Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Feature engineering through two-level genetic algorithm

Aditi Gulati [a], Armin Felahatpisheh [b], Camilo E. Valderrama [b,c,*]

[a] Computer Science and Engineering Department, Indira Gandhi Delhi Technical University for Women, Delhi, 110006, India
[b] Department of Applied Computer Science, University of Winnipeg, 515 Portage Avenue, Winnipeg, R3B 2E9, MB, Canada
[c] Department of Community Health Sciences, Cumming School of Medicine, University of Calgary, 3280 Hospital Drive NW, Calgary, T2N 4Z6, AB, Canada

ARTICLE INFO

Keywords:
Feature engineering
Multi-objective genetic algorithms
Ensemble learning
Machine learning
Feature selection
Model interpretability

ABSTRACT

Deep learning models are widely used for their high predictive performance, but often lack interpretability. Traditional machine learning methods, such as logistic regression and ensemble models, offer greater interpretability but typically have lower predictive capacity. Feature engineering can enhance the performance of interpretable models by identifying features that optimize classification. However, existing feature engineering methods face limitations: (1) they usually do not apply non-linear transformations to features, ignoring the benefits of non-linear spaces; (2) they usually perform feature selection only once, failing to reduce uncertainty through repeated experiments; and (3) traditional methods like minimum redundancy maximum relevance (mRMR) require additional hyperparameters to define the number of selected features. To address these issues, this study proposed a hierarchical two-level feature engineering approach. In the first level, relevant features were identified using multiple bootstrapped training sets. For each training set, the features were expanded using seven non-linear transformation functions, and the minimum feature set maximizing ensemble model performance was selected using the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In the second level, candidate feature sets were aggregated using two strategies. We evaluated our approach on twelve datasets from various fields, achieving an average F1 score improvement of 1.5% while reducing the feature set size by 54.5%. Moreover, our approach outperformed or matched traditional filter-based methods. Our approach is available through a Python library (*feature-gen*), enabling others to benefit from this tool. This study highlights the utility of evolutionary algorithms to generate feature sets that enhance the performance of interpretable machine learning models.

1. Introduction

Data-driven applications are ubiquitous in various fields, including critical areas such as military defense, healthcare, and banking. This widespread use of data-driven approaches is possible due to advances in machine learning and artificial intelligence, which allow learning from data sets with minimal human involvement. Specifically, deep learning has become the preferred option for data-driven systems due to their ability to produce accurate predictions. However, although deep learning models can achieve high performance in prediction tasks, most cannot explain their predictions, making the detection of untrustworthy, unreliable, unethical, and biased predictions challenging (Arrieta et al., 2020).

An alternative approach to provide more interpretation is to use classical machine learning models, such as logistic regression, decision trees, or rule-based, which can explain the rationale behind their predictions. However, the problem with using these simpler machine learning models is that they usually achieve a lower performance than deep learning models (Aceves-Fernandez, 2020). To address this disadvantage, additional tasks can be performed to increase the performance of these machine learning models. In particular, feature engineering, a task relegated after the emergence of deep learning, can improve the performance of machine learning models (Nargesian et al., 2017).

Feature engineering involves selecting, creating, or transforming features to improve the accuracy of the model. This process requires domain knowledge to define a set of candidate features, which are manually tested in a trial-and-error approach. As this trial-and-error approach is both challenging and time-consuming, feature engineering is not always included in the preprocessing steps for building machine learning models (Khurana, 2018).

Given the benefits but also the challenges of performing feature engineering manually, it is worth exploring automated approaches to select and produce relevant features. These automated approaches

* Corresponding author at: Department of Applied Computer Science, University of Winnipeg, 515 Portage Avenue, Winnipeg, R3B 2E9, MB, Canada.
E-mail addresses: aditi021btcse21@igdtuw.ac.in (A. Gulati), felahatpisheh-a@webmail.uwinnipeg.ca (A. Felahatpisheh), c.valderrama@uwinnipeg.ca (C.E. Valderrama).

https://doi.org/10.1016/j.mlwa.2025.100696
Received 20 April 2025; Received in revised form 10 June 2025; Accepted 21 June 2025
Available online 4 July 2025

require exploring a set of features to derive or select a set of features that maximize predictive performance. A common practice for this is to use heuristic methods that explore combinations of features and select those with higher predictive performance.

Among heuristic approaches, filter-based feature selection methods are the most widely used. These methods evaluate the relationship between each feature and the response variable, often through correlation analysis or hypothesis testing. However, they do not account for potential interactions or combinations of features. Additionally, users must pre-define the number of features to select, a task that can be challenging without prior knowledge. This limitation often necessitates tuning extra hyperparameters, which further exacerbates computational complexity and increases the number of required operations.

An alternative to filter-based feature selection is genetic algorithms (GAs), which allow for exploring feature combinations while eliminating the need to pre-define the number of features to select (Li et al., 2017; Zhang & Yang, 2008). Additionally, GAs can simultaneously optimize multiple objectives, making them particularly valuable for producing feature sets that not only maximize predictive performance, but also minimize the number of selected features (Rostami et al., 2021; Wang et al., 2020). This latter property is especially valuable, as reducing the feature set facilitates understanding how input variables influence predictions, thereby improving the interpretability and explainability of the model outputs (Campagner & Cabitza, 2020).

