

Enhancing Model Generalization in Index Futures Markets via Evolutionary Optimization and Data Leakage-Free Feature Engineering

Otávio Zucchetti Dalla Costa[1], Bruno L. Dalmazo[1], Viviane Leite Dias de Mattos[1], Richard F. Pinto[1], Diego Renan Bruno[2], Eduardo N. Borges[1], Giancarlo Lucca[1], Fabian C. Cardoso[3], and Rafael A. Berri[1(✉)]

[1] Federal University of Rio Grande (FURG), Rio Grande, Brazil
{otaviozdc,dalmazo,vivianemattos,richard_pinto,eduardoborges, giancarlo.lucca,rafaelberri}@furg.br
[2] São Paulo State University (UNESP), São José do Rio Preto, Brazil
diego.bruno@unesp.br
[3] University of Rio Verde (UniRV), Rio Verde, Brazil
fabian@unirv.edu.br

Abstract. This study investigates the directional prediction of the Mini Bovespa Index Futures (WIN), which are short-term derivative contracts based on Brazil's main stock index that traders use to speculate or hedge utilizing rapid intraday price movements. A major hurdle in this domain is the tendency of machine learning models to overfit to market noise. To mitigate this, we propose an optimization pipeline that integrates Sequential Forward Selection (SFS) and evolutionary hyperparameter optimization via Genetic Algorithms (GA). The evaluation focused on how the XGBoost and Multi-Layer Perceptron (MLP) architectures respond to evolutionary tuning. It was observed that the need for GA optimization depends on the stability of the model. For example, the XGBoost baseline (default hyperparameters) presented a generalization gap, which the GA-based regularization wrapper managed to reduce to 0.0179, giving it an F1-score of 0.7220. Additionally, the GA demonstrated superior computational efficiency over Random Search, by navigating the hyperparameter space through a directed search that minimizes the evaluation of suboptimal architectures. On the other hand, MLP already demonstrated stability with default settings, although GA brought its gap to near-zero indices and guaranteed it an F1-score of 0.7182, yielding marginal improvements. These results suggest that evolutionary algorithms are important for controlling variance models like XGBoost but serve as a refinement step for robust structures like MLP.

Keywords: Genetic Algorithms · XGBoost · Multi-Layer Perceptron · Generalization Gap · Financial Forecasting

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16768, pp. 550–568, 2027.
https://doi.org/10.1007/978-3-032-30488-9_34

1 Introduction

The Brazilian financial market, especially the Mini Bovespa Index Futures (WIN), is characterized by liquidity and volatility. For algorithmic trading systems, this domain presents a low signal-to-noise ratio, where market noise obscures directional trends [8]. Although Machine Learning (ML) models are standard for capturing non-linear dependencies in such time series, a vulnerability is imposed: overfitting. In intraday trading, common ML approaches often adjust to historical noise instead of learning structural patterns, leading to discrepancies between retrospective testing performance and real-time execution, which this study defines as the **Generalization Gap** [13].

In algorithmic trading, minimizing the generalization gap proves more important than increasing raw predictive accuracy (such as the F1-Score). Poor retrospective performance coupled with a large gap indicates that the model adjusted for market noise rather than capturing structural signals. In a real trading environment, this instability imposes financial drawdowns and operational risks on investors, as the strategy fails when exposed to unknown market regimes. Consequently, prioritizing structural robustness over maximum historical accuracy becomes a requirement for viable automated trading. To address this instability, an integrated data optimization flow is proposed to mitigate overfitting. A Sequential Forward Selection (SFS) wrapper associated with chronological normalization is used, ensuring that the selection of technical indicators is entirely free from look-ahead bias.

Furthermore, an Evolutionary Hyperparameter Optimization framework using Genetic Algorithms (GA) is presented. Instead of relying on common metrics that can reward overfitting, the evolutionary process is driven by a personalized fitness function. This function penalizes the variance between training and validation scores, exerting selective pressure that favors stable architectures over merely precise ones. This methodology is applied to two distinct families of learners: Gradient Boosting Decision Trees (XGBoost) and Neural Networks (Multilayer Perceptron - MLP), evaluating how evolutionary tuning impacts models with fundamentally different inductive biases.

This study demonstrates that evolutionary optimization is context-dependent. While necessary for reducing the generalization gap in variance models like XGBoost, it offers diminishing returns to stable MLP architectures. Moreover, it provides noisy financial markets with a reproducible and statistically sound framework for feature selection and model fitting. By bridging theoretical predictive power and practical stability, this study lays the groundwork for autonomous trading systems that can maintain viability over long periods in dynamic environments [23].

The remainder of this paper is organized as follows: Sect. 2 introduces the theoretical concepts, Sect. 3 details the proposed optimization methodology, Sect. 4 presents the experimental results, and Sect. 5 concludes the study with future research directions.

2 Theoretical Concepts

This section establishes the theoretical foundation on which the proposed optimization flow is based. First, the central principles of GAs are presented, detailing the biomimetic approach, focused on navigating non-linear search spaces for model fitting. Subsequently, the logical foundation, attributed to the feature engineering process, is outlined, formalizing the specific technical indicators, categorized into volatility, momentum, and mean reversion metrics, that are assigned the role of extracting predictive signals from the intraday financial time series environment.

2.1 Genetic Algorithm (GA)

Genetic Algorithms (GAs) represent a category of metaheuristic optimization structures based on the biological principles of natural selection and evolution [10]. These algorithms operate by computational emulation of the evolutionary life cycle, in which a population of candidate solutions is subjected to iterative refinement via selection, crossover, and mutation operators. Candidates that demonstrate favorable performance, as quantified by an evaluation metric, are given a higher probability of propagating characteristics to subsequent generations, directing the system towards convergence on effective solutions.

