Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Efficient GGO-XGBoost Optimization for Heart Failure Survival Prediction

A. Fallah*[1], H. Pirkhedri[2] and M. M. Momen[3]

[1]Department of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran.
[2]Department of Computer Engineering, Bu-Ali Sina University, Hamedan, Iran.
[3]Department of Computer Engineering, Isfahan University of Technology, Isfahan, Iran.

ABSTRACT

Heart failure mortality-outcome prediction requires models that are both clinically sensitive and methodologically reliable, especially when class imbalance and data leakage can distort performance estimates. This study proposes an efficient leakage-controlled GGO-XGBoost framework for binary mortality-outcome prediction using the UCI heart failure clinical records dataset. The outcome is defined by `DEATH_EVENT`, where class 0 indicates survival during follow-up and class 1 indicates death during follow-up. The proposed framework combines pipeline-based preprocessing, training-only imbalance handling, and stability-aware Greylag Goose Optimization for tuning XGBoost hyperparameters. Logistic Regression, SVM, Random Forest, and baseline XGBoost are used as comparative models under the same evaluation protocol. On the independent test set, the optimized GGO-XGBoost model achieved AUC 0.8768, recall 0.7895, F1-score 0.7317, accuracy 0.8167, and precision 0.6818. These results suggest that constrained metaheuristic optimization can improve clinically relevant classification behavior in imbalanced heart failure outcome prediction, provided that leakage control and conservative model selection are carefully maintained.

Keyword: Heart failure prediction; Mortality prediction; Greylag Goose Optimization; XGBoost; Clinical machine learning.

ARTICLE INFO

Article history:
Research paper
Received 11 June 2026
Accepted 02 July 2026
Available online 07 July 2026

AMS subject Classification: 68T05, 90C59.

*Corresponding author: A. Fallah. Email: amirhossein.fallah@email.kntu.ac.ir

1 Introduction

Heart failure mortality-outcome prediction is a clinically meaningful binary classification problem because the outcome is directly related to follow-up prioritization, early warning decisions, and patient monitoring. In this paper, the phrase "survival prediction" refers to prediction of the observed binary survival/death outcome label in the dataset. It does not refer to classical survival analysis models such as Cox regression or Kaplan-Meier estimation. This distinction is important because the implemented model predicts `DEATH_EVENT` as a class label rather than estimating a time-dependent hazard function.

In this setting, false negatives are particularly costly: a patient who later experiences a death event may be classified as low risk and may therefore receive insufficient clinical attention. Recent studies on heart disease prediction have emphasized that medical machine-learning models must be developed with careful preprocessing, robust validation, and explicit attention to class imbalance rather than relying only on the choice of classifier [1]. Sensor-oriented and Internet of Medical Things frameworks also show that cardiac prediction models should be reproducible, compact, and interpretable enough to support practical decision-making environments [2].

The present work focuses on the public UCI heart failure clinical records dataset. The prediction target is `DEATH_EVENT`: class 0 means that the patient survived during the follow-up period, whereas class 1 means that the patient died. The input space contains 12 clinical attributes: age, anaemia, creatinine phosphokinase, diabetes, ejection fraction, high blood pressure, platelets, serum creatinine, serum sodium, sex, smoking status, and follow-up time. The dataset contains 299 records and is moderately imbalanced, with 203 survival cases and 96 death-event cases. These properties make leakage control, fold-level stability, and conservative model tuning essential.

This study proposes an efficient GGO-XGBoost pipeline. The implementation downloads the dataset directly from the UCI repository, splits the data before fitting any preprocessing object, and embeds imputation, scaling, encoding, class balancing, model training, hyperparameter optimization, and evaluation in a unified pipeline. Logistic Regression, SVM, Random Forest, and baseline XGBoost are retained as baselines under the same data split, preprocessing procedure, balancing mechanism, and evaluation metrics. The final XGBoost model is tuned by Greylag Goose Optimization (GGO). The contribution is therefore not merely a tuned classifier, but a compact and reproducible binary mortality-outcome prediction workflow for tabular clinical data.

2 Related Work

Prior research on cardiac prediction has moved from conventional tabular machine learning toward hybrid optimization, reduced-feature models, deep learning, and explainable AI. A systematic review of machine-learning models for heart disease prediction reports that performance is strongly affected by preprocessing, feature selection, validation design, and class-imbalance handling, especially when public datasets are heterogeneous or class-imbalanced [1]. This finding is important for the present study because the UCI heart failure dataset is class-imbalanced; a model can easily appear stronger than it is if preprocessing, sampling, or feature engineering is performed before the train-test split.

