GENETIC ALGORITHM FOR THE EVOLUTION OF CLASSIFIER'S WEIGHTS

ALGORITMO GENÉTICO PARA LA EVOLUCIÓN DE LOS PESOS DEL CLASIFICADOR

Alejandro Moreno Martínez
Universidad Autónoma del Estado de México / Centro Universitario Valle de México, México
amorenom758@alumno.uaemex.mx

Asdrúbal López Chau
Universidad Autónoma del Estado de México / Centro Universitario Zumpango, México
alchau@uaemex.mx

Víctor Manuel Landassuri Moreno
Universidad Autónoma del Estado de México / Centro Universitario Valle de México, México
vmlandassurim@uaemex.mx

Saturnino Job Morales Escobar
Universidad Autónoma del Estado de México / Centro Universitario Valle de México, México
sjmoralese@uaemex.mx

Reception: 15/December/2025 **Acceptance**: 31/March/2026

Abstract

One of the main challenges in machine learning is obtaining high-performance predictive models for classification tasks. Traditionally, this goal is achieved by adjusting hyperparameters; however, little attention has been paid to assigning weights at the instance level in training sets. This work proposes the use of genetic algorithms (GA) with real representation to evolve and optimize these weights, allowing classifiers to focus on more complex or relevant examples. The methodology consisted of applying a GA to three widely used classifiers (Naïve Bayes, Logistic Regression, and Support Vector Machines) on five public datasets. The process included the initial generation of weights, their evolution using genetic operators, and the evaluation of accuracy in multiple runs. The results show significant improvements in accuracy, with increases of up to 20 percentage points compared to the baseline, demonstrating the potential of the proposed approach.

Keywords: Classifier, Genetic algorithm**,** Weight assignment, Weight evolution.

Resumen

Uno de los principales retos en el aprendizaje automático es obtener modelos predictivos con alto rendimiento en tareas de clasificación. Tradicionalmente, este objetivo se alcanza ajustando hiperparámetros; sin embargo, se ha prestado poca atención a la asignación de pesos a nivel de instancia en los conjuntos de entrenamiento. Este trabajo propone el uso de algoritmos genéticos (GA) con representación real para evolucionar y optimizar dichos pesos, permitiendo que los clasificadores se concentren en ejemplos más complejos o relevantes. La metodología consistió en aplicar un GA sobre tres clasificadores ampliamente utilizados (Naïve Bayes, Regresión Logística y Máquinas de Vectores de Soporte) en cinco conjuntos de datos públicos. El proceso incluyó la generación inicial de pesos, su evolución mediante operadores genéticos y la evaluación de precisión en múltiples ejecuciones. Los resultados muestran mejoras significativas en precisión, con incrementos de hasta 20 *puntos porcentuales frente al baseline, evidenciando el potencial del enfoque propuesto.*

Palabras clave: *Algoritmo genético, asignación de pesos, clasificador, evolución de pesos.*

1. Introduction

Classification is one of the core tasks in machine learning, with applications ranging from medical diagnosis to educational assessment. Traditionally, improvements in classifier performance have been achieved by tuning hyperparameters or selecting features. However, much less attention has been paid to the assignment of instance-level weights, despite its potential to emphasize difficult or underrepresented cases. Manually determining such weights is highly impractical, which opens the door to optimization-based approaches.

Previous research has shown the effectiveness of genetic algorithms (GA) in related tasks. For example, Padilla [2017], used GA for feature selection in cardiac arrhythmia classification, achieving improvements in accuracy and noise reduction. Similarly, Mosquera [2018], applied GA for dimensionality reduction in the prediction of psychosocial risks, reporting significant gains with SVM and Naïve Bayes. More

recently, Aparicio [2022], combined GA with logistic regression for predicting heart disease associated with diabetes, confirming the suitability of evolutionary strategies in medical domains.

