Multi-objective Feature Selection Optimization Using NSGA-2

Hamza Rangwala  
 Department of Science  
 Wilfrid Laurier University  
 Waterloo, ON, Canada  
hamzarangwala51@gmail.com

Azam Bidgoli  
 Faculty of Science  
 Wilfrid Laurier University  
Waterloo, ON, Canada   
aasilianbidgoli@wlu.ca

Shahryar Rahnamayan  
 Department of Engineering  
 Brock University  
St. Catherines, ON, Canada   
srahnamayan@brocku.ca

ABSTRACT

Feature selection is critical in machine learning, particularly with high-dimensional datasets. This paper introduces an innovative approach using the NSGA-II (Non-dominated Sorting Genetic Algorithm II) for feature selection, aiming to minimize the number of features and classification error simultaneously. A custom function was implemented to facilitate this process. The K-Nearest Neighbors (KNN) classifier evaluated the selected features across multiple runs and folds to ensure robustness. The methodology involved optimizing feature selection on training sets and evaluating on validation sets, identifying Pareto front solutions, and testing on the test set for generalization capability. Results showed effective feature reduction while maintaining or improving classification accuracy. Key metrics, such as average training error, minimum validation error, and test error, were tracked to highlight consistency. Non-dominated binary vectors of optimal feature subsets were analyzed for further feature selection via voting. This framework aims to narrow the disparity between the classification error observed on the training and test datasets, thereby enhancing model generalization. This study provides a robust framework for feature selection in high-dimensional data, leveraging evolutionary algorithms and cross-validation techniques to enhance machine learning model performance.

CCS CONCEPTS

• Insert CCS text here • Insert CCS text here   • Insert CCS text here

KEYWORDS

Insert keyword text, Insert keyword text, Insert keyword text, Insert keyword text

ACM Reference format:

FirstName Surname, FirstName Surname and FirstName Surname. 2018. Insert Your Title Here: Insert Subtitle Here. In *Proceedings of ACM Woodstock conference (WOODSTOCK’18). ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/1234567890

1 INTRODUCTION

In the era of big data, the exponential growth of available information has led to increasingly high-dimensional datasets, posing significant challenges for machine learning applications. These datasets, characterized by many features, often include redundant, irrelevant, or noisy data, which can negatively impact the performance of machine learning models. Consequently, feature selection has emerged as a critical preprocessing step to enhance model accuracy, reduce computational complexity, and improve interpretability. Given the recent explosion in the dimensionality of real-world datasets, which now often contain thousands or even millions of features, the importance of feature selection methods has surged dramatically. [1]

It aims to identify the most relevant subset of features that contribute significantly to the predictive accuracy of a model. By eliminating unnecessary features, the dimensionality of the data is reduced, leading to simpler models that are easier to interpret and faster to train. Moreover, effective feature selection can mitigate the risk of overfitting, thereby enhancing the generalization capability of the model on unseen data. We define the feature selection as a problem including two competing objectives and we try to find a set of optimal solutions so called Pareto-optimal solutions instead of a single optimal solution.[2]

Traditional feature selection methods, such as filter, wrapper, and embedded techniques, have been widely used. However, these methods often struggle with the curse of dimensionality and may not efficiently explore the vast search space of potential feature subsets. To address these limitations, evolutionary algorithms have gained popularity due to their ability to perform global searches and handle complex optimization problems. Among these, the Non-dominated Sorting Genetic Algorithm II (NSGA-II) has shown promise in multi-objective optimization tasks, making it a suitable candidate for feature selection.

Multi-objective optimization algorithms such as genetic algorithms (GAs) have been successfully applied to various optimization problems, ranging from evolutionary design [3]. This study introduces an innovative approach utilizing NSGA-II for feature selection, with a dual objective of minimizing the number of features and the classification error simultaneously.

Simulation results on difficult test problems show that the proposed NSGA-II, in most problems, is able to find much better spread of solutions and better convergence near the true Pareto-optimal front compared to Pareto-archived evolution strategy and strength-Pareto EA—two other elitist MOEAs that pay special attention to creating a diverse Pareto-optimal front [4] .The NSGA-II algorithm is well-suited for this task as it can effectively balance the trade-off between these conflicting objectives by identifying a set of Pareto optimal solutions. Each solution on the Pareto front represents a potential feature subset that optimizes the trade-off between the two objectives.

