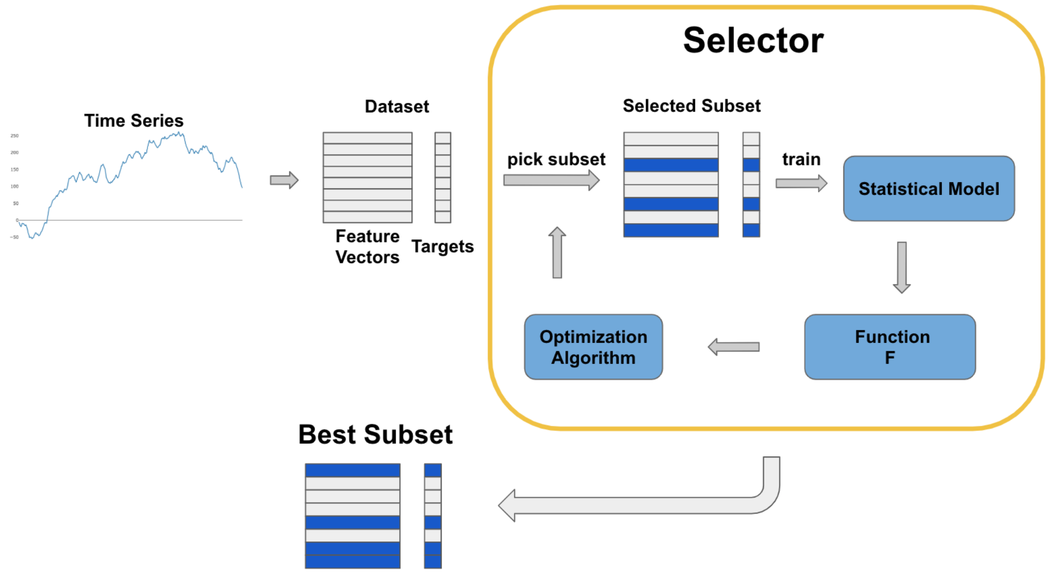


Figure 3.2: Optimization Procedure

The process is composed of three main entities. The statistical model chosen, the scoring function $F$, and the optimization algorithm. This subsection explains the specifics of the model and the function $F$. The optimization procedure is the main engine of the framework and so deserves a subsection of it's own.

### 3.2.1
### Statistical Model

As previously mentioned, the whole point of the selector is finding a subset of data more suitable to be predicted. In order to verify this property in a given series, a statistical model has to be defined. As an example, lets use the simple case of a linear regression.

The first step is to create the feature representation of the time series. Instead of working with the raw values, a common approach is to use the log return series, computed as

$$r_t = \log \frac{s_t}{s_{t-1}} \tag{3-2}$$

Where $s_t$ is the value for the time-series at instant $t$. The features to be used at instant $t$ can be, for instance, the last $n$ observed returns. The target to be predicted can be the next return $r_{t+1}$. Once the series is processed in this features/target form, the optimization procedure can start. At it, in each iteration different candidate subsets are selected according to the optimization procedure. The selected data is then used to train the statistical model, which in turn is used to compute the score $F$ of the given candidate. Fig. 3.3 displays the example with $n = 15$.

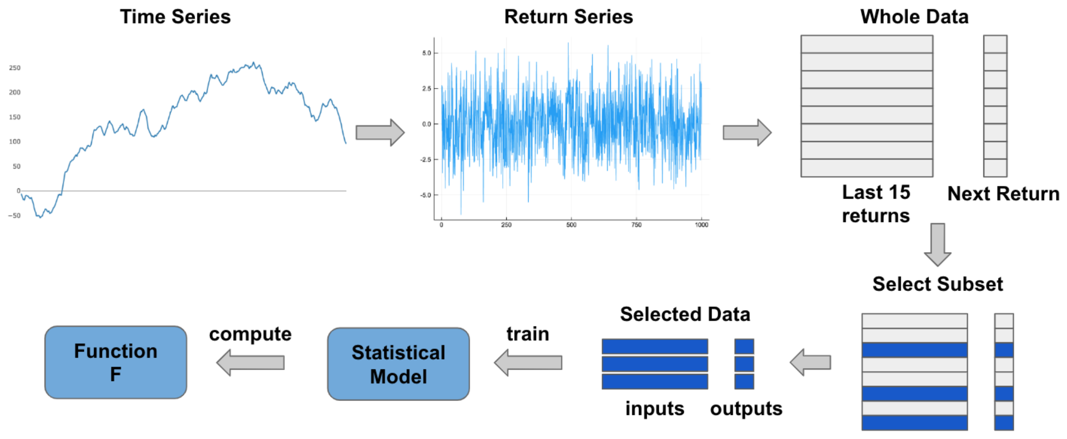


Figure 3.3: Example of Model Setup

In the case of the statistical model being a simple linear regression using the described features, the model to be trained would be as in Eq. 3-3, where the next return $r_{t+1}$ is a sum of previous returns with appropriate coefficients plus a constant term $\beta$. The training can also include pre-processing schemes, such as feature selection. For example, the LASSO algorithm [16] can be used to choose a subset of features to include in the regression.

$$r_{t+1} = \beta + \sum_{i=1}^{n} \alpha_i r_{t-i} \tag{3-3}$$

Once the model has been fitted for a particular subset, it can be used in the scoring function evaluation, described next.

### 3.2.2 The Fitness Function

The scoring function has the purpose of quantifying how accurately the candidate subset can be represented by the chosen statistical model. To be consistent with the naming convention in the optimization literature, hereafter it will be referred to as the fitness function.

With the statistical model trained at a specific subset, an error metric is used to access how much the model could adjust to the trained data. The Mean Absolute Scaled Error (MASE) error metric is selected and computed according to 3-4.

$$MASE(y, \hat{y}) = \frac{\frac{1}{n}\sum_{i=0}^{n} |y_i - \hat{y_i}|}{\frac{1}{n}\sum_{i=0}^{n} |y_i|} \tag{3-4}$$

Where $y_i$ is the target and $\hat{y}_i$ is the model's prediction. The MASE is chosen because of two properties. First, as it uses absolute differences it is less sensible to outliers than possible alternatives such as the Mean Squared Error (MSE) [16]. Second, as it is normalized by the absolute value of the returns, it relates directly with how much uncertainty the model was able to explain.

If the chosen subset was used to both train the model and compute the MASE, this could be a problem. The model could achieve a low error at the specific subset while not being able to correctly represent out-of-samples observations. To alleviate this risk, the model evaluation in each subset is computed using a cross-validation scheme [16]. In it, each subset is divided in 3 parts, two are used for training, while the last one is used to compute the metric. This procedure is repeated until each fold had the opportunity to be used in the evaluation phase, and the values are averaged. The resulting metric is referred to as $MASE_{CV}$, and Fig. 3.4 helps to illustrate the process.

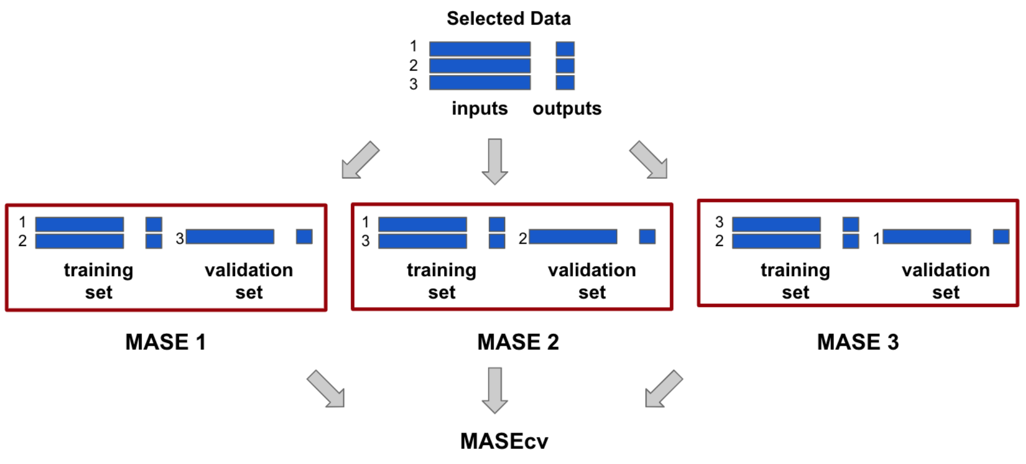


Figure 3.4: Computation of $MASE_{CV}$

Using the $MASE_{CV}$ directly as a fitness score might lead the optimization to search a point where the model can learn correctly, but it does not encourage the mechanism to explore further regions and find all the predictable samples. Without that, the framework can discard a valuable number of "good" intervals. In order to avoid this, a measure of subset size is incorporated. The chosen measure is the proportion of the size of subset $\mathbb{P}$ in respect to the whole dataset $\mathbb{D}$, $\frac{|\mathbb{P}|}{|\mathbb{D}|}$. This choice not only has the same order of magnitude as the MASE, but also ensures a metric unrelated to that particular dataset size. The weight of this new factor influences a lot on the results, and two versions are proposed in Eqs. 3-5 & 3-6.

$$F_{exploit}(\mathbb{P},\mathbb{D}) = (1 - MASE_{CV}) + 0.1 * \frac{|\mathbb{P}|}{|\mathbb{D}|} \tag{3-5}$$

$$F_{explore}(\mathbb{P},\mathbb{D}) = (1 - MASE_{CV}) + 0.3 * \frac{|\mathbb{P}|}{|\mathbb{D}|} \tag{3-6}$$

The constant 1 in both definitions makes no practical difference, being present only to bring some interpretability to the values. Namely, a positive value in the first term of both functions signify a model capable of achieving a MASE smaller than 1, revealing enhancement over the unconditional absolute values of the series. The equations differ in the constant multiplying the $\frac{|\mathbb{P}|}{|\mathbb{D}|}$ term. The 0.1 and 0.3 constants where chosen based on the experimentation of a range of values between 0 and 1, to bring different properties to the optimization procedure. Eq. 3-5 assigns a lighter weight to $\frac{|\mathbb{P}|}{|\mathbb{D}|}$, which influences the framework to focus more on good solutions than on exploring new ones. Eq. 3-6 gives three times more importance to the same term, favoring the exploration phase in the optimization procedure. For this reason, they are respectively named $F_{exploit}$ and $F_{explore}$. Both equations will be further explored in the Results section with aid from a simulation study.

Besides the mathematical representation, the definition of the fitness function also needs a minimum number of samples $n$ in the chosen subset. This restriction is obvious, as an arbitrarily small subset would be unfitted to be used on the training of a statistical model. If a model with less than $n$ samples is evaluated, it is assigned the smallest value possible, so that any interval with a number of samples greater than $n$ is preferred over it.

## 3.3 Optimization Procedure

Gathering what was exposed, it is possible to assign fitness scores to each possible candidate subset. The selector then proceeds to choose which new subsets to evaluate, using what it learned from the previous choices. The optimization procedure is the algorithm responsible to make this choice. Although generally speaking the P-Craw admits any algorithm capable of dealing with this task, in the present context two specific methods are implemented and discussed. They are the Genetic Algorithm (GA) and the Particle Swarm Optimization (PSO).

Genetic Algorithms are a class of optimization procedures that take inspiration on the dynamics of natural selection to choose how to explore the parametric space. They first appeared in [33], and operate on a chromosome representation of the model parameters. Although not strictly necessary, usually this representation is made in a binary form, such that the final chromosome is an array of 0s and 1s.

The Particle Swarm Optimization is another algorithm capable of optimizing a fitness function only by computing point values of that function. In this method, each candidate is assigned a velocity vector together with its position, and so the solutions "wander" through the parametric topology. At each iteration the particles update their velocities based on the best position they have already encountered, and on the best position encountered by their neighborhood. This dynamic mimics both the behavior of some group of animals finding food and that of human behavior, where individuals proceed based in their beliefs and that of their social group [34].

Both models are traditional, taking different approaches to the task. They were both implemented in order to understand how the proposed optimization problem behaves in different scenarios. In other to better adapt those algorithms to the present context, modifications are imposed to each one. In the GA approach, a technique named KNN-GA is presented. For the PSO, a variant named Perturbed PSO is discussed. The next subsections present those algorithms in more detail.

### 3.3.1 Genetic Algorithm (GA)

This subsection describes the chosen implementation for the regular genetic algorithm, from the chromosome representation to the restarting mechanism.

#### 3.3.1.1 Chromosome Representation

The chromosome representation in binary form of a GA is a 0 and 1 encoding of the parameters. To create this representation for the present context, the whole dataset will be joined into groups of size equal to 2% of the number of available points $N$. If 2% of the size is not a integer, the group size will be that number rounded up, and the remainder of the division of the size by the group size will compose an offset group that starts the series. For example, for a series with of size 320, 2% is 6.4, the group size is then 7, and the offset is $320 \mod 7 = 5$. Each group will be represented by either a 0 or a 1. A 0 means the whole group is not present at the subset, while a 1 signifies the inclusion of that group.

This 2% group size representation is chosen in order to impose some restrictions to the way subsets are selected, and to make the number of genes in the final chromosome well behaved. Just for the sake of visualization, Fig. 3.5 drops this restriction and represents how a series with 100 samples would be represented with using a group size of 30. This setup is not the one described, and is only shown here to display groups an offset without visual clutter.

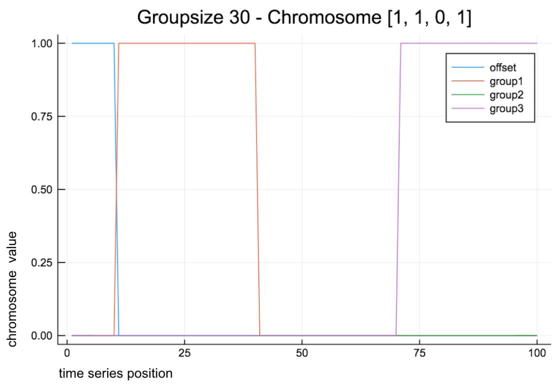


Figure 3.5: Group Chromosome Representation

### 3.3.1.2
### Initialization

The initialization of an optimization algorithm is an important factor in the overall performance [35]. Recall that when the fitness function was discussed, it was stated that a minimum number of samples $n$ for the subset had to be defined. The initialization procedure takes this number as a parameter. To avoid initial solutions far from the possibility of choosing all available data, the initial sets are restrained to have the smallest number samples greater than $n$ possible.

Two initialization methods are tested. In the first, each individual solution is initialized independently of each other. In the second one, the process happens sequentially. When the first individual is created, the intervals active in it are discarded from the possible ones to be active in the next, and so on. When all the intervals are taken, all of them become available again and the process continues. This method was designed to ensure an initialization with broad coverage of the parametric space.

### 3.3.1.3
### Selection Mechanism

Regarding the selection scheme, the tournament method is implemented, as it can be shown to have better rates of convergences than the original proportionate selection mechanism [36] . In the tournament selection, a group of individuals is randomly chosen to participate in a match (the number is called the selection pressure, and is usually 2), and the candidate with the highest value among the participants is selected. This method is then chosen to be used in the present context along with its usual pressure value.

### 3.3.1.4
### Crossover & Mutation

The evolution of one generation of possible solutions happens by means of crossover between existing individuals. Every time a couple is selected for mating, there is a probability that a crossover operation will be used to generate two offspring from those individuals. If they are not chosen by the operator, they proceed unaltered to the next generation. Two distinct operators are presented and tested in the model. In the classical Single-Point crossover, a random index on the chromosomes vector is chosen, and the final offspring are the parents with genes exchanged from that point on. In the Two-Point variant, two points are chosen instead of one, and the genes are exchanged in the interval between the two points. Fig . 3.6 illustrates both processes.

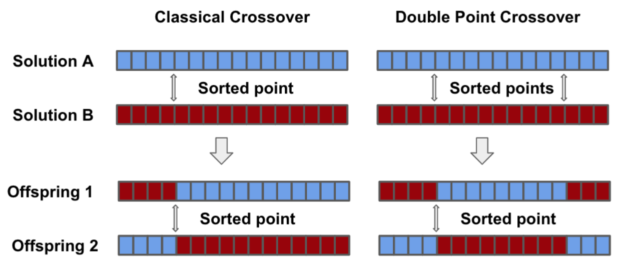


Figure 3.6: Crossover Methods

A simple implementation of the mutation method is used. If the individual is indeed selected to mutate, each gene in the string of that solution will have a probability $(1/l) * 10$ of being flipped (with $l$ being the length of the chromosome). This provides a mutation affecting in average 10% of the genes of the individual. This value was used to ensure a significant change in a mutated chromosome. The probabilities for both mutation and crossover are constant during the evolution of the algorithm, for the sake of simplicity.

#### 3.3.1.5
#### Survival

In order to avoid losing the best solutions in the evolutionary process, the survival operator carries the best individuals from one generation to the next. The proposed form of the Genetic Algorithm uses an elitist approach, carrying only the best individual at each step in the pool. This approach has a positive impact on the convergence properties of the GA, making it more likely to the algorithm to achieve the global maxima and even guarantying asymptotic convergence in some scenarios [35].

