STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **15**, February 2026, pp 1099–1118.
Published online in International Academic Press (www.IAPress.org)

A Mathematical Programming Model with Cost Sensitivity in the Objective Function for Imbalanced Datasets Challenges

Redouane Hakimi*, Badreddine Benyacoub, Mohamed Ouzineb

Institut National de Statistique et d'Economie Appliquée, Laboratory MASAFEQ, Rabat, Morocco

Abstract This paper introduces CS-MSD, a cost-sensitive deviation minimization model designed to address a typical issue of imbalanced datasets in binary classification, which is a major problem in machine learning tasks across various areas. Imbalanced datasets, in which a single class significantly dominates the other, frequently generate biased models that neglect the minority class, making them extremely important in practical sectors like health services and financial services. The Traditional re-sampling techniques, including under-sampling and over-sampling, have associated limitations, such as information loss and over-fitting. CS-MSD overcomes these limitations by combining external deviations with cost sensitivity, which produces a perfect balance of minority and majority class costs. The model outperforms Decision Tree and Radial SVM, achieving a Recall of 0.958 on the win dataset, alongside Specificity and G-mean metrics. With CPU and wall times of 0.052 s and 0.054 s, respectively, CS-MSD also surpasses Random Forest and Bagging in computational efficiency, making it ideal for time-sensitive tasks. Its combined flexibility and processing speed establish CS-MSD as a vital solution for enhancing classification performance across diverse domains.

Keywords Mathematical programming, Binary classification, Imbalanced data, Cost sensitivity, MSD.

AMS 2010 subject classifications 90C90, 68Q32

DOI: 10.19139/soic-2310-5070-2549

1. Introduction

Binary classification is a key approach in machine learning (ML) that divides data into two separate groups based on different characteristics and patterns [1, 2, 3]. The main task is to differentiate between two outcomes, often referred to as positive and negative classes. Regardless of the algorithm used, the goal is to frequently adjust model parameters to minimize the gap between predicted and actual outcomes. However, binary classification faces substantial challenges when applied to imbalanced datasets, where class distributions are uneven. These datasets create significant hurdles, leading to biased models that favor the majority class while overlooking the minority class [4, 5, 6]. To mitigate this, techniques such as under-sampling and over-sampling have been developed [7, 8, 9, 10, 11, 12, 13, 14]. Under-sampling reduces the size of the majority class to balance the datasets, but this approach may result in the loss of important information and a bias toward the minority class. In contrast, over-sampling increases the size of the minority class by replication or synthesis, which can lead to over-fitting and reduced generalization. Additionally, cost-sensitive learning methods have gained prominence by accounting for the varying costs of classification errors [15]. By minimizing the costs of specific misclassifications, these methods improve sensitivity to errors in the minority class [16, 17].

The goal of this study is to explore cost-sensitive learning techniques through the application of mathematical programming (MP) technique. Non-parametric discriminant approaches using linear programming have been

*Correspondence to: Redouane Hakimi (Email: r.hakimi@insea.ac.ma). Institut National de Statistique et d'Economie Appliquée, B.P.:6217 Rabat-Instituts Rabat, Morocco.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

widely examined in the field of discriminant analysis. Glover [18, 38] compared these methods with traditional linear discriminant techniques, demonstrating LP's effectiveness in minimizing deviations beyond the two-group boundary, suggesting it as a competitive alternative approach. In another study, [19] investigated the integration of resampling methods into statistical discriminant MP models, highlighting their ability to improve classification performance and address limitations of deterministic models. Moreover, [39] emphasized the critical role of normalization in ensuring MP discriminant models do not yield functions with zero coefficients and constants. Additionally, [40] discussed the use of an objective function in MP discriminant analysis to minimize deviations and optimize classification accuracy within defined neighborhoods. This method, supported by constraints that ensure correct classification both within and outside these neighborhoods, enhances overall classification accuracy. While MP discriminant analysis techniques offer the advantage of building classification models based on variables specific to each observation [20], they are particularly suited for small-scale datasets.

Our goal in investigating the concepts of cost sensitivity is to create a model called cost-sensitive deviation minimization model (CS-MSD), which combines cost recognition with robust MP to successfully address diverse imbalanced datasets. This study is vital in many sectors, including health and medicine, where it provides an important role in job duties such as breast cancer classification. In this situation, correctly recognizing minority cases is critical to avoiding harmful results. Furthermore, in physics, chemistry and biology, our goal is to provide practical answers to real-world challenges, so pushing progress and stimulating innovation in these fields.