To facilitate the feature selection process, a custom function was developed which is used to diversify the initial population based on the number of features and generating random values to set the cells of the individuals and integrated with the NSGA-II algorithm. The K-Nearest Neighbor Algorithm is the simplest of all machine learning algorithms. It is based on the principle that the samples that are similar, generally lies in close vicinity [5]. The selected feature subsets were evaluated using the K-Nearest Neighbors (KNN) classifier, a widely used non-parametric method known for its simplicity and effectiveness. The performance of the KNN classifier, in conjunction with the selected features, was assessed across multiple runs and folds of the dataset, ensuring the robustness and generalizability of the results.

The methodology employed in this study involved dividing the dataset into training and validation sets. The feature selection process was optimized on the training set, and the selected features were then evaluated on the validation set to identify Pareto front solutions. These solutions were further analyzed for their performance on the validation set, and the subsets with the minimum classification error were tested on a separate test set to evaluate their generalization capability.

The results of this study demonstrated that the proposed approach effectively reduced the number of features while improving classification accuracy. Key performance metrics, such as average training error, minimum validation error, and test error which were tracked across multiple runs, highlighting the consistency and reliability of the feature selection process. Additionally, the non-dominated binary vectors representing the optimal feature subsets were saved and analyzed for further selection using a voting mechanism, providing valuable insights into the feature selection process.

In summary, this research offers a robust framework for feature selection in high-dimensional data, leveraging the strengths of evolutionary algorithms and rigorous cross-validation techniques. The findings contribute to the field by offering a scalable and effective solution for feature selection, potentially improving the performance of various machine learning models.

3 METHODOLOGY

Our methodology integrates evolutionary algorithms with the K-Nearest Neighbors (KNN) classifier to identify optimal feature subsets. By minimizing both the number of features and classification error, we aimed to enhance model performance while ensuring robustness through cross-validation.

3.1 NSGA-II Algorithm

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) is a widely used evolutionary algorithm designed for solving multi-objective optimization problems. NSGA-II improves upon its predecessor, NSGA, by addressing key issues such as computational complexity, lack of elitism, and the need for specifying a sharing parameter. NSGA-II outperforms two other contemporary MOEAs: Pareto-archived evolution strategy (PAES) and strength Pareto EA (SPEA) in terms of finding a diverse set of solutions and in converging near the true Pareto-optimal set [4]. NSGA-II operates by simulating the process of natural evolution. The algorithm maintains a population of potential solutions that evolve over successive generations. Each solution in the population is evaluated based on multiple objective functions, which, in this study, are the minimization of feature count and classification error.

During the initialization of the algorithm the population needs to be diverse, it will ensure that the initial population is covers the overall solution space and converges to an optimal set of solutions. We generate individuals with a value in between 1 and n where n represents the total number of features in the dataset. After doing that we set n random cells of the individual to 1, where n represents the number of selected features in the individual. This is being reiterated multiple times to suffice the number of individuals required to accommodate the initial population size. The algorithm is thus initialized by generation random population to give optimal solutions on the Pareto front. The population is sorted into different fronts based on Pareto dominance. This technique is called the Non-Dominated Sorting (NDS). A solution is said to dominate another if it is no worse in all objectives and better in at least one. The first front consists of non-dominated solutions, the second front is dominated only by the first front, and so on, and this is true for all fronts. Our aim is to filter out the Non-Dominant solutions from the fronts which are the closest match to our objective functions.

Within each front, solutions are assigned a crowding distance, which is a measure of how close a solution is to its neighbors. This helps maintain diversity in the population by favoring solutions in less crowded areas of the objective space. Solutions are selected based on a combination of rank (front number) and crowding distance. Lower-ranked solutions are preferred, and within the same rank, solutions with higher crowding distances are chosen. Selected solutions undergo crossover and mutation operations to create offspring for the next generation. Crossover combines parts of two parent solutions to produce one or more offspring, while mutation introduces small random changes to a solution. After that Bitflip mutation is also implemented, it helps in introducing variation into the population, helping to explore new solutions and maintain genetic diversity. Where for each bit in the binary vector, a random number between 0 and 1 is generated. If this number is less than pmp\_mpm​, the bit is flipped (i.e., a 0 is changed to a 1, and a 1 is changed to a 0). The parent and offspring populations are combined and sorted again. The best solutions, based on rank and crowding distance, are chosen to form the next generation. This elitism ensures that the best solutions are preserved over generations. After each generation there is a fitness evaluation to balance the trade-off between minimizing the number of features and the classification error. The new offspring are evaluated using the same fitness functions, and the process repeats over multiple generations.