While these studies highlight the potential of GA to enhance classification, they focused mainly on feature selection or general parameter optimization. In contrast, few works have addressed the evolution of instance-specific weights, which could directly influence how classifiers handle complex data. This paper proposes the use of a real-valued GA to assign and evolve weights for each training instance across three classifiers: Naïve Bayes, Logistic Regression, and Support Vector Machines. Experiments on five public datasets demonstrate that this strategy consistently improves precision, providing evidence of the value of GA for automated weight optimization and highlighting its applicability in socially relevant domains such as health, education, and industry.

2. Methods

Classification algorithms

Classification algorithms are machine learning methods that analyze patterns and features in data to estimate the class or category to which an instance belongs. Among the many available classifiers, this study focuses on three in particular: Naïve Bayes, Logistic Regression, and Support Vector Machines (SVM). These algorithms fall under the category of supervised learning, which relies on labeled data to train predictive models. Each classifier has distinct characteristics and configurations that make it well-suited for specific classification tasks:

- **Naïve Bayes:** Naive Bayes is a probabilistic classification method based on Bayes' theorem, which assumes conditional independence among the predictor variables (features). The classifier operates in the following steps: first, data are used to estimate the prior probabilities of each class; second, the conditional probabilities of the features given each class are calculated. Finally, when a new, unseen sample is introduced, Bayes' theorem is applied to determine the most probable class, as shown in Equation 1.

$$P(A|B) = \big(P(B|A) \cdot P(A)\big)/P(B) \quad (1)$$

Where:

$P(A|B)$: represents the posteriori probability of A given B.

$P(B|A)$: denotes the probability of B occurring given that A has occurred.

$P(A)$: represent the prior probabilities of event A.

$P(B)$: represent the prior probabilities of event B.

Logistic regression: Logistic regression is a statistical method used to estimate the probability that an instance belongs to a particular class. It is based on the logistic model, which applies the logistic (sigmoid) function to transform a linear combination of input features into a probability value. Logistic regression models the relationship between the independent variables and the dependent (categorical) variable, enabling class predictions based on estimated probabilities. Equation 2 presents the general form of logistic function. Where x is the Input variable (feature) and e Euler's number.

$$f(x) = 1/(1 + e^{(-x)}) \quad (2)$$

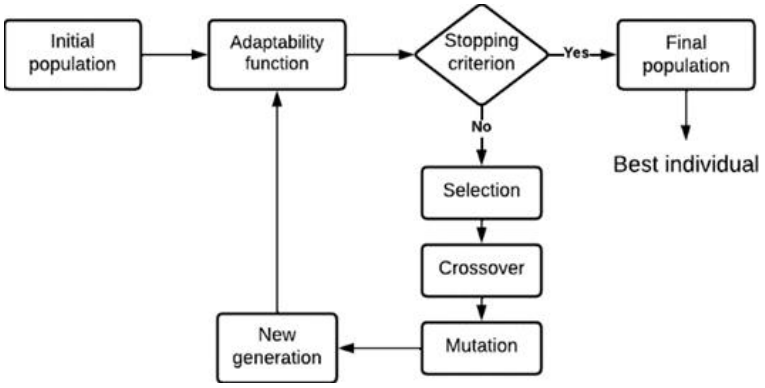
- **Support Vector Machine:** The Support Vector Machine (SVM) is a widely used algorithm for both classification and regression tasks. Its primary goal is to identify an optimal hyperplane that effectively separates instances of different classes. This hyperplane is chosen to maximize the margin, defined as the minimum distance between the hyperplane and the nearest data points from each class, known as support vectors. During training, the algorithm optimizes the weights and coefficients associated with these support vectors to define the optimal decision boundary. This optimization process involves minimizing a cost function that penalizes classification errors while maximizing the margin between classes.