#### 3.3.1.6
#### GA with Restarting

In [37] it is proposed a technique to enhance the common GA setup. In genetic algorithms, as with every search task, there is a trade-off between exploration as exploitation [38]. In the early stages, the population is exploring the search space, not paying particular attention to any specific solution. When a promising region is found, this discovery triggers an exploitation phase, where the exploration of new areas is compromised in favor of the further refinement of what was learned. If the exploitation pattern is reached too early, the method might get stuck in local maxima, not being able to find better solutions. To avoid that, the study proposes to restart the population at every $N$ generations, but keeping a portion of the original participants. This way, the knowledge acquired would be maintained while avoiding early convergence.

The $N$ parameter defines how often the restarting will occur. In the present context, the restart is set to happen at the middle of the evolutionary process, so that the population will have equal opportunities to evolve both before and after it. For example, if the population is allowed to evolve for 100 generations, at generation 51 it will be restarted. The idea will be implemented in the current context in an elitist fashion, keeping only the best solution at that generation. For the rest of the population, the restarting will discard all the solutions and replace them with a new population created with the same initialization method chosen for the first individuals, being either the group or random initialization.

### 3.3.2 KNN-GA

This subsection presents the KNN-GA. The method is a modification on the regular GA created with the current context in mind. The process of trying on a different combinations of subsets, computing MASE metrics, and choosing the best one, can lead to biased results. In fact, if one tries with enough noisy combinations, one of them will eventually display a deceivingly predictable pattern. One hypothesis on how to repel this possibility is to explore the neighborhood of the solutions. If a fitness score was a product of a spurious combination of noise, it is likely to have a sensitive response to any change in its configuration. On the other hand, if the corresponding subset actually contained more predictable intervals them otherwise, small changes, although capable of interfering, are expected to be less destructive.

To explore this reasoning, the neighborhood of the candidates is incorporated in the score. Once all the fitness numbers are computed for the population, each individual's fitness is replaced with the average of the k nearest neighbors (itself included), where k is a parameter to be defined. The distance is computed in the space of the chromosome representation, and the Hamming distance is used [39], with the closest genes being the ones with the smaller number of genes different from the chosen one.

This way, the value of a candidate solution will represent how well is it's surrounding area, and if a spurious result occurs in a noisy region, it will have its survivability greatly reduced. This works in a smoothing fashion, reducing the importance of noise and helping the optimization of the underlying structure. This implementation will be named KNN-GA.

### 3.3.3 Particle Swarm Optimization (PSO)

This subsection describes the implementation used for the regular PSO. The Particle Swarm Optimization represents the possible solutions as wandering particles. At each iteration, each particle updates it's velocity vector the solutions explored so far. The update equation used is the same presented at [40] as is that of Eq. 3-7.

$$v_{t+1} = wv_t + U(0, c_p)(p_t - x_t) + U(0, c_g)(l_t - x_t) \tag{3-7}$$

Where $v_t$ is the velocity at iteration t, $x_t$ is the position at iteration $t$, $p_t$ is the best know position so far by that specific particle, $U(0, c)$ is a uniform distribution in the range 0 and c and $l_t$ is the best know position found in the neighborhood of that particle. The values for $w$, $c_p$, and $c_g$ were defined by experimentation, and are displayed at Eq. 3-8.

$$\begin{aligned} w &= .5 \\ c_p &= .5 \\ c_g &= 1.0 \end{aligned} \tag{3-8}$$

The neighborhood of the candidate is a set of N other solutions chosen randomly at the initialization. Each time an unsuccessful iteration occurs (one that doesn't raise the global maximum found so far), the network is scrambled and a new neighborhood is assigned to every participant. The number of neighbors was also chosen by experimentation an is defined as $N = 3$ in the present work.

#### 3.3.3.1 Position Representation

The PSO concept assumes a continuous search space, and so the binary representation used in the GA chromosomes can not be used. To be able to apply velocities to the representation of solutions while not diverging drastically from what was proposed previously, the values of each gene in the GA vector are given a new interpretation. Instead of defining whether the whole interval will be included or not, they now characterize the chance every sample at the interval has of being included in the subset. This way, if a interval has a parameter of 0.5, each sample in that group has a 50% probability if being included in the subset. Every time that particle is evaluated, this sampling occurs to every group. This way, evaluating a particle different times will produce different results, as different samples will arise. Fig. 3.7 displays one possible particle for a data-set of 100 samples and a group size of 30.

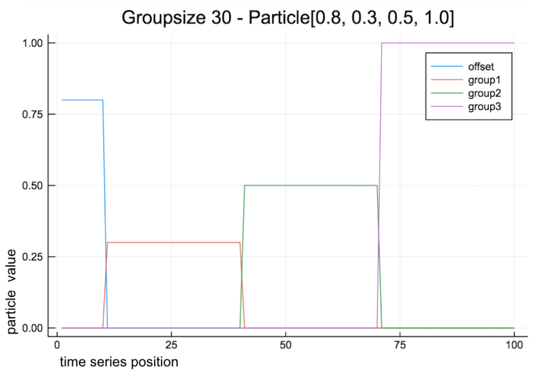


Figure 3.7: PSO Particle Representation

It is important to notice the randomness associated with this representation. Each time a particle is activated, it generates a different subset and, of course, this property will influence the exploitation-exploration dynamics of the algorithm. As a matter of fact, the exploration aspect will be favored. Since the impossibility of locking in one specific solution arises, the method is more prone to continue searching in other areas. This can cause an effect similar to the one aimed at with the KNN-GA formulation; on average, the metric at a given position will be influenced by a neighborhood of subsets around that position.

#### 3.3.3.2 Initialization

The initialization of the algorithm takes the equivalent approaches described to the GA variant, one being aware of the whole population and one being completely independent for each particle. In the independent approach, each dimension in each particle is initialized by the sampling of a uniform distribution between the values 0.0 and 0.4. This range is determined to keep the initial particles far from the solution of selecting the whole database. In the group aware approach, as in the GA, each particle sorts a number of intervals to have the value 1, and the other values set to 0. For the next particle, the intervals already taken are discarded from the sampling so on. In both approaches the initial velocities of each dimension are sampled from a uniform distribution between -1 and 1.

### 3.3.4 Perturbed PSO

This subsection now describes the variation proposed to the regular PSO algorithm, enumerating the motivations and reasoning behind them. In the context of the proposed PSO, the sampling happens every time a particle position is evaluated. When that happens, a particle that had a value of .35 for a given interval may, for example, choose half the samples of the group to be included in the subset. There are two ways to proceed about that. In the first, the fitness function is computed to the sampled subset, and the algorithm continues. In the other option, the position of the particle is altered to conform to the current sampling. In the previous example with half the samples in the group chosen, the value of that group would be altered from .35 to 0.5.

This way, the particle suffers a random perturbation in it's position every time it has it's value computed. This perturbation $\epsilon$ is independent for each coordinate, and as it arrives from the trial of $n$ Bernoulli's variables with probability $p$ of success (where n is the group size and p is the current value) the distribution of $\epsilon$ is:

$$\mathrm{P}(\epsilon) = \binom{n}{n(\epsilon+p)} p^{n(\epsilon+p)}(1-p)^{n(1-\epsilon-p)} \tag{3-9}$$

The perturbed version of the PSO involves a simple modification then. At the end of each iteration, when the positions have already been updated, perform the sampling at that position and then perturb the particles according to the outcome.

Fig 3.8 shows the distribution of those perturbations as the value $p$ in the group varies. As it can be seen, the functions are not symmetric, and for values near the 0 and 1 boundaries, they concentrate their probability mass in a way to "push" to particles towards those boundaries. In this sense they make the 0 and 1 values "sticky", making it more difficult, but not improbable, that the particle escapes them. This can be seen as a desirable property, because it helps to discard bad intervals and keep good ones. It also has a higher variance when the candidate is "in doubt" (values of p near 0.5) in respect to that group, prompting the solution to test one of the sides.

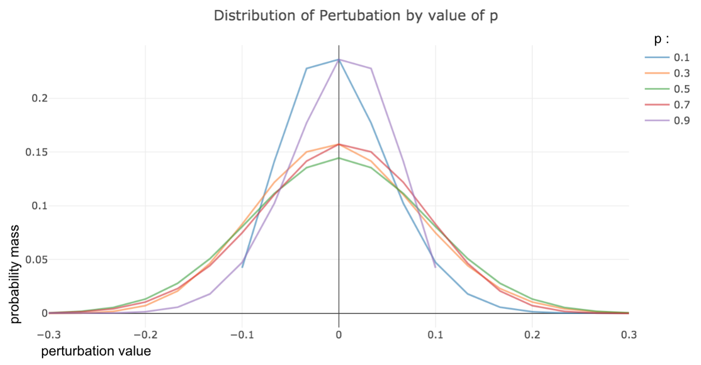


Figure 3.8: Perturbation Distribution by current value of group p

## 3.4 Forecaster

With the chosen subset at hand, the samples are used to train the forecasting module in a supervised learning setup. The task is to train a statistical model only in the set proposed by the Selector. This model doesn't necessarily has to be the same as the one used in the Selector's optimization procedure. As previously exposed, the predictability identification and series forecasting are totally separated tasks in the P-Craw methodology.

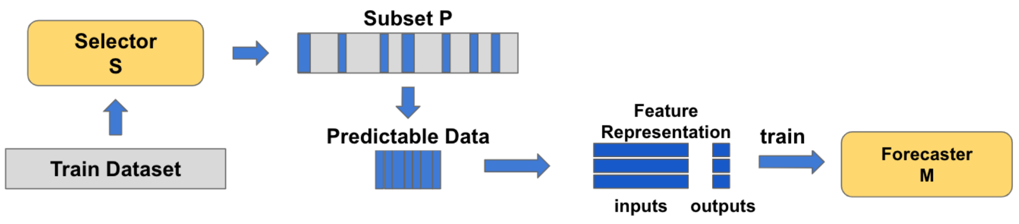


Figure 3.9: Training of the Forecaster

Not only the model can be different, but even the feature representation can change. The feature representation in Fig. 3.9 is not tied up to be the same as the one in Fig. 3.3. The Forecaster has the role of predicting the movement of the price series. This prediction can be regarding the next value, but it also can be a classification regarding if the prices will go up or down.

In the present study, 3 models are used as Forecaster in the Results Chapter, all of them focusing on predicting return values. In this case, the models used the same feature representation as the one used for the optimization procedure of the Selector. The linear regression coupled with the LASSO for feature selection, the Gradient Boosted Trees for regression, and the Random Forests [16, 41].

## 3.5 Classifier

Once the best subset has been identified, a label is assigned to each row of the feature matrix representing whether it is on the chosen subset or not (in other words, if it is considered as predictable or not). This label is used as a target for a new task. This time, supervised learning is used to train a model capable of identifying new samples on the test phase which posses the same dynamic as the ones in the selected subset. As in the Forecaster, the features used don't need to be the same used in any other building block of the framework. Fig. 3.10 displays the training process.

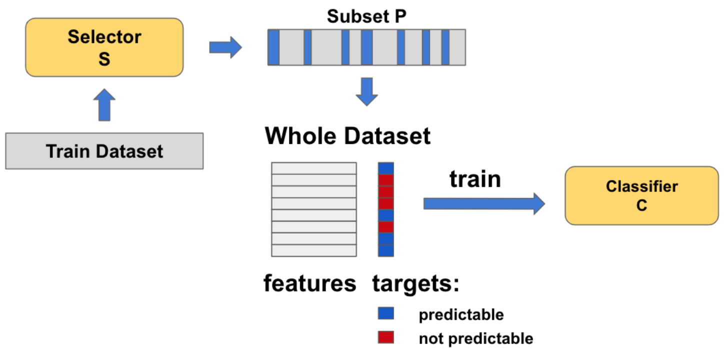


Figure 3.10: Training of the Classifier

In the present context the model used to classification was the Gradient Boosted Trees for classification [16]. For the sake of simplicity, the Classifier used the same feature representation as the Forecaster and the Selector.

# 4
# Results

The results of the study are divided into three sections. The first introduces the setup of a simulation constructed to study the proposed framework. The second displays the results when the P-Crawl is tested in the simulated data for both the PSO and GA optimization procedures, as well as the KNN-GA and Perturbed PSO variations. The third part is a real-life application of the method on data from Brazil's stock market BMF&Bovespa.

## 4.1
## Simulation Setup & Fitness Function

### 4.1.1
### Setup

A simulated series is created o help investigate the usefulness of the proposed framework. In this simulation, three different regimes are concatenated together. The first one follows an ARIMA(1, 1, 0) process when transformed by the logarithm function, with coefficient $\phi = .7$ and $\sigma = 2$. The differences $\Delta y_t$ of the transformed process evolve according to Eq. 4-1, where $y_t$ are the values of the time series.

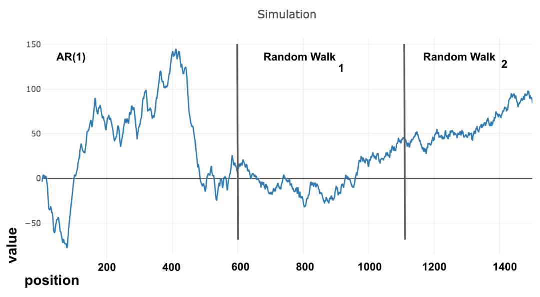
$$\begin{aligned} \Delta \log y_t &= \phi \Delta \log y_{t-1} + \epsilon_t \\ \epsilon_t &\sim N(0, \sigma) \end{aligned} \tag{4-1}$$

The other two regimes are random walks when transformed by the logarithm function. The innovations of those regimes have mean $\mu = 0$ and variances $\sigma_1 = 2$ and $\sigma_2 = 6$ respectively. They evolve according to 4-2.

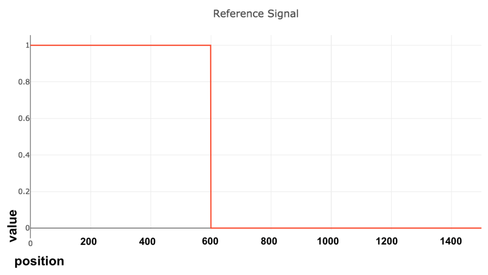
$$\begin{aligned} \Delta \log y_t &= \epsilon_t \\ \epsilon_t &\sim N(0, \sigma_i),\ i = 1, 2 \end{aligned} \tag{4-2}$$

Following the example of subsection 3.2.1, the log return of the series is used. It is important to notice that $\Delta \log y_t = r_t$, and so the returns in Eqs. 4-1 and 4-2 follow respectably a AR(1) process and a random Gaussian noise. To create the feature representation, a vector of past returns is used using the last 15 observations. Every simulation generated has 2015 observations, creating a feature and target representations of 2000 entries.

Before proceeding to next discussions, the concept of purity needs to be introduced. In the current study, the purity of a dataset will be defined as the percentage of predictable regimes present in it (AR(1) in the simulation case). If a simulation has 300 predictable samples and 700 unpredictable ones, it will have a purity of 30%. Subsets of a given simulation can have a different purity and so it is important to differentiate between subset and simulation purity. When the simulation purity is discussed, it will be referred to as total purity, to emphasize its calculation regarding all the dataset. Simulations with different total purity are generated. In each new simulation, a reference signal is generated together with the X and Y matrices with the predictable rows classified as 1 and the others as 0. Fig. 4.1 displays an example of a simulated series and reference vector.



(a) Simulated Series



(b) Corresponding Reference Signal

Figure 4.1: Simulation Example

### 4.1.2 Fitness Function

As discussed in Section 3.2, the fitness function needs to be associated with a specific statistical model. Following the example presented is this section, a simple linear regression is used. As mentioned, the model can pre-process the data before training. In the simulations, a feature selection was performed using the LASSO algorithm.

As presented in Chapter 3, the model is trained in a subset and used to compute the fitness function. Two different forms are presented and studied for this function, and are again displayed at Eqs. 4-3 and 4-4. The $MASE_{CV}$ is the MASE error metric computed by a 3-fold cross-validation. The quantity $\frac{|\mathbb{P}|}{|\mathbb{D}|}$ is the ratio of the subset size in respect to the whole data.

$$F_{exploit}(\mathbb{P},\mathbb{D}) = (1 - MASE_{CV}) + 0.1 * \frac{|\mathbb{P}|}{|\mathbb{D}|} \tag{4-3}$$

$$F_{explore}(\mathbb{P},\mathbb{D}) = (1 - MASE_{CV}) + 0.3 * \frac{|\mathbb{P}|}{|\mathbb{D}|} \tag{4-4}$$

Given a fitness function and a generated simulation, it is interesting to visualize the landscape of function on that dataset. To be able to display that, each subset can be represented in a X and Y-axis according to their purity and size, with the fitness function computed and encoded in the Z-axis.

It is important not to confuse the total purity with the subset purity. The subset purity is ratio of predictable samples in the subset with the total samples in it (it's size), while the total purity is computed in regard to all the data. Fig. 4.2 displays both $F_{exploit}$ and $F_{explore}$ in simulations with varying total purity, and a number of important observations can be inferred.