Previous studies have highlighted the potential of GAs for feature engineering, emphasizing their ability to optimize multiple objectives, such as maximizing classification performance while minimizing the number of selected features. However, previous studies face some limitations. Many approaches have been tested primarily on small to moderately sized datasets, limiting their generalizability across diverse domains and larger datasets. Moreover, prior research has ignored the use of statistical techniques, such as bootstrap resampling, to explore diverse alternative solutions and identify consistent features. Additionally, the use of non-linear transformations to improve predictive performance has been largely unexplored. Given the success of non-linear transformations in deep learning, integrating similar strategies into feature engineering for traditional machine learning models holds promise for improved performance. Therefore, further research is needed to develop GA-based feature engineering approaches that incorporate non-linear transformations, use robust statistical techniques, and evaluate their effectiveness across extensive and diverse datasets.

In this study, we advance this field by introducing a hierarchical feature engineering algorithm based on multi-objective genetic algorithms (GAs). Our approach employs bootstrap sampling to identify and select robust and consistent features across diverse training samples. Additionally, it incorporates non-linear transformations, including logarithmic, cubic, and sigmoid functions, to enhance predictive performance. To balance accuracy and interpretability, we use Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002) to simultaneously maximize classification performance and minimize the feature set size. Overall, the main contributions of our work are summarized as follows:

- Propose a hierarchical, multi-objective GA framework that incorporates bootstrap resampling and nonlinear feature transformations to produce a reduced feature set that enhances the performance of interpretable machine learning models.
- Validate the proposed approach on 12 diverse datasets, demonstrating its effectiveness across multiple domains.
- Provide an open-source Python (*feature-gen*) implementation to promote adoption and reproducibility in practical machine learning workflows (Felahatpisheh et al., 2025).

2. Literature review

Previous studies have explored the potential of evolutionary algorithms for feature engineering (Li et al., 2017; Zhang & Yang, 2008). For example, Ali and Saeed (2023) proposed a hybrid method that combines filter-based feature selection techniques, such as information gain and Chi-squared, with GAs for selecting features in cancer classification using high-dimensional microarray datasets. While their method improved classification accuracy, its focus on specific domains like bioinformatics limits its applicability to other types of data. Similarly, Shi and Saad (2023) incorporated GAs within AutoML frameworks to dynamically generate and select features through iterative optimization, which involves transformation, combination, and synthesis. However, this approach faces challenges related to computational complexity and reliance on specific AutoML tools, restricting its scalability.

Other research has examined multi-objective optimization in feature engineering (Rostami et al., 2021; Wang et al., 2020), utilizing evolutionary algorithms like the Non-dominated NSGA-II to balance competing objectives. These studies illustrate that evolutionary algorithms can effectively manage the trade-offs between model accuracy and simplicity, leading to more efficient feature selection.

In addition to feature selection via GAs, an important aspect of identifying optimal feature subsets is leveraging machine learning models to assess their predictive capacity. Kiziloz (2021) compared individual machine learning classifiers with ensemble methods for feature selection and found that ensemble approaches yield more robust feature sets. In particular, a meta-classifier composed of Logistic Regression, Support Vector Machines (SVM), Extreme Learning Machine (ELM), Naive Bayes, and Decision Trees outperformed single-model approaches, such as Gradient Boosting trees.

Building on these foundations, our study introduces a hierarchical GA-based feature engineering framework. We use the NSGA to optimize two objectives simultaneously: maximizing predictive performance and minimizing the number of selected features, which enhances both interpretability and efficiency. We utilize bootstrap resampling to explore diverse feature combinations and evaluate candidate subsets through ensemble modeling. Finally, we validate the generalizability of our approach across twelve datasets spanning various domains.

3. Methodology

Fig. 1 shows the flowchart of our proposed GA-based feature engineering method. This approach employed a hierarchical, two-level algorithm. At the first level, feature selection was performed independently on three bootstrap sample sets derived from the training dataset. The decision to use three bootstrap sets was based on selecting the smallest odd number greater than one, as odd numbers help resolve ties when aggregating features. Additionally, using a small number of bootstrap sets reduces computational demands, making the approach more practical and resource-efficient.

At the second level, the selected features from these bootstrap sets were aggregated to identify an optimal feature set. Two different strategies were explored for the combination of the features. The following subsections provide a detailed description of each component of the micro-and-macro genetic algorithm feature selection approach.

3.1. Feature transformation

To explore non-linear transformations that could enhance the predictive performance of our models, we applied transformation techniques to the numerical variables, using seven distinct techniques: quadratic, cubic, square root, cubic root, logarithmic, sigmoid, and tanh. These transformations address non-normally distributed features by mapping them to a new space that facilitates classification. For example, logarithmic transformations can be effective for skewed distributions, while quadratic and cubic transformations help capture non-linear relationships. Moreover, similar to deep learning models, the transformed features help capture non-linear data patterns, thereby improving classification performance.