Within this biomimetic paradigm, the specific optimization challenge constitutes the environment, while members of the population are assigned the role of potential solutions. The evolutionary pressure exerted by the environment is operationalized through a fitness function. This function assigns a fitness score to each individual, measuring the usefulness of the proposed solution in relation to the problem's constraints. Consequently, GAs are reserved for robustness in navigating complex problem spaces where conventional deterministic approaches prove insufficient [7].

2.2 Feature Engineering and Rationalization

To mitigate the low signal-to-noise ratio of the financial time series, raw closing prices were transformed into a multidimensional attribute space, representing volatility positioning, momentum changes, and mean reversion. Volatility is modeled using the Bollinger Bands Relative Position [2], defined as $BB_{pos} = \frac{C_t - LB_t}{UB_t - LB_t}$, where C_t denotes the closing price in the time interval t while UB_t and LB_t represent the Upper and Lower Bands respectively. This transformation maps absolute prices onto a stationary amplitude metric, generally contained between 0 and 1, which is considered important for the convergence and stability of nonlinear models, such as MLP and XGBoost. Analytically, values close to 1 are associated with the indication of overbought regions, while values close to 0 are linked to the suggestion of oversold conditions, allowing the algorithm to dynamically identify reversal points. Currently, quantification is applied through the Relative Strength Index (RSI) [22], which is decomposed into its current state (RSI_t), its first-order difference ($RSI_{diff} = RSI_t - RSI_{t-1}$) and its

immediate history ($RSI_LAG1 = RSI_{t-1}$). The inclusion of RSI_{diff} acts as an indicator of trend acceleration or exhaustion, while coupling the current state with RSI_LAG1 allows the model to differentiate between the accumulation and distribution phases of a trend, explicitly reconstructing the temporal inertia of price changes. Finally, short-term serial correlation and mean reversion [12,15] are captured by two specific features: $CLOSEvsSMA_n$, which calculates the percentage distance between the closing price and an n-period Simple Moving Average (SMA_n), expressed as $Dist_n = \frac{C_t - SMA_n}{SMA_n}$, and $RETURN_LAG1$, representing the lagged simple return of the previous 5-minute candle, $R_{t-1} = \frac{C_{t-1} - C_{t-2}}{C_{t-2}}$. Significant deviations in $Dist_n$ identify "stretched" price levels where the asset is statistically likely to undergo a pullback, whereas the lagged return captures the immediate historical shock. Together, this multi-scale feature engineering provides the learning architectures with a robust representation of recent market behavior and short-term autocorrelation [9,18].

3 Methodology

The proposed methodology establishes a flow designed to mitigate two primary risks in financial machine learning, look-ahead bias, and overfitting. As illustrated in Fig. 1, the architecture is sequential in nature, ensuring that feature selection and data normalization are restricted to the chronological limits of training data. This workflow was realized through the Python language [17], utilizing the XGBoost library [4] for boosting models, while the Scikit-learn library [16] and the TensorFlow framework [1] provided the necessary structures for data processing and neural networks.

3.1 Input Data

The experimental framework uses high-frequency intraday data extracted from the BovDBV2 dataset [20], focusing on Mini Bovespa Index Futures (WIN). The dataset covers the first half of 2024 (January 1st to June 30th), organized into 5-minute intervals containing Open, High, Low, Close, and Volume vectors. This chronological sequence is partitioned into two isolated sets, a training and optimization period within the sample (corresponding to 70% of the data, dedicated to adjustment procedures) and a test period outside the sample (the final 30% of the chronological sequence). This chronological division ensures that the final evaluation metrics remain unbiased in relation to the hyperparameter optimization process.

3.2 Feature Engineering

The construction of the input space constitutes the fundamental layer of the proposed methodology. Based on the theoretical definitions established in

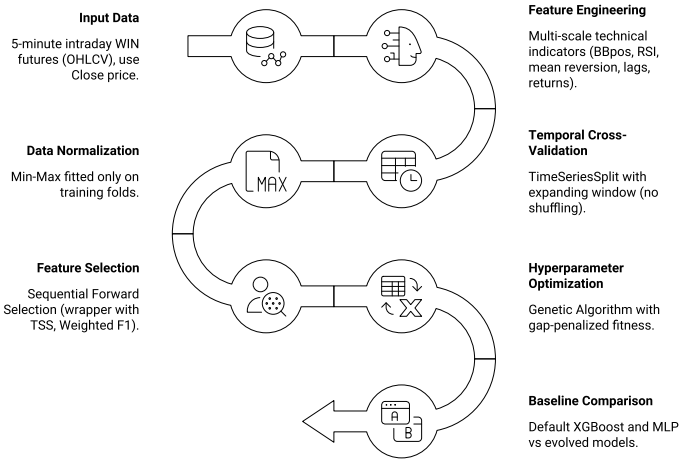

Fig. 1. Proposed methodological flow, detailing the temporal progression from the acquisition of raw data to the hyperparameter optimization phase, based on genetic methods.

Sect. 2.2, the isolated closing price vector was transformed into a multidimensional attribute matrix. The initial set of candidates was generated by calculating the central indicators, BB_{pos}, RSI, and mean reversion metrics, across a spectrum of time windows. This multi-scale approach ensures that the model captures market dynamics at varying granularities, from immediate volatility shocks to exhaustion of structural trends. Additionally, to incorporate the rate of change and temporal dependencies, first-order differences (such as RSI_{diff}) and lagged return values (RETURN_LAG1) were included.