Genetic Algorithms

Genetic Algorithms (GAs) are a type of evolutionary algorithm that uses binary strings to represent individuals in a population. They apply genetic operators—selection, crossover, and mutation—iteratively across generations. In each generation, individuals are evaluated to compute their fitness values, which are used

to identify the best candidate. If an optimal solution is not yet found, the selection operator chooses individuals based on their fitness or, in some cases, at random. The crossover operator is then applied to the selected individuals, recombining their genetic material to produce offspring and form a new generation. Finally, the mutation operator introduces random alterations to the genes, promoting diversity in the population. This process repeats until a solution with optimal or satisfactory fitness is discovered.

Figure 1 presents the general structure of a Genetic Algorithm (GA). The process begins with the generation of an initial population of individuals, typically selected at random. Next, each individual is evaluated using a fitness function to assess how well it solves the target problem. A stopping criterion is then checked to determine whether the algorithm should continue evolving or terminate upon reaching a satisfactory solution. If the criterion is not met, the genetic operators (selection, crossover, and mutation) are applied to produce a new generation (offspring population), which replaces the current (parent) population. This cycle continues until the stopping condition is satisfied.



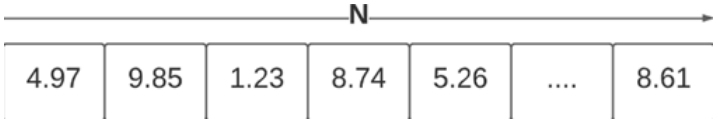
Source: own elaboration.

Figure 1 General diagram of a Genetic Algorithm (GA).

A population refers to a set of individuals that potentially represent candidate solutions to the problem. In a conventional Genetic Algorithm, the genotype is typically encoded using binary strings, where each gene is represented as a binary digit. However, in this study, a real-valued representation is employed, where each gene corresponds to a weight expressed as a real number. This choice aligns with

the nature of the classifiers used, which operate on numerical input, and supports the objective of assigning higher weights to instances that are more difficult to classify, thereby increasing their influence during training.

As a result, each run of the GA may produce individuals with different genome lengths, depending on the number of instances involved in the classification task. Logistic Regression and Support Vector Machine classifiers support both positive and negative weights, while Naive Bayes allows only positive values. An example of this representation is shown in Figure 2, where the genome consists of N weights corresponding to N instances, forming a weight vector. In this approach, the list of weights has a dimensionality equal to the number of instances in the test dataset.



Source: own elaboration.

Figure 2 Chromosome representation: list of weights for an individual.

Initialization refers to the process of generating individuals at the start of the algorithm. Typically, this involves random generation; however, incorporating prior knowledge or heuristics can help accelerate convergence. In this study, initialization is performed by randomly generating the list of weights for everyone in the population. Selection is the genetic operator responsible for choosing the parents that will generate the next offspring population. In this case, rank-based selection is applied, where individuals are sorted in ascending order according to their fitness values [Valencia, 1997]. Crossover is performed using arithmetic crossover, which is suitable for real-valued genomes. A new offspring is created by combining two parent solutions, $parent_1$ and $parent_2$, using a scalar factor a, where $a \in [0, 1]$, as shown in Equation 3.

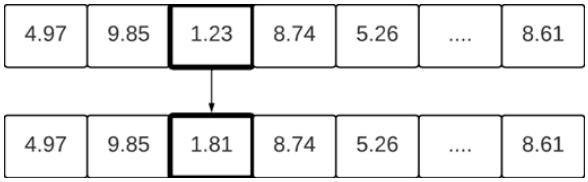
$$child = a * parent_1 + (1 - a) * parent_2 \quad (3)$$

Mutation is a key operator that enables the algorithm to escape local minima and explore new regions of the search space. In this study, each gene in the weight list has a 30% probability of undergoing mutation.

When selected, the gene is modified according to a predefined mutation strategy, this process is illustrated in Equation 4.

$$gen = gen + mutationValue \tag{4}$$

Where $mutationValue$ is a randomly generated real number in the range $[-1, 1]$. For example, Figure 3 illustrates a case in which the third gene was selected for mutation. A random value of 0.58 was generated and added to the original value of the third gene, resulting in a new value of 1.81. This modification produced a new individual, as shown in the bottom section of Figure 3.