First, the function increases as the purity of the subset increases, which is a desirable property. Second, the relative size term makes the function values

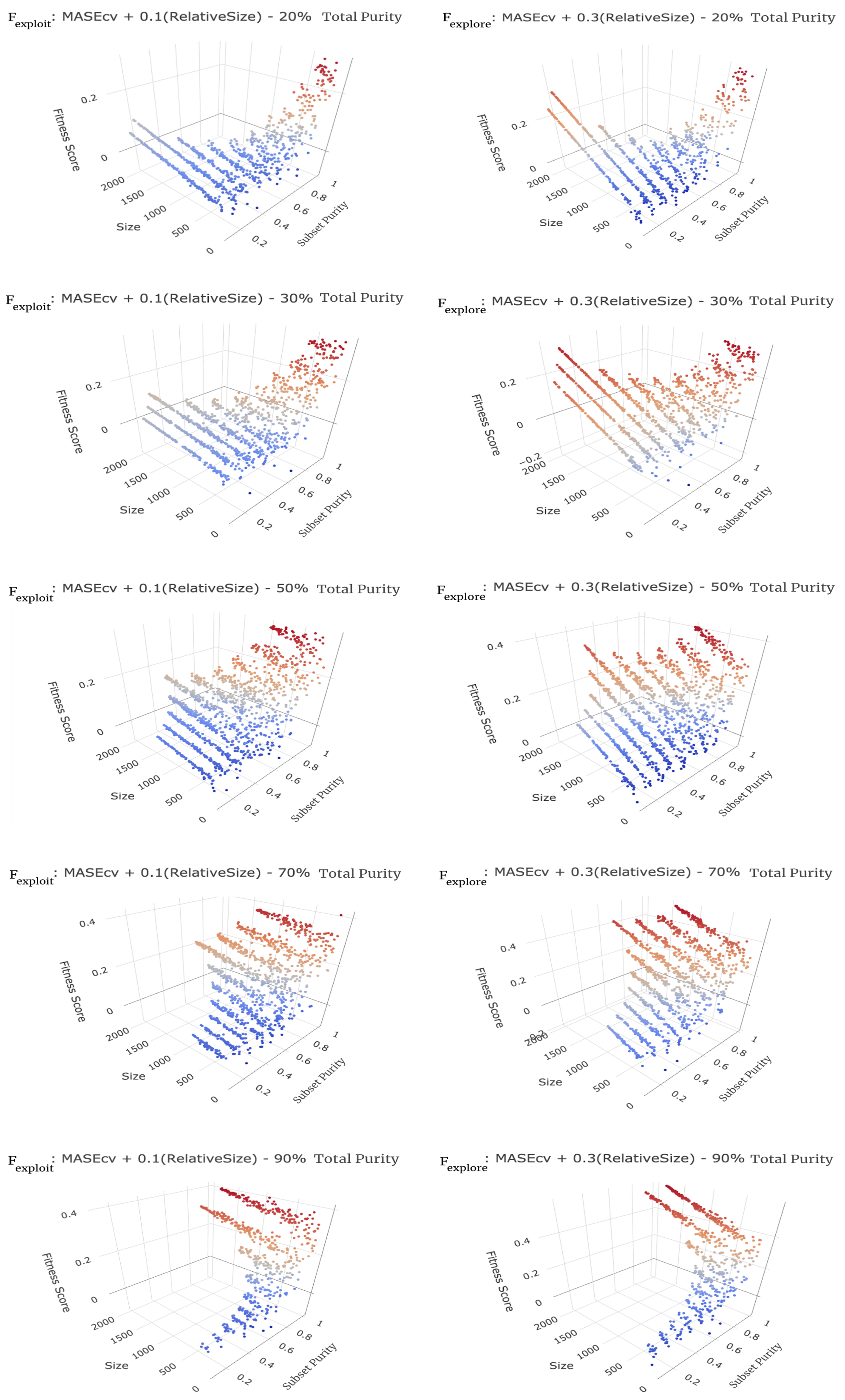


Figure 4.2: Proposed Fitness Functions

rise as the number of samples in the chosen subset increases, placing the global maxima exactly in the subset correctly containing all of the predictable periods. Third, function values get erratic for lower subset sizes, and can display spurious high values at those regions, especially for low sizes with high purity.

The version $F_{explore}$ enforces the importance of covering all the predictable samples, but also raises the values of large subsets with poor purity. There is a local maxima in the solution of picking the whole subset in all the versions, and at the 20% and 30% mark this solution gets dangerously close to the global maxima in $F_{explore}$. Actually, looking at the landscape of this function at those values, an optimization algorithm might be forced to make a choice of whether to go to large subsets or small ones, and can get trapped if the wrong choice is made.

This line of thought is the reason why two fitness functions were proposed. Function $F_{explore}$ is more adequate for datasets with a high total purity, but can dangerously benefit large and noisy intervals when it is difficult to discern predictable patterns. Function $F_{exploit}$ alleviates this effect, but gives a poor incentive at exploring subsets with greater number of samples.

A question that naturally arises then is how to identify each case, since in a real situation the total purity is unknown. If the images in Fig .4.2 could be plotted in a actual problem the total purity of any given dataset could be identified, but that is not the case. The unknown part that makes it impossible is, of course, the purity axis. Even so, the images could still be visualized projected in the "Size" axis, a projection that is bound to carry some information of the whole picture.

To visualize this, the point of 1000 samples in the "Size" axis, representing subsets with half the size of the whole data, was chosen. For each simulation, random subsets of 1000 samples were drawn and had their MASE evaluated after being used to train the selected model (with cross-validation). The histogram of those values were computed and compared for simulations with different total purity, and the result is displayed in Fig 4.3.

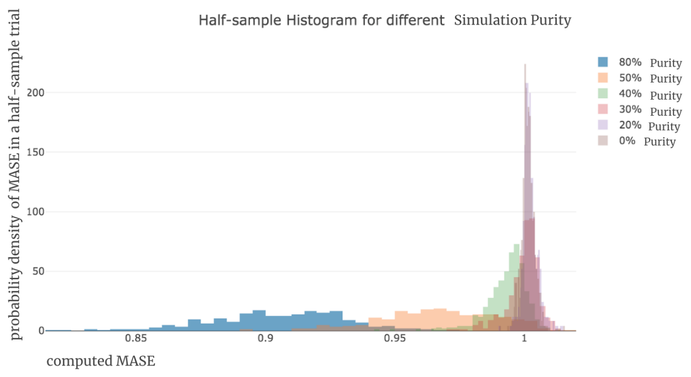


Figure 4.3: Different Simulation Histograms

As it can be seen, simulations with different purity display very different histograms, both concerning the mean as well as the variance of the distribution. The first key observation that can be drawn is that datasets with a high percentage of predictable samples concerning the chosen forecasting model can be easily identified by their ability to have MASEs significantly lower than 1.

This might not always be the case though, as this property depends on the portion of variance the model can explain in the predictable samples. For example, in a real series even models trained on predictable sets may not be able to achieve a 0.85 MASE value. The actual error metrics obtained are not the important fact, but the possible decrease in error if different subsets are trained.

Another important clue is found in the spread of the distribution. In the 0% purity case, where the model can not be consistently better or worse depending on the outcome of the sampling, the variance of the distribution is the smallest. Although no strict test has been developed to draw formal conclusions, Fig. 4.3 can serve as a guide in drawing preliminary hypotheses about the data at hand. The two factors to analyse when computing the histograms of Fig. 4.3 are the mean and spread of the distribution. Lower means and higher spreads pointing towards high total purity cases, while the opposite characteristics being an indicative of low total purity.

## 4.2 Optimization Procedure

To evaluate the performance of the P-Craw framework, the simulations presented in the last section were used together with their reference signal. Simulations with different total purity were used in order to have a broader picture of the capabilities and limitations to each configuration. On a normal evaluation of a optimization procedure, two quantities are of interest: the best and mean fitness value of the population for each iteration. In this section, two metrics will be added. The first is the purity for the best solution at each generation (again, not to be misinterpreted as the total purity) and the second is the relative size of the best solution.

The relative size is not in respect to the whole dataset, but only to the predictable periods present in the simulation, being different from the quantity $\frac{|\mathbb{P}|}{|\mathbb{D}|}$ . This way, if a simulation possesses 2000 observations but only 600 predictable ones and a given subset has size 300, the relative size is 0.5. This makes it possible to identify the optimal subset in any situation as the one with purity and relative size of 1, regardless of the total purity and size of the simulation. The relative size can be greater than 1, for example, if the subset is greater than the size of the inefficient time-frames, but this is not desired.

The purity and relative size metrics relate with the exploitation and exploration properties of the algorithms. The purity metric at each generation convey how effectively the method was when exploiting the signal found from the predictable samples, but a high purity might be found in a relatively small portion of the data. The relative size is a metric of how extensively the algorithm explored the possible subsets. For example, if the purity is close to 1 but the relative size is significantly bellow 1, the procedure was able to achieve a predictable solution, but not to sufficiently explore the possible subsets.

In the subsequent comparisons, every configuration will run 30 times, and best fitness value, mean fitness value, purity and relative size will be compared. Not only that, but their final distribution among runs and their evolution throughout every generation will be displayed when convenient.

Each optimization algorithm will start at an specific configuration, described at the beginning of the tests. At each comparison, a different part of the configuration will be studied and changed if it is beneficial to the method. When any parameter is altered, the others remain as the previous stage unless explicitly said otherwise.

### 4.2.1 Particle Swarm Optimization

For the beginning of simulations, the PSO setup uses the regular Particle Swarm Optimization with the random initialization and 100 particles running for 100 iterations. As mentioned in Chapter 3, It uses the update equation as in Eq. 4-5, with values $w = 0.5$, $c_p = .5$ and $c_g = 1.0$.

$$v_{t+1} = wv_t + U(0, c_p)(p_t - x_t) + U(0, c_g)(l_t - x_t) \tag{4-5}$$

In the random initialization the position of each dimension in each particle is initialized by a sampling of a uniform distribution between 0.0 and 0.4, while the velocities are sampled uniformly between -1 and 1. The minimum number of samples in both the $F_{exploit}$ and $F_{explore}$ functions is set to 200.

### 4.2.2 Perturbed PSO vs Regular PSO

The first major comparison will be between the Perturbed Version of the PSO and the regular one. Fig. 4.4 displays the outcome for the simulation. As is easily noted, the perturbed version was superior in every metric. Besides, it also showed a smaller spread between the paths of each run, specially in the purity and relative sizes curve.

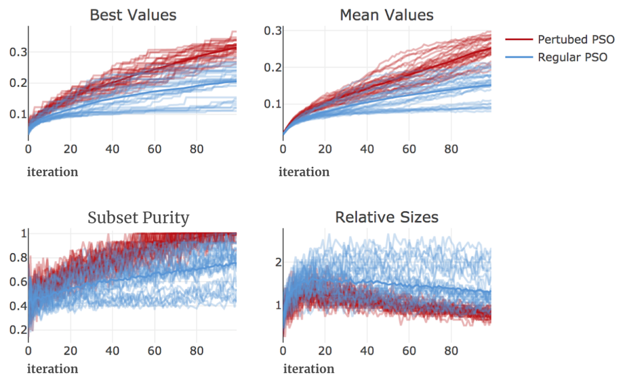


Figure 4.4: PSO Evolution - Perturbed vs Regular

This can be a consequence of the "sticky edges" property of this implementation, which helps good or bad intervals to be included or discarded with greater consistency in each generation. This version also seems to find better values for the score function more easily, establishing superiority not only in the exploitation of good solution, but also in exploration of good candidates. The Perturbed version was, in average, able to output a subset with 99% of predictable samples, while gathering roughly 80% of all such samples available. The final results are displayed in table 4.1.

| Configuration | Avg Mean Value | Avg Best Value | Avg Purity | Avg Relative Size |
|---|---|---|---|---|
| Pertubed PSO | 0.25 | 0.31 | 0.99 | 0.8 |
| Regular PSO | 0.15 | 0.21 | 0.76 | 1.34 |

Table 4.1: PSO - Perturbed vs Regular

As the proposed modification in the PSO algorithm showed to improve the results, the Perturbed PSO is elected the preferred version and will be used in the following benchmarks.

### 4.2.3 Initialization

The initialization scheme for the algorithm is studied next. With the simulation setup held fixed, group initialization is performed and the results are displayed in Fig 4.5.

Looking at the Best Values graph, the group initialization gets an average a better solution at the first rounds. The problem, it seems, is that this initial confidence triggers the exploitation of those solutions too early on, leading to diverging results in each run, as it is made clear by the purity and relative size progressions. Clusters of different paths can be noticed, referring to trials where the algorithm bet on the larger intervals and trials where small ones were further investigated. As previously discussed in subsection 4.1.2, it seems that getting this choice wrong has serious consequences and is difficult to correct later.

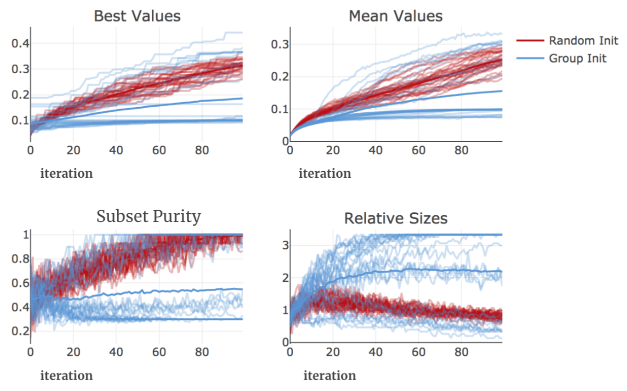


Figure 4.5: PSO Evolution - Random vs Group Initialization

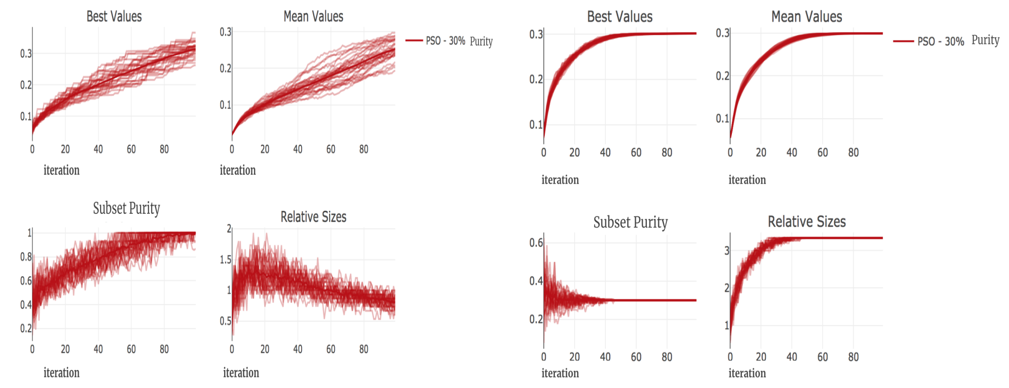
| Configuration | Avg Mean Value | Avg Best Value | Avg Purity | Avg Relative Size |
|---|---|---|---|---|
| Random Init | 0.25 | 0.31 | 0.99 | 0.8 |
| Group Init | 0.16 | 0.19 | 0.55 | 2.18 |

Table 4.2: PSO - Random vs Group Initialization

The random initialization seems to be more reasonable in the PSO case, displaying results not only more well-behaved but significantly better overall, as it can be seen in Table 4.2. This technique is then chosen and used on subsequent comparisons.

#### 4.2.3.1 Total Purity Variation and Fitness Function

The two possible fitness functions are used in the PSO context and the results are discussed. Fig 4.6 displays the comparison at the 30% scenario. The $F_{explore}$ version of the optimization objective lead the algorithm to pick the whole set of possible samples. In the PSO case, this tendency is very consistent, as it can be seen by the low dispersion between runs in the evolution of all the metrics. The $F_{exploit}$ version on the other hand showed very good results, being able to conquer a high purity and a relative size close to one.

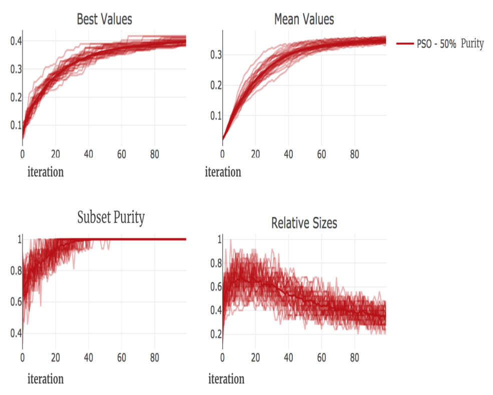


(a) $F_{exploit}$ (b) $F_{explore}$

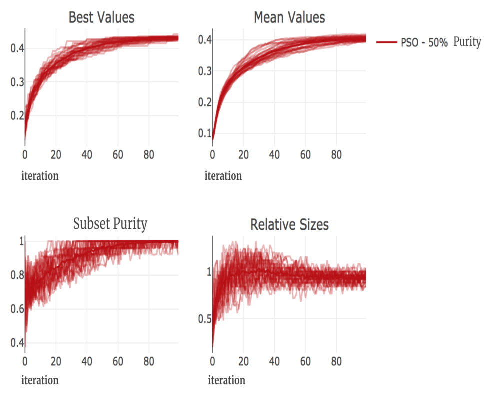
Figure 4.6: 30% Purity

It is interesting to note the evolution of those metrics in Fig 4.6 (a). While the purity starts at a lower value and steadily increases, the relative size first goes through a peak, to then slowly decrease towards the 1.0 mark. This reveals a specific path taken by the algorithm. It first try larger subsets, to then steadily filter noisy observations and reduce the size of the candidates, converging towards the optimal choice.