This strategy is associated with the production of a set of high-dimensional attributes. Instead of applying manual filtering based on discretionary heuristics to this data, this extensive dataset reserved the role of raw input for the SFS (described in Sect. 3.5), allowing the algorithm to identify the subset of variables that maximizes generalization performance.

3.3 Temporal Cross-Validation

In financial time series forecasting, the inherent serial correlation of the data is attributed to the inadequacy of common random validation methods [3]. Consequently, K-Fold cross-validation is not permitted since the shuffling of observations would result in the destruction of the temporal structure and the introduction of anticipation bias, allowing the model to inadvertently train with future data to predict past events. To solve this problem, a TimeSeriesSplit (TSS) cross-validation scheme [16] was used, strictly applied to the training and optimization period within the sample. This technique utilizes an "anchored expanding window" approach that strictly respects chronological order. For each split k, this in-sample data is partitioned so that the training set comprises the historical

sequence defined by the indices $[0, \ldots, n]$. The validation set is then formed by the immediately subsequent horizon $[n+1, \ldots, n+m]$, where n represents the cutoff point and m the validation window size. This architecture strictly isolates future information, effectively simulating a realistic Walk-Forward Analysis where the model is periodically re-calibrated as new market data become available [14].

3.4 Data Normalization

To guarantee input consistency and strictly prevent the leakage of future volatility information into the final evaluation, a strict chronological normalization strategy was employed [13]. Within this framework, the Min-Max normalization identifies the boundary values (min, max) exclusively within the in-sample training period (the 70% partition) [16]. These fixed parameters are then applied to transform the isolated out-of-sample test set, ensuring that the final evaluation is never exposed to global statistics derived from future data. The transformation is formally defined as:

$$x' = \frac{x - \min(X_{train})}{\max(X_{train}) - \min(X_{train})} \tag{1}$$

Where $\min(X_{train})$ and $\max(X_{train})$ denote the minimum and maximum values observed strictly within the distribution of the in-sample training period. This transformation subjects all attributes to a scale contained within the bounded interval [0,1]. By fixing these parameters, the system is forced to simulate a real trading environment in which extreme future prices are unknown at the time of the forecast. Moreover, by applying normalization to the input before the attribute selection phase, it is ensured that the SFS algorithm evaluates the predictive power of the indicators based on relative structural patterns, rather than absolute magnitudes [6].

3.5 Sequential Forward Selection (SFS)

Given the size of the initial attribute space, the presence of irrelevant or redundant variables can introduce noise that impairs the performance of nonlinear models. To solve this problem, an SFS wrapper was used to obtain a representation of market data.

The SFS algorithm employs a greedy search strategy, starting from an empty set, the algorithm iteratively adds the attribute associated with an improvement in the objective function. This process continues until the performance metric stabilizes or declines.

In the proposed structure, the quality assessment of the subset is assigned to a supervised learning classifier that acts as a wrapper estimator [11]. To ensure consistency with temporal constraints, candidate subsets are assigned scores in chronological validation folds (TSS), preventing look-ahead bias during the selection phase [14]. The algorithm prioritizes attribute configurations associated with an increase in a classification metric in the validation set.

Fundamentally, instead of imposing a selection based on standard parameters, a hierarchical validation approach is adopted. Promising subsets are allowed to advance to the Hyperparameter Optimization stage (Sect. 3.6). This dynamic ensures that the final decision is based on the potential of each configuration. The GA then is reserved the role of arbiter, distinguishing the attributes to which a robust predictive signal is associated from those that only impose complexity without comprehensive benefit [6].

Although wrapper methods are inherently biased by their underlying estimator, XGBoost was specifically selected for the SFS phase due to its computational efficiency and resistance to nonlinearities without requiring prior data scaling. Running a dedicated SFS wrapper using a Neural Network would impose a computational bottleneck. Therefore, the XGBoost wrapper serves as a technically viable global filter for noise reduction in a core set of structurally informative attributes. By design, this same subset of attributes is provided to both the XGBoost and MLP architectures during the evolutionary phase. This strategy is linked to the establishment of a comparison base, ensuring that differences in final generalization performance originate exclusively from structural optimization and the fundamental inductive biases of each model, rather than variations in the input data.

3.6 Hyperparameter Optimization

After identifying the candidate attribute subsets, in the previous step, the final phase involves using a GA to optimize the hyperparameters of each configuration independently. This optimization strategy is important within the proposed hierarchical structure by subjecting each candidate subset to evaluation under favorable parameter definitions, the bias associated with default model configurations is eliminated and it is ensured that the final selection is based on the potential of each attribute set.

Fitness Function and Generalization. To counteract overfitting, we introduced a Fitness Function $F(\theta)$ that explicitly penalizes the divergence between training and validation scores, ensuring structural stability [13]. The F for an individual θ is defined as:

$$F(\theta) = \bar{S}_{val} - \lambda \cdot \max(0, \bar{S}_{train} - \bar{S}_{val}) \tag{2}$$

where:

- $\bar{S}_{val}$ and $\bar{S}_{train}$ denote the mean Weighted F1-Scores across the TSS validation and training folds, respectively.
- The term $\max(0, \bar{S}_{train} - \bar{S}_{val})$ represents the **Overfitting Gap**. We utilize the max function to ensure that the model is only penalized when training performance artificially exceeds validation performance.
- λ is the penalty coefficient.