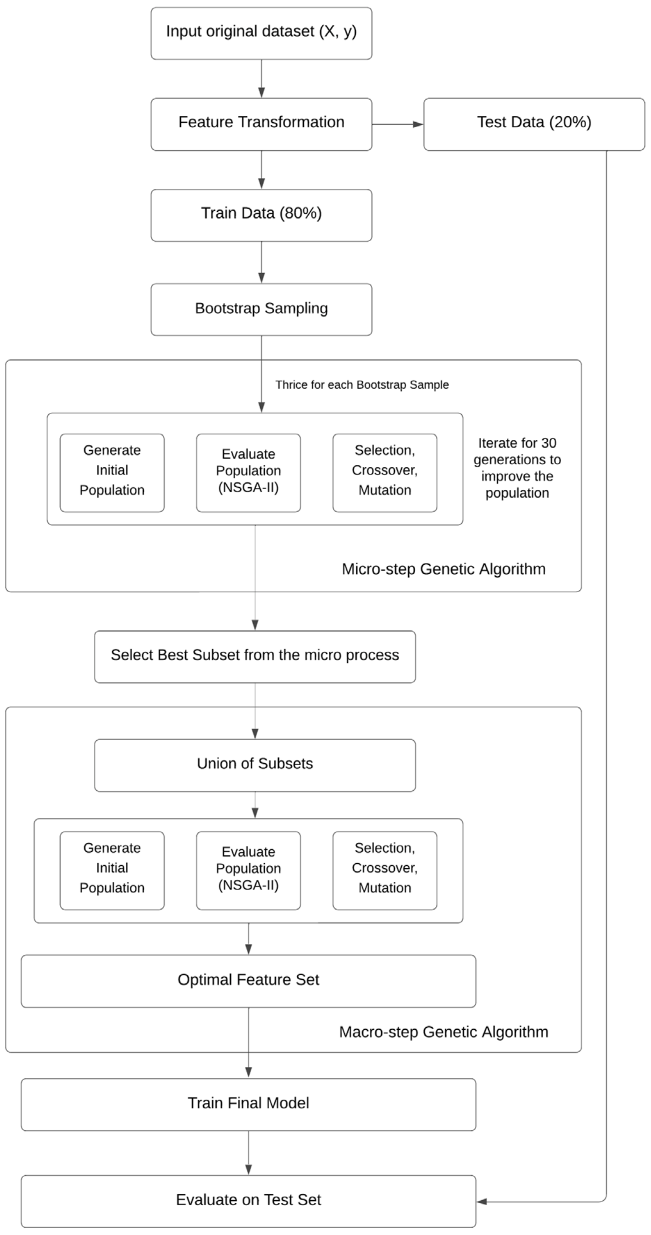
Input original dataset (X, y)
Feature Transformation
Test Data (20%)
Train Data (80%)
Bootstrap Sampling
Thrice for each Bootstrap Sample
Generate Initial Population
Evaluate Population (NSGA-II)
Selection, Crossover, Mutation
Iterate for 30 generations to improve the population
Micro-step Genetic Algorithm
Select Best Subset from the micro process
Union of Subsets
Generate Initial Population
Evaluate Population (NSGA-II)
Selection, Crossover, Mutation
Optimal Feature Set
Macro-step Genetic Algorithm
Train Final Model
Evaluate on Test Set

Fig. 1. Flowchart used to select the best features throughout the macro–micro genetic algorithm.

Categorical variables (i.e., non-numerical variables) were transformed using one-hot encoding. This encoding creates a separate column for each unique value of the categorical variable.

These transformations were concatenated to the original feature set. Thus, the feature dimensionality was extended from $|F|$ to $|F_{numeric}| + 7 \times |F_{numeric}| + \sum_{f \in F_{categorical}} |V_f|$, where $|F|$ was the original number of features in the dataset, $|F_{numeric}|$ was the number of numerical features, and $|V_f|$ was the total number of categories of the fth categorical feature.

3.2. Training and test split

After applying the transformations, the dataset was split into training and test sets with an 80% to 20% ratio. The splitting process was stratified to maintain the same class distribution in both the training and test sets.

3.3. Bootstrap sampling

To create multiple subsets of the original dataset, we applied bootstrap sampling on training data $\mathcal{D}$ to generate three independent datasets D_1, D_2, and D_3, as follows:

$$\begin{tikzcd} & D_1 \\ \mathcal{D} \arrow[ur] \arrow[r] \arrow[dr] & D_2 \\ & D_3 \end{tikzcd}$$

Bootstrap sampling is a statistical technique that generates new datasets from the original data by random sampling with replacement, ensuring diversity in each generated sample. We used this technique to enable a more robust exploration of feature combinations by deriving candidate feature sets independently from each bootstrap sample. These candidate sets were then aggregated in the second stage to identify features that were consistently selected across all training samples.

3.4. Micro-step genetic algorithm

In the first level, named the 'micro' step, we applied our GA-based approach to each of three bootstrap sampling (D_1, D_2, and D_3). For each of these bootstrap samples, the approach comprised four main steps:

3.4.1. Step 1: Generate initial population

The first step involved generating the initial population, which consisted of 40 individuals, also known as chromosomes. Each chromosome had a length equal to the total number of features and represented a feature combination. The inclusion or exclusion of a feature was indicated by a binary bit. For example, assuming a total of ten features, the binary sequence '1000001111' corresponded to a feature vector containing the first and the last four features.

3.4.2. Step 2: Evaluation of the population

The second step used the NSGA-II algorithm to evaluate each of the 40 chromosomes by considering two objectives. The first objective was to maximize the prediction performance using F1 score, while the second objective was to minimize the dimensionality of the feature set. Specifically, for each of the 40 chromosomes, the bootstrap samples were used to train an ensemble model using the features indicated by the chromosome. Then, the samples contained in the training data $\mathcal{D}$ but not in the bootstrap sampling (out-of-bag samples) were used to compute the F1-score. The ensemble classifier was composed of logistic regression, SVM, and XGBoost. The prediction of the models was aggregated using majority voting. The dimensionality of the feature set was quantified as the total number of ones in the chromosome, representing the selected features.

3.4.3. Step 3: Genetic operations

In the third step, the NSGA-II performed a binary tournament selection to choose pairs of chromosomes for crossover and mutation. For crossover, parents with the better fitness scores were selected, and a new child (chromosome) was created by using single-point and multi-point crossover, where genes were swapped between parents to produce new combinations. For mutation, two individuals were chosen at random, and the one with the best fitness was selected as a parent. This parent produced a new chromosome by randomly changing some of its genes. The probability of mutation was predefined, ensuring variety within the population.

3.4.4. Step 4: Generation iterations

The new generation was evaluated again by the NSGA-II following the previously described procedure. This cycle was repeated for 30 generations. At the conclusion of the 30th generation, the chromosome with the highest fitness score was selected as the optimal feature set for the corresponding bootstrap sample.