Feature-sensitized and sensor-based prediction systems have highlighted the value of compact models that can be connected to practical monitoring environments [2]. Optimization-based feature selection has also been explored for early cardiovascular detection, including a snake-optimization feature-selection framework designed to improve rapid disease identification [3]. App-based reduced-feature prediction and explainable AI systems show a related trend: a clinically useful cardiac model should be accurate, light enough to deploy, and understandable to users rather than only optimized for a single headline score [4]. These studies support the design choice of combining a disciplined pipeline with an optimization strategy rather than training an isolated classifier.

Explainable and advanced learning approaches have expanded this literature. XAI-HD introduced an

explainable artificial-intelligence framework for heart disease detection and reinforced the need for transparent predictions in medical classification [5]. Transformer-based deep learning has been used for incident heart-failure prediction, showing that more expressive architectures may be useful when richer longitudinal clinical records are available [6]. Stacking ensembles with K-fold cross-validation have also been applied to improve stability in heart-disease prediction [7]. Other studies have investigated ant-colony-optimized neural models, hybrid deep learning with wavelet-based medical-imaging features, and reliable interpretable AI for acute myocardial infarction prognosis [8–10].

The gap addressed in the present work is narrower and practical. Instead of proposing a high-capacity deep architecture, the study focuses on a public clinical dataset and asks whether a carefully constrained metaheuristic search can improve clinically relevant metrics while preserving leakage control. The resulting framework connects five components that are often treated separately: split-before-fit validation, pipeline-based preprocessing, training-only class balancing, stability-aware GGO optimization, and independent test-set reporting.

3 Materials and Methods

3.1 Dataset and target definition

The dataset used in this work is the `heart_failure_clinical_records_dataset`. It is downloaded directly inside the code from the UCI repository; therefore, no local dataset upload is required to reproduce the experiment. After reading the data, the target column `DEATH_EVENT` is separated from the 12 predictors. Class 0 denotes survived/no death during the follow-up interval, and class 1 denotes death event. The stratified 80/20 split produces 239 training records and 60 independent test records. The training subset contains 162 survival cases and 77 death-event cases, while the test subset contains 41 survival cases and 19 death-event cases.

The feature `time` is retained as an input in the implemented version because it is part of the original dataset and the supplied code uses all predictors after separating `DEATH_EVENT`. However, `time` represents the follow-up duration recorded after patient inclusion. Therefore, from a deployment perspective, its clinical availability at the intended prediction moment must be checked carefully. The present article reports the model exactly according to the implemented code, while noting that future prospective use should define whether `time` is known at prediction time or should be excluded for a strictly baseline-risk setting.

3.2 Preprocessing and class-imbalance handling

All transformations are performed inside a pipeline. Numeric predictors are imputed with the median and scaled using RobustScaler. This choice is appropriate for clinical variables such as creatinine phosphokinase, platelets, serum creatinine, and follow-up time, which may contain skewness or outlying values. Categorical predictors, including anaemia, diabetes, high blood pressure, sex, and smoking, are imputed with the most frequent value and converted by one-hot encoding. The preprocessing object is never fitted on the full dataset before splitting.

The training minority-class ratio is approximately 0.322, so the code activates SMOTEENN. SMOTEENN combines synthetic minority oversampling with edited-nearest-neighbor cleaning. In this study, it is inserted after preprocessing and before the classifier inside the imbalanced-learning pipeline. Consequently, resampling is performed only during training folds or the final training fit. The validation folds and independent test set are not resampled. Baseline models include Logistic Regression, SVM, Random Forest, and baseline XGBoost. Each baseline is trained and assessed under the same split, preprocessing, balancing, cross-validation, and test-set evaluation protocol as the final model.

3.3 GGO-XGBoost optimization

The final model is an XGBoost classifier optimized by Greylag Goose Optimization, a nature-inspired population-based algorithm [11]. Each candidate solution represents a nine-dimensional XGBoost hyperparameter vector: number of estimators, maximum depth, learning rate, subsample ratio, column-sampling ratio, L2 regularization, minimum child weight, gamma, and L1 regularization. The search space is intentionally conservative to reduce overfitting on a clinical tabular dataset: estimator count is limited to 50–220, maximum depth to 2–4, learning rate to 0.01–0.12, subsampling and feature-sampling ratios to 0.70–0.95, L2 regularization to 1–50, minimum child weight to 3–15, gamma to 0–5, and L1 regularization to 0–10.