Furthermore, to address the class imbalance present in the intraday market data, a double-layer strategy was adopted. During the training phase, the loss functions of both XGBoost and MLP were weighted inversely proportional to the class frequencies (Cost-Sensitive Learning) [21], requiring that the gradient descent penalize classification errors in the minority class more rigorously. During the evolutionary evaluation, the Weighted F1-Score was reserved the role of primary metric ($\bar{S}$), ensuring that the selection pressure favors solutions that balance precision and recall, rather than focusing efforts on accuracy in the majority class.

To perform the evolutionary adjustment, the Genetic Algorithm was assigned a configuration with specific hyperparameters, suitable for the computational complexity of each model. Since the XGBoost model has lower processing requirements, this system was allowed to explore the search space on a larger scale compared to that applied to the MLP. The evolution of candidate solutions is handled through generational epochs driven by stochastic operators, specifically tournament selection, two-point crossing, and bit inversion mutation, adapted to the binary representation scheme. Table 1 details these configurations including population size, number of generations, mutation rates, and selection strategies used for both architectures during the optimization phase.

Table 1. Evolutionary Algorithm Configuration for XGBoost and MLP

Parameter	XGBoost (Tree-based)	MLP (Neural Network)
Encoding Strategy	Binary	Binary
Population Size	20 Individuals	10 Individuals
Generations	1000	200
Crossover Probability	70%	70%
Mutation Probability	20% (Individual) / 5% (Bit)	20% (Individual) / 5% (Bit)
Selection Method	Tournament ($k = 3$)	Tournament ($k = 3$)
Penalty Coefficient (λ)	0.5	0.5

3.7 Experiment Configuration

To quantify the impact of evolutionary optimization, non-optimized benchmarks were established for both learning architectures. These benchmark models serve as a control group for measuring the inherent tendency of the algorithms to overfit before applying any genetic tuning. This approach aims to obtain a performance benchmark against which the results obtained after the stochastic search process can be compared.

The reference configuration for the XGBoost model is detailed in Table 2. Based on standard hyperparameters, this structure produces a variance estimator that requires reduced regularization. This arrangement serves as a benchmark

for quantifying the model's susceptibility to overfitting within the financial data environment, allowing the analysis to incorporate the distinction between data noise and the signal captured by optimization.

Table 2. Default Hyperparameters for XGBoost Baseline

Hyperparameter	Value	Description
`n_estimators`	100	Number of boosting rounds
`max_depth`	6	Maximum tree depth
`learning_rate`	0.3	Step size shrinkage (η)
`subsample`	1.0	% of samples used per tree
`colsample_bytree`	1.0	% of features used per tree
`gamma`	0	Min. loss reduction
`reg_lambda` (L2)	1	L2 regularization term
`reg_alpha` (L1)	0	L1 regularization term

For the neural network, the reference architecture is assigned the structure of a common feedforward Multi-Layer Perceptron (MLP). As shown in Table 3, this topology requires the use of two hidden layers, the application of dropout regularization, and the use of the Adam optimizer. This configuration serves as a fundamental reference for evaluating the structural stability inherent in neural networks in this specific domain, allowing the necessary comparison between baseline models and models subjected to evolutionary refinement to be added to the study.

Table 3. Default Hyperparameters for MLP Baseline

Hyperparameter	Value/Setting	Description
`Architecture`	$[32, 16]$	Neurons in hidden layers
`Activation`	ReLU	Rectified Linear Unit
`Output Activation`	Sigmoid	Probability estimation $(0, 1)$
`Optimizer`	Adam	Adaptive Moment Estimation
`Learning Rate`	0.001	Standard starting rate
`Dropout`	$[0.2, 0.1]$	Dropout rates for Layer 1 and Layer 2
`Initialization`	HeNormal	Weights initialization strategy
`Batch Size`	32	Samples per gradient update

Additionally, to validate the computational efficiency and search strategy of the GA, a Random Search (RS) algorithm was implemented as an optimization

benchmark for the XGBoost architecture. The RS was configured with a computational budget of 20,000 independent model evaluations, which is associated with mathematical equivalence to the GA population composed of 20 individuals subjected to evolution over 1,000 generations. The RS technique allowed sampling based on the limits of continuous and discrete hyperparameters, used by the GA (as detailed in Table 7), applying to this process the temporal cross-validation scheme and the fitness function identical to those of the evolutionary method.

4 Results

This section presents the experimental results obtained. Although evolutionary optimization and feature selection were confined to the 70% in-sample period using a 5-fold TSS cross-validation structure, the final comparative metrics, generalization gap and confusion matrices presented here are exclusively based on the isolated 30% out-of-sample test set. This evaluation ensures the absence of data leakage from the adjustment phase. The analysis follows a chronological structure: first, the identification of the feature subsets generated by the SFS is verified, subsequently, the evolutionary convergence process is detailed, a comparative analysis between the reference models and the models subjected to evolution is performed, and finally, the GA algorithm is assigned the role of reference for comparison with the RS algorithm.

4.1 Selected Features and SFS Analysis

To isolate indicators with predictive capacity and reduce dimensionality, the SFS execution employed an XGBoost classifier as a wrapper estimator. Weighted F1 was used as the primary evaluation metric due to its ability to handle class imbalances in financial datasets, providing the results with the harmonic mean between precision and recall [19].

The model's performance trajectory as attributes are sequentially added is illustrated in Fig. 2. The analysis reveals a saturation point in the predictive power beyond which the inclusion of new variables does not result in improved performance metrics.

As can be seen in Fig. 2 the F1-Score shows an improvement during the initial iterations, indicating the gain in information provided by the volatility and momentum indicators. However, the curve shows a plateau after the inclusion of the tenth attribute and beyond this point the addition of variables is linked to reduced returns and increased variance, which is consistent with the phenomenon of the "curse of dimensionality" [6].