Source: own elaboration.

Figure 3 Example of the mutation operator.

Elitism is an operator that preserves the best-performing individuals by passing them unchanged to the next generation. This mechanism ensures that the highest-quality solutions identified in early generations are retained throughout the evolutionary process. It is recommended that elitism be applied to no more than 10% of total population.

Assigning weights to instances

Assigning weights to instances during the training of a machine learning model is an effective technique for addressing class imbalance or emphasizing specific samples. This approach enables the model to adapt its learning process to the characteristics of the problem at hand. Table 1 shows the data set used in this study.

Experimental setup

The genetic algorithm was implemented in Python, and the scikit-learn library was used to build and evaluate the classifiers. The parameters used for the genetic algorithm are detailed below.

Table 1 Datasets characteristics.

Datasets	Number of instances	attributes
Breast-cancer [Zwitter, 1988]	286	9
Diabetes [Kahn, 1999]	768	9
Hepatitis [Hepatitis, 1988]	155	19
Ionosphere [Sigillito, 1989]	351	34
Lymphography [Zwitter, 1988]	148	18

Source: own elaboration.

For each classifier, 80% of the dataset was used for training, while the remaining 20% was reserved for testing. The genetic algorithm assigned weights only to the training instances. After training the predictive model, its performance was evaluated on the test set.

The reported results represent the average performance over ten independent runs of each experiment. As a baseline for comparison, each classifier was also evaluated without the genetic algorithm and without instance weighting. It is important to note that accuracy was used as the fitness function to guide the evolution of individuals in the genetic algorithm.

Classifiers

The classifiers were implemented using their default configurations, without tuning any parameters that could potentially improve their performance. The classifiers were used with their default hyperparameter settings as provided by scikit-learn. The configurations are as follows:

- Naive Bayes: $priors = None, var_{smoothing} = 1e-09$
- Logistic Regression: $penalty =' l2', dual = False, tol = 0.0001, C = 1.0,$ $fit_{intercept} = True, intercept_{scaling} = 1, class_{weight} = None, random_{state} = None,\ solver =' lbfgs', \underset{iter}{max} = 10000, multi_{class} =' auto', verbose = 0, warm_{start} = False, n_{jobs} = None, l1_{ratio} = None$
- Support Vector Machine (SVM): $C = 1.0, kernel =' rbf', degree = 3,$ $gamma =' scale', coef0 = 0.0, shrinking = True, probability = False,\ tol = 0.001,\ cache_{size} = 200, class_{weight} = None, verbose = False, \underset{iter}{max} = -1,\ decision_{function_{shape}} =' ovr', break_{ties} = False, random_{state} = None$

This configuration was used both in the baseline experiments and in the implementations of the classifiers with the Genetic Algorithm. The performance of the algorithms was evaluated using the precision metric, as defined in Equation 5. Where $True\ positives\ (TP)$ refer to instances correctly classified as positive, while $False\ positives\ (FP)$ refer to instances incorrectly classified as positive.

$$Precision = \frac{True\ positives}{(True\ positives + False\ positives)} \tag{5}$$

It is important to note that in this study, precision is used as the primary performance metric. Precision, also known as the positive predictive value, is defined in Equation 5. This metric should not be confused with accuracy, which also considers true negatives.

Genetic Algorithm

For the Genetic Algorithm (GA), the experimental setup used a stopping criterion based on a maximum of 5000 generations. Preliminary tests showed that increasing the number of generations to 10000 resulted in significantly longer execution times without a meaningful reduction in error, justifying the choice of 5000. Each generation consisted of 10 individuals. The crossover rate was set to 80%, and the mutation probability per gene was 30%, with mutation values randomly drawn from the range $[-1, 1]$. Elitism was applied to 10% of the population, and during initialization, each gene in the weight list was randomly assigned a value between 0 and 10.