When the simulation is created in a 50% split between predictable and unpredictable regimes, the results are quite different. This time, the $F_{explore}$ version was able to find nearly all the AR1 samples, maintaining a small number of random innovations on the chosen subsets. The function $F_{exploit}$ on the other side was efficient about choosing valid samples but found only a fraction of those samples.



(a) $F_{exploit}$



(b) $F_{explore}$

Figure 4.7: 50% Purity

Tables 4.3 and 4.4 consolidate the results for both the $F_{exploit}$ and $F_{explore}$ fitness functions when evaluated at different purity.

| Configuration | Avg Mean Value | Avg Best Value | Avg Purity | Avg Relative Size |
|---|---|---|---|---|
| 20% Total Purity | 0.1 | 0.1 | 0.2 | 5 |
| 30% Total Purity | 0.25 | 0.31 | 0.99 | 0.8 |
| 40% Total Purity | 0.38 | 0.43 | 1 | 0.43 |
| 50% Total Purity | 0.35 | 0.4 | 1 | 0.35 |
| 70% Total Purity | 0.41 | 0.46 | 1 | 0.31 |
| 90% Total Purity | 0.38 | 0.42 | 1 | 0.35 |
| 100% Total Purity | 0.42 | 0.46 | 1 | 0.32 |

Table 4.3: PSO - $F_{exploit}$ Function

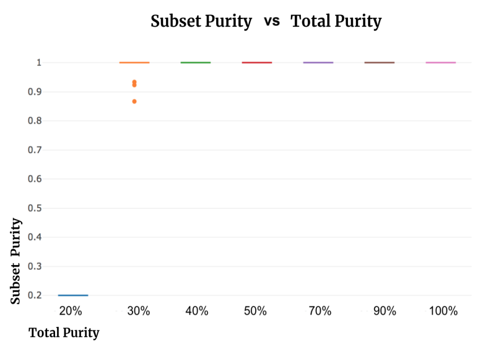
| Configuration | Avg Mean Value | Avg Best Value | Avg Purity | Avg Relative Size |
|---|---|---|---|---|
| 20% Total Purity | 0.3 | 0.3 | 0.2 | 5 |
| 30% Total Purity | 0.3 | 0.3 | 0.3 | 3.33 |
| 40% Total Purity | 0.35 | 0.39 | 0.91 | 0.99 |
| 50% Total Purity | 0.4 | 0.43 | 0.99 | 0.94 |
| 70% Total Purity | 0.5 | 0.51 | 1 | 0.98 |
| 90% Total Purity | 0.55 | 0.56 | 1 | 0.97 |
| 100% Total Purity | 0.61 | 0.62 | 1 | 0.99 |

Table 4.4: PSO - $F_{explore}$ Function

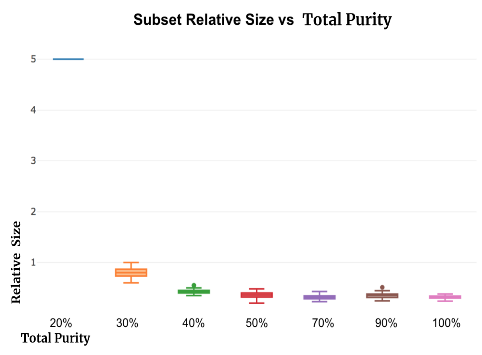
The PSO showed to be very effective for scenarios where the predictable regime prevailed. For total purity of 40% or greater, the Particle Swarm together with the $F_{explore}$ fitness function were able to correctly classify nearly every sample on the simulations.

Figs. 4.8 and 4.9 displays the box-plots for subset purity and relative size at each total purity configuration for functions $F_{exploit}$ and $F_{explore}$ respectively. For the $F_{explore}$ function, it is evidenced the consistency of results for total purity above 40%. The dispersion for subsets purity is very low, with all results being concentrated near 1, with only the 40% presenting a spread of values decreasing at most to the the 0.70 mark. The situation is the same for the relative size, with all results above the 40% total purity case highly clustered near 1, with a greater but tolerable dispersion in the 40% scenario.

On the other side, the algorithm performed poorly on the low purity regimes. Even with the $F_{exploit}$ loss function, the method chooses the whole dataset for a 20% scenario. The $F_{explore}$ function is chosen as a more appropriate fit for the PSO, specializing the method in high total purity situations.

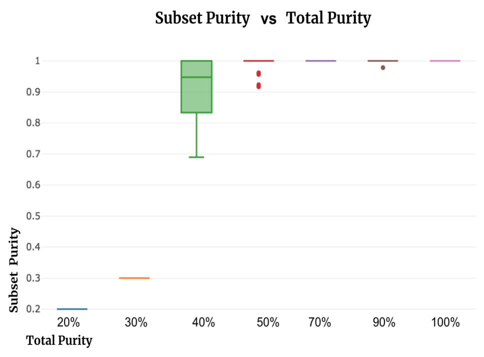


(a) Purity

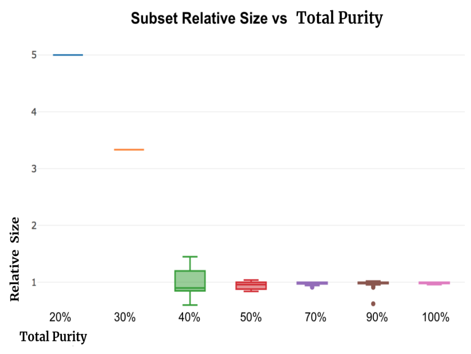


(b) Relative Sizes

Figure 4.8: PSO - $F_{exploit}$ Fitness Function Box-Plots



(a) Purity



(b) Relative Sizes

Figure 4.9: PSO - $F_{explore}$ Fitness Function Box-Plots

### 4.2.4 Genetic Algorithm (GA)

The Genetic Algorithm comparison starts with the regular GA implementation with Single-Point crossover, group aware initialization and 100 genes running for 100 generations. The selection mechanism is the tournament selection with a size of 2, the crossover and mutation probabilities are fixed during the evolution, and are 80% and 10% respectively. When a chromosome is chosen for mutation, every gene has 10% chance of being flipped. The high rates used for mutation were chosen to enhance the exploration properties of the GA. The best individual is kept at each generation, in a elitist fashion. As previously mentioned, the group aware initialization creates each solution sequentially, and the genes set as 1 in the previous chromosomes are removed from the possible genes to be set as 1 in the next ones. The minimum number of samples in both the $F_{exploit}$ and $F_{explore}$ functions is set to 200. Again, when any parameter is changed, the others remain as they previously were unless explicitly said otherwise.

#### 4.2.4.1 KNN-GA vs Regular GA

The first test performed concerns the KNN variant described in the previous section. To evaluate the modification proposed, a simulation with 30% total purity was employed, and the $F_{exploit}$ function was used. The regular GA implementation was tested against a version with K=5 of the nearest neighbors' approach. The number of neighbors was chosen after some simple initial tests, testing values between 1 and 12. The evolution of both algorithms is displayed in Fig. 4.10, with the results compiled in Table 4.5. In order to compare corresponding metrics, the Best Values and Mean Values displayed are not the KNN ones, but rather the original scores before the neighborhood average.

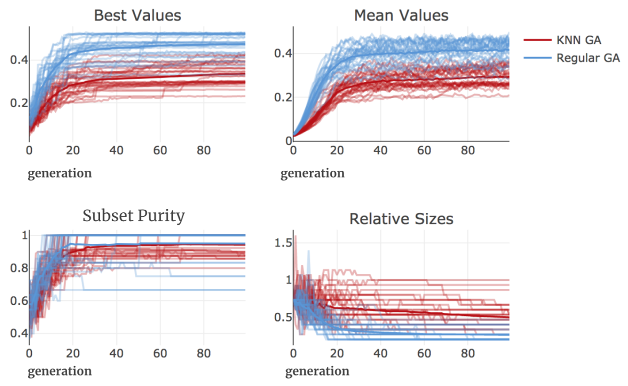


Figure 4.10: GA Evolution - KNN vs Regular

| Configuration | Avg Mean Value | Avg Best Value | Avg Purity | Avg Relative Size |
|---|---|---|---|---|
| KNN GA | 0.29 | 0.34 | 0.94 | 0.5 |
| Regular GA | 0.42 | 0.47 | 0.95 | 0.26 |

Table 4.5: Evolutionary Algorithm - KNN vs Regular Table

The first interesting observations concerns the values obtained for the fitness function in both versions. Even though the regular version was able to obtain higher scores, this advantage did not guarantee better results in the chosen subsets. On the contrary, the Nearest Neighbors version was able to cover roughly twice the number of correct samples (0.5 relative size vs 0.25) while keeping the same purity as the regular one (0.94 purity vs 0.95).

This result confirms that the filtering proportioned by the neighborhood averaging approach helps the algorithm to search for regions with a better overall structure rather than optimizing for spurious high values. It can be recalled from Subsection 4.1.2 that in regions with high purity and low size, the fitness function was prone to have random high values, what could trap an optimization strategy in not looking for larger solutions.

As the KNN-GA shows clear advantage in the comparison, it is chosen as the GA version to be used in the next tests. The proposed variation successfully achieved the desired enhancements for which it was designed.

#### 4.2.4.2 Initialization

For the 40% total purity, the initialization scheme was changed in order to see how much that choice could impact the final result. The resulting figure (4.11) doesn't show a significant change in the evolution of metrics, and so no clear winner is found. The group aware initialization is chosen on a tight margin.

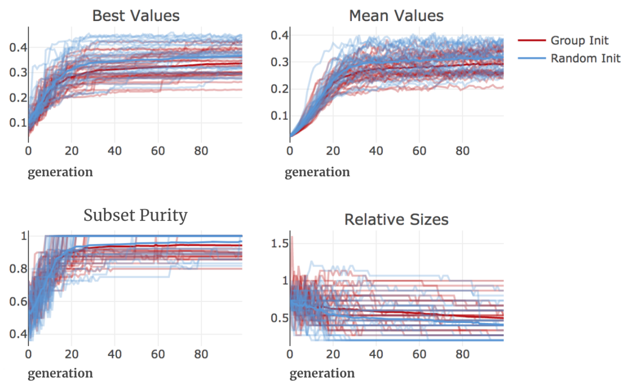
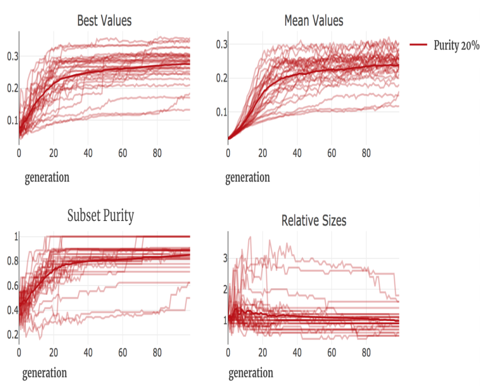


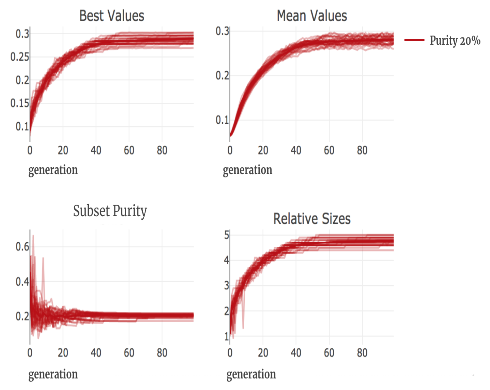
Figure 4.11: GA Evolution - Random vs Group Aware Initialization

#### 4.2.4.3 Total Purity Variation and Fitness Function

Next, the algorithm was tested using both $F_{exploit}$ and $F_{explore}$ as fitness functions, and in simulations with varying total purity. As in the GA case, the variation on this number is done to ensure the method is probed in all possible scenarios regarding the "contamination" of the data with unpredictable samples.

Fig. 4.12 displays the evolution of the 4 metrics in each one of the 30 runs for a dataset with 20% of predictable samples for both $F_{exploit}$ and $F_{explore}$. The alarming fact is the poor performance of $F_{explore}$. With 20% total purity, choosing a subset with relative size 5 means choosing the whole available data. Obviously, this solution has a subset purity of 20%, the same as the total purity. Function $F_{explore}$ consistently fails in performing any filtering them, outputting roughly the whole data in each single run. This behavior was conjectured in subsection 4.1.2 and depicts the choice the algorithm has to make in low purity scenarios about going for large or small subsets. On the other hand though, the $F_{exploit}$ function had a robust behavior, being able to close every run with a relative size reasonably close to one while maintaining a good purity.

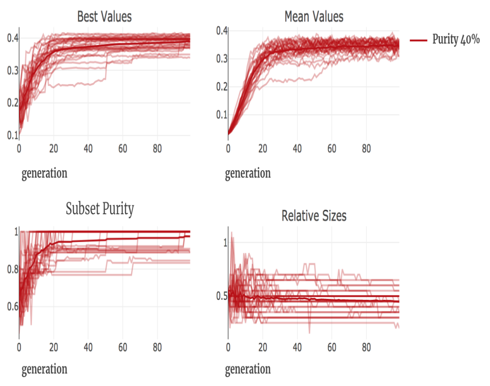


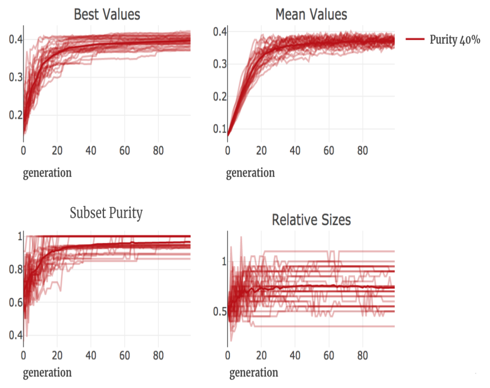


(a) $F_{exploit}$ (b) $F_{explore}$

Figure 4.12: 20% Total Purity

Fig. 4.13 shows the results for the 40% purity dataset case. Both functions, in this case, maintain a high portion of predictable samples in their selected subsets. However, in this case, $F_{explore}$ manages to accomplish this high purity while acquiring a larger amount of such samples, as it can be seen by the relative size curves.





(a) $F_{exploit}$ (b) $F_{explore}$

Figure 4.13: 40% Total Purity

Tables 4.6 and 4.7 display the consolidated results for different total purity for $F_{exploit}$ and $F_{explore}$ respectively. As can be seen, function $F_{exploit}$ have a better performance in lower purity, while $F_{explore}$ is more capable of exploring scenarios with a greater portion of predictable samples. Even so, the performance at those scenarios is not satisfactory even for $F_{explore}$. The algorithm is able to use less than half of available samples in a 90% total purity scenario, for example. Function $F_{exploit}$, on the other hand, gets stable results at total purity as low as 20%. Figures 4.14 and 4.15 shows the Box Plots for relative size and total purity at each fitness function.