In the most crucial computations, CS-MSD performs well and strikes an efficient compromise between recall and precision. CS-MSD offers a more balanced approach with competing outcomes, which correlates to the top model in terms of sensitivity and specificity, even though alternative models can also obtain high AUCs. Furthermore, it is notable for its high computation efficiency, superior design from models like RF and bagging with a very short processing time. Strong results are consistently produced by CS-MSD across a variety of datasets, making it an excellent choice for time-sensitive assignments involving imbalanced datasets.

The following is how the paper is organized: The literature on technical resampling and cost sensitivity is reviewed in Section 2. We describe the sum of deviations model for binary classification that minimizes costs in Section 3. In Section 4, the study's methodology is described. In Section 5, the findings are addressed and presented. The research's conclusions are finally summed up in Section 6.

2. Related work

Resampling is an essential statistical technique used in binary classification to analyses samples and comprehend the general properties of a datasets. This strategy extracts useful insights by generating fresh datasets from a random selection of observations. Resampling techniques, notably under- and over-sampling, are critical for correcting class imbalances in imbalanced datasets. These algorithms reorganize the data to better reflect all classes. Furthermore, the idea of cost sensitivity is important in this context since it measures how weights influence decision-making. This judgment is especially important when working with data having unequal distributions. Overall, resampling and cost sensitivity combine to increase classification performance.

2.1. Resampling approach

Under-sampling is a frequent strategy in ML for dealing with datasets that are imbalanced. It seeks to balance sample sizes by randomly deleting instances from the majority class; however, this might result in the loss of critical information from the majority class, impeding optimization attempts. Several under-sampling strategies have been proposed, including KNN_Near [7], KNN_Order [8] and KNN_Ru [11]. Empirical investigations have demonstrated that KNN_Near, which employs a k-nearest neighbors technique, performs well. KNN_Order uses neighbor counts to balance datasets.

Furthermore, KNN_Ru is more effective than other approaches because it incorporates random sub-sampling and k-nearest neighbor analysis, which helps retain critical points close to the majority class. Random under-sampling has advantages, but it can also miss important information that is necessary for the induction process [21]. While the Scala under-sampling library [22] offers a comprehensive solution for managing imbalanced datasets, addressing

class distribution imbalances and challenges associated with skewed data, remainder balancing methods employ heuristics to address these issues.

On the other hand, oversampling is another popular ML technique for dealing with imbalanced datasets. The aim is to make synthetic samples in order to reduce class imbalance and increase minority class. When trying to handle imbalanced datasets in the context of classification, various over-sampling techniques are used.

A notable contribution to the field of imbalanced data management is presented in [23], which conducts a comprehensive comparative evaluation of 66 oversampling methods. The study innovatively employs two distinct ranking techniques - Borda Count and Kruskal-Wallis test - to systematically assess method effectiveness. Their analysis reveals that MCT, Polynom-fit-SMOTE, and CBSO emerge as consistently superior performers, demonstrating better results than traditional SMOTE across multiple datasets and classifiers. This work makes two key contributions: first, it highlights the critical role of ranking methodology selection when evaluating imbalance-handling techniques; second, it provides practical, evidence-based guidance for practitioners in selecting appropriate data management approaches.

SMOTE is a well-known method that helps reduce over-fitting. The main idea behind SMOTE [24, 25] is to generate synthetic samples for the minority class by interpolating between neighboring minority class instances. However, one limitation of SMOTE is that it treats all minority class samples equally, ignoring variations in difficulty or distribution. SMOTE-D [10] improves on this by consistently creating synthetic items for the minority class while removing random samples. Another method, SWIM [14], utilizes the distribution of the majority class to generate synthetic, comparable instances, enhancing the classifier's effectiveness for the minority class.

Recent work [26] proposes SMOTE-DB-FSVM, an enhanced approach for diabetes detection in imbalanced datasets that refines traditional SMOTE by first applying density-based filtering and confidence scoring to minority class instances, reducing noisy synthetic data generation. The confidence scores are then integrated into a Density-Based Fuzzy SVM (DB-FSVM) optimization, prioritizing reliable synthetic samples, while a genetic algorithm optimizes the SVM kernel parameters for improved classification performance. This method addresses SMOTE's noise limitations by selectively generating higher-quality synthetic samples.

In other studies investigating ML approaches to address class imbalance in diabetes prediction using large-scale health datasets, [27] examined algorithmic solutions and data augmentation techniques (including oversampling, undersampling, and hybrid methods) applied to the BRFSS dataset, where diabetic cases constitute a significant minority. Their work demonstrates that strategic dataset balancing prior to model training enhances sensitivity to the minority class, yielding more clinically reliable predictions.