3. Results

This section presents the experimental analysis of the precision obtained on each dataset, both without applying the Genetic Algorithm (baseline) and with the proposed GA-based approach. When classifiers were trained and tested on each dataset without instance weighting, the results obtained are shown in Table 2.

As shown in Table 2, the Support Vector Machine (SVM) achieved the highest precision scores on the Breast Cancer, Diabetes, Ionosphere, and Lymphography datasets, while Logistic Regression produced the best result for the Hepatitis

dataset. In contrast, when instance weights were assigned and optimized using the Genetic Algorithm (GA), an overall improvement in precision was observed across all experiments, as presented in Table 3.

Table 3 reports the average precision across the entire population after completing the evolutionary process. In contrast, Table 4 presents the best precision values obtained from the best individual across ten independent runs for each experiment.

Table 2 Precision scores of classifiers in the baseline experiment (no instance weighting).

	Naïve Bayes	Logistic Regression	SVM
Breast-cancer	61.94	56.72	65.85
Diabetes	66.5	71.3	71.82
Hepatitis	62.93	79.46	62.93
Ionosphere	89.68	91.62	96.88
Lymphography	46.21	58.33	58.73

Source: own elaboration.

Table 3 Average precision over ten runs with GA-optimized instance weights.

	Naïve Bayes	Logistic Regression	SVM
Breast-cancer	76.7	90.6	93.2
Diabetes	77	78.1	80
Hepatitis	88.3	96.8	94.4
Ionosphere	98.8	98.5	98.9
Lymphography	54.9	93.8	66.5

Source: own elaboration.

Table 4 Best precision across ten runs with GA-optimized instance weights.

	Naïve Bayes	Logistic Regression	SVM
Breast-cancer	80.39	80.39	94.4
Diabetes	84.9	79.06	80.16
Hepatitis	88.33	97.91	96
Ionosphere	98.9	98.9	98.9
Lymphography	60.41	98.03	98.03

Source: own elaboration.

When comparing Tables 2 and 3, it is evident that in all cases, the average precision obtained with evolved weights outperformed the baseline results, where SVM consistently outperforms Naïve Bayes and Logistic Regression. Furthermore, when comparing the best individual (Table 4) against both the average precision values with evolved weights (Table 3) and the baseline results (Table 2), improvements are

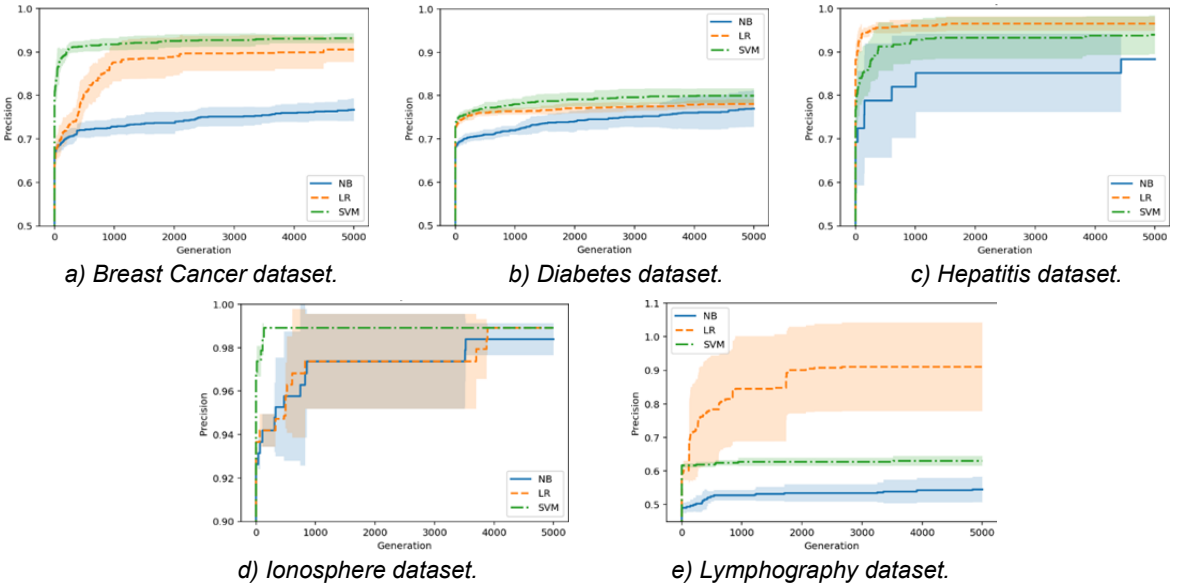
observed in all cases. This indicates that assigning a list of weights to training instances and optimizing them using a Genetic Algorithm yield enhanced and more effective weight values for each classifier.