3.4.5. Outcome: Micro-step candidate feature sets

After executing the genetic algorithm on each bootstrap sample, the micro-step yielded three optimal feature sets $[F_1, F_2, F_3]$, as:

$$\begin{tikzcd} D_1 \arrow[r] & F_1 \\ D_2 \arrow[r] & F_2 \\ D_3 \arrow[r] & F_3 \end{tikzcd}$$

3.5. Macro-step genetic algorithm

In the macro-step, the three feature sets obtained for each bootstrap sample in the micro-step were combined to further identify an optimal feature set, F_{macro}. Two different strategies were considered for combining these three feature sets (F_1, F_2, and F_3): bitwise union and the bitwise median.

The union operator was applied for each feature across the three feature sets, resulting in 1 when at least that feature was selected in any of the feature sets. On the other hand, the median operator only selected a feature if at least two feature sets selected that feature. The union operator allowed more candidates to be kept from the feature set, whereas the median focused more on selecting features that were more consistent across all three candidate feature sets. Regardless of the operator, the macro-step aggregated the micro outputs as:

$$\begin{tikzcd} F_1 \arrow[dr] & & \\ F_2 \arrow[r] & F_{\text{macro}} \arrow[r] & F_Z \\ F_3 \arrow[ur] & & \end{tikzcd}$$

Then, F_{macro} was passed through the same four steps of the genetic algorithm with a new bootstrap sample to find the final feature set, F_z. Like the micro-step, in the macro-step, the GA found F_z by maximizing the F1 score and minimizing the dimensionality of the final feature set.

3.6. Train final model

The optimal feature set F_Z, identified by the micro–macro hierarchical algorithm, was used to train a final ensemble model using the training dataset. This final model was then evaluated on the test data, which had been kept separate during the micro–macro steps.

4. Experiments

To evaluate the performance of our method, we conducted experiments on twelve different datasets. We compared our proposed GA-based feature selection approach with three widely used filter methods: chi-square, mutual information, and mRMR. To promote adoption and reproducibility in practical machine learning applications, we have also made our GA-based method available as an open-source Python library, *feature-gen* (Felahatpisheh et al., 2025).

4.1. Datasets

Table 1 presents the datasets used in this study, spanning diverse domains such as healthcare, finance, video games, and biology. These datasets include both binary and multi-class classification scenarios, with varying numbers of features, instances, and class distributions. The diversity of these datasets allows for a comprehensive evaluation of the NSGA-II algorithm to balance our two objectives: maximizing classification performance while minimizing the feature set size. Furthermore, this varied dataset collection provided a robust framework for assessing the algorithm's scalability, adaptability, and generalizability across different contexts.

For all datasets, rows with missing or undefined values were removed to maintain data quality and avoid errors in processing. Additionally, we dropped rows with positive and negative infinity values.

Table 1
Details of the datasets used in the experiments.

Dataset	Dataset ID	Number of features	Number of classes	Number of instances
Covertype	CT	54	7	581,012
Mushrooms	MR	22	2	8124
Spambase	SB	57	2	4601
Nursery	NU	8	5	12,960
Connect-4 opening	C4	42	3	67,557
Waveform	WF	21	3	5000
Financial	FI	3	2	17,108
Pima indian diabetes	PM	8	2	768
Breast cancer	BC	9	2	699
Ionosphere	IO	34	2	351
Wisconsin breast cancer	WBC	30	2	569
Musk (Version 2)	MU	168	2	6598

4.2. Micro–macro approach performance

For each dataset, we compared the performance obtained using the original feature set with that provided by our micro–macro algorithm. The only transformation applied to the original feature set was one-hot encoding for categorical variables, as machine learning models require numerical inputs. To evaluate performance, we used accuracy and F1 score, with the latter providing a less biased metric for imbalanced data. Additionally, we measured feature reduction by comparing the original and final feature sets. This evaluation was conducted using the two macro-step strategies: the union and median operators.

4.2.1. Comparison with filter feature selection methods

We also compared our macro–micro method with three common filter feature selection methods. These three methods were chi-square, mutual information, and mRMR methods. To ensure consistency and a fair comparison, we trained an ensemble model for each dataset using the same training set used for the macro–micro approach. Moreover, similar to the original dataset, the only transformation applied in the filter-based method was one-hot encoding for the categorical variables. The 20%, 30%, and 50% top features provided by the chi-square, mutual information, and mRMR were selected, and the final accuracy was calculated on the test set.

4.3. The feature-gen library

The micro–macro approach was implemented in a publicly accessible library named feature-gen. The feature-gen library is a Python-based tool that simplifies feature engineering for classification tasks. With a user-friendly API, it enables users to define their dataset, target column, and desired ensemble methods while providing full control over configuration parameters such as the number of generations, population size, and optimization settings. The feature-gen library is available on the PyPI repository (Felahatpisheh et al., 2025).

5. Results

This section presents the results obtained from evaluating our Micro–Macro Approach on twelve datasets (see Table 1). The most frequently selected feature transformations by the approach are also reported. Finally, a comparison with traditional filter-based feature selection methods is provided.