2.2. Cost-sensitivity technique

Recent interest has surged in developing ML models adept at capturing complex variable interplay while considering associated expenditures. Cost-sensitive learning, a subset of ML, directly incorporates relative misclassification costs, preventing biased models, especially crucial in medical diagnoses for misclassification.

Incorporating cost considerations into the feature selection process represents a significant advancement in improving the capabilities of ML algorithms [28]. Yotsawat proposed a novel framework called cost-sensitive extreme gradient boosting, specifically designed to tackle class imbalance in bankruptcy prediction models. This approach proves to be superior to other ensemble methods in identifying bankruptcy cases in real datasets [17]. The ICET method optimizes cost-sensitive classification decisions [29] by using a genetic algorithm that balances testing costs and classification errors through decision trees, allowing for effective resource allocation with conditional costs. A novel method in Support Vector Machines additionally addresses asymmetric misclassification costs [30] by limiting the number of features chosen while establishing upper bounds on false positive and negative rates. In terms of the main costs related to inductive concept learning are misclassification errors, testing costs, undesirable results and computational costs, which include size complexity [31]. Cost-sensitive training of individual base classifiers is optional when using a novel model for credit card fraud detection which includes cost-sensitive learning into a meta-learning ensemble [32]. Cost-sensitive logistic regression provides as the meta-learner in this model, which uses decision trees, multi-layer neurons and k-nearest neighbors as baseline learners.

A cost-sensitive Long Short-Term Memory (CSLSTM) model is proposed for pressurizer water level prediction in marine PWRs [33] to deal with the issues of class imbalance and temporal correlation shifts in time-series data.

Compared with standard LSTM and SVR models, this model handles better because it uses a unique cost-sensitive factor to weight training examples according to their temporal position and fluctuation level.

To address the challenges associated with present classification-based approaches, "cost-sensitive three-way class-specific" attribute reduction suggests a monotonous result cost for three-way decisions in a decision-theoretic rough set framework [34]. Algorithms based on addition-deletion and deletion strategies are developed to construct class-specific minimum cost reducts. Experiments on eight UCI datasets demonstrate the monotonicity of the proposed result cost and reveal that class-specific reducts outperform classification-based reducts in terms of misclassification and test costs. This approach underscores the value of tailoring attribute reduction to individual decision classes for improved cost-sensitive decision-making.

To enhance market power abuse identification in the Chinese electricity market, the KNBN-CSSVM-DBTC algorithm is proposed, effectively addressing data imbalance and high dimensionality issues [35]. This method integrates K-Nearest Bound Neighbor support vector pre-selection, a cost-sensitive SVM and a distance-based parameter optimization technique. Evaluations on constructed and real-world datasets demonstrate superior speed and recall compared to standard SVM approaches, achieving over 95% recall in most scenarios. Such efficiency facilitates real-time monitoring and proactive interventions against market manipulation.

An approach is introduced that integrates cost sensitivity directly into the learning process of four common ML algorithms by modifying their objective functions, thereby avoiding data manipulation and preserving the original data distribution [36]. Experiments on four medical datasets demonstrate the effectiveness of this cost-sensitive method, particularly in improving the prediction of the minority class compared to standard algorithms. This contribution aligns with our research on cost-sensitive learning by showcasing the practical benefits of algorithmic modifications to address class imbalance. While our work emphasizes mitigating majority-class bias through integrated cost-sensitive error minimization within the objective function, their approach provides a comparative perspective on balancing computational efficiency and performance improvements. The simplicity of implementing standard algorithms and datasets is a notable strength, though the reliance on a heuristic class weighting scheme remains a limitation. Exploring advanced weighting strategies could further enhance predictive performance.

The challenge of early dementia detection is addressed through the prediction of Subjective Cognitive Decline (SCD) using the imbalanced BRFSS dataset [37]. Their work contributes significantly to the field of MP, focusing on cost sensitivity in imbalanced datasets by combining data-resampling techniques (e.g., SMOTE-NC, ADASYN) with cost-sensitive training methods and ensemble learning. This approach results in a notable improvement in sensitivity (58% compared to a 3% baseline), highlighting the potential of using simpler, population-level data for early risk identification.