Table 4 shows that, for the Ionosphere dataset, all classifiers achieved the same precision. In contrast, Naïve Bayes and SVM were the lowest-performing classifiers on the Hepatitis dataset. Similarly, Naïve Bayes did not perform as well as the other classifiers on the Lymphography dataset. For the remaining datasets, Naïve Bayes achieved the highest precision on Diabetes, while SVM outperformed the other models on Breast Cancer. A high standard deviation is observed in small datasets such as Hepatitis and Lymphography. This is primarily due to the limited number of instances, which makes each data split exert a strong influence on performance. Furthermore, the presence of imbalanced classes amplifies variability across runs. In contrast, larger datasets such as Breast Cancer or Diabetes exhibit more stable curves, as the sample size is sufficiently large to mitigate the impact of randomness during training.

As an example, the precision performance of all classifiers using the Genetic Algorithm (GA) is illustrated in Figure 4. In this figure, NB and LR refer to Naïve Bayes and Logistic Regression, respectively. Figure 4a shows the results for the Breast Cancer dataset, where it can be observed that the Naïve Bayes classifier consistently underperforms compared to the other classifiers throughout the evolutionary process. Most classifiers reach their peak precision around generation 4000. Figure 4b presents the results for the Diabetes dataset. In this case, Logistic Regression and SVM exhibit a nearly parallel increase in precision over the generations, while Naïve Bayes shows a steady but more gradual improvement across the entire evolution. Figure 4c corresponds to the Hepatitis dataset. Here, Naïve Bayes achieves its highest precision early in the process, while Logistic Regression and SVM show a progressive increase up to around generation 3000, after which their performance stabilizes. By the end of the evolutionary process, both classifiers outperform Naïve Bayes on this dataset. In Figure 4d, which shows the results for the Ionosphere dataset, most classifiers reach high precision values early in the process, around generation 1000. After generation 3000, all classifiers

converge to their maximum performance. For the Lymphography dataset (not shown), a similar pattern is observed: most classifiers reach peak precision within the first 1000 generations and exhibit little change thereafter, consistent with the behaviors seen in Figure 4.

To further analyze the GA's behavior during evolution, the Diabetes dataset with the Naïve Bayes classifier was selected. A full sample of 5000 generations was examined, and the results are shown in Figure 5, which presents the evolution of the best, average, and worst individuals based on their precision values.

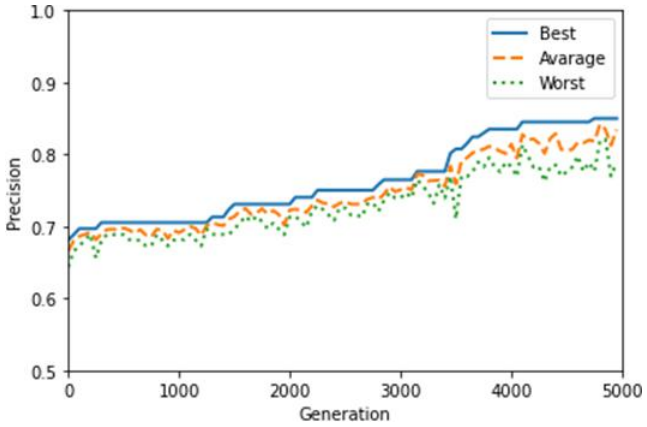


a) Breast Cancer dataset. *b) Diabetes dataset.* *c) Hepatitis dataset.*

d) Ionosphere dataset. *e) Lymphography dataset.*

Source: own elaboration.