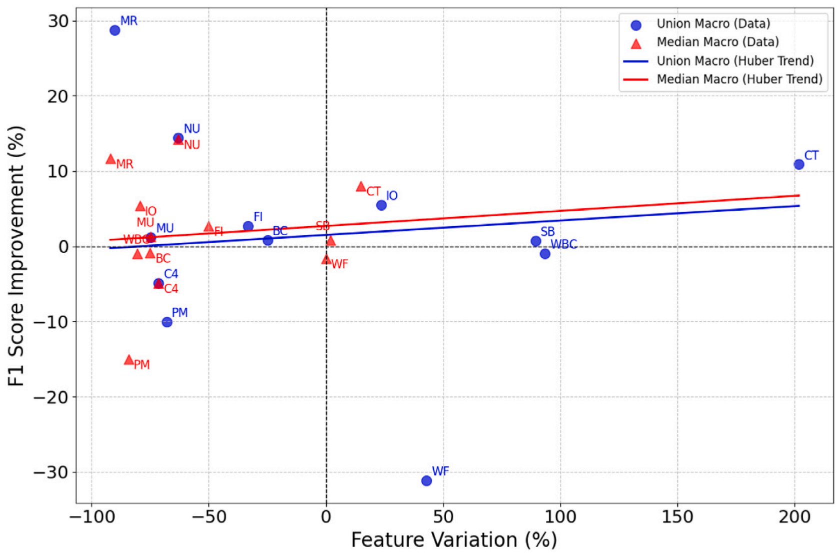

Fig. 2. Trade-off between F1 score and the number of feature variations for the union operator (blue line) and median operator (red line).

Table 2
Performance of the ensemble model composed of logistic regression, SVM, and XGBoost using all the original features on the twelve datasets. For each dataset, accuracy, F1 score, and feature count are reported. Categorical features were transformed using one-hot encoding.

Dataset	Train samples	Test samples	Accuracy	F1 score	No. of features
CT	464,809	116,203	77.2	52.2	54
MR	6499	1625	55.1	47.4	112
SB	3680	921	94.1	93.8	57
NU	9823	2456	94.1	74.4	27
C4	54,045	13,512	78.7	54.2	126
WF	4000	1000	83.9	83.8	21
FI	8000	2000	97.3	69.7	3
PM	614	154	63.6	59.4	1262
BC	559	140	96.4	96.1	20
IO	280	71	91.5	90.4	34
WBC	455	114	98.2	98.1	31
MU	5278	1320	99.1	98.2	6866
Average	46,504	11,626	85.8	76.5	717.8

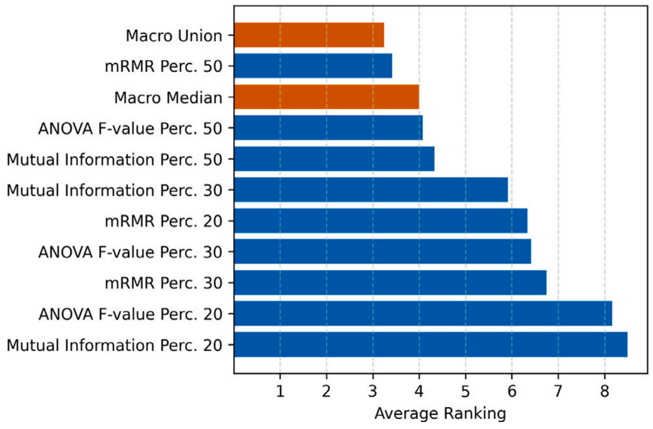

Fig. 3. Average rankings across 12 datasets based on F1 scores, comparing the micro–macro approaches (orange bars) with filter-based feature selection methods (blue bars), including Chi-square, Mutual Information, and mRMR.

5.1. Micro–macro approach performance

Table 2 presents the performance of the ensemble model using the original feature sets, while Tables 3 and 4 show the results of the micro–macro approach with the union and median operators, respectively. With the union operator, the accuracy and F1 score improved by 1.4% and 1.5%, respectively, across the 12 datasets. However, this improvement came at the cost of a 2.1% increase in average feature dimensionality. In contrast, the median operator not only achieved a slightly higher average predictive performance increase (1.6%) but also significantly reduced the average feature dimensionality by 54.5%. This highlights the median operator's ability to balance predictive performance and feature reduction effectively.

Fig. 2 shows the relationship between the F1 score and the number of feature variations achieved by the micro–macro approach. Both methods exhibited a positive trend, showing that an increase in the number of features increased the F1 score. However, the trend line indicates that reducing the number of features by 1% to 70% did not significantly affect the F1 score. In fact, within this range, the F1 score variation was positive for most of the tested datasets.

5.1.1. Feature transformations

Tables 5 and 6 present the percentage of features selected for each transformation. For the union operator, the logarithmic transformation was the most frequently selected, followed by the cubic power and sigmoid transformations. For the median operator, the quadratic power transformation was the most effective, followed by the sigmoid and cubic power transformations. For datasets containing only categorical variables (FI, C4, PM, MR, and NU), no transformations were selected as the transformations were only applied to numerical variables. The datasets WBC, SB, and WF used the highest proportion of transformed features, with less than 12% of the original features (NT) selected.

5.1.2. Comparison with traditional feature selection methods

Tables 7, 8, 9 present the accuracy and F1 score performance across 12 datasets for the chi-square, mutual information, and mRMR filtering methods, each selecting the top 20%, 30%, and 50% of the features. All three methods achieved strong classification results, with average accuracy exceeding 82% and F1 scores surpassing 72%. The 50th percentile consistently delivered the best performance, indicating that retaining half of the features produced an optimal balance between performance and dimensionality reduction.

Fig. 3 shows the average rankings based on F1 scores for the micro–macro algorithm compared to nine filter-based feature selection methods. The micro–macro approach using the union operator achieved the highest overall ranking across the 12 benchmark datasets.

Table 3
Performance of the GA-based hierarchical micro–macro approach across 12 datasets using the union operator in the macro step and an ensemble model composed of logistic regression, SVM, and XGBoost. For each dataset, accuracy, F1 score, and feature count are reported. The last three columns show variations compared to the results obtained using the original features (Table 2).