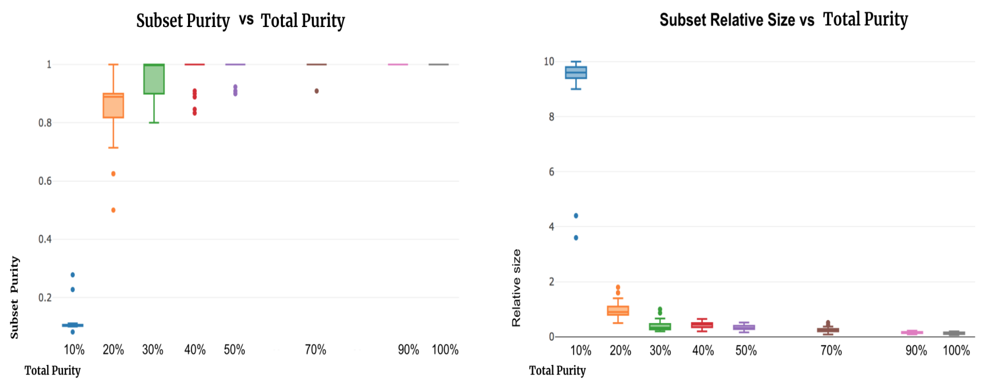
| Configuration | Avg Mean Value | Avg Best Value | Avg Purity | Avg Relative Size |
|---|---|---|---|---|
| 10% Total Purity | 0.09 | 0.09 | 0.11 | 9.22 |
| 20% Total Purity | 0.24 | 0.27 | 0.85 | 0.98 |
| 30% Total Purity | 0.4 | 0.45 | 0.96 | 0.4 |
| 40% Total Purity | 0.34 | 0.39 | 0.97 | 0.45 |
| 50% Total Purity | 0.37 | 0.41 | 0.98 | 0.33 |
| 70% Total Purity | 0.39 | 0.42 | 0.99 | 0.25 |
| 90% Total Purity | 0.44 | 0.47 | 1 | 0.16 |
| 100% Total Purity | 0.47 | 0.49 | 1 | 0.13 |

Table 4.6: Evolutionary Algorithm - $F_{exploit}$ table

| Configuration | Avg Mean Value | Avg Best Value | Avg Purity | Avg Relative Size |
|---|---|---|---|---|
| 20% Total Purity | 0.28 | 0.29 | 0.2 | 4.77 |
| 30% Total Purity | 0.37 | 0.4 | 0.93 | 0.72 |
| 40% Total Purity | 0.37 | 0.4 | 0.97 | 0.747 |
| 50% Total Purity | 0.39 | 0.41 | 0.98 | 0.68 |
| 70% Total Purity | 0.44 | 0.46 | 0.99 | 0.67 |
| 90% Total Purity | 0.53 | 0.54 | 1 | 0.49 |
| 100% Total Purity | 0.56 | 0.57 | 1 | 0.91 |

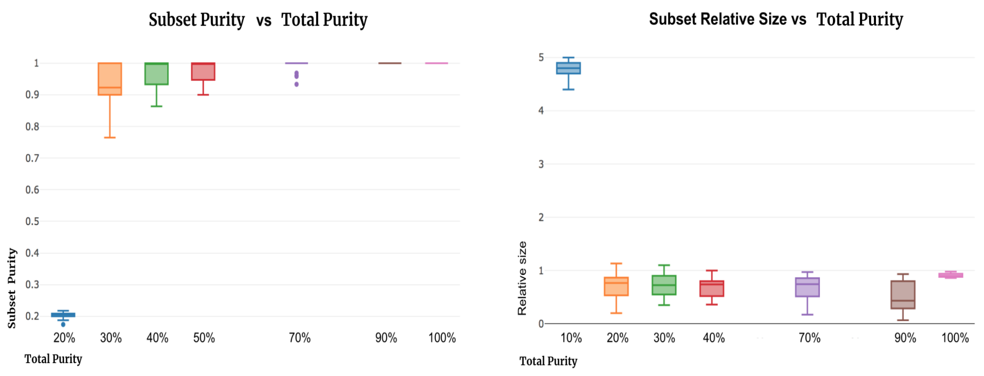
Table 4.7: Evolutionary Algorithm - $F_{explore}$ table

As in the PSO, function $F_{explore}$ showed overall better results. Even so, the KNN-GA algorithm when used with $F_{explore}$ doesn't bring any improvement when compared to the Perturbed PSO. On the other hand, the KNN-GA coupled with $F_{exploit}$ is capable of probing datasets with only 20% and 30% of predictable samples without diverging towards picking the whole aviable data. Function $F_{exploit}$ is chosen as a more suitable match for the KNN-GA. Although function $F_{explore}$ has a overall performance at higher total purity scenarios, neither one of the versions is satisfactory in that case. Version $F_{exploit}$ is then chosen for the GA setup to specialize in the low total purity cases, and will be used then on the next evaluations.



(a) Subsets Purity (b) Relative Sizes

Figure 4.14: Genetic Algorithm - $F_{exploit}$ BoxPlots



(a) Subsets Purity (b) Relative Sizes

Figure 4.15: Genetic Algorithm - $F_{explore}$ BoxPlots

### 4.2.4.4 Restarting Mechanism & Minimum Sample Size

The previous results were obtained using a Single-Point crossover scheme and a minimum acceptable sample size in the fitness function of 200 samples. In the context of the algorithm being suitable to low total purity scenarios, a question arises. How would the algorithm react if the minimum number of samples required by the fitness functions was to be changed? In order to answer that question, the next comparisons on the Genetic Algorithm setup are made also varying this number.

To test the Restarting Mechanism, a minimum number of 400 samples is used in the Fitness Functions with the 30% total purity scenario (where 600 of 2000 values innovations follow the AR1 structure). The restart happens at the 51º generation, near the middle of the evolution process. Fig. 4.16 shows the evolution of the metrics.

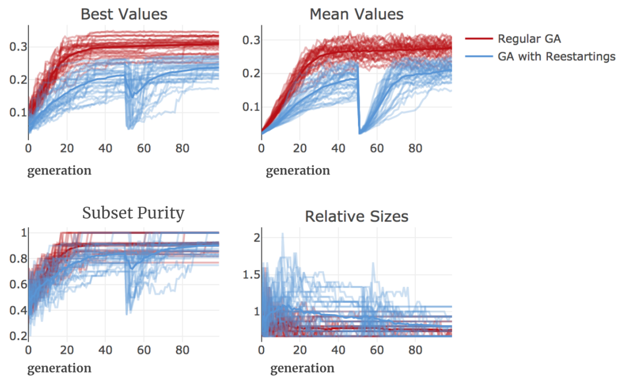


Figure 4.16: GA Evolution - Restarting Mechanism

In a regular scenario, the restarting of the population would affect only the mean values of the generation, as the best score is kept. However, the highest nearest neighbor score is the one being preserved, while the unfiltered fitness score is the one being displayed. The value kept at the restarting is the best after the KNN smoothing, which is not necessarily the best fitness in the population. This can cause the Best Value curve to drop, as it can be seen in Fig. 4.16.

| Configuration | Avg Mean Value | Avg Best Value | Avg Purity | Avg Relative Size |
|---|---|---|---|---|
| Regular GA | 0.27 | 0.31 | 0.92 | 0.75 |
| GA with Reestartings | 0.21 | 0.24 | 0.9 | 0.8 |

Table 4.8: Restarting Mechanism

The results don't show a clear winner. The restating brought an average small increase in relative size with an also small decrease in purity. Without enough evidence to be adopted, the restarting mechanism is discarded, and the regular GA proceeds as the preferred implementation.

Concerning the change in the minimum number of samples required, the algorithm seems to have responded well. The final relative sizes acquired by the simulations raised significantly, from 0.4 in the previous comparison to 0.75, with a small decrease in the subset purity from 0.96 to 0.92 . The average Relative Size increased as expected, but without significantly compromising the purity of the selected subsets, which is the best result that could be expected.

### 4.2.4.5
### Crossover Mechanism & Minimum Sample Size

To further stress the behaviour of the GA regarding the minimum size acceptable for subsets this number is raised to 1000 in the next comparison. The test concerns the crossover mechanism. So far, the Single-Point crossover has been used, but the Two-Point version was also implemented.

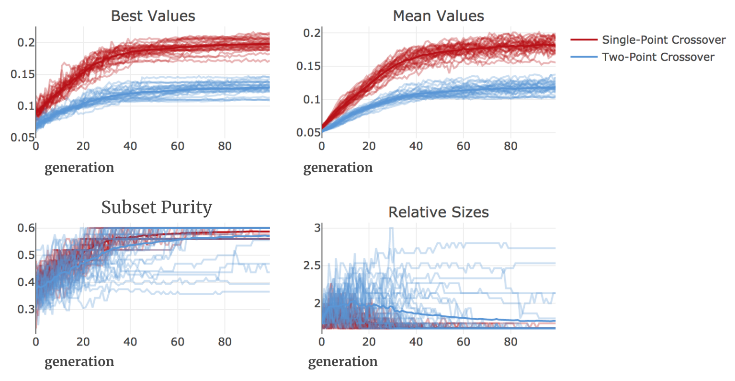


Figure 4.17: GA Evolution - Crossover Mechanism

Both variations are tested and compared. Despite the fact the the results displayed in Table 4.9 are not significantly different at first sight, the visual inspection at Fig. 4.17 shows that the two-point crossover imposes a much more unpredictable outcome, having good results in some runs and bad outcomes on others.

| Configuration | Avg Mean Value | Avg Best Value | Avg Purity | Avg Relative Size |
|---|---|---|---|---|
| Single-Point Crossover | 0.18 | 0.2 | 0.59 | 1.67 |
| Two-Point Crossover | 0.12 | 0.13 | 0.57 | 1.76 |

Table 4.9: Crossover Mechanism

This unpredictable behavior in the Double-Point technique might be explained by the nature of the mechanism. When compared to the Single-Point variation it tends to change a smaller portion of the parent genes when creating the offspring, which in turn favors exploitation over exploration in the algorithm. The algorithm then is more prone to exploit interesting areas too early on in the evolution, leading to a pattern where the final result changes drastically depending on the initial conditions. The Single-Point crossover is thus preferred in this case.

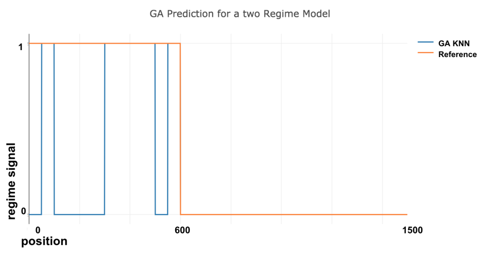
At the same time, once again the minimum numbers of samples imposed seemed to pose the least damage possible. With 1000 minimum samples and only 600 predictable, the best case scenario would be to choose the minimum number while gathering all of the predictable ones. This would mean a solution of relative size $\frac{1000}{600} = 1.66$ (1000 minimum samples chosen over 600 predictable samples). At the same time, the corresponding purity would be the number of predictable samples in the subset chosen (at max 600) over the chosen subset size 1000, namely $\frac{600}{1000} = 0.6$. This means that at the used 30% purity simulation, the 1000 sample restriction could at the very best have numbers close to the displayed in Table 4.9.

The Genetic Algorithm can them be concluded to show stable results when used with the $F_{exploit}$ fitness function on datasets with a low purity configuration.

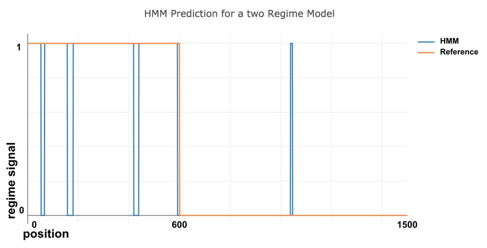
#### 4.2.4.6
#### Hidden Markov Model (HMM) Benchmark

To help illustrate the potential of the developed tool, it is compared to the classical regime switching model of the Hidden Markov Model. For this purpose, a simulation containing 600 AR1 samples followed by 900 noisy innovations is created. The algorithm is trained once on the simulation with the final GA algorithm and $F_{exploit}$ function, and the classification of the intervals is compared with the reference. The same is done using a Gaussian Hidden Markov Model (HMM) of two states, like the one used in [26].

The HMM solution does considerably better, as depicted in Fig. 4.18. If this simple setup was to be expected in a real scenario, there would be no reason not to use the HMM version. To slightly complicate the challenge, instead of 900 samples of Gaussian with null mean and variance 2, the random regime is composed of 700 samples like that followed by more 700 of the same type but with variance 6. To be fair with the HMM competitor, it is configured to be trained for 3 regimes, while the GA suffers no change.



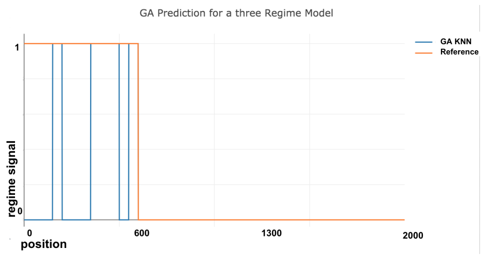
(a) Final GA in Simulation



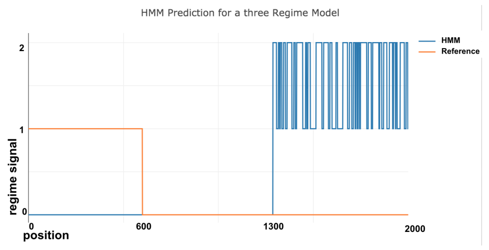
(b) HMM in Simulation

Figure 4.18: Simulation Comparisons

Fig. 4.19 display the results. Even with the unrealistic knowledge of the number of regimes, and with all those regimes unrealistically following simple Gaussian innovations processes, the rise in the number of possible choices and in the simulation size alone were sufficient to largely degrade the performance of the Hidden Markov algorithm. At the same time, the performance of the GA was totally unaffected. This example does not fully explore the potential of the new approach. Although the Hidden Markov Model is modelled with Gaussian innovations in mind, the P-Crawl does not make assumptions on the structure of the data, and thus is less likely to have its performance degraded should the data follow a more complicated dynamic.



(a) Final GA in Second Simulation



(b) HMM in Second Simulation

Figure 4.19: Simulation Comparisons

### 4.2.5
### Summary

In the previous sections, both the GA and PSO were studied in their use as optimization procedures in the P-Crawl framework. The comparisons shows a complementary relationship between the two methods. The GA is coupled with the $F_{exploit}$ fitness function in order to become an appropiate algorithm for datasets with a low total purity. The results displayed a good performance of that setup in scenarios ranging from 20% to 40% total purity.

The PSO algorithm displayed a more consistent exploration of the subset's space. Although this exploration revealed to be dangerous in low total purity scenarios, where it could lead the algorithm to pick the whole dataset, it performed very well in datasets ranging from 40% total purity to 100%.

Together, both algorithms cover ranges from 20% to 100% of predictable intervals. The difficulty is then to detect which one to use, since the total purity is not an available information in a real situation. The final discussion in Subsection 4.1.2 can help to elucidate this matter, but does not define a clear decision rule.

In every scenario where the PSO algorithm failed to search the optimal subset, it diverged toward the whole dataset. This fact can be used to create a simple rule of thumb. Test the PSO first, and visualize the output. If it is a smaller subset than the whole dataset, use it. If not, use both the PSO and the GA and see which one gives better results in the test phase of the framework.

## 4.3 Real Case Study - Brazilian Stock Market Intraday Data

### 4.3.1 Dataset

In the real case study, the dataset used was from the Brazilian Stock Market (Bovespa). The data was obtained from the official FTP site (ftp://ftp.bmf.com.br/MarketData/) with help from the R package GetHF-Data [42]. The files consist of trade information from every transaction in the Limit Order Book from 2018-07-02 to 2019-03-3, compromising 8 months of data. Each day of transactions in the Bovespa corresponds to a file of in average 300MB, so the whole information available represents a disk space of nearly 50GB. To narrow down the scope of the data the movement of trade orders was analyzed on the first day available, and the 20 stocks with the most trading volume for this day were chosen. Fig. 4.20 shows the number of trades for each one of those, as well as their corresponding symbol.

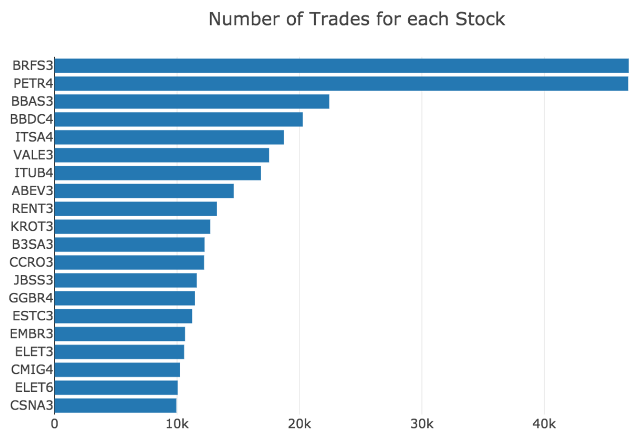


Figure 4.20: 20 Most Traded Stocks

The trading data represents each time a specific stock was traded, at which price, and at what volume (along with other information). This type of record can be aggregated in different time scales. To the current study, those aggregations were done for each stock and each day in the 5-minute, 60-minute and daily windows. The time series of interest is the closing price at the end of each 5-minute window interval for a particular stock. Once the trades are aggregated in this time frame, the resulting series has approximately 9.300 observations for each stock. The next step is to define a feature vector capable of summarizing enough information about each moment in the stock dynamics. For this purpose, a number of technical indicators were studied and computed for the series.

One of the key aspects of the chartists' theory in the market concerns levels of support and resistance for the price. Those levels are defined roughly as the last locals minima and maxima of the traded values, and are believed to contain information concerning the current market belief about those stocks. If a price approaches a support region, wherein the recent past represented the lowest price at which buyers were willing to acquire that paper, the stock will be tested to see if this belief (that the asset is worth more than that quantity) still hold amongst participants. If the test fails and the price "breaks the support", it could be an indicator that the stock is losing value, and it might be a good time to sell. The strategy of keeping track of support and resistance levels falls in a technical concept referred to as Price Action.

In chartist theory, there are other types of strategy and indicators. Two important categories are the Trend-following and Reversal types. Trend Following indicators try to detect a emerging trend in price levels, and operate in its favor before it is consolidated by the rest of the market. An example of this type of strategy would be the moving average indicators discussed earlier.

Reversal strategies, on the other hand, try to identify moments where a trend will shift. One example of those indicators are the Boillinger Bands for the given stock. Those bands are drawn according to a rolling estimate of the standard deviation of price innovations, and are a key factor in many Technical Analysis strategies. They are used to operate a mean-reversion strategy, where every time the price gets above the upper band or below the lower one, it is believed to proceed to the contrary band, sometimes stopping at the middle level instead. Fig. 4.21 displays the bands for a fraction of the ITS4A five-minute price series as well as support and resistance levels.