Consequently, the subset with $K = 10$ was identified as the appropriate operating configuration. Nevertheless, the subset with $K = 11$ attributes was assigned an equivalent score (within the standard deviation), allowing the introduction of RSI_LAG1, which provides context relative to the previous state.

To ensure a rigorous evaluation, both subsets were allowed to advance to the evolutionary optimization stage.

Table 4 details the compositions related to the two subsets of attributes that showed satisfactory performance. It should be noted that the values in parentheses indicate the retrospective period (n) used to calculate each technical indicator (for example, $BB_{pos}(9)$ is associated with a calculation window of 9 periods).

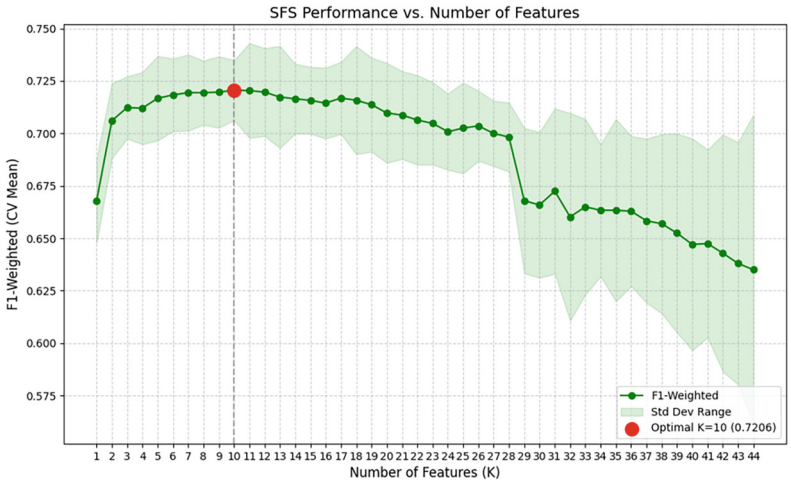

Fig. 2. Sequential Forward Selection (SFS) Performance.

Feature Analysis. The selected subsets are attributed a tendency towards Volatility Indicators, since most attributes are linked to variations in the positions of Bollinger Bands (BB_{pos}) arranged in different time windows. This empirical finding suggests that, for Mini Index futures contracts, the relative price position within its volatility range plays a more significant role as a predictor of immediate direction than trend-following metrics.

Additionally, the inclusion of $RETURN_LAG1$ and RSI_{diff} confirms the need to capture the immediate rate of change and historical shock. The second subset ($K = 11$) reinforces this structure by adding RSI_LAG1 allowing the model to observe the temporal trajectory of the moment. This composition of attributes is linked to the purpose of providing the system with a multi-scalar perception of price oscillations, ensuring that both the amplitude of volatility and the inertia of the previous movement are incorporated into the technical analysis.

4.2 Evolutionary Hyperparameter Optimization

After the attribute selection stage, the subsets of attributes considered suitable ($K = 10$ and $K = 11$) were assigned for use in the GA evaluation, which was

Table 4. Optimal Feature Subsets ($K = 10$ and $K = 11$) Selected by SFS

Rank	Feature Name	Technical Category
1	BB_{pos} (9)	Volatility
2	BB_{pos} (20)	Volatility
3	$RETURN_LAG1$	Microstructure / Autocorrelation
4	$CLOSEvsSMA_{20}$	Mean Reversion Distance
5	BB_{pos} (12)	Volatility
6	BB_{pos} (10)	Volatility
7	RSI_{diff}	Momentum Acceleration
8	BB_{pos} (25)	Volatility
9	RSI (6)	Momentum
10	BB_{pos} (18)	Volatility
Additional Feature for $K = 11$		
11	RSI_LAG1	Momentum Trajectory

configured to maximize a fitness function with a gap penalty (Eq. 2). To maintain computational feasibility, the evolutionary parameters (Table 1) were adapted to each architecture, the XGBoost model, with lower processing requirements, allowed for an expanded-scale exploration of the search space, while the MLP model was reserved a restricted optimization structure.

Both models were associated with the display of constant evolutionary convergence. The algorithm was assigned the application of penalties to configurations with initial overfitting directing the population to regions of stability in the hyperparameter space to which the reduction of the generalization gap and the preservation of predictive accuracy are linked, as detailed in the subsequent comparative analysis. This search process aims to ensure that the final result incorporates the robustness necessary for operation in data environments subject to fluctuations.

4.3 Comparative Analysis: Baseline vs. Evolved Models

To quantify the impact of evolutionary optimization, performance analysis was linked to reference models equipped with standard hyperparameters, in contrast to the architectures to which the evolutionary process was applied. The evaluation criterion in this research considers not only the test score but also the stability associated with the Generalization Gap ($|Train - Test|$).

XGBoost Baseline. To establish a performance benchmark, the XGBoost model was initially evaluated using its standard hyperparameters, without applying evolutionary adjustment. This configuration, detailed in Table 2, is characterized as a high-variance learner to which reduced regularization is applied.

The reference model was shown to be overfit in both attribute subsets. For $K = 10$, the training F1 score of 0.9129 dropped to 0.7230 in unobserved data, resulting in a Generalization Gap of 0.1899. Similarly, for the subset $K = 11$, the training score decreased from 0.9169 to a test score of 0.7212, equivalent to a gap of 0.1957. These discrepancies indicate that the model memorized market noise rather than learning structural patterns, as detailed in Table 5. This phenomenon highlights the need to look for configurations that demonstrate a superior ability to handle the complexity of financial data.

Table 5. Confusion Matrices for the XGBoost Baseline models, illustrating the classification performance on the unseen test set for both feature subsets.