Figure 4 Precision performance of the best individual obtained.



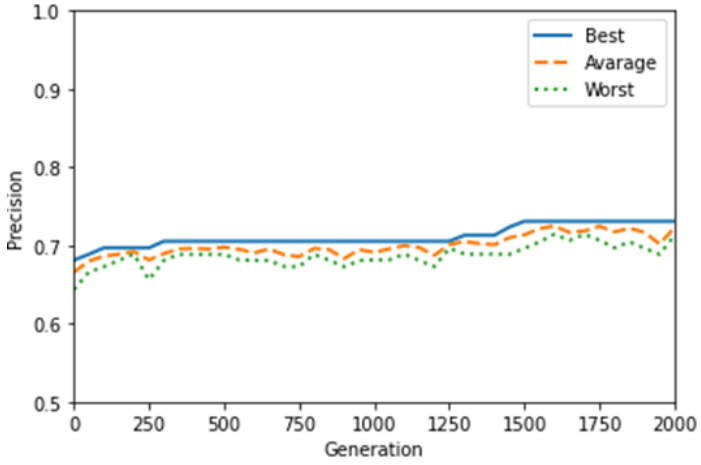
Source: own elaboration.

Figure 5 Evolution of the best, average, and worst individuals for the Diabetes dataset.

As shown in Figure 5, the average and worst individuals in the population initially achieve relatively high precision values, which may later decrease before improving again in subsequent generations. This fluctuation in performance may result from several factors, including the use of rank-based selection and arithmetic crossover, where offspring are generated by combining 80% of the first parent's genes with 20% from the second parent. It is also important to note that offspring have a 30% probability of undergoing mutation, which introduces random changes in their genes. These changes may lead to either improvement or degradation in performance. While the mutation operator facilitates exploration of new regions in the search space—sometimes producing less fit individuals—improvements are preserved over generations through the elitism operator, which ensures that the best-performing individuals are carried over unchanged.

Additionally, it is possible for the best individual in a population to be crossed with lower-performing individuals, which in some cases leads to offspring that outperform their parents. This behavior is reflected in the gradual improvement of both the average and worst individuals over generations, as illustrated in Figure 5.

Figure 6 provides a clearer view of the oscillation between improvements and declines in precision over generations by zooming in on the first 2000 generations from Figure 5. It is important to note that Figure 5 was subsampled to enhance the clarity of the graphical representation.



Source: own elaboration.

Figure 6 Zoomed-in view of the first 2000 generations from Figure 5.

4. Discussion

In terms of the results obtained, significant improvements are observed in all classifiers when using the Genetic Algorithm to adapt the weights of each instance. It is important to note that the execution time varies considerably for each dataset, ranging from a couple of hours to approximately 10 hours per experiment. This wide range is due to the large number of instances and features present in each dataset. The Genetic Algorithm approach proves particularly beneficial compared with manually finding the optimal weights for each instance, which would be highly complex and time-consuming for a specialist.

Regarding precision, it is observed that a rapid adaptation of the values for each dataset occurs in the first few generations, typically around 2000 generations. From that point onward, accuracy gradually improves as the generations progress.

Making a comparison with the literature, as far as the authors know, no similar performance has been reported for the Lymphography dataset. In Zwitter [1988], no previous citation or performance metric for this dataset is reported either. In Goix [2016], the Least Squares Anomaly Detection (LSAD) algorithm achieved an AUC (Area Under the Curve – ROC) score of 0.978 for the Ionosphere dataset. Note that ROC AUC values range from 0 to 1. In Quinn [2014], another anomaly-detection method reported an AUC of 0.958 for the same dataset.