By fusing a strong mathematical optimization framework with cost-sensitive principles, the suggested CS-MSD model expands on previous developments. In contrast to ensemble-based methods [17, 32], CS-MSD directly optimizes misclassification costs to reduce majority-class bias, guaranteeing better minority class detection performance. Its versatility and scalability are further demonstrated by its use in a variety of fields, such as industrial systems, finance and health. Although current approaches [33, 35] are effective in particular situations, CS-MSD's focus on computational efficiency and generalizability makes it a flexible option for imbalanced datasets.

The CS-MSD addresses a number of costs concurrently, such as misclassification and computation time, while ICET uses evolutionary algorithms to optimize decision trees and CSLSTM adds unique cost-sensitive balancing to time-series data. Given the mathematical decision strategy, CS-MSD may provide more thorough and flexible solutions for a variety of fields.

3. Proposed model

The classification method can be formulated using MP with a matrix $X \in \mathbb{R}^{n\times r}$, represented as $X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$, where $n = n_B + n_G$ indicates the total number of observations, with n_B corresponding to the observations

identified as "bad" and n_G those identified as "good". The dataset's observations are denoted by a row vector $X_i = (x_{i1}, \ldots, x_{ir}) \in \mathbb{R}^r$. The acronym r represents the number of variables (features).

A scalar b, referred to as the intercept or bias term, is established as a threshold to distinguish the majority class G_G from the minority class G_B. In cases where the problem is linearly separable, the classes G_G and G_B do not overlap, meaning that each observation X_i belongs exclusively to either G_G or G_B.

Mathematically, the objective of an MP model is to determine the values for b and non-zero W that satisfy:

$$\begin{cases} \sum_{j=1}^{r} x_{ij}.w_j < b & \forall i \in G_B \\ \sum_{j=1}^{r} x_{ij}.w_j \geq b & \forall i \in G_G \end{cases} \tag{1}$$

Where $W = \begin{bmatrix} w_1 \\ \vdots \\ w_r \end{bmatrix}$ is a vector of real numbers in $\mathbb{R}^r$, representing the weights assigned to each feature. The goal is to identify the optimal values for W and b that effectively distinguish between the classes while minimizing classification errors.

We provide an extensive description of the terminology used in the mathematical model:

- **Matrix representation:** The dataset is represented by the matrix:

$$X = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1r} \\ x_{21} & x_{22} & \ldots & x_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nr} \end{bmatrix}$$

  of dimensions $(n \times r)$, where:

  - n is the total number of observations, where $n = n_B + n_G$.
  - n_B represents the number of observations classified as "bad."
  - n_G represents the number of observations classified as "good."
  - r is the total number of features (variables).

  The matrix X could alternatively be represented as:

$$X = (X_1, X_2, \ldots, X_n)^t$$

  where the symbol t denotes the transpose operation. Transposing a matrix swaps its rows and columns, each row vector X_i represents an observation consisting of r features:

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{ir}).$$

  Here, x_{ij} refers to the value of the j-th feature for the i-th observation, with $j \in \{1, \ldots, r\}$ and $i \in \{1, 2, \ldots, n\}$.
- **Intercept term:** The scalar b is the intercept or bias term of the model. It helps define the decision boundary separating the "good" class from the "bad" class.
- **Weight vector:** The weight vector W consists of coefficients assigned to each feature.
- **Class groups:** The sets G_G and G_B represent the "good" and "bad" classes, respectively.
  The goal is to find the optimal values of W and b that best distinguish the classes. The weight vector affects the decision boundary and its values are changed for reducing classification errors.

3.1. Model (MSD) for Minimizing the Sum of Deviations

Researchers [18] introduced deviation variables into the MP model to account for allowable margins of error, denoted by $\tilde{\epsilon}_i$ and ϵ_i. When a "bad" observation X_i is misclassified, ϵ_i indicates its distance from the boundary b. Similarly, when a "good" observation X_i is misclassified, $\tilde{\epsilon}_i$ measures the distance from b. These deviation terms are referred to as external deviations in [19].