(a) XGBoost Baseline $K = 10$

Actual\Pred	**Neg**	**Pos**
Neg	1537	624
Pos	648	1781

(b) XGBoost Baseline $K = 11$

Actual\Pred	**Neg**	**Pos**
Neg	1533	628
Pos	652	1777

Evolutionary Optimization for XGBoost. To optimize the XGBoost model, the fitness function with gap penalty (Eq. 2) was used. As can be seen in Figs. 3 and 4 both attribute configurations were linked to exhibiting convergence in the first 200 generations, preceding entry into a refinement phase focused on locating regions of stability in the hyperparameter space.

Furthermore, the observed variance between maximum and average fitness over 1000 generations demonstrates that the mutation operators were responsible for maintaining genetic diversity. This mechanism is associated with the prevention of premature convergence, while the algorithm was tasked with mapping the appropriate structural definitions. This behavior is attributed to the fact that the search process incorporates an exploration of the solution space, which ensures that the configuration is linked to the stability necessary for operation on data subject to fluctuations.

For the $K = 10$ subset, the GA was assigned the identification of an architecture linked to a training F1 score of 0.7331 and a test F1 score of 0.7192, resulting in a generalization gap of 0.0139. Similarly, the $K = 11$ subset was associated with a training F1 score of 0.7399 and a test F1 score of 0.7220, leaving a gap of 0.0179. These results confirm that the optimization process mitigated overfitting in different attribute sets. Table 6 contains a summary of the final classification performance obtained in the unobserved test set, as shown in the confusion matrices.

These results demonstrate that the GA successfully traded off the "perfect memorization" of the training set for a model that maintains consistent predictive power in the noisy environment of the Mini Futures Index.

The structural preferences of the evolved models are detailed in Table 7. Compared to the broad search space with the convergence optima, it is evident

Table 6. Confusion Matrices for the evolved XGBoost models, illustrating the classification performance on the unseen test set for both feature subsets.

(a) Evolved XGBoost $K = 10$

Actual\Pred	Neg	Pos
Neg	1494	667
Pos	621	1808

(b) Evolved XGBoost $K = 11$

Actual\Pred	Neg	Pos
Neg	1520	641
Pos	635	1794

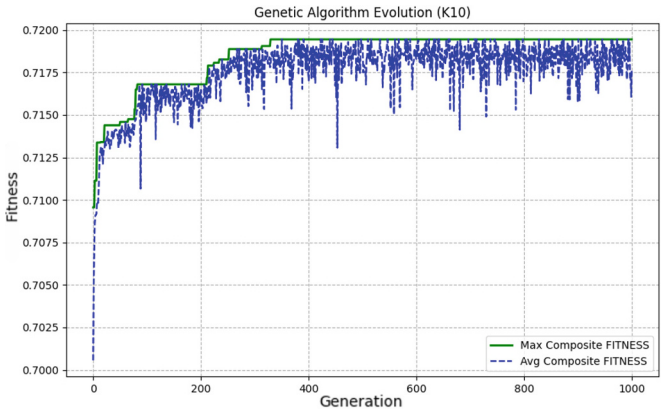

Fig. 3. GA Evolution for $K = 10$, showing Maximum and Average Fitness.

that the algorithm favored distinct regularization strategies for each subset. For instance, the $K = 10$ model minimized complexity via a shallow tree depth (3) and higher gamma, whereas $K = 11$ balanced slightly deeper trees (5) with increased L1 regularization (`reg_alpha`).

MLP Baseline. Similarly, the MLP was assigned the evaluation through a reference architecture to establish the neural network's performance parameter. The network topology and training configuration, detailed in Table 3, represent a common feed-forward network for binary classification.

Quantitatively, the MLP reference model for the subset $K = 10$ was linked to a training F1 score of 0.7270 and a test F1 score of 0.7140, resulting in a generalization gap of 0.0130. The $K = 11$ subset was associated with stability with training and test scores of 0.7292 and 0.7196, respectively, leaving a gap of 0.0096. These reduced margins confirm that even without evolutionary adjustment, the neural network architecture incorporates the mitigation of overfitting observed in the tree-based model. Table 8 presents the Confusion Matrices for the reference MLP models.

Evolutionary Optimization for MLP. For the Neural Network, the GA fine-tuned the balance between capacity (neurons) and regularization (Dropout/L2).

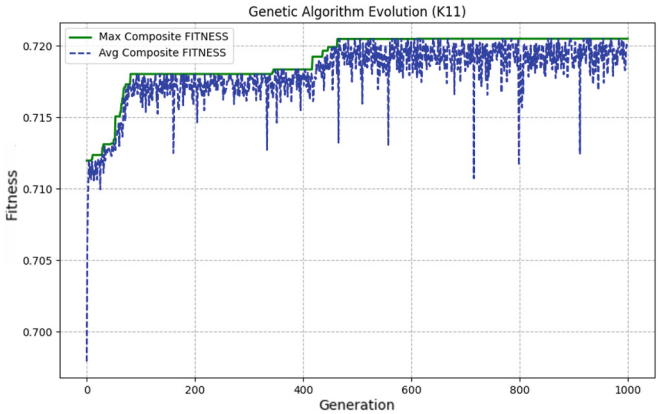

Fig. 4. GA Evolution for $K = 11$, showing Maximum and Average Fitness.