Dataset	Micro-Macro, Union operator			Accuracy	F1	No. feature
	Accuracy	F1 score	No. features	variation (%)	variation(%)	variation(%)
CT	77.9	57.9	163	1.0	10.9	201.9
MR	65.4	61.0	11	18.6	28.8	−90.2
SB	94.7	94.4	108	0.6	0.7	89.5
NU	92.5	85.1	10	−1.7	14.4	−63.0
C4	76.7	51.5	36	−2.4	−4.9	−71.4
WF	83.4	57.7	30	−0.6	−31.1	42.9
FI	97.2	71.6	2	−0.10	2.69	−33.3
PM	61.7	53.4	404	−3.1	−10.1	−68.0
BC	97.1	96.9	15	0.7	0.8	−25.0
IO	95.8	95.3	42	4.6	5.5	23.5
WBC	97.4	97.1	60	−0.9	−1.0	93.5
MU	99.7	99.4	1739	0.6	1.2	−74.7
Average	86.6	76.8	218.3	1.4	1.5	2.1

Table 4
Performance of the GA-based hierarchical micro–macro approach across 12 datasets using the median operator in the macro step and an ensemble model composed of logistic regression, SVM, and XGBoost. For each dataset, accuracy, F1 score, and feature count are reported. The last three columns show variations compared to the results obtained using the original features (Table 2).

Dataset	Micro-Macro, Median operator			Accuracy	F1	No. feature
	Accuracy	F1 score	No. features	variation (%)	variation(%)	variation(%)
CT	77.8	56.4	62	0.8	8.0	14.8
MR	63.8	52.9	9	15.6	11.7	−92.0
SB	93.4	93.1	58	−0.8	0.8	1.8
NU	92.3	84.9	10	−2.0	14.2	−63.0
C4	76.7	51.5	36	−2.4	−4.9	−71.4
WF	82.5	82.5	21	−1.7	−1.6	0.0
FI	97.2	71.6	2	−0.1	2.7	−50.0
PM	60.4	50.4	199	−5.1	−15.0	−84.2
BC	95.7	95.2	5	−0.7	−0.9	−75.0
IO	95.8	95.2	7	4.6	5.4	−79.4
WBC	97.4	97.1	6	−0.9	−1.0	−80.6
MU	99.7	99.4	1739	0.6	1.3	−74.7
Average	86.0	77.5	179.5	0.7	1.6	−54.5

Table 5
Percentage of features selected by each of the seven transformations and the original features (NT: no transformation) from the feature set produced by the micro–macro approach using union operator. The last row shows the average across 12 datasets.

Dataset	Union operator for macro step							
	Quadratic	Cubic	log	Square root	Cubic root	tanh	Sigmoid	NT
CT	9.2	14.7	15.3	11.0	14.1	11.7	11.7	12.3
MR	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
SB	13.0	9.3	13.9	13.0	13.0	17.6	9.3	11.1
NU	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
C4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
WF	3.3	6.7	20.0	13.3	20.0	13.3	13.3	10.0
FI	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
PM	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
BC	20.0	13.3	6.7	6.7	0.0	6.7	13.3	33.3
IO	9.5	11.9	21.4	9.5	11.9	9.5	16.7	9.5
WBC	11.7	21.7	11.7	15.0	6.7	15.0	10.0	8.3
MU	2.5	2.5	2.4	2.5	2.6	2.2	1.8	83.4
Average	5.8	6.7	7.6	5.9	5.7	6.3	6.3	55.7

Table 6
Percentage of features selected by each of the seven transformations and the original features (NT: no transformation) from the feature set produced by the micro–macro approach using median operator. The last row shows the average across 12 datasets.

Dataset	Median operator for macro step							
	Quadratic	Cubic	log	Square root	Cubic root	tanh	Sigmoid	NT
CT	16.1	6.5	16.1	14.5	11.3	06.5	17.7	11.3
MR	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
SB	13.8	13.8	20.7	19.0	12.1	10.3	5.2	5.2
NU	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
C4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
WF	9.5	14.3	0.0	9.5	4.8	28.6	23.8	9.5
FI	50.0	0.0	0.0	0.0	0.0	0.0	0.0	50.0
PM	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
BC	20.0	20.0	0.0	0.0	0.0	0.0	0.0	60.0
IO	28.6	14.3	0.0	0.0	14.3	14.3	14.3	14.3
WBC	16.7	16.7	0.0	16.7	16.7	0.0	33.3	0.0
MU	2.5	2.5	2.4	2.5	2.6	2.2	1.8	83.4
Average	13.1	7.3	3.3	5.2	5.1	5.2	8.0	52.8

Similarly, the variant employing the median operator performed competitively, securing third place among all methods evaluated. Among the filter-based techniques, the mRMR method, when selecting the top 50% of features, demonstrated the best performance in its category, ranking second overall.

6. Discussion

Our results indicate that the micro–macro genetic algorithm approach can find a feature set that maximizes classification performance while minimizing the number of selected features. By employing a bootstrapping strategy, the approach can simultaneously identify candidate feature sets across different samples through the genetic algorithm, which are subsequently aggregated and refined during the macro step. Notably, our approach can enhance F1 score performance, which is a critical indicator for handling imbalanced datasets, tackling a prevalent challenge in machine learning. This improvement occurs alongside a reduction in the number of features, thereby supporting interpretability, as a smaller feature set makes it easier to understand how input variables influence predictions.

Table 7
Accuracy and F1 score performance across the 12 datasets for the ANOVA F-value filtering method, selecting the top 20%, 30%, and 50% of the features.