The baseline XGBoost configuration is included as one member of the initial population so that the optimizer begins from a credible reference point rather than from purely random candidates. The code evaluates five GGO meta-configurations: population/iteration pairs of 6/6, 8/8, 10/8, 10/10, and 12/10. The best selected configuration is population size 12 and 10 iterations, with best cross-validation fitness 0.843299. The selected XGBoost parameters are `n_estimators`=142, `max_depth`=2, `learning_rate`=0.112159, `subsample`=0.791598, `colsample_bytree`=0.838472, `reg_lambda`=1.054215, `min_child_weight`=4, `gamma`=0.012208, and `reg_alpha`=0.815770.

The implemented fitness function is computed from stratified five-fold cross-validation on the training data. It rewards discrimination, sensitivity, F1-score, and accuracy, while penalizing fold-level instability and excessive complexity. For a candidate hyperparameter vector $\boldsymbol{x}$, the base score is

$$B(\boldsymbol{x}) = 0.40\overline{AUC} + 0.30\overline{Recall} + 0.20\overline{F1} + 0.10\overline{Accuracy}. \tag{1}$$

The stability penalty is based on the standard deviations of AUC and F1-score across folds:

$$P_s(\boldsymbol{x}) = 0.10\sigma_{AUC} + 0.05\sigma_{F1}. \tag{2}$$

The complexity component follows the code-level definition:

$$C(\boldsymbol{x}) = 0.30\frac{n_{est}}{220} + 0.30\frac{d_{max}}{4} + 0.20\frac{1}{1+\lambda+\alpha} + 0.20\frac{1}{w_{min}}, \tag{3}$$

where n_{est} is the number of estimators, d_{max} is the maximum tree depth, λ and α are the L2 and L1 regularization terms, and w_{min} is the minimum child weight. The complexity penalty is then

$$P_c(\boldsymbol{x}) = 0.03C(\boldsymbol{x}), \tag{4}$$

and the final objective maximized by GGO is

$$F(\boldsymbol{x}) = B(\boldsymbol{x}) - P_s(\boldsymbol{x}) - P_c(\boldsymbol{x}). \tag{5}$$

This formulation matches the code: it rewards high AUC and recall most strongly, uses F1-score and accuracy as secondary criteria, and discourages candidates that are unstable across folds or unnecessarily complex.

3.4 Leakage-control and validation protocol

Leakage control is a central part of the implemented method. The dataset is split before preprocessing, resampling, hyperparameter search, and model selection. During cross-validation, the preprocessing transformers are fitted only on the corresponding training fold. Median imputation, RobustScaler, most-frequent categorical imputation, and one-hot encoding are therefore estimated from training data only. The fitted transformations are then applied to the validation fold.

The same principle is used for class balancing. SMOTEENN is placed inside the imbalanced-learning pipeline, not applied to the full dataset. Therefore, synthetic examples and edited-nearest-neighbor cleaning are generated only from training data. The independent test set is held out from preprocessing fit, resampling,

GGO fitness evaluation, model selection, and threshold-dependent reporting until the final evaluation. This matters because applying resampling or scaling before the split would allow test-set information to influence the learned representation and could inflate performance estimates.

The baseline and final models are compared using the same five test-set metrics: AUC, recall, F1-score, accuracy, and precision. No threshold optimization was applied; recall, F1-score, accuracy, and precision were computed using the default 0.5 decision threshold. Cross-validation on the training partition is used only inside model development and GGO fitness evaluation; the reported performance comparison is based on the independent test partition. The same random state is used for the train-test split, cross-validation, SMOTEENN, tree-based models, and GGO search, which supports reproducibility of the reported results.

3.5 Code-derived implementation summary

The computational procedure follows the notebook blocks directly. Table 1 summarizes how the main code blocks map into the implemented research workflow. Algorithm 1 then gives the formula-based GGO update and selection logic used to tune XGBoost.

Table 1: Code-derived workflow used to build and evaluate the final model.