Table 7. XGBoost Hyperparameter Search Space and GA-Optimized Values

Hyperparameter	Search Space (Min - Max)	Optimal ($K = 10$)	Optimal ($K = 11$)
Structural Parameters			
`n_estimators`	[100, 1200]	807	894
`max_depth`	[3, 12]	3	5
Learning Dynamics			
`learning_rate`	[0.01, 0.30]	0.0612	0.1226
`subsample`	[0.50, 1.00]	0.6290	0.8548
`colsample_bytree`	[0.50, 1.00]	0.9355	0.9194
Regularization			
`gamma`	[0, 10]	7.3016	5.5556
`reg_alpha` (L1)	[0, 10]	0.0000	1.5294
`reg_lambda` (L2)	[0, 10]	2.2353	2.3529

As illustrated in Figs. 5 and 6, the algorithm efficiently navigated the search space, finding a robust local optimum within early generations and minimizing the generalization gap without requiring prolonged evolutionary exploration.

The parameters selected for the best MLP individuals are listed in Table 9. The algorithm favored moderate network sizes and specific L2 regularization, smoothing the decision boundary to prevent overfitting. The MLP optimization was highly successful in ensuring stability. For the $K = 10$ subset, the training and test F1-scores reached 0.7188 and 0.7182, respectively, producing a negligible generalization gap of 0.0006. The $K = 11$ subset yielded similarly robust results (0.7178 training, 0.7174 test, 0.0004 gap). However, because the baseline MLP already demonstrated strong inherent stability without evolutionary tuning, the GA functioned primarily as a fine-tuning mechanism. These marginal improve-

Table 8. Confusion Matrices for the MLP Baseline models, illustrating the classification performance on the unseen test set for both feature subsets.

(a) MLP Baseline $K = 10$

Actual\Pred	Neg	Pos
Neg	1553	608
Pos	706	1723

(b) MLP Baseline $K = 11$

Actual\Pred	Neg	Pos
Neg	1495	666
Pos	620	1809

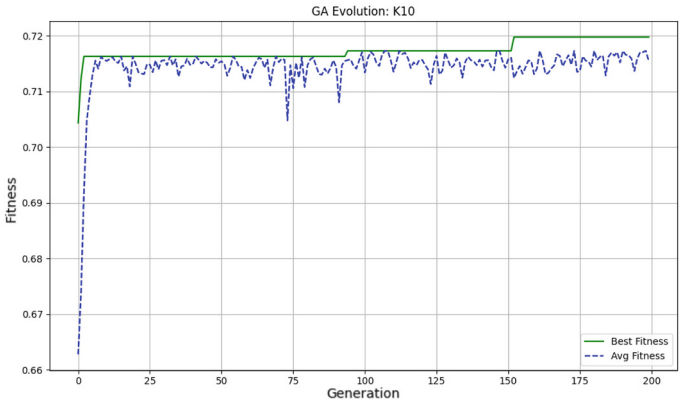

Fig. 5. Genetic Algorithm Evolution for MLP ($K = 10$), showing Maximum and Average Fitness.

ments over the default settings suggest that the substantial computational cost of evolving neural topologies offers diminishing returns for this specific architecture.

Genetic Algorithm vs. Random Search. To ensure that the evolutionary approach provided benefits over standard stochastic methods, the GA was benchmarked against the RS algorithm specifically for the XGBoost architecture. Under an identical budget of 20,000 evaluations, both methods successfully mitigated baseline overfitting and achieved statistically equivalent predictive power. For instance, in the $K = 10$ subset, the GA produced a test F1-score of 0.7192 with a generalization gap of 0.0139, compared to the RS's test F1-score of 0.7217 and a gap of 0.0151. Similarly, in the $K = 11$ configuration, GA yielded a test F1-score of 0.7220 with a generalization gap of 0.0179, while RS achieved an F1-score of 0.7174 with a gap of 0.0166.

However, the execution revealed a consistent superiority of the GA in computational efficiency, although the margin of this advantage proved to be highly context-dependent. For the $K = 10$ configuration, the RS required an execution time nearly 92% longer than the GA to exhaust the same budget. In contrast, in the $K = 11$ subset the performance margin decreased, with RS taking approximately 10% longer than GA to complete the optimization. This variation

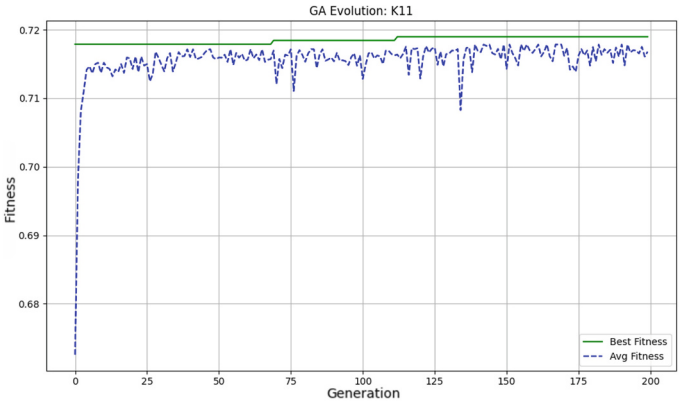

Fig. 6. Genetic Algorithm Evolution for MLP ($K = 11$), showing Maximum and Average Fitness.

Table 9. MLP Hyperparameter Search Space and GA-Optimized Values

Hyperparameter	Search Space	Optimal ($K = 10$)	Optimal ($K = 11$)
Architecture & Optimization			
`neurons_layer1`	$[16, 64]$	33	53
`learning_rate`	$[1e^{-5}, 5e^{-3}]$	0.0024	0.0026
`batch_size`	$\{16, 32, 64\}$	16	32
Regularization			
`l2_alpha`	$[1e^{-4}, 1e^{-2}]$	1.18×10^{-4}	7.89×10^{-4}
`dropout_rate`	$[0.20, 0.60]$	0.32	0.21

indicates a characteristic of the evolutionary process. When a satisfactory architecture is found in shallow regions with regularization (as in $K = 10$), the GA reduces the time spent by applying penalties to complex trees at the beginning of execution. On the other hand, when the evolutionary pressure converges to an architecture with structural complexity, such as the topology required for the $K = 11$ subset, the time to train each candidate increases, reducing the efficiency difference. However, even in this more demanding scenario, the GA's targeted search provided structural stability more quickly than random sampling.