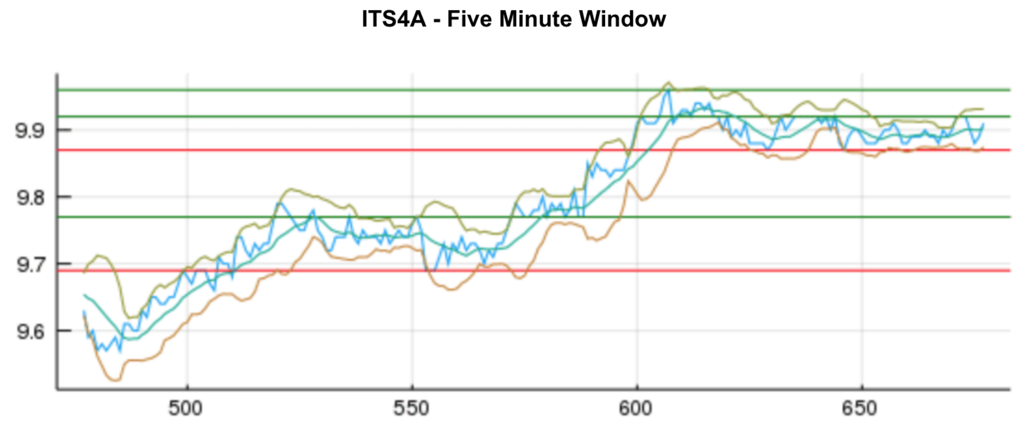


Figure 4.21: Supports, Resistances, and Boillinger Bands

Yet another important category for technical metrics is that of the momentum indicators. The indicators at this category try to grasp the strength of market movements. An example would be the volumes of stocks traded. If the volume of transactions for a certain stock starts to increase, it may indicate a sudden interest of the market participants for that particular stock, pointing towards the incorporation of new information. This can help in the early identification of trends, for instance.

Table 4.10 compiles all the indicators used in the construction of the feature matrix. Unless explicitly written, those metrics were all computed in reference to the 5-minute interval windows. The absolute values for Boillinger Bands, Support and Resistance lines, and Moving Averages were not used, as the level of those values is not as interesting as their value with respect to their price. Instead, the quantities used were the log returns of those indicators in respect to the price series. This way, a small value with respect to the last support value means that the series is close to that support line. The Pivot Points mentioned are yet another technical indicator, computed using the values for the closing price, maximum price and minimum price registered in the last trading day. Instead of the simple moving average discussed earlier, a Exponential Moving Average (EMA) is used. This form o moving average uses a weight decay that assigns greater importance to more recent values. The angular coefficient used is computed in respect of the last 120 values in the 60-minute series. The target for each row is the next log return for the price series after 30 minutes, or a horizon of 6 observations in the 5-minutes price series. This was chosen upon the hypothesis that big movements might be more behaved to predict than small 5-minute oscillations, and may as well lead to bigger profits for a trader.

| Feature | Function |
|---|---|
| Last 9 Absolute Log Returns | Heteroscedasticity of Returns |
| Last 3 Log Returns to the 3 last Support levels | Price Action Indicators |
| Last 3 Log Returns to the 3 last Resistance levels | Price Action Indicators |
| Pivot Points | Price Action Indicators |
| Last 3 Log Returns to the Upper Boillinger Band | Reversal Indicators |
| Last 3 Log Returns to the Lower Boillinger Band | Reversal Indicators |
| Last 3 Log Returns to the Mid Boillinger Band | Reversal Indicators |
| Last 3 Volumes of Stock Traded in each interval | Momentum Indicators |
| Angular Coefficient from the 60-Minute Series | Trend Following Indicators |
| Last 3 values of the MACD Technical Indicator | Momentum Indicators |
| Last 3 values of the RSI Technical Indicator | Momentum Indicators |
| Last 3 Log Returns to the VWAP | Trend Following Indicators |
| Last 3 Log Returns to the EMA of 9 periods | Trend Following Indicators |
| Last 3 Log Returns to the EMA of 27 periods | Trend Following Indicators |
| Last 3 Log Returns to the EMA of 200 period | Trend Following Indicators |

Table 4.10: Feature Vector

As feature selection is a crucial step in machine learning procedures, the results in the Experiments section will be highly influenced by the chosen representation summarized in Table 4.10. This specific representation was chosen in a heuristic manner, aiming to encompass an exhaustive set of technical principles. Other feature sets can be evaluated in future works.

### 4.3.2 Experiments

#### 4.3.2.1 BRFS3 Stock Prices

Initially, the stock with greater volume at the first date available is chosen for exploration. This stock is the BRFS3, and represents a large brazilian company in the food sector. The first choice to be made in order to apply the P-Crawl is regarding the regression model used in the fitness function evaluation. As a non-linear relationship among the elements of the feature vector is expected, the chosen model must be able to learn such structure. At the same time, the choice needs to take into consideration the training time, as each interaction of the optimization mechanism requires training in a hundred different subsets. With those restrictions in mind, the Boosted Trees implementation of [41] is used. In order to make the learning of the model fast during the optimization phase and avoid a higher variance to the possible outcomes, it is restricted to train only up to the first 10 trees. The parameters of the model are the max depth of the regressors, the shrinking factor, and the amount of sub-sampling to be done when training each new tree. Those parameters are chosen in a grid-search, using cross validation to assert the best combination.

The final series has a total of 9.543 observations. Unfortunately, the proposed model has limitations about how to find predictable samples in such a large number while keeping a granularity on the group size. If the usual imposition of a group size of 2% of the total was imposed on big samples, the ability to detect spikes of predictability or noisy behavior would be degraded. A better solution in those cases could be employing the optimization in moving windows, but here this complexity was avoided. In order to mimic the simulation setup, only the first 2000 observations are probed for predictable periods. Fig. 4.22 displays the series used for training.

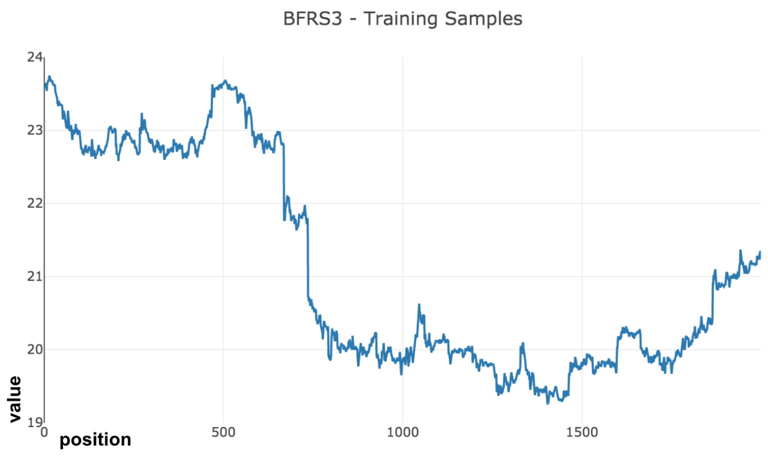


Figure 4.22: 5 Min Price Series - Training Samples

Before actually scanning the interval, a choice must be made though. In the simulation part of the Result Section two models were chosen, one was the Perturbed PSO coupled with the $F_{explore}$ fitness functions for high purity scenarios, and the other was the KNN-GA using $F_{exploit}$ for datasets with a low proportion of predictable regimes. To further study which one to choose, the procedure proposed in the fitness function simulation is repeated with the data at hand. Half the available samples for training are repeatedly drawn and the cross-validated MASE of the boosted trees is computed for each trial. The histogram of those values is displayed at Fig. 4.23.

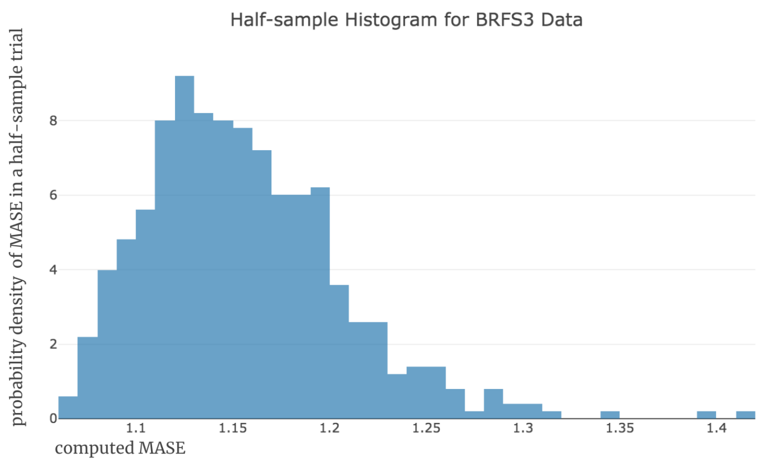
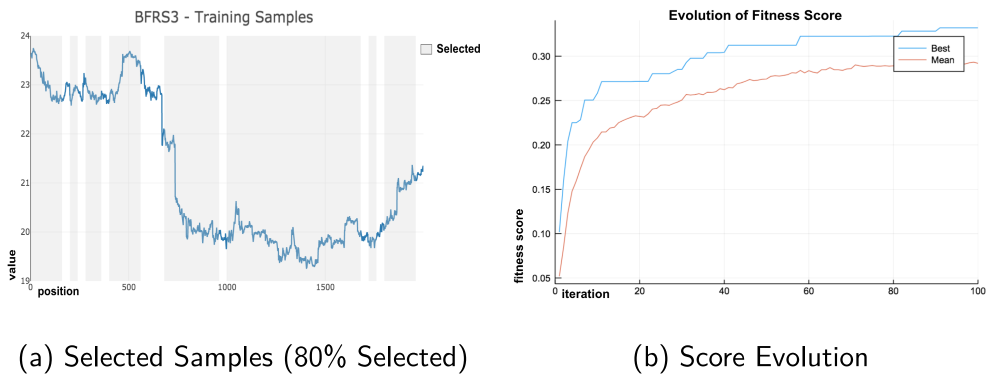


Figure 4.23: Half-Sample MASE Histogram - Boosted Trees

As expected, the pattern displayed does not directly link to any one of the examples in the fitness function simulation section. This is because financial price series are notoriously hard to predict, and so it would be unwise to expect MASE values much lesser than 1. On the other hand, the histogram displays another characteristic mentioned in the same context; the high variance of the distribution. The value of the error metric is revealed to depend largely on the chosen subset, a characteristic showed by the simulations with greater purity. Nevertheless, this inspection is not formal, and further methods are required to decide how to proceed. The rule of thumb proposed in Subsection 4.1.2, and the Perturbed PSO with the $F_{explore}$ function is employed first at the data.

When the Perturbed PSO is used the results obtained are that of Fig. 4.24. The final signal covers 80% of the available dataset, a configuration linked to good results in the simulation studies. At first sight then, the model seems to be a consistent choice.



(a) Selected Samples (80% Selected)

(b) Score Evolution

Figure 4.24: PSO Algorithm used in the BRFS3 Training Set

To further assert this conclusion, another comparison is employed. The whole point of the optimization routine is to find a subset with a better cross-validated error metric. The obvious sanity check is thus to evaluate if the final solution indeed carries this property. For that purpose, the Boosted Tree model was trained using both the whole available samples and only the samples chosen by the algorithm. The model had the same number of trees as in the optimization procedure, and the parameters were chosen separately for each case with the same method as well. In each case, the trained model MASE was computed by a k-fold cross validation and the results are displayed at Table 4.11.

| Metric | Whole Dataset | Selected Samples |
|---|---|---|
| Cross Validated MASE | 1.15 | 1.08 |

Table 4.11: MASE Comparison (Cross Validation)

As it can be noticed, the selected subset brought a decrease in the error metric, confirming that the Selector successfully performed it's task. Next, labels are generated from the output and used to train the Classifier as described in the Proposal section. As already mentioned this could be done with any given model, and even with a whole different feature representation. In the spirit of keeping things simple, though, the same matrix will be used, and the chosen model will be the Boosted Classification Trees which are also implemented at [41]. For this task, the parameters were chosen in order to maximize the cross-validated accuracy of the training predictions, with the same grid-search used formerly.

The output of the classifier is set to be the probability of the sample being predictable, and so is bounded between 0 and 1. A natural way to select predictable samples would be to elect those above the 0.5 mark. In the present context though, a higher standard is chosen. With the goal of being as selective as possible, the predictable samples will be only those at which the output of the classifier is above 0.99. Even with this strict selection mechanism, 44% of the test set is chosen for forecasting, as displayed in Fig. 4.25.

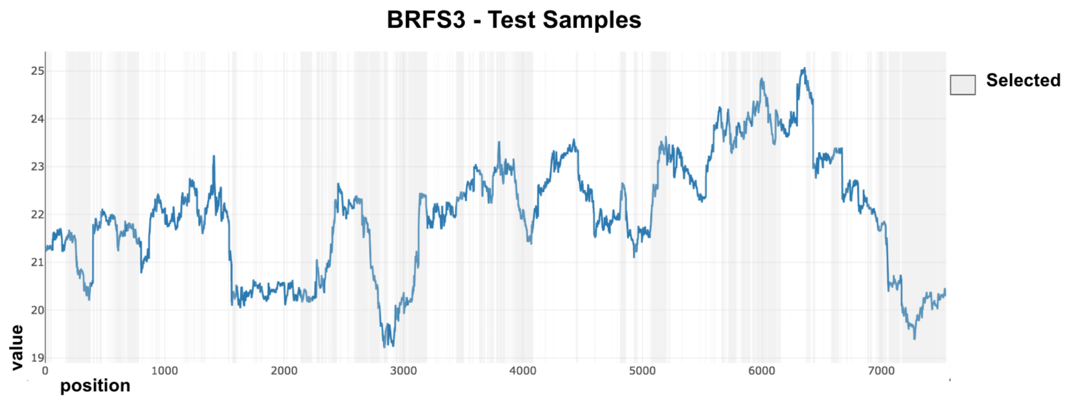
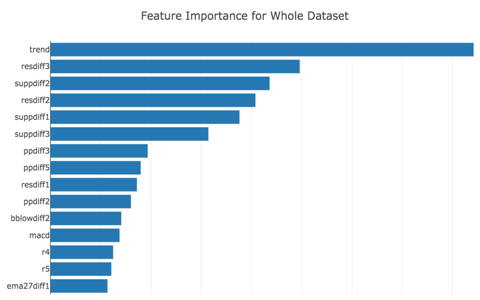
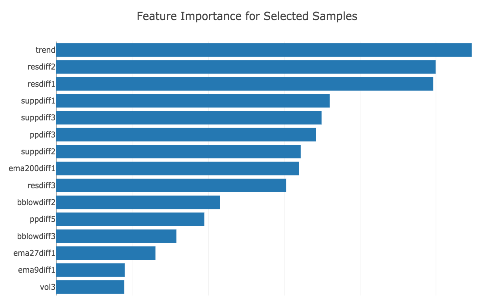


Figure 4.25: Periods in the Test Set Chosen for Prediction (44% Selected)

As a tree based algorithms is used, another useful information can be provided. By averaging the improvement at each split of each tree, this algorithm is able to produce a measure of the relative importance of each feature [16]. Fig. 4.26 shows the feature importance for the models trained in both the whole dataset and in the selected samples. It is possible to see that in both cases, the trend computed is the most important feature. In the first case though, this feature is much more important than all of the others. In the selected subset scenario the feature importance is more evenly distributed, with other features obtaining a relevance close to the trend. This significant change in the importance happens even with the selected subset holding a number of samples not much smaller than the original one (80%).



(a) Whole Dataset



(b) Selected Samples

Figure 4.26: Relative Feature Importance

In both cases the following positions are occupied by the support and resistance levels. Those levels are of key importance for chartists, and the present study indeed points them as influential. A key difference that can be noticed between the two though is that in the selected subset case, all of the moving averages appear among the top 15 most important features. This agrees with the findings of [26], where this indicator is argued to have a higher importance in specific moments more suitable to technical trading. Pivot Points also seem to be important to both situations, consolidating the fact that, in general, Price Action indicators seem to be very relevant when compared to the other indicators in the feature vector. Lower Boillinger Bands also appear in both scenarios, suggesting that this metric might be more effective when used in respect to those than when used in respect to the upper ones.

Now the Forecaster can be trained and evaluated. As stated in the Proposal section, the forecasting model doesn't need to be the same used in the Selector. Nevertheless, for this test the same model is employed to check if the specific pattern detected by the Perturbed PSO algorithm holds on new observed samples. The model is then evaluated in two different ways. First, it is trained on the whole dataset, and the MASE is computed using the whole test set. Second, the same model is trained only in the selected samples in the training set, and only the selected samples in the test set are used to compute the error.

| Metric | Whole Dataset | Selected Samples |
|---|---|---|
| Test Set MASE | 1.1 | 1.11 |

Table 4.12: MASE Comparison (Test Set)

Table 4.12 shows that differently from what happened at the training set, the MASE metric was not enhanced in the test set. This fact makes it evident that lowering the cross-validation score in-sample doesn't necessarily means it will be lowered out-of-sample.