Code stage	Role in the experimental workflow
Dataset loading and target split	Download the UCI heart failure records, define `DEATH_EVENT` as the binary label, and separate predictors from the target.
Train-test split and preprocessing	Create the stratified 80/20 split, then build numeric and categorical preprocessing branches inside a single pipeline.
Balancing and baselines	Activate SMOTEENN when the training minority ratio is below 0.40 and evaluate Logistic Regression, SVM, Random Forest, and baseline XGBoost under the same protocol.
GGO fitness and search	Encode XGBoost hyperparameters as normalized candidate positions, evaluate them by five-fold training cross-validation, and optimize the stability-aware fitness function.
Final evaluation and export	Refit the selected GGO-XGBoost pipeline on the full training set, evaluate the independent test set, save the model, and export the metric table and plots.

4 Results and Discussion

4.1 Baseline and final-model comparison

Table 2 reports the independent test-set performance of the four baseline models and the final optimized GGO-XGBoost model. All rows are evaluated on the same held-out test partition, using the same preprocessing and leakage-controlled evaluation protocol. Baseline XGBoost achieved the strongest AUC, whereas the optimized GGO-XGBoost model produced the best recall, F1-score, and accuracy. This comparison keeps the final claim precise: the optimization did not improve global discrimination measured by AUC, but it improved the threshold-dependent detection profile for the positive death-event class.

The baseline rows in Table 2 show that baseline XGBoost produced the highest AUC, equal to 0.908858, but its recall was 0.684211. For mortality-outcome prediction, this limitation is meaningful because the positive class corresponds to death-event cases. Random Forest produced an AUC of 0.863286, while Logistic Regression and SVM achieved AUC values of 0.783055 and 0.759949, respectively. These values indicate that nonlinear ensemble models were better suited to the dataset than the linear or kernel baselines under the current preprocessing and balancing protocol.

The final GGO-XGBoost row in Table 2 shows a different trade-off. Its AUC was 0.876765, which is lower than the baseline XGBoost AUC, but its recall increased from 0.684211 to 0.789474. Its F1-score also

Algorithm 1 Formula-based GGO optimization for XGBoost hyperparameters.

Require: Training data $\mathcal{D}_{tr}$, search domain $\Omega = [0,1]^9$, population size N, iterations T
Ensure: Best normalized position $\boldsymbol{x}^*$, decoded XGBoost parameters $h(\boldsymbol{x}^*)$, and best fitness F^*

Initialize population $\mathcal{P}^0 = \{\boldsymbol{x}_1^0, \ldots, \boldsymbol{x}_N^0\}$ with $\boldsymbol{x}_i^0 \in \Omega$.
Encode the baseline XGBoost configuration and set it as $\boldsymbol{x}_1^0$.
Evaluate $F(\boldsymbol{x}_i^0)$ for all candidates using the leakage-controlled training pipeline.
Set $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}_i^0 \in \mathcal{P}^0} F(\boldsymbol{x}_i^0)$ and $F^* = F(\boldsymbol{x}^*)$.
for $t = 0, 1, \ldots, T-1$ **do**
    Compute progress $q_t = t/T$ and exploration factor $a_t = 1.5(1 - q_t)$.
    Compute the flock mean $\bar{\boldsymbol{x}}^t = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i^t$.
    for $i = 1, 2, \ldots, N$ **do**
        Draw $\boldsymbol{r}_1, \boldsymbol{r}_2 \sim U(0,1)^9$ and $u \sim U(0,1)$.
        if $u < 0.50$ **then**
            Generate a guided candidate

$$\boldsymbol{z}_i^t = \boldsymbol{x}_i^t + a_t \boldsymbol{r}_1 \odot (\boldsymbol{x}^* - \boldsymbol{x}_i^t) + 0.30 \boldsymbol{r}_2 \odot (\bar{\boldsymbol{x}}^t - \boldsymbol{x}_i^t).$$

        else
            Generate a local exploratory candidate

$$\boldsymbol{z}_i^t = \boldsymbol{x}^* + \mathcal{N}\left(\boldsymbol{0},\ [0.08(1 - q_t)]^2 I\right).$$