To prevent trivial solutions (S) Eq. 2, we use Glover's [41] normalization approach and introduce the constraint Eq. 3. The sum over j from 1 to r of the product of n_B and the sum of x_{ij} for indices i belonging to the good class G_G, minus n_G times the sum of x_{ij} for indices i in the bad class G_B, all multiplied by w_j, equals 1.

$$S = \{w_j = 0\ \forall j,\ \tilde{\epsilon}_i = \epsilon_i = 0\ \forall i,\ b = 0\} \tag{2}$$

$$\sum_{j=1}^{r} \left(n_B \times \sum_{\substack{i=1 \\ i \in G_G}}^{n_G} x_{ij} - n_G \times \sum_{\substack{i=1 \\ i \in G_B}}^{n_B} x_{ij} \right) \times w_j = 1 \tag{3}$$

The MSD model is expressed as follows:

$$\begin{cases} \text{Min} & \displaystyle\sum_{i=1}^{n_G} \tilde{\varepsilon}_i \ + \sum_{i=1}^{n_B} \varepsilon_i \\ \text{S.T:} & \displaystyle\sum_{j=1}^{r} x_{ij}.w_j < b + \varepsilon_i \qquad \forall i \in G_B \\ & \displaystyle\sum_{j=1}^{r} x_{ij}.w_j \geq b - \tilde{\varepsilon}_i \qquad \forall i \in G_G \\ & \displaystyle\sum_{j=1}^{r} (n_B. \sum_{\substack{i=1 \\ i \in G_G}}^{n_G} x_{ij} - n_G. \sum_{\substack{i=1 \\ i \in G_B}}^{n_B} x_{ij}).w_j = 1 \\ & b \in \mathbb{R}, w_j \in \mathbb{R}, \qquad \forall j \\ & \varepsilon_i, \tilde{\varepsilon}_i \geq 0, \qquad \forall i \end{cases} \tag{4}$$

3.2. CS-MSD Model

In this section, we introduce the CS-MSD model, an innovative approach that first calculates the cost values for both good and bad groups, then integrates these values into the objective function of the standard MSD model. To achieve this, we assign cost values C_B and C_G to the G_B and G_G classes, respectively. The cost associated with the majority class is calculated as $C_G = \frac{n}{2 \cdot n_G}$, where n represents the total number of samples in the dataset, n_G denotes the number of samples in the majority class and the denominator's 2 corresponds to the number of classes [42]. For the minority class, we normalize costs so that they sum to one, which implies $C_B = 1 - C_G$. This relationship guarantees an inverse proportionality between the class costs: as C_G decreases for the majority class, C_B correspondingly increases for the minority class [43]. This cost structure intentionally imposes stronger penalties for misclassifying minority class observations.

The model can manage to adjust for minority class costs in comparison to majority class costs because of this balanced structure. In order to take these modifications into consideration, the MSD expression in Eq. 4 is reformulated as CS-MSD (Eq 5).

$$\begin{cases} \text{Min} \sum\limits_{\substack{i=1 \\ i\in G_B}}^{n_B} C_B.\varepsilon_i \ + \sum\limits_{\substack{i=1 \\ i\in G_G}}^{n_G} C_G.\tilde{\varepsilon_i} \\ \text{S.T:} \\ \quad \sum\limits_{j=1}^{r} x_{ij}.w_j < b + \varepsilon_i \qquad \forall i \in G_B \\ \quad \sum\limits_{j=1}^{r} x_{ij}.w_j \geq b - \tilde{\varepsilon_i} \qquad \forall i \in G_G \\ \quad \sum\limits_{j=1}^{r} (n_B. \sum\limits_{\substack{i=1 \\ i\in G_G}}^{n_G} x_{ij} - n_G. \sum\limits_{\substack{i=1 \\ i\in G_B}}^{n_B} x_{ij}).w_j = 1 \\ \quad b \in \mathbb{R}, w_j \in \mathbb{R}, \qquad \forall j \\ \quad \varepsilon_i, \tilde{\varepsilon_i} \geq 0, \qquad \forall i \end{cases} \tag{5}$$

Analysis of the CS-MSD Model:

1. **Objective Function:**
   The objective function is derived by minimizing the expected risk, incorporating the cost-sensitive terms C_B and C_G:
   $$\mathcal{R} = C_B \cdot \mathbb{P}(\text{FN}) + C_G \cdot \mathbb{P}(\text{FP})$$
   Where $\mathbb{P}(\text{FN})$ and $\mathbb{P}(\text{FP})$ represent the probabilities of false negatives and false positives, respectively. This formulation aligns with Bayes-optimal prediction, as established in [43]. Based on this, the objective function is expressed as follows:
   $$\text{Min} \quad \sum_{\substack{i=1 \\ i\in G_B}}^{n_B} C_B \cdot \varepsilon_i + \sum_{\substack{i=1 \\ i\in G_G}}^{n_G} C_G \cdot \tilde{\varepsilon_i}$$
   This function represents the comprehensive costs associated with the misclassification of "bad" and "good" observations. Here, ε_i represents the error terms for observations in the bad class G_B, while $\tilde{\varepsilon_i}$ pertains to the good class G_G. The values C_B and C_G represent the costs linked to the misclassification of minority and majority classes, respectively.
2. **Constraints:**
   CS-MSD includes three essential constraints:
   (a) **The first constraints is:**
   $$\sum_{j=1}^{r} x_{ij} \cdot w_j - \varepsilon_i < b \quad \forall i \in G_B$$
   By adjusting for the error, this constraint guarantees that the weighted sum of features for the bad instances is less than a constant b. By linking the model's output to the predetermined threshold b, it enables a controlled assessment of misclassifications.
   (b) **The second constraint is:**
   $$\sum_{j=1}^{r} x_{ij} \cdot w_j + \tilde{\varepsilon_i} \geq b \quad \forall i \in G_G$$
   Similar to the first, this constraint applies to good instances. It guarantees that the weighted sum, now including the positive error $\tilde{\varepsilon_i}$, great or equal the same constant b. This equivalency across classes ensures that the model treats both classifications equitably in relation to the decision threshold.