Dataset	ANOVA F-value					
	Percentile 20		Percentile 30		Percentile 50	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
CT	70.9	47.1	73.0	45.1	75.6	50.2
MR	64.1	56.2	64.8	62.4	59.9	52.6
SB	89.3	88.6	89.1	88.5	91.5	91.1
NU	81.8	62.0	86.6	65.8	91.5	83.9
C4	74.6	47.7	75.4	49.1	77.5	52.6
WF	66.0	64.9	70.4	69.5	78.0	77.8
FI	97.2	68.2	97.3	70.6	97.2	71.6
PM	58.4	48.4	64.3	58.6	67.5	64.4
BC	95.0	94.5	95.0	94.5	95.7	95.3
IO	97.2	96.9	94.4	93.8	98.6	98.4
WBC	96.5	96.1	97.4	97.2	97.4	97.1
MU	97.7	95.3	98.3	96.5	98.7	97.4
Average	82.4	72.2	83.8	74.3	85.8	77.7

Table 8
Accuracy and F1 score performance across the 12 datasets for the mutual information filtering method, selecting the top 20%, 30%, and 50% of the features.

Dataset	Mutual information					
	Percentile 20		Percentile 30		Percentile 50	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
CT	72.5	40.9	73.9	46.1	77.2	50.5
MR	64.9	56.1	63.1	57.5	58.7	51.7
SB 92.2	91.8	92.2	91.6	91.9	91.3	
NU	81.7	61.9	86.6	65.8	86.9	69.6
C4	73.5	47.0	74.6	47.9	77.5	52.5
WF	65.9	65.0	70.4	69.5	79.3	79.3
FI	97.2	68.2	97.2	68.2	97.3	69.7
PM	64.3	55.9	63.6	52.7	63.0	56.6
BC	95.0	94.5	95.7	95.3	95.7	95.3
IO	91.5	90.4	94.4	93.8	97.2	96.9
WBC	96.5	96.1	97.4	97.2	97.4	97.1
MU	99.3	98.7	99.3	99.1	99.9	99.9
Average	82.9	72.2	84.0	73.7	85.2	75.9

Table 9
Accuracy and F1 score performance across the 12 datasets for the mRMR filtering method, selecting the top 20%, 30%, and 50% of the features.

Dataset	mRMR					
	Percentile 20		Percentile 30		Percentile 50	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
CT	72.3	42.0	74.0	47.0	75.9	51.4
MR	63.7	61.8	62.3	56.9	58.8	51.2
SB	92.3	91.9	92.9	92.6	93.6	93.2
NU	80.8	61.3	85.2	64.7	91.4	70.9
C4	71.8	44.4	74.4	47.5	75.9	50.5
WF	76.8	76.7	79.2	79.1	80.9	80.9
FI	97.2	70.7	97.2	70.7	97.2	71.6
PM	60.4	43.0	63.6	52.7	63.0	60.7
BC	95.7	95.3	95.0	94.5	95.7	95.3
IO	97.2	96.9	94.4	93.8	97.2	96.9
WBC	97.4	97.1	95.6	95.2	97.4	97.1
MU	98.9	97.9	99.5	99.1	99.6	99.3
Average	82.9	72.2	84.0	73.7	85.2	75.9

The choice between union and median operator for the aggregation in the macro-step offers a trade-off between performance boost and feature reduction. The union operator retains more features, leading to a higher performance score but a lower feature reduction. In fact, using the union operator as the macro step increased the average number of features across the datasets. On the other hand, the median operation ensures the consideration of only those features shown to be relevant in at least two of the bootstrap samples of the micro-step, thus ensuring the inclusion of critical features in the last step. This selective inclusion allowed for a reduction in the number of features but slightly decreased the performance score compared to the union operator.

As a practical contribution, our results suggest that applying transformations such as sigmoid and quadratic improves model performance. The reason behind this improvement is the capacity of the transformations to capture data relationships that are not noticeable in the original feature space. This is particularly true for complex datasets, where such transformations can better represent nonlinear patterns and interactions, leading to a higher predictive score.

Regarding filter-based feature selection methods, the micro–macro genetic algorithm approach demonstrated superior performance in most cases. The enhanced performance of our proposed approach can be attributed to its ability to effectively balance competing objectives, making it well-suited for diverse datasets and domains. Moreover, unlike traditional filtering methods, in which the user needs to specify the percentile number of features to select, our approach can optimize this by itself, avoiding the necessity of tuning additional hyperparameters during the training process.

Overall, the proposed hierarchical genetic algorithm framework contributes by offering an approach that automates feature selection and engineering. The results indicate that the framework delivers comparable or superior performance to traditional feature selection methods by effectively integrating genetic algorithms, ensemble learning, and feature transformations. Additionally, the reduced feature set supports machine learning model interpretability, addressing the growing need for explainability in predictive modeling.

6.1. *Comparison with previous works*

Similar to previous studies (Li et al., 2017; Shi & Saad, 2023; Zhang & Yang, 2008), our approach highlights the role of feature transformations in capturing nonlinear patterns. However, unlike previous studies, we showed the effectiveness of these feature transformations for multi-objective optimization settings. Our work also supports the findings of Kiziloz (2021), demonstrating that leveraging ensemble methods is more reliable for feature selection than traditional filtering methods. Specifically, the use of ensemble learning with Logistic Regression, SVM, and XGBoost ensures broad applicability across linear, boundary-based, and nonlinear problems. However, in contrast to Kiziloz (2021), we extended the approach by incorporating an ensemble classifier into a two-level hierarchical algorithm to optimize two goals: performance and feature reduction. This extension enables the reduction of the feature set's dimensionality without sacrificing accuracy.

In comparison to Rostami et al. (2021) and Wang et al. (2020) that also used the NSGA-II to optimize two objectives, we extended their work by performing the procedure multiple times through bootstrapping. This statistical technique allowed us to expose the NSGA-II to different configurations of the training data, thus exploring more different feature combinations. These combinations were then combined ('macro-step'), thus allowing the exploitation of these candidate feature sets to derive an optimal feature set.