        end if
        if $U(0,1) < 0.05$ **then**
            Apply mutation $\boldsymbol{z}_i^t \leftarrow \boldsymbol{z}_i^t + \mathcal{N}(\boldsymbol{0}, 0.10^2 I)$.
        end if
        Project to the feasible domain $\tilde{\boldsymbol{x}}_i^t = \Pi_\Omega(\boldsymbol{z}_i^t)$.
        Decode $\tilde{\boldsymbol{x}}_i^t$ to XGBoost parameters $h(\tilde{\boldsymbol{x}}_i^t)$ and compute $F(\tilde{\boldsymbol{x}}_i^t)$.
        if $F(\tilde{\boldsymbol{x}}_i^t) > F(\boldsymbol{x}_i^t)$ **then**
            $\boldsymbol{x}_i^{t+1} \leftarrow \tilde{\boldsymbol{x}}_i^t$.
        else
            $\boldsymbol{x}_i^{t+1} \leftarrow \boldsymbol{x}_i^t$.
        end if
        if $F(\boldsymbol{x}_i^{t+1}) > F^*$ **then**
            $\boldsymbol{x}^* \leftarrow \boldsymbol{x}_i^{t+1}$ and $F^* \leftarrow F(\boldsymbol{x}_i^{t+1})$.
        end if
    end for
end for
return $\boldsymbol{x}^*$, $h(\boldsymbol{x}^*)$, and F^*.

Table 2: Independent test-set performance of baseline models and the final GGO-XGBoost model.

Model	AUC	Recall	F1-score	Accuracy	Precision
Baseline XGBoost	0.908858	0.684211	0.684211	0.800000	0.684211
Random Forest	0.863286	0.631579	0.631579	0.766667	0.631579
Logistic Regression	0.783055	0.631579	0.615385	0.750000	0.600000
SVM	0.759949	0.631579	0.585366	0.716667	0.545455
GGO-XGBoost (final)	0.876765	0.789474	0.731707	0.816667	0.681818

increased from 0.684211 to 0.731707, and its accuracy increased from 0.800000 to 0.816667. Precision remained essentially comparable, changing from 0.684211 for baseline XGBoost to 0.681818 for GGO-XGBoost. Therefore, the optimized model shifted the decision behavior toward improved positive-class detection without a meaningful precision loss. This is the central result of the experiment and avoids overclaiming superiority across all metrics.

4.2 Confusion matrix and curve-based analysis

Figure 1 reports the independent test-set confusion matrix. The final GGO-XGBoost model correctly classified 34 of the 41 survival/no-death cases and 15 of the 19 death-event cases. It produced 7 false positives and 4 false negatives. From a clinical screening perspective, the false-negative count is especially important because it represents death-event patients predicted as lower-risk. The model missed 4 such cases and detected 15, which matches the recall value of 0.789474 reported in Table 2.

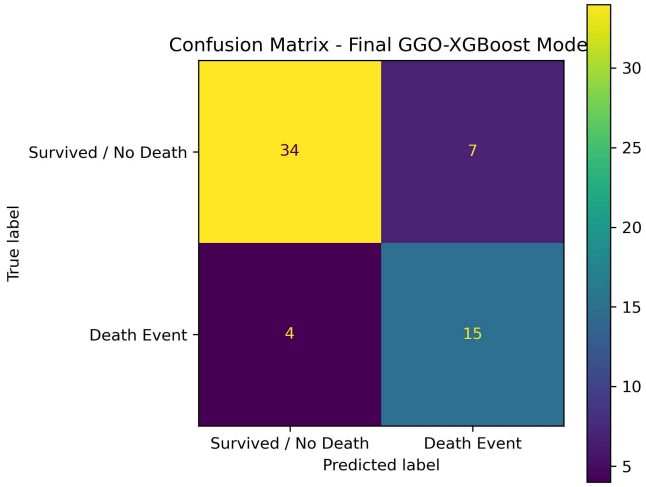

Figure 1: Confusion matrix of the final GGO-XGBoost model on the independent test set. The matrix contains 34 true negatives, 7 false positives, 4 false negatives, and 15 true positives.

The ROC curve in Fig. 2 reports AUC 0.88, showing that the optimized model maintains useful ranking ability between survival and death-event cases across classification thresholds. This value is consistent with the exact test AUC of 0.876765 in Table 2. Although baseline XGBoost produced a higher AUC in Table 2, the optimized model produced a more recall-oriented threshold-dependent profile, which is valuable when the cost of missing positive cases is high.

The Precision-Recall curve in Fig. 3 reports average precision 0.72. This curve is particularly informative for the present task because the death-event class is less frequent than the survival class. While ROC-AUC measures ranking ability across both classes, Precision-Recall analysis focuses on retrieval quality for the positive class. The AP value of 0.72 confirms that the final model provides a usable positive-class retrieval profile, although it also shows that the model is not perfect and should be externally validated before clinical deployment.