To determine whether the solutions we found out where in fact the best solutions possible we calculate the HV on the Pareto Optimal Front. The hypervolume (HV) is a performance metric used to evaluate the quality of solutions on the Pareto optimal front in multi-objective optimization problems. It measures the volume of the objective space dominated by the solutions on the Pareto front, relative to a reference point. The hypervolume is the size of the space covered by the Pareto front solutions when compared to a predefined reference point, which is typically chosen to be worse than the worst objective values found by the algorithm. A higher hypervolume indicates a better-quality Pareto front, as it suggests that the solutions collectively cover a larger and more optimal region of the objective space. In the context of feature selection, a larger hypervolume would indicate a better trade-off between minimizing feature count and classification error.

By utilizing these mechanisms, NSGA-II effectively addresses the challenges of multi-objective optimization, providing a robust method for feature selection in high-dimensional datasets. The algorithm's ability to handle multiple conflicting objectives makes it well-suited for applications where balancing trade-offs is crucial, such as in minimizing both feature count and classification error in machine learning models.

3.2 Tailored Feature Selection Methodology

We developed a function for the feature selection process which is designed to optimize two primary objectives:

1. Minimizing the Number of Features.
2. Minimizing the Classification Error.

**Minimizing Feature Count**: This function calculates the total number of features selected in each solution. The fewer the features, the better the solution is considered in this context.

**Minimizing Classification Error**: This function evaluates the classification error using the K-Nearest Neighbors (KNN) classifier. It identifies the selected features from the binary vector and trains the KNN classifier on these features. The classification error is computed as one minus the accuracy of the classifier on the training data.

The initialization of the function depends upon the number of decision variables corresponding to the total number of features in the dataset, two objective functions, and one inequality constraint. The constraint ensures that at least one feature is selected in each solution. This prevents the algorithm from considering solutions with no selected features, which would be meaningless for the classification task. The evaluation function processes the population of solutions, calculating the values for both objective functions and the constraint for each solution. It evaluates the number of features selected, the classification error, and ensures the constraint is satisfied. This function's evaluation method is invoked to assess the fitness of each solution in the population. This involves computing the number of features, the classification error, and applying the constraint to ensure validity. The algorithm iterates through multiple generations, continuously evaluating and evolving the population towards optimal solutions.

This function is integrated with the NSGA-II algorithm to achieve an effective multi-objective optimization for feature selection. The following details describe the implementation and integration process.

3.3 Evaluation Metrics

Evaluating the performance of the feature selection process is critical to ensure that the selected features lead to an effective and robust machine learning model. In this study, several key metrics are used to assess the quality of the feature selection process, including average training error, minimum validation error, and test error after voting. Each of these metrics provides unique insights into the model's performance at different stages of the selection and evaluation process.

Average training error is the mean classification error observed on the training dataset over multiple runs and folds. It provides an indication of how well the model fits the training data. A lower average training error suggests that the model is able to capture the underlying patterns in the training data effectively.

Average Training Error =

where is the training error for the i-th run or fold, and N is the total number of folds.

Average minimum validation error is the lowest classification error observed on the validation dataset during the feature selection and model training process. Hyper parameter tuning guides the tuning process to find the optimal settings that minimize the error on the validation set, reducing the risk of overfitting. It serves as a critical metric for model selection and hyperparameter tuning. Helps in selecting the model configuration (e.g., feature subset, hyperparameters) that generalizes best to unseen data.

Average Minimum Validation Error = min {}

After the optimization process using NSGA-II, the feature selection framework includes a voting mechanism to further refine the selection of features. This step involves identifying the most frequently selected features across the binary vector solution having the lowest classification error on the validation pareto front and using these to build the final model. The test error after voting is a crucial metric to evaluate the effectiveness of this approach in generalizing to unseen data. This approach aims to produce robust and efficient machine learning models suitable for high-dimensional datasets.