(c) **The third constraint is:**

$$\sum_{j=1}^{r}\left(n_B\cdot\sum_{\substack{i=1\\ i\in G_G}}^{n_G}x_{ij}-n_G\cdot\sum_{\substack{i=1\\ i\in G_B}}^{n_B}x_{ij}\right)\cdot w_j=1$$

represents a constraint in an optimization framework where n_B and n_G are constants associated with different groups within the dataset, specifically the G_B and G_G classes, respectively. The term $\sum_{\substack{i=1\\ i\in G_G}}^{n_G} x_{ij}$ captures contributions from the good class, while $\sum_{\substack{i=1\\ i\in G_B}}^{n_B} x_{ij}$ represents those from the bad class.

The multiplication by w_j, a weighting factor, indicates that each summation component j has a unique relevance or influence on the overall equation. By imposing a specific relationship between the weighted contributions from both groups and requiring the total sum to equal 1, this constraint guides the optimization process to produce meaningful and non-trivial solutions. Consequently, it provides information about the connections being studied and ensures that the final model incorporates important datasets features.

The objective of the CS-MSD model is to reduce the errors associated with misclassifications in a binary classification context. Ensuring applicable solutions and addressing imbalanced datasets requires regulating minority and majority classes, thereby enhancing classification performance without compromising overall precision. Differing from SMOTE's resampling or boosting's [44] weight adjustments, we explicitly minimize misclassification cost through optimization constraints.

4. Experimental setup

To ensure that the characteristics of the data are clearly understood, we provide thorough explanations of the data 4.1 used in this section. We also present the classifiers 4.2 used in this study and describe the evaluation metrics 4.3 used to evaluate the models' performance.

4.1. Datasets

As indicated in Table 1, we examine the details of six imbalanced datasets that were downloaded from the UCI repository [45]. Each dataset shows a unique scenario with distinct levels of class imbalance. A key measure of distinction between the majority and minority classes within each dataset is the imbalance ratio (IR=$\frac{n_B}{n_G}$). This IR provides essential details about the level of class imbalance. The number of observations in the minority class divided by the number of observations in the majority class is the way it is defined.

Table 1. Summary of the datasets

Dataset	Abbreviation	Subject Area	Feature Type	Instances	Features	IR	Minority percentage
Wine	Wine	Physics and Chemistry	Integer, Real	178	13	0,496	33, 10%
Wisconsin Breast Cancer	WBC	Health and Medicine	Integer	699	09	0,526	34, 50%
Yeast	Yeast	Biology	Real	1484	08	0,453	31, 20%
Auction Verification	Auction	Computer Science	Integer	2043	07	0,147	12, 80%
Obesity	Obesity	Health and Medicine	Integer	2111	16	0,148	12, 90%
Predict students dropout	P.Student	Academic	Integer, Real	4424	36	0,473	32, 10%

The CS-MSD model is designed specifically for binary classification, so we applied targeted preprocessing steps to enhance its integration within our experimental framework. To achieve this, we used the one-class-versus-rest (OvR) technique, which allows the model to handle binary distinctions effectively. By using OvR, we separate one class from all others, which simplifies the binary classification process.

4.2. Classification models

A comparative analysis is used to evaluate our proposed CS-MSD model to other known ML models. Each model is executed with specific packages and optimized using appropriate hyper-parameters, as shown in Table 2.

To rigorously evaluate each model, a 5-fold cross-validation approach is applied, splitting the dataset into five equal parts. In each iteration, one segment serves as the test set while the others are used for training.