The classification report further supports this interpretation. For the survival/no-death class, precision was 0.89, recall was 0.83, and F1-score was 0.86 with support 41. For the death-event class, precision was 0.68, recall was 0.79, and F1-score was 0.73 with support 19. The weighted F1-score was 0.82. These values show that the final model is stronger at identifying survival cases, but the GGO-tuned configuration improves positive-class detection compared with the baseline XGBoost threshold profile. Overall, the results suggest a balanced and clinically reasonable trade-off: the model sacrifices a small amount of AUC relative to baseline XGBoost but gains sensitivity and F1-score for the clinically critical death-event class.

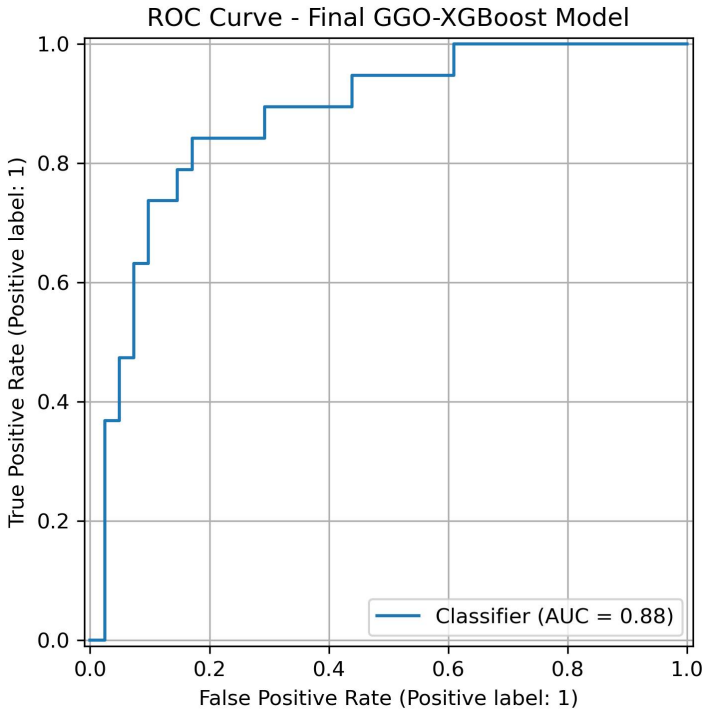

Figure 2: ROC curve of the final GGO-XGBoost model on the independent test set. The curve reports AUC 0.88, consistent with the exact test AUC of 0.876765.

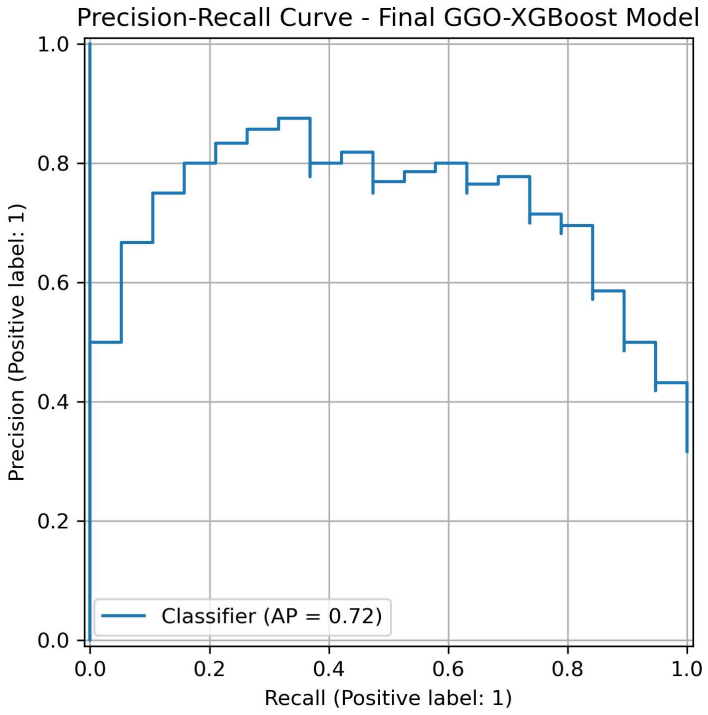

Figure 3: Precision-Recall curve of the final GGO-XGBoost model on the independent test set. The average precision is 0.72 for the positive death-event class.