To the best of our knowledge, our approach is the first to expand the original with nonlinear transformations, thus enabling the projection of the features on alternative spaces in which class prediction can be enhanced. Indeed, our results indicated that the optimal solution generated by our macro–micro approach selected, on average, 45% of nonlinear transformed features (see Table 5). Thus, this finding suggests that incorporating nonlinear features into traditional machine learning models can significantly enhance their performance.

6.2. *Limitations and future work*

We note that our micro–macro genetic algorithm was implemented specifically for classification problems using tabular data. We did not evaluate its applicability to regression problems and non-structured data, such as text, images, or signals. However, the methodology presented here could be adapted to other types of data and regression tasks by modifying the objectives of the NSGA-II algorithm. Future research should explore the potential of the micro–macro genetic algorithm for regression problems to evaluate its feasibility and effectiveness.

Additionally, we note that the iterative nature of the micro–macro genetic algorithm requires computational resources, which may limit its scalability to very large datasets. This condition could be addressed by extending the computational framework to support distributed systems. The distributed system would introduce load balancing and task scheduling mechanisms, ensuring efficient utilization of computational resources.

As feature distributions vary across datasets, we recognize that a predefined set of seven transformations may not be optimal for all scenarios. Additionally, we applied transformations only to numerical features, without exploring transformation options for categorical features. Future work should explore transformation methods that adapt to the unique characteristics of each dataset, potentially developing mechanisms to analyze feature distributions and relationships to select the most appropriate transformation.

7. Conclusion

This paper presents a two-level hierarchical genetic algorithm for feature selection, aiming to maximize classification performance while minimizing the number of selected features. This proposed method introduces three key contributions: (1) the integration of repeated bootstrapping with feature transformations to enhance robustness and solution diversity, (2) the application of NSGA-II for effective multi-objective optimization, balancing accuracy and interpretability, and (3) the provision of an open-source Python library (Felahatpisheh et al., 2025), enabling reproducibility and practical application in real-world scenarios.

Our approach achieves performance comparable to or surpassing traditional feature selection techniques, highlighting its robustness and versatility as a feature engineering tool. However, the computational costs associated with bootstrapping and genetic operations remain a constraint, particularly for large-scale or extremely high-dimensional datasets. To address these challenges, future research should explore alternative evolutionary operators to enhance efficiency, investigate real-time feature adaptation for streaming contexts, and further optimize scalability. Additionally, benchmarking the algorithm across diverse domains will help solidify its position within the field and guide subsequent advancements.

CRediT authorship contribution statement

Aditi Gulati: Methodology, Writing – original draft. **Armin Felahatpisheh:** Conceptualization, Methodology, Data curation, Software. **Camilo E. Valderrama:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

AG was supported by a Mitacs Globalink (Project ID: 32844).

Data availability

Data will be made available on request.

References

Aceves-Fernandez, M. A. (2020). *Advances and Applications in Deep Learning*. IntechOpen, http://dx.doi.org/10.5772/intechopen.87786.

Ali, W., & Saeed, F. (2023). Hybrid filter and genetic algorithm-based feature selection for improving cancer classification in high-dimensional microarray data. *Processes*, *11*(2), http://dx.doi.org/10.3390/pr11020562.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115. http://dx.doi.org/10.1016/j.inffus.2019.12.012.

Campagner, A., & Cabitza, F. (2020). Back to the feature: A neural-symbolic perspective on explainable AI. In A. Holzinger, P. Kieseberg, A. M. Tjoa, & E. Weippl (Eds.), *Machine learning and knowledge extraction* (pp. 39–55). Springer International Publishing, ISBN: 978-3-030-57321-8.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. http://dx.doi.org/10.1109/4235.996017.

Felahatpisheh, A., Gulati, A., & Valderrama, C. E. (2025). Feature-gen. URL: https://pypi.org/project/feature-gen/ (Accessed 13 January 2025).

Khurana, U. (2018). Transformation-based feature engineering in supervised learning: Strategies toward automation. In *Feature engineering for machine learning and data analytics* (pp. 221–243). CRC Press.

Kiziloz, H. E. (2021). Classifier ensemble methods in feature selection. *Neurocomputing*, *419*, 97–107. http://dx.doi.org/10.1016/j.neucom.2020.07.113.

Li, Y., Li, T., & Liu, H. (2017). Recent advances in feature selection and its applications. *Knowledge and Information Systems*, *53*(3), 551–577. http://dx.doi.org/10.1007/s10115-017-1059-8.

Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E. B., & Turaga, D. (2017). Learning feature engineering for classification. In *Proceedings of the twenty-sixth international joint conference on artificial intelligence, IJCAI-17* (pp. 2529–2535). http://dx.doi.org/10.24963/ijcai.2017/352.

Rostami, M., Berahmand, K., & Forouzandeh, S. (2021). A novel community detection based genetic algorithm for feature selection. *Journal of Big Data*, *8*(1), 2. http://dx.doi.org/10.1186/s40537-020-00398-3.

Shi, K., & Saad, S. (2023). Automated feature engineering for automl using genetic algorithms. In *The 20th international conference on security and cryptography* (pp. 450–459).

Wang, H., He, C., & Li, Z. (2020). A new ensemble feature selection approach based on genetic algorithm. *Soft Computing*, *24*(20), 15811–15820. http://dx.doi.org/10.1007/s00500-020-04911-x.

Zhang, Z., & Yang, P. (2008). An ensemble of classifiers with genetic algorithm-based feature selection. *Deakin University Journal Contribution*, URL: https://hdl.handle.net/10536/DRO/DU:30017964.