Table 2. Mapping of model name abbreviations and hyper-parameters

Model Name	**Abbreviation**	**Hyper-parameter**	**Python Package**
XGBoost [46]	XGB	max_depth (Max tree depth),scale_pos_weight, *	xgboost
Light GBM [47]	LGBM	max_depth,class_weight, *	lightgbm
Gradient Boosting [48]	GB	max_depth, n_estimators,class_weight, *	sklearn.ensemble
Random Forest [25, 49]	RF	max_depth,class_weight, *	sklearn.ensemble
Decision Tree [50]	DT	max_depth,class_weight, *	sklearn.tree
Bagging [51]	Bagg	max_depth, n_estimators,class_weight, *	sklearn.ensemble
Linear SVM [52]	SVML	kernel=linear,class_weight, *	sklearn.svm
Radial SVM [52]	SVMR	kernel=rbf,class_weight, *	sklearn.svm
Logistic Regression [9]	LR	class_weight, *	sklearn.linear_model

*:Other default parameters.

The objective is to assess and compare the performance and applicability of these models to the CS-MSD model in two scenarios 5.1. The CS-MSD model, given in Section 3, optimizes weights and intercepts to effectively separate classes. It is solved using the Gurobi Optimizer, version 10. The values are then used in a function that determines classification binary probabilities.

This function initially calculates logits as a linear mixture of input features and their weights, plus a bias term. The logits are transformed to probabilities using the sigmoid function, resulting a 2D array of probabilities for classes 0 and 1. To assign class labels, a threshold based on the mean of the positive class probabilities is employed.

4.3. Performance evaluation metrics

In ML, algorithm performance evaluation usually depends on a confusion matrix. This matrix arranges the results of predictions by showing the actual class values in its rows and the predicted class values in its columns. In a binary classification context, as illustrated in Table 3 below, the confusion matrix defines four key categories: True Negatives (TN), False Positives (FP), False Negatives (FN) and True Positives (TP).

In classification, TN represents cases where negative examples are correctly classified, while FP indicates negative examples that are incorrectly classified as positive. Conversely, FN signifies positive examples that are incorrectly identified as negative, whereas TP represents positive examples that are correctly classified.

Table 3. Confusion matrix

	Predicted Negative	**Predicted Positive**
Actual Negative	TN	FP
Actual Positive	FN	TP

A classification model's performance is assessed using a variety of indicators. Specificity (Eq. 6), commonly known as the true negative rate, assesses the fraction of accurately detected negative cases among all negative instances.In contrast, recall (Eq. 7), also known as sensitivity or the true positive rate (TPR), assesses the model's capacity to recognize all positive cases. The F1_score (Eq. 9) provides a consistent evaluation of a classifier's performance by taking the harmonic mean of precision (Eq. 8) and recall. The F1-score balances precision with recall, making it a widely used for classification tasks, particularly in circumstances with class imbalance.

The Geometric mean, referred to as G-mean (Eq. 10), evaluates both sensitivity and specificity simultaneously, making it ideal for imbalanced classification situations. This metric estimates a classifier's effectiveness based on its ability to successfully classify both positive and negative observations while minimizing any effects of class imbalance. However, it is essential to recognize that accuracy (Eq. 11) may not be an adequate metric for evaluating models trained on imbalanced data. It may return high values even if all samples are classified as the majority class, as indicated in [53].

Another widely used measure in classification modeling is the area under the curve (AUC) , which serves as a key assessment tool in this field (Eq. 12). A higher AUC score on a range of 0 to 1, indicates higher model performance, Where 0.5 denotes a random event and values nearing 1 denote excellent performance.

$$\text{Specificity}(S_{pec}) = \frac{TN}{TN + FP} \tag{6}$$

$$\text{Recall}(S_{ens}) = \frac{TP}{TP + FN} \tag{7}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{8}$$

$$\text{F1\_score} = 2 \times \frac{Precision \times S_{ens}}{Precision + S_{ens}} \tag{9}$$

$$\text{G-mean} = \sqrt{S_{pec} \times S_{ens}} \tag{10}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR}) \tag{12}$$

where:

$$\text{FPR} = 1 - \text{Specificity} = \frac{\text{FP}}{\text{FP+TN}} \text{ (False Positive Rate)}$$

Both wall time and CPU time must be simultaneously evaluated in order to evaluate a computing model's performance. When assessing computing model performance, wall time: measures the actual elapsed time for a task, including all delays (Input/Output, sleep(), other processes), is a crucial measure to take into account. On the other hand, CPU time measures actual processor usage, divided into user-level execution and kernel-level operations.