Test Error After Voting​ = 1− ​*I* ( =

Systematically evaluating these metrics, the research demonstrates the effectiveness of the proposed feature selection method in improving the overall performance of machine learning models, making it a valuable contribution to the field.

4  Experimental Setup

This section includes the details of the datasets that are being used, the steps taken to carry out this unique framework for splitting, evaluation and optimization of the datasets for feature selection as well as the models and techniques used to bridge the gap between the training and test error which has been significant earlier.

4.1 Dataset Description

This study utilized a diverse set of biological datasets to evaluate the feature selection process. The datasets vary in size, number of features, and classification tasks, providing a comprehensive assessment of the proposed method's performance. Primarily the datasets have been taken from a category representing the biological data with ranges of instance between 50 to 174 and the feature ranges in between 2000 and 24481, it spans a broad range in terms of both the number of instances and features, providing a comprehensive basis for evaluating the proposed feature selection method across various biological contexts. The classification of the datasets includes both binary and multi-class.

By covering a wide range of instances, features, and classification types, these datasets provide a robust framework for assessing the efficacy and generalizability of the proposed feature selection method in various biological contexts.

4.2 Classifier Used

The K-Nearest Neighbors (KNN) classifier is a simple, yet effective machine learning algorithm used for both classification and regression tasks. It operates based on the principle that similar data points are likely to exist in proximity within a feature space. The KNN algorithm makes predictions by identifying the 'k' nearest data points in the training set to a given input and then voting for the most common class label (in classification) or averaging the values (in regression) of these neighbors [6]. This parameter determines the number of nearest neighbors to consider when making a prediction. A small 'k' value can lead to a model sensitive to noise, while a large 'k' value can smooth out predictions but may overlook small but significant patterns [7]. The weight parameter specifies whether all neighbors contribute equally to the prediction ('uniform') or whether closer neighbors should have a greater influence ('distance’) [8].

KNN is a lazy learner, meaning it does not require a separate training phase. This can be advantageous when dealing with dynamic datasets where new data points continuously become available. By choosing an appropriate value of 'k', KNN can be robust to outliers, which is crucial when dealing with noisy biological data. KNN was configured with k=5k=5k=5 (5 nearest neighbors) to balance between overfitting and underfitting. The Euclidean distance metric was used to measure similarity, and uniform weights were applied to ensure equal contribution from each neighbor. This configuration provided a robust framework for evaluating the performance of the feature subsets selected by the NSGA-II optimization algorithm.

4.3 Experimental Procedure

This study follows a comprehensive experimental process to evaluate the effectiveness of the proposed feature selection method using NSGA-II. The process includes dataset splitting, optimization using the NSGA-II, validation on train, and testing after validation. The dataset is initially split into training and test sets using train\_test\_split with a 70-30 ratio. This ensures that 30% of the data is reserved for final testing. The split is stratified based on class labels to maintain the class distribution in both sets. The training set is further divided into training and validation sets using K-Fold cross-validation. This ensures that different portions of the data are used for training and validation in each fold, promoting robustness and generalizability. We will be taking the training sets after cross-validation for multiple iterations and feeding that data to the algorithm for optimization and getting solutions from the pareto optimal front of the algorithm.

A function has been designed which evaluates each solution from the optimization function of the algorithm on the validation set, computing the number of selected features and validation error. After doing that, we perform NDS (Non-Dominant Sorting) on the solutions to plot a pareto front on the validation set. This plot helps us in identifying the solutions with the minimum error rate and calculating the average error rate on validation across multiple runs, which will help in further analysis. The non-dominated binary vectors representing optimal feature subsets are also saved for further analysis on independent test data for frequent feature selection optimization. The identification of the solution having the minimum error rate is done from these solutions and using that solution we evaluate it on the test data set which had been initially split and stored on the local machine. The test data is firstly retrieved using the pickle library and after that used to evaluate on the solution containing the minimum error rate on validation pareto front. The accuracy calculated using these solutions helps us in calculating the average test error for every run and the features associated with them.