5 Conclusions

The study presented an optimization flow aimed at reducing overfitting in the directional forecast of the Mini Bovespa Future Index (WIN). By integrating forward SFS with evolutionary parameter tuning, the study analyzed the balance between model complexity and generalization capacity in the XGBoost and MLP architectures.

The experimental results reveal a dichotomy with respect to the need for evolutionary adjustment. For the XGBoost architecture, GA proved to be a necessary component for viability. The reference models were assigned a generalization gap greater than 0.18. In a domain where the reliability of backtests is valued, such a discrepancy renders the model unreliable for real-world execution. The evolutionary process served as a dynamic regularization tool, which reduced this gap to less than 0.02 and established a balance between theoretical predictive power and operational safety.

In contrast, MLP demonstrated its own structural stability. Although GA provided refinement to the generalization gap to reduced levels, the performance gain suggests a scenario of diminishing returns. Even though GA provides targeted and efficient search, the reference stability of the neural network indicates that parameter tuning offers limited practical benefits. This contrast indicates that, while evolutionary optimization is necessary to impose limits on high-variance models like XGBoost, it serves the fine-tuning function for architectures with natural stability.

Moreover, the results observed in the Mini Bovespa Index suggest that the application of this methodology can encompass other financial assets. The structure that integrates the sequential selection of attributes with evolutionary adjustment allows the model's conformation to the noise levels and volatility variations inherent in different instruments of the futures market.

Future research will focus on expanding the attribute space to incorporate order flow dynamics and exploring Multi-Objective Evolutionary Algorithms (MOEAs) [5] to optimize financial metrics such as the Sharpe Ratio. Finally, by prioritizing robustness over memorability and considering market friction, this study provides a basis for autonomous trading systems capable of maintaining viability in fluctuating financial environments [23].

Acknowledgment. The authors would like to thank FAPERGS (24/2551-0001396-2, 23/2551-0000773-8), CNPq (307416/2025-9), FAPERGS/CNPq (23/2551-0000126-8) and the Fesurv-University of Rio Verde.

References

1. Abadi, M., et al.: TensorFlow: Large-scale machine learning on heterogeneous distributed systems (2015). Software available from tensorflow.org
2. Bollinger, J.: Bollinger on Bollinger Bands. McGraw-Hill Professional (2002)
3. Cerqueira, V., Torgo, L., Mozetič, I.: Evaluating time series forecasting models: an empirical study on performance estimation methods. Mach. Learn. **109**(11), 1997–2028 (2020)
4. 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 (2016)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-ii. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)

6. Dhal, P., Azad, C.: A comprehensive survey on feature selection in the various fields of machine learning. Appl. Intell. **52**(4), 4543–4581 (2022)
7. Goldberg, D.E.: Genetic Algorithms in Search. Addison-Wesley, Optimization and Machine Learning (1989)
8. Gu, S., Kelly, B., Xiu, D.: Empirical asset pricing via machine learning. Rev. Fin. Studies **33**(5), 2223–2273 (2020)
9. Hasbrouck, J.: Empirical Market Microstructure: The Institutions, Economics, and Econometrics of Securities Trading. Oxford University Press (2007)
10. Katoch, S., Chauhan, S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimedia Tools Appl. **80**(5), 8091–8126 (2021)
11. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**(1–2), 273–324 (1997)
12. Krauss, C., Do, X.A., Huck, N.: Deep neural networks, gradient-boosted trees, random forests: statistical arbitrage on the s&p 500. Eur. J. Oper. Res. **259**(2), 689–702 (2017)
13. Lopez de Prado, M.: Advances in Financial Machine Learning. John Wiley & Sons (2018)
14. Lopez de Prado, M.: Machine learning for asset managers. Cambridge University Press (2020)
15. Nti, I.K., Adekoya, A.F., Weyori, B.A.: A systematic review of fundamental and technical analysis of stock market predictions. Artif. Intell. Rev. **53**(4), 3007–3057 (2020)
16. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
17. Python Software Foundation: Python language reference, version 3.10 (2021). https://www.python.org
18. Sezer, O.B., Gudelek, M.U., Ozbayoglu, A.M.: Financial time series forecasting with deep learning: a systematic literature review: 2005–2019. Appl. Soft Comput. **90**, 106181 (2020)
19. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Information processing & management **45**(4), 427–437 (2009)
20. Souza, A.S., Lucca, G., Borges, E.N., Cardoso, F.C., Dalmazo, B.L., Berri, R.: Dataset for Intraday Analysis of B3 stock prices (2024). https://doi.org/10.7910/DVN/TMB4IG
21. Thabtah, F., Hammoud, S., Kamalov, F., Gonsalves, A.: Data imbalance in classification: experimental evaluation. Inf. Sci. **513**, 429–441 (2020)
22. Wilder, J.W.: New Concepts in Technical Trading Systems. Trend Research, Greensboro, NC (1978)
23. Yang, H., Liu, X.Y., Zhong, S., Walid, A.: Deep reinforcement learning for automated stock trading: An ensemble strategy. In: Proceedings of the First ACM International Conference on AI in Finance, pp. 1–8 (2020)