In the present study, a value of 98.9 was obtained for the best individual found through the evolution of the weight list (Table 4), using all four classifiers. This supports the importance of weight assignments and their evolution.

5. Conclusion

In this work, a Genetic Algorithm (GA) was proposed to improve the performance of four classifiers by assigning a list of instance-specific weights for various datasets. The evolved weights enabled the classifiers to handle complex cases more accurately than when the same models were used without weighting. The results support the conclusion that assigning individual weights to each instance—and adapting them through a Genetic Algorithm tailored to the specific classifier and dataset—effectively improves precision. In this context, GAs serves as a powerful

tool for automatically discovering near-optimal weight configurations, overcoming the impracticality of manual weight tuning by experts, especially considering that each weight must be optimized for every instance and classifier combination.
Given the observed performance improvements, further research is encouraged to extend this approach by incorporating additional datasets, classifiers, and possibly exploring other evolutionary strategies to enhance scalability and generalization.

Acknowledgments

The authors acknowledge the computing resources provided by the Centro Universitario UAEM Valle de México (CLU-CUUAEM-VM) at the Universidad Autónoma del Estado de México (UAEMex), which were essential for carrying out the experiments in this study. The authors also express their gratitude to the Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT), now the Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI), for the scholarship support that made this research possible.

6. References and Bibliography

[1] Aparicio-Montelong, I., Celaya-Padilla, J. M., Luna-García, H., Galván-Tejada, C. E., Galván-Tejada, J. I., & Gamboa-Rosales, H. Predicción de Enfermedades Cardíacas Derivadas de Diabetes mediante Algoritmos Genéticos: Caso de Estudio. Research in Computing Science, No. 151, pp. 159–172, 2022.

[2] Goix, N., Drougard, N., Brault, R., & Chiapino, M. One Class Splitting Criteria for Random Forests. Asian Conference on Machine Learning, 2016.

[3] Hepatitis, (1988). UCI Machine Learning Repository. https://doi.org/10.24432/C5Q59J.

[4] Kahn, M., (1999). Diabetes [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5T59G.

[5] Mosquera, R., Castrillón, O. D., & Parra, L. Máquinas de Soporte Vectorial, Clasificador Naïve Bayes y Algoritmos Genéticos para la Predicción de

Riesgos Psicosociales en Docentes de Colegios Públicos Colombianos. Información Tecnológica, Vol. 29, No. 6, pp. 153–162, 2018.

[6] Padilla-Navarro, C., González-Reyna, S., Aguilera-González, G., Ortega-Yepez, M., Bombela-Jiménez, S., Rangel-Huerta, M., & Lino-Ramírez, C. Algoritmos Genéticos Aplicados a la Optimización de Características en la Clasificación de Arritmias Cardiacas Utilizando los Clasificadores KNN y Naïve Bayes. Research in Computing Science, No. 134, pp. 55–68, 2017.

[7] Quinn, J. A., & Sugiyama, M. A Least-Squares Approach to Anomaly Detection in Static and Sequential Data. Pattern Recognition Letters, No. 40, pp. 36–40, 2014.

[8] Sigillito, V., Wing, S., Hutton, L., & Baker, K., (1989). Ionosphere. UCI Machine Learning Repository. https://doi.org/10.24432/C5W01B.

[9] Valencia, P. E. Optimización mediante Algoritmo Genético. Anales del Instituto de Ingenieros de Chile, Vol. 109, No. 2, pp. 83–92, 1997.

[10] Zwitter, M., & Soklic, M., (1988). Breast Cancer. UCI Machine Learning Repository. https://doi.org/10.24432/C51P4M.

[11] Zwitter, M., & Soklic, M., (1988). Lymphography. UCI Machine Learning Repository. https://doi.org/10.24432/C54598.