4.4 Voting Mechanism

In this study, a voting mechanism is implemented to further refine the selection of non-dominated binary vectors representing optimal feature subsets. Each run of the optimization process results in a set of non-dominated binary vectors, representing feature subsets that have the minimum error rate on the validation set. These binary vectors are saved to files for each run and later loaded for analysis. An array is initialized to count the number of times each feature is selected across all runs. For each solution in the Pareto front, the corresponding feature positions in the array are incremented by one. Features are sorted based on their selection frequency in descending order, and the top 100 features are identified.

The top features based on their selection frequency are used to create a new feature subset. This subset is then used to train a K-Nearest Neighbors (KNN) classifier on the training data having the most frequent features selected, and the classifier's performance is evaluated on the test set which has been restored from the machine to determine the minimum error rate on test after voting and choosing the most frequent features. This process is repeated iteratively for the top 100 features, starting by selecting the most frequently occurring feature in the first iteration and subsequently adding the next most frequent feature in each subsequent run.

The user must style this paragraph in **ParaContinue** style, which follows immediately after the **DisplayFormula** (numbered equation). The **DisplayFormula** style is applied only in case of a numbered equation. A numbered equation always has a number to its right. Insert paragraph text here. **Display Formula without Number**



The **DisplayFormulaUnnum** style is applied only in case of an unnumbered equation. An unnumbered display equation never contains an equation number to its right, and this unique property distinguishes it from a numbered equation.



Figure 1: Figure Caption and Image above the caption [In draft mode, Image will not appear on the screen]

**Theorem/Proof/Lemma.** Insert text here for the enunciation or Math statement. Insert text here for the enunciation or Math statement. Insert text here for the enunciation or Math statement. Insert text here for the enunciation or Math statement. Insert text here for the enunciation or Math statement.

....Insert text here for the Quotation or Extract, Insert text here for the Quotation or Extract, Insert text here for the Quotation or Extract, Insert text here for the Quotation or Extract, Insert text here for the Quotation or Extract, Insert text here for the Quotation or Extract.

1.1 Heading Level 2

In the below paragraph, it is explained how alt-txt value is placed in **MS Word 2010**. To add alternative text to a picture in Word 2010, follow these steps:

1. In a Word 2010 document, insert a picture.
2. Right click on the inserted picture and select the **Format Picture** option.
3. Select the **Alt Txt** option from the left-side panel options.
4. In the "Title:" and "Description:" text boxes, type the text you want to represent the picture, and then click "Close".

Below are steps to place alt-txt value in **MS Word 2013/2016**. To add alternative text to a picture in Word 2013/2016, follow these steps:

1. In a Word 2013/2016 document, insert a picture.
2. Right click on the inserted picture and select the **Format Picture** option.
3. In the settings at the right side of the window, click on the "Layout & Properties" icon (3rd option).
4. Expand **Alt Txt** option.
5. In the "Title:" and "Description:" text boxes, type the text you want to represent the picture, and then click "Close".

*1.1.1 Heading Level 3.* Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here.

*1.1.1.1 Heading Level 4.*Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here.

ACKNOWLEDGMENTS

Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here.

REFERENCES

[1] G. Chandrashekar and F. Sahin, “A survey on feature selection methods,” *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014, doi: <https://doi.org/10.1016/j.compeleceng.2013.11.024>.

[2] G. Goos *et al.*, “Lecture Notes in Computer Science”.

[3] Bentley, P.J., Wakefield, J.P.: Generic evolutionary design. In: Chawdhry, P.K., Roy, R., Pant, R.K. (eds.) Soft Computing in Engineering Design and Manufacturing, pp. 289–298. Springer, London (1998). <https://doi.org/10.1007/978-1-4471-0427-8_31>

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Computat.*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: [10.1109/4235.996017](https://doi.org/10.1109/4235.996017).

[5] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inform. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, doi: [10.1109/TIT.1967.1053964](https://doi.org/10.1109/TIT.1967.1053964).

[6] N. S. Altman, “An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression,” *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992, doi: [10.1080/00031305.1992.10475879](https://doi.org/10.1080/00031305.1992.10475879).

[7] R. O. Duda, P. E. Hart, and D. G. Stork, “Pattern Classiﬁcation (2nd ed.)”.

[8] D. W. Aha, D. Kibler, and M. K. Albert, “Instance-based learning algorithms,” *Mach Learn*, vol. 6, no. 1, pp. 37–66, Jan. 1991, doi: [10.1007/BF00153759](https://doi.org/10.1007/BF00153759).

[6]