5 Conclusion

This paper developed a leakage-controlled GGO-XGBoost pipeline for heart failure mortality-outcome prediction as a binary classification task. The workflow directly downloads the UCI dataset, separates the `DEATH_EVENT` target, applies preprocessing and SMOTEENN only inside the training pipeline, evaluates four baseline models, and optimizes XGBoost using a stability-aware GGO procedure. The methodology explicitly protects the independent test set from preprocessing fit, resampling, optimizer selection, and model-development decisions. This point is essential for medical datasets, where even minor leakage can create optimistic results.

On the independent test set, the final model achieved AUC 0.876765, recall 0.789474, F1-score 0.731707, accuracy 0.816667, and precision 0.681818. The model detected 15 of 19 death-event cases and missed 4. These results show that the optimized model favors detection of death-event cases, which is appropriate for a screening-oriented binary mortality-prediction task. The main claim is therefore deliberately limited: GGO-XGBoost improved recall, F1-score, and accuracy over baseline XGBoost, but it did not improve AUC. Future work should test the same pipeline on larger external cohorts, assess calibration and threshold selection in clinical settings, and examine whether the follow-up-time variable is available at the intended prediction moment.

References

[1] Banerjee, T. and Paçal, I., A systematic review of machine learning in heart disease prediction. Turkish Journal of Biology, 49(5), 600–634, 2025. DOI: 10.55730/1300-0152.2766. PMCID: PMC12614364.

[2] Heart disease prediction with a feature-sensitized interpretable framework for the Internet of Medical Things sensors. Frontiers in Digital Health, 7, 10.3389/fdgth.2025.1612915, 2025.

[3] A snake optimization algorithm-based feature selection framework for rapid detection of cardiovascular disease in its early stages. Biomedical Signal Processing and Control, 102, 107417, 2025.

[4] Advancing Heart Disease Prediction in App-Based Systems using Optimized Machine Learning Model with Reduced Features and Explainable AI. In Proceedings of the IEEE International Conference on Quantum Photonics, Artificial Intelligence, and Networking (QPAIN), IEEE Xplore, 2025.

[5] Talukder, M. A. R., Talaat, A. S., Kazi, M., and Khraisat, A., XAI-HD: an explainable artificial intelligence framework for heart disease detection. Artificial Intelligence Review, 58, 2025.

[6] Rao, S., Li, Y., Ramakrishnan, R., Hassaine, A., Canoy, D., Cleland, J., Lukasiewicz, T., Salimi-Khorshidi, G., and Rahimi, K., An explainable transformer-based deep learning model for the prediction of incident heart failure. IEEE Journal of Biomedical and Health Informatics, 26(7), 2022.

[7] Sultan, S. Q., Javaid, N., Alrajeh, N., and Aslam, M., Machine learning-based stacking ensemble model for prediction of heart disease with explainable AI and K-fold cross-validation: A symmetric approach. Symmetry, 2025.

[8] Xia, B., Innab, N., Kandasamy, V., Ahmadian, A., and Ferrara, M., Intelligent cardiovascular disease diagnosis using deep learning enhanced neural network with ant colony optimization. Scientific Reports, 2024.

[9] Palanisamy, C., Pachamuthu, K., and Ramamoorthy, A. K., Heart disease prediction using hybrid deep learning and medical imaging with wavelet-based feature extraction. International Journal of Reconfigurable and Embedded Systems (IJRES), 15(1), 183–193, 2026. DOI: 10.11591/ijres.v15.i1.pp183-193.

[10] Kim, M., Kang, D., Kim, M. S., Choe, J. C., Lee, S. H., Ahn, J. H., Oh, J.-H., Choi, J. H., Lee, H. C., Cha, K. S., Jang, K., Bong, W. R., Song, G., and Lee, H., Acute myocardial infarction prognosis prediction with reliable and interpretable artificial intelligence system. Journal of the American Medical Informatics Association, 2024.

[11] El-kenawy, E.-S. M., Khodadadi, N., Mirjalili, S., Abdelhamid, A. A., Eid, M. M., and Ibrahim, A., Greylag Goose Optimization: Nature-inspired optimization algorithm. Expert Systems with Applications, 238, 122147, 2024. DOI: 10.1016/j.eswa.2023.122147.