#### 4.3.2.2 Most Traded Stocks

With the preliminary tests carried at BRFS3 stock, a more complete study is performed. This evaluation is carried in all of the 20 chosen papers. In order to do that, the exact same procedure taken before will be repeated for each one of the selected stocks. For each series, the percentage of test set coverage will be displayed along with other metrics. Those metrics are: the MAE, MASE, Correct Directional Changes, and the naive strategy error. They are evaluated for both the whole dataset and selected subset cases. The Correct Directional Changes (CDC) asserts how often the model was able to correctly predict if the price would rise or fall, and is computed as

$$CDC = \frac{1}{n}\sum_{i=1}^{n} D_t \qquad D_t = 1 \text{ if } \hat{y}_t * y_t > 0, 0 \text{ otherwise} \tag{4-6}$$

In the first run, the comparison is carried using the same boosted trees models employed so far. The results are summarized in Table 4.21. As it can be analyzed in this table, the increase in MASE does not represent a general tendency, instead the mean of the metric for the select samples improves overall. The other two metrics of interest (MAE and CDC) also show to be enhanced by the technique. To test whether or not the framework brought significant improvements, a Student-T Paired Test is performed. The test pairs the computed values for both cases, and take the differences from the numbers obtained when using the whole dataset and the ones obtained with the selected samples. The null hypothesis is that those differences have zero mean, and an one-sided test is performed to assert the chance of this mean being great or equal to zero for the MAE and MASE, and less or equal for the CDC. A 5% significance level will be require to refute the null case and state a positive result. The mean value $\mu_d$ and standard deviation $\sigma_d$ of those differences is computed, and the following test statistic is calculated

$$t = \frac{\bar{\mu_d}}{\frac{\sigma_d}{\sqrt{n}}} \tag{4-7}$$

The statistic $t$ follows a t-Student distribution with $n - 1$ degrees of freedom (19 in this case) and is computed for the MAE, MASE and CDC. Table 4.13 compiles the results. Although the improvements in the CDC and MASE does not pass the 5% significance level, the MAE successfully conquers the mark, displaying evidence of improvement in this metric. Even though every metric displayed better results when the framework was employed, the only significant change confirmed is the enhancement in the absolute error. Table 4.21 colors the cases where these numbers enhance and where they not to facilitate the visual inspection.

| MAE | CDC | MASE |
|---|---|---|
| 0.017 | 0.168 | 0.221 |

Table 4.13: Paired Tests P-values for the Boosted Trees model

Once the enhancement in the Mean Absolute Error is settled, it remains necessary to quantify it. The question to be answered then is how much improvement did the framework provide. Diving deeper in the results, the scatter plot between the whole dataset trained models' errors and the predictable samples ones' is computed. Fig. 4.27 shows the arising pattern. Its is easy to notice that a linear relationship can be inferred, with the metrics evaluated at the selected intervals being in average 0.81 times the regular ones. This signifies an reduction of almost 20% the original value, demonstrating the effect of the methodology.

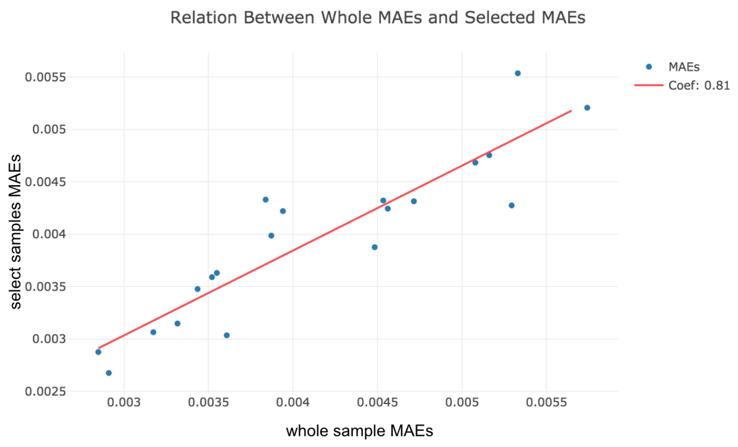


Figure 4.27: Regular and Enhanced MAEs for the Boosted Trees model

Table 4.14 summarizes the statistics for the regression performed. It is worth noticing that the results reinforce the significance of the improvement, with the 95% confidence interval for the angular coefficient being entirely bellow 1.

| Coefficient | Estimate | Std. Error | Lower Bound | Upper Bound |
|---|---|---|---|---|
| Intercept | 0.0006 | 0.0004 | -0.0002 | 0.0014 |
| Angular | 0.8104 | 0.0861 | 0.6295 | 0.9913 |

Table 4.14: Statistics for the regression between metrics - Boosted Trees

In the previous results, the same model was used for both the optimization stage and the forecasting of select samples (although with different parameters). A question that arises then is if this improvement is restrict to this model or if the selected signal indeed unveils a structure intrinsic to the series that can be exploited by other techniques.

To answer that question the signals computed are tested again with two different models. The first, the Random Forest (RF), another tree based variant. The second, a linear regression paired with the Least Absolute Square Shrinkage Operator (LASSO) model for feature selection [16]. The parameters presented in those models will be computed separately for each series as in the previous case. For the RF model this computation will be also done with the aid of cross validation, and for the LASSO the Bayesian Information Criterion (BIC) will be used. The comparison regarding those models is carried on exactly as it was with the first one, and Tables 4.22 and 4.23 contemplate the findings with the same colored aids for visual evaluation.

The linear model does better than the Boosted Tress overall. Due to the complicated dynamics of the price series, a simpler regressor as this one can be safer, being less prone to over-fitting. For this model again the same pattern is observed for the enhancements in MAE and CDC. The MASE metric on the other hand actually increases this time. As formerly done, paired-t tests are performed to access the significance of those differences.

Table 4.15 displays the results, reinforcing the only significant change as being the enhancement in MAE, with this metric displaying strong evidence of improvement. The relation of values between the two scenarios is again displayed in Fig 4.28.

| MAE | CDC | MASE |
|---|---|---|
| 0.004 | 0.111 | 0.915 |

Table 4.15: Paired Tests P-values for the linear model

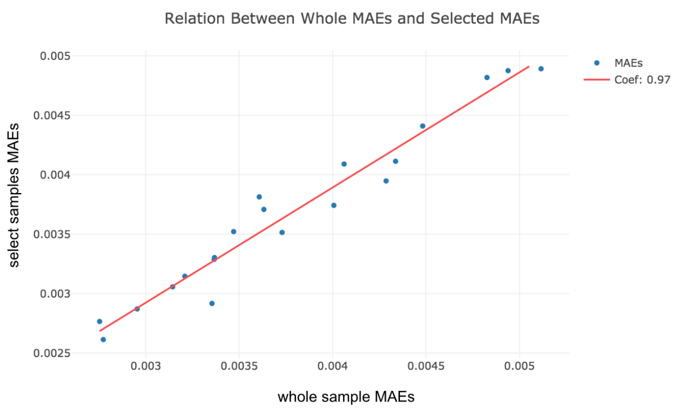


Figure 4.28: Regular and Enhanced MAEs for the linear model

This time the decrease in the mean absolute error is significantly smaller. A conjecture that could explain this is that the model used in the optimization stage has advantage to explore the resulting structure at the online phase. Another reason might be linked to the better performance already obtained by the linear model, as it is natural to assume that the better the results, the more difficult to enhance it. Nevertheless, the selection of samples was able to reduce by 3% the model error. Table 4.16 displays the regression details.

| Coefficient | Estimate | Std. Error | Lower Bound | Upper Bound |
|---|---|---|---|---|
| Intercept | 0 | 0.0002 | -0.0004 | 0.0004 |
| Angular | 0.9683 | 0.0498 | 0.8637 | 1.0729 |

Table 4.16: Statistics for the regression between metrics - linear model

Finally, the Random Forest model is applied to the stocks. Concerning the overall performance, this model does better than the boosted one, but slightly worse than the linear model. The benchmark in Table 4.23 depicts once again an improvement in the MAE and CDC with an apparent deterioration of the MASE. The same tests already performed for the other models are repeated and summarized in Table 4.17. Again the enhancement in the mean absolute error is confirmed. This time though, the worsening in the MASE is also significant (the p-value greater than 0.95 shows moderate evidence in favor of the null hypothesis).

| MAE | CDC | MASE |
|---|---|---|
| 0.018 | 0.179 | 0.982 |

Table 4.17: Paired Tests P-values for the Random Forest model

In respect to the relation of MAEs between the models using all of the samples and using only the select data points, Fig. 4.29 displays another strong linear relation, with a coefficient very close to the one computed for the linear model. The details of this regression are displayed at Table 4.18.

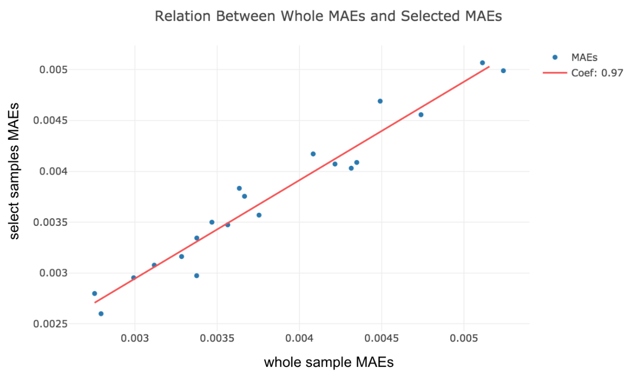


Figure 4.29: Regular and Enhanced MAEs for the Random Forest model

| Coefficient | Estimate | Std. Error | Lower Bound | Upper Bound |
|---|---|---|---|---|
| Intercept | 0 | 0.0002 | -0.0004 | 0.0004 |
| Angular | 0.9685 | 0.0518 | 0.8597 | 1.0773 |

Table 4.18: Statistics for the regression between metrics - Random Forest

The improvement observed when using the Boosted Trees model in the mean absolute error was significant in the other two variants tested. The perceived deterioration in the MASE for those other results revealed that the variance of the selected samples is, in general, smaller than in the whole set. The results so far are gathered in Table 4.19 for comparison.

| Model | MAE P-value | CDC P-value | MASE P-value | MAE Enhancement |
|---|---|---|---|---|
| Boosted Trees | 0.017 | 0.168 | 0.221 | 19% |
| LASSO | 0.004 | 0.111 | 0.915 | 3.3% |
| Random Forests | 0.018 | 0.179 | 0.982 | 3.3% |

Table 4.19: Models Results

To increase the power of the conclusions the combined 60 points of the models are compiled together and the paired tests are performed on this aggregated data. The outcomes help to uncover significance for the CDC improvement, displaying moderate evidence for this metric, while showing very strong evidence for the decrease in MAE and no evidence at all for the MASE deterioration. These results show that the effect of the methodology was not only to find smaller variance intervals, but ones where the direction of price changes is also more predictable. Although reasonable that the original model was more capable of exploring the signal structure, the fact that this is not unique to it shows the successful exposure of a more structured series in the chosen subset. This conclusions favors the Adaptive Markets Hypothesis, with the results shedding more light in the efficiency discussion.

| MAE | CDC | MASE |
|---|---|---|
| 0.0004 | 0.0248 | 0.6119 |

Table 4.20: Paired Tests P-values for the Aggregated Models Results

| Stock | Selected Fraction | Whole MAE | Selected MAE | Whole CDC | Selected CDC | Whole Naive | Selected Naive | Whole MASE | Selected MASE |
|---|---|---|---|---|---|---|---|---|---|
| BRFS3 | 0.44 | 0.00394 | 0.00422 | 0.54 | 0.54 | 0.0036 | 0.0038 | 1.1 | 1.11 |
| PETR4 | 0.68 | 0.00384 | 0.00433 | 0.52 | 0.53 | 0.0033 | 0.0033 | 1.15 | 1.33 |
| BBAS3 | 0.2 | 0.00352 | 0.00359 | 0.51 | 0.49 | 0.0034 | 0.0033 | 1.05 | 1.09 |
| BBDC4 | 0.75 | 0.00332 | 0.00315 | 0.5 | 0.52 | 0.0031 | 0.003 | 1.07 | 1.04 |
| ITSA4 | 0.26 | 0.00291 | 0.00268 | 0.55 | 0.57 | 0.0028 | 0.0025 | 1.06 | 1.06 |
| VALE3 | 0.31 | 0.00344 | 0.00348 | 0.52 | 0.51 | 0.0032 | 0.0031 | 1.07 | 1.11 |
| ITUB4 | 0.56 | 0.00285 | 0.00288 | 0.51 | 0.51 | 0.0028 | 0.0028 | 1.04 | 1.04 |
| ABEV3 | 0.45 | 0.00317 | 0.00307 | 0.53 | 0.52 | 0.0029 | 0.0029 | 1.08 | 1.07 |
| RENT3 | 0.65 | 0.00453 | 0.00432 | 0.52 | 0.5 | 0.0041 | 0.004 | 1.12 | 1.09 |
| KROT3 | 0.66 | 0.00516 | 0.00476 | 0.56 | 0.56 | 0.0043 | 0.0044 | 1.19 | 1.08 |
| B3SA3 | 0.66 | 0.00355 | 0.00363 | 0.52 | 0.54 | 0.0035 | 0.0035 | 1.02 | 1.04 |
| CCRO3 | 0.42 | 0.00472 | 0.00432 | 0.55 | 0.55 | 0.0042 | 0.0039 | 1.11 | 1.11 |
| JBSS3 | 0.44 | 0.00448 | 0.00388 | 0.55 | 0.58 | 0.0037 | 0.0035 | 1.2 | 1.11 |
| GGBR4 | 0.53 | 0.00387 | 0.00399 | 0.53 | 0.52 | 0.0036 | 0.0037 | 1.07 | 1.08 |
| ESTC3 | 0.66 | 0.00508 | 0.00468 | 0.52 | 0.54 | 0.0045 | 0.0045 | 1.12 | 1.05 |
| EMBR3 | 0.17 | 0.00361 | 0.00304 | 0.51 | 0.53 | 0.0033 | 0.0029 | 1.08 | 1.05 |
| ELET3 | 0.58 | 0.00574 | 0.00521 | 0.52 | 0.53 | 0.0051 | 0.0048 | 1.12 | 1.08 |
| CMIG4 | 0.3 | 0.00529 | 0.00428 | 0.53 | 0.56 | 0.004 | 0.0037 | 1.33 | 1.17 |
| ELET6 | 0.94 | 0.00533 | 0.00554 | 0.52 | 0.5 | 0.0049 | 0.0049 | 1.08 | 1.14 |
| CSNA3 | 0.37 | 0.00456 | 0.00424 | 0.56 | 0.55 | 0.0043 | 0.0041 | 1.05 | 1.04 |

Table 4.21: Framework Performance in All Chosen Stocks with the Boosted Trees model

| Stock | Selected Fraction | Whole MAE | Selected MAE | Whole CDC | Selected CDC | Whole Naive | Selected Naive | Whole MASE | Selected MASE |
|---|---|---|---|---|---|---|---|---|---|
| BRFS3 | 0.44 | 0.00361 | 0.00381 | 0.56 | 0.56 | 0.0036 | 0.0038 | 1.01 | 1.01 |
| PETR4 | 0.68 | 0.00337 | 0.00329 | 0.52 | 0.52 | 0.0033 | 0.0033 | 1.01 | 1.01 |
| BBAS3 | 0.2 | 0.00337 | 0.0033 | 0.52 | 0.54 | 0.0034 | 0.0033 | 1 | 1 |
| BBDC4 | 0.75 | 0.00314 | 0.00306 | 0.51 | 0.53 | 0.0031 | 0.003 | 1.02 | 1.01 |
| ITSA4 | 0.26 | 0.00277 | 0.00261 | 0.56 | 0.56 | 0.0028 | 0.0025 | 1.01 | 1.03 |
| VALE3 | 0.31 | 0.00321 | 0.00315 | 0.5 | 0.51 | 0.0032 | 0.0031 | 1 | 1.01 |
| ITUB4 | 0.56 | 0.00275 | 0.00277 | 0.52 | 0.53 | 0.0028 | 0.0028 | 1 | 1 |
| ABEV3 | 0.45 | 0.00295 | 0.00287 | 0.54 | 0.53 | 0.0029 | 0.0029 | 1 | 1 |
| RENT3 | 0.65 | 0.00406 | 0.00409 | 0.51 | 0.54 | 0.0041 | 0.004 | 1 | 1.03 |
| KROT3 | 0.66 | 0.00448 | 0.00441 | 0.56 | 0.57 | 0.0043 | 0.0044 | 1.03 | 1 |
| B3SA3 | 0.66 | 0.00347 | 0.00352 | 0.53 | 0.51 | 0.0035 | 0.0035 | 1 | 1.01 |
| CCRO3 | 0.42 | 0.00429 | 0.00395 | 0.54 | 0.55 | 0.0042 | 0.0039 | 1.01 | 1.01 |
| JBSS3 | 0.44 | 0.00373 | 0.00352 | 0.57 | 0.57 | 0.0037 | 0.0035 | 1 | 1.01 |
| GGBR4 | 0.53 | 0.00363 | 0.00371 | 0.55 | 0.56 | 0.0036 | 0.0037 | 1 | 1 |
| ESTC3 | 0.66 | 0.00483 | 0.00482 | 0.54 | 0.53 | 0.0045 | 0.0045 | 1.07 | 1.08 |
| EMBR3 | 0.17 | 0.00335 | 0.00292 | 0.54 | 0.56 | 0.0033 | 0.0029 | 1 | 1.01 |
| ELET3 | 0.58 | 0.00512 | 0.00489 | 0.52 | 0.51 | 0.0051 | 0.0048 | 1 | 1.01 |
| CMIG4 | 0.3 | 0.00401 | 0.00374 | 0.56 | 0.55 | 0.004 | 0.0037 | 1.01 | 1.02 |
| ELET6 | 0.94 | 0.00494 | 0.00488 | 0.51 | 0.51 | 0.0049 | 0.0049 | 1 | 1 |
| CSNA3 | 0.37 | 0.00434 | 0.00411 | 0.57 | 0.57 | 0.0043 | 0.0041 | 1 | 1 |

Table 4.22: Framework Performance in All Chosen Stocks with the linear model

| Stock | Selected Fraction | Whole MAE | Selected MAE | Whole CDC | Selected CDC | Whole Naive | Selected Naive | Whole MASE | Selected MASE |
|---|---|---|---|---|---|---|---|---|---|
| BRFS3 | 0.44 | 0.00363 | 0.00383 | 0.54 | 0.54 | 0.0036 | 0.0038 | 1.02 | 1.01 |
| PETR4 | 0.68 | 0.00356 | 0.00347 | 0.52 | 0.52 | 0.0033 | 0.0033 | 1.06 | 1.07 |
| BBAS3 | 0.2 | 0.00338 | 0.00334 | 0.51 | 0.54 | 0.0034 | 0.0033 | 1.01 | 1.01 |
| BBDC4 | 0.75 | 0.00312 | 0.00308 | 0.51 | 0.52 | 0.0031 | 0.003 | 1.01 | 1.01 |
| ITSA4 | 0.26 | 0.00279 | 0.0026 | 0.56 | 0.56 | 0.0028 | 0.0025 | 1.01 | 1.03 |
| VALE3 | 0.31 | 0.00328 | 0.00316 | 0.5 | 0.51 | 0.0032 | 0.0031 | 1.03 | 1.01 |
| ITUB4 | 0.56 | 0.00276 | 0.0028 | 0.52 | 0.51 | 0.0028 | 0.0028 | 1 | 1.01 |
| ABEV3 | 0.45 | 0.00299 | 0.00295 | 0.53 | 0.52 | 0.0029 | 0.0029 | 1.02 | 1.03 |
| RENT3 | 0.65 | 0.00408 | 0.00417 | 0.51 | 0.52 | 0.0041 | 0.004 | 1.01 | 1.05 |
| KROT3 | 0.66 | 0.00449 | 0.00469 | 0.56 | 0.56 | 0.0043 | 0.0044 | 1.03 | 1.07 |
| B3SA3 | 0.66 | 0.00347 | 0.0035 | 0.53 | 0.52 | 0.0035 | 0.0035 | 1 | 1.01 |
| CCRO3 | 0.42 | 0.00432 | 0.00403 | 0.54 | 0.55 | 0.0042 | 0.0039 | 1.02 | 1.03 |
| JBSS3 | 0.44 | 0.00375 | 0.00357 | 0.55 | 0.56 | 0.0037 | 0.0035 | 1.01 | 1.02 |
| GGBR4 | 0.53 | 0.00367 | 0.00375 | 0.54 | 0.55 | 0.0036 | 0.0037 | 1.01 | 1.01 |
| ESTC3 | 0.66 | 0.00474 | 0.00456 | 0.53 | 0.52 | 0.0045 | 0.0045 | 1.05 | 1.02 |
| EMBR3 | 0.17 | 0.00338 | 0.00297 | 0.54 | 0.55 | 0.0033 | 0.0029 | 1.01 | 1.03 |
| ELET3 | 0.58 | 0.00524 | 0.00499 | 0.52 | 0.52 | 0.0051 | 0.0048 | 1.03 | 1.03 |
| CMIG4 | 0.3 | 0.00422 | 0.00407 | 0.56 | 0.54 | 0.004 | 0.0037 | 1.06 | 1.11 |
| ELET6 | 0.94 | 0.00511 | 0.00507 | 0.52 | 0.52 | 0.0049 | 0.0049 | 1.04 | 1.04 |
| CSNA3 | 0.37 | 0.00435 | 0.00409 | 0.55 | 0.58 | 0.0043 | 0.0041 | 1 | 1 |

Table 4.23: Framework Performance in All Chosen Stocks with the Random Forests model

# 5 Conclusion and Future Work

The present work discussed the efficiency of the market as well as the studies that inspired the proposed idea. The simulations performed showed that the P-Crawl is capable of performing well in a large variety of conditions. Regarding the proposed optimization procedures, the Perturbed PSO displayed better exploration-exploitation capabilities for the task at hand than the regular PSO. The same result was obtained for the KNN-GA, showing superior results than the regular Genetic Algorithm. To access the performance of the framework, different simulations where made. The term purity was used to refer to the percentage of predictable samples in each simulation.

The Perturbed PSO and the KNN-GA coupled with the two fitness functions proposed exhibited good results for a range of contamination going from 20% to 100% purity. Two different score functions were proposed, $F_{exploit}$ and $F_{explore}$, with their difference being in the importance they give to the size of the selected subsets. The KNN-GA with the $F_{exploit}$ function showed robust results in total purity from 20% to 40%, while the Perturbed PSO coupled with the $F_{explore}$ variant consistently discovered the right patterns in total purity scenarios from 40% to 100%. Although some route of investigation and rule of thumb were discussed, the problem of how to identify the total purity configuration of a real-life dataset with respect to a specific model remains open.

The relative importance of some major technical indicators were compared. The trend of the price series was selected as the most important feature, and the Price Action technical indicators exhibited a relevant importance. This information can be used to refine the feature representation for intra-day stock data in future work. Different indicators and feature representations can also be explored, since this choice has a major impact in finding predictable samples in the series.

When applied to the brazilian stock market intraday data, the analysis elected the Perturbed PSO variant as adequate for the task. The 20 most traded stocks at the beginning of the studied period were used as a benchmarks, and the MAE, CDC and MASE metrics were evaluated with and without the use of the proposed framework. The P-Crawl significantly increased the performance on the MAE and CDC, asserting the utility of the proposed methodology. The same result was not achieved for the MASE. The reductions where the greatest when the same model was used for both the probing and forecasting of samples, achieving a contraction of MAE as high as 19%.

Not only the selectivity in time-frames enhanced the evaluation metrics, but the period scanned for predictability in respect to a particular model was more prone to be exploited by it. Although the framework was not able to improve the MASE metric, the decrease in the correct directional changes showed that the selected intervals had a more predictable price dynamic. The outcome is favorable to the AMH, aggregating more evidence in favor a floating efficiency level in the market.

Regarding future works, many opportunities are available. To make the P-Crawl more feasible, the speed of the proposed algorithms is of great importance. A more efficient implementation of the optimization procedure is fundamental for the framework to be more easily employed. Computing each iteration concurrently in different CPUs or in the computing units of a GPU could considerably reduce the time required for the procedure. The matter of how to derive conclusions about the total purity of a real dataset in regard to a statistical model is also open.

Every building block of the P-Crawl can be fiddled with. A broad range of statistical models can be tested in the Selector, Forecaster and Optimizer. New fitness functions can be proposed, such as using financial gains instead of error metrics for fitness scores.

The feature representations also allow for interesting experiments in future work. One possible approach could be to construct a vector using predictability metrics, building on the studies of [17, 18, 19, 21] and others. This way, the optimization procedure would be able to combine a series of predictability signals to discover the best possible indicator.

The implementation used for the GA and PSO is also an important factor. The PSO showed more robust exploration properties, and a possible cause can be the randomness introduced at the particle representation. A similar approach can be taken for the GA algorithm, using Quantum-inspired evolutionary algorithms such as in [43, 44, 45].

# 6
# References

[1] NTI, I. K.; ADEKOYA, A. F. ; WEYORI, B. A.. **A systematic review of fundamental and technical analysis of stock market predictions**. Artificial Intelligence Review, p. 1–51, 2019. 1

[2] ISLAM, S.; OTHERS. **Fundamental and technical analysis: Tools used in stock market**. 2019. 1

[3] DE ZWART, G.; MARKWAT, T.; SWINKELS, L. ; VAN DIJK, D.. **The economic value of fundamental and technical information in emerging currency markets**. Journal of International Money and Finance, 28[4]:581–604, 2009. 1

[4] FAMA, E. F.. **The behavior of stock-market prices**. The journal of Business, 38[1]:34–105, 1965. 1, 2.1, 2.4

[5] EFENDI, R.; ARBAIY, N. ; DERIS, M. M.. **A new procedure in stock market forecasting based on fuzzy random auto-regression time series model**. Information Sciences, 441:113–132, 2018. 1

[6] MOGHADDAM, A. H.; MOGHADDAM, M. H. ; ESFANDYARI, M.. **Stock market index prediction using artificial neural network**. Journal of Economics, Finance and Administrative Science, 21[41]:89–93, 2016. 1, 2.4

[7] TALARPOSHTI, F. M.; SADAEI, H. J.; ENAYATIFAR, R.; GUIMARÃES, F. G.; MAHMUD, M. ; ESLAMI, T.. **Stock market forecasting by using a hybrid model of exponential fuzzy time series**. International Journal of Approximate Reasoning, 70:79–98, 2016. 1

[8] LAHMIRI, S.. **Minute-ahead stock price forecasting based on singular spectrum analysis and support vector regression**. Applied Mathematics and Computation, 320:444–451, 2018. 1

[9] KAUR, J.; DHARNI, K.. **Predicting daily returns of global stocks indices: Neural networks vs support vector machines**. Journal of Economics, Management and Trade, p. 1–13, 2019. 1

[10] JIANG, S.-L.; WU, W.-J.. **Forecasting stock market with social media sentiment based on adaptive network fuzzy inference system**. DEStech Transactions on Economics, Business and Management, [ssemr], 2019. 1

[11] FINN, F. J.; WHITTRED, G.. **On the use of naive expectations of earnings per share as experimental benchmarks**. Economic Record, 58[2]:169–173, 1982. 1

[12] MCLEAN, R. D.; PONTIFF, J.. **Does academic research destroy stock return predictability?** The Journal of Finance, 71[1]:5–32, 2016. 1

[13] SCHWERT, G. W.. **Anomalies and market efficiency**. Handbook of the Economics of Finance, 1:939–974, 2003. 1, 2.1

[14] CZAPKIEWICZ, A.; ZAREMBA, A. ; SZCZYGIELSKI, J. J.. **Predicting the performance of equity anomalies in frontier emerging markets: a markov switching model approach**. Economic Research-Ekonomska Istraživanja, 32[1]:3077–3093, 2019. 1

[15] LO, A. W.. **The adaptive markets hypothesis**. The Journal of Portfolio Management, 30[5]:15–29, 2004. 1, 2.1

[16] HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. ; FRANKLIN, J.. **The elements of statistical learning: data mining, inference and prediction**. The Mathematical Intelligencer, 27[2]:83–85, 2005. 1, 2.3, 3.2.1, 3.2.2, 3.4, 3.5, 4.3.2.1, 4.3.2.2

[17] QIAN, B.; RASHEED, K.. **Hurst exponent and financial market predictability**. In: IASTED CONFERENCE ON FINANCIAL ENGINEERING AND APPLICATIONS, p. 203–209, 2004. 1, 2.4, 5

[18] MOLGEDEY, L.; EBELING, W.. **Local order, entropy and predictability of financial time series**. The European Physical Journal B-Condensed Matter and Complex Systems, 15[4]:733–737, 2000. 1, 2.4, 5

[19] MAASOUMI, E.; RACINE, J.. **Entropy and predictability of stock market returns**. Journal of Econometrics, 107[1-2]:291–312, 2002. 1, 2.4, 5

[20] KIM, J. H.; SHAMSUDDIN, A. ; LIM, K.-P.. **Stock return predictability and the adaptive markets hypothesis: Evidence from century-long us data**. Journal of Empirical Finance, 18[5]:868–879, 2011. 1

[21] LETTAU, M.; VAN NIEUWERBURGH, S.. **Reconciling the return predictability evidence: The review of financial studies: Reconciling the return predictability evidence**. The Review of Financial Studies, 21[4]:1607–1652, 2007. 1, 2.4, 5

[22] GROSSMAN, S. J.; STIGLITZ, J. E.. **On the impossibility of informationally efficient markets**. The American economic review, 70[3]:393–408, 1980. 2.1

[23] ENGEL, C.; HAMILTON, J.. **Long swings in the dollar: Are they in the data and do markets know it?** American Economic Review, 80:689–713, 02 1990. 2.2, 2.2

[24] MITRA, S.; DATE, P.. **Regime switching volatility calibration by the baum–welch method**. Journal of Computational and Applied Mathematics, 234[12]:3243–3260, 2010. 2.2

[25] ENGEL, C.. **Can the markov switching model forecast exchange rates?** Journal of International Economics, 36[1-2]:151–165, 1994. 2.2

[26] REITZ, S.. **On the predictive content of technical analysis**. North American Journal of Economics and Finance, [17]:127–137, 2006. 2.3, 2.3, 2.4, 4.2.4.6, 4.3.2.1

[27] MURPHY, J. J.. **Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications**. New York Institute of Finance, 1999. 2.3

[28] YANG, S.-R.; BRORSEN, B. W.. **Nonlinear dynamics of daily futures prices: conditional heteroskedasticity or chaos?** Journal of Futures Markets, 13[2]:175–191, 1993. 2.4

[29] KIMOTO, T.; ASAKAWA, K.; YODA, M. ; TAKEOKA, M.. **Stock market prediction system with modular neural networks**. In: 1990 IJCNN INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, p. 1–6. IEEE, 1990. 2.4

[30] WHITE, H.. **Economic prediction using neural networks: The case of ibm daily stock returns**. 1988. 2.4

[31] SAAD, E. W.; PROKHOROV, D. V. ; WUNSCH, D. C.. **Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks**. IEEE Transactions on neural networks, 9[6]:1456–1470, 1998. 2.4

[32] ABU-MOSTAFA, Y. S.; MAGDON-ISMAIL, M. ; LIN, H.-T.. **Learning from data**, volumen 4. AMLBook New York, NY, USA:, 2012. 2.4

[33] HOLLAND, J. H.; OTHERS. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. MIT press, 1992. 3.3

[34] EBERHART, R.; KENNEDY, J.. **Particle swarm optimization**. In: PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, volumen 4, p. 1942–1948. Citeseer, 1995. 3.3

[35] FOGEL, D. B.. **Evolutionary computation: toward a new philosophy of machine intelligence**, volumen 1. John Wiley & Sons, 2006. 3.3.1.2, 3.3.1.5

[36] THIERENS, D.; GOLDBERG, D.. **Convergence models of genetic algorithm selection schemes**. In: INTERNATIONAL CONFERENCE ON PARALLEL PROBLEM SOLVING FROM NATURE, p. 119–129. Springer, 1994. 3.3.1.3

[37] BELIGIANNIS, G. N.; TSIROGIANNIS, G. A. ; PINTELAS, P. E.. **Restartings: A technique to improve classic genetic algorithms' performance.** In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE, p. 404–407. Citeseer, 2004. 3.3.1.6

[38] ČREPINŠEK, M.; LIU, S.-H. ; MERNIK, M.. **Exploration and exploitation in evolutionary algorithms: A survey**. ACM Computing Surveys (CSUR), 45[3]:35, 2013. 3.3.1.6

[39] ROTH, R.. **Introduction to coding theory**. Cambridge University Press, 2006. 3.3.2

[40] CLERC, M.. **Beyond standard particle swarm optimisation**. In: INNOVATIONS AND DEVELOPMENTS OF SWARM INTELLIGENCE APPLICATIONS, p. 1–19. IGI Global, 2012. 3.3.3

[41] CHEN, T.; GUESTRIN, C.. **Xgboost: A scalable tree boosting system**. In: PROCEEDINGS OF THE 22ND ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, p. 785–794. ACM, 2016. 3.4, 4.3.2.1, 4.3.2.1

[42] PERLIN, M.; RAMOS, H.. **GetHFData: A R Package for Downloading and Aggregating High Frequency Trading Data from Bovespa**, 2016. 4.3.1

[43] DA CRUZ, A. A.; VELLASCO, M. M. B. R. ; PACHECO, M. A. C.. **Quantum-inspired evolutionary algorithm for numerical optimization**. In: HYBRID EVOLUTIONARY ALGORITHMS, p. 19–37. Springer, 2007. 5

[44] DE PINHO, A. G.; VELLASCO, M. ; DA CRUZ, A. V. A.. **A new model for credit approval problems: A quantum-inspired neuro-evolutionary algorithm with binary-real representation**. In: 2009 WORLD CONGRESS ON NATURE & BIOLOGICALLY INSPIRED COMPUTING (NABIC), p. 445–450. IEEE, 2009. 5

[45] VELLASCO, M.; CRUZ, A. ; PINHO, A.. **Quantum-inspired evolutionary algorithms applied to neural network modeling**. In: IEEE WORLD CONFERENCE ON COMPUTATIONAL INTELIGENCE, PLENARY AND INVITED LECTURES, p. 125–150, 2010. 5