The hardware configuration utilized for this study comprises an Intel® Core™ i5-4330M CPU, operating at 2.80 GHz with 2 physical cores,4 logical processors and 8.00 GB of RAM, enabling effective multitasking through hyper-threading technology.

5. Design process and experimental results

5.1. Design process

As shown in Fig. 1, all six datasets follow the same preprocessing steps in our pipeline. The pipeline automatically separates features from targets, processing numeric features through mean imputation and standardization (StandardScaler) while categorical features preprocessing use the mode imputation and one-hot encoding, integrated via ColumnTransformer with preserved feature names. Highly correlated features (absolute correlation > 0.98) are removed to reduce preprocessing use the multicollinearity before reconstructing each dataset by merging processed features with their respective targets.

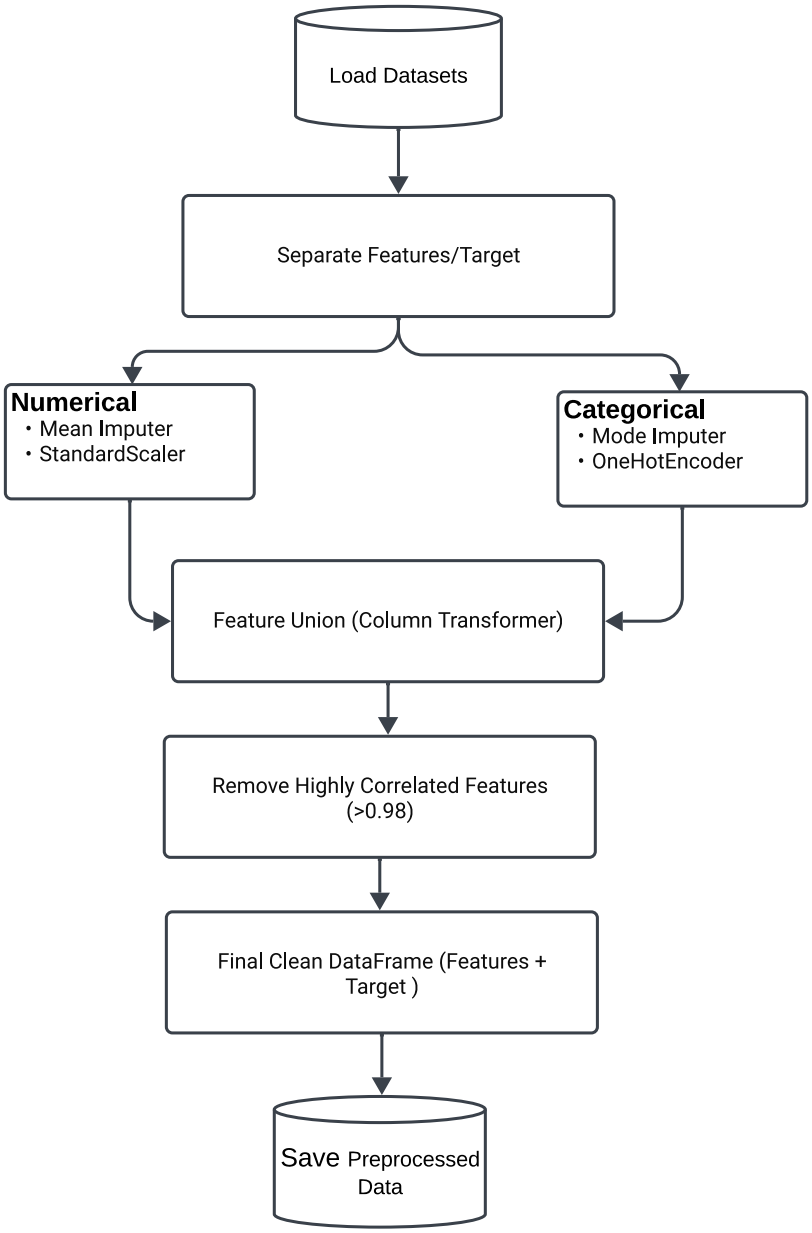

Figure 1. Unified data preparation workflow.

Two subsets of the preprocessed datasets are now created: a training and a testing set. Twenty percent of the data is kept for testing, with the remaining 80% used for training. The 80%-20% splitting method is a prominent methodology in ML research, striking a compromise between reliable evaluation and powerful training. Despite its widespread use and being a heuristic guideline this ratio is not always optimum. The "Pareto principle" [54] generally supports the 80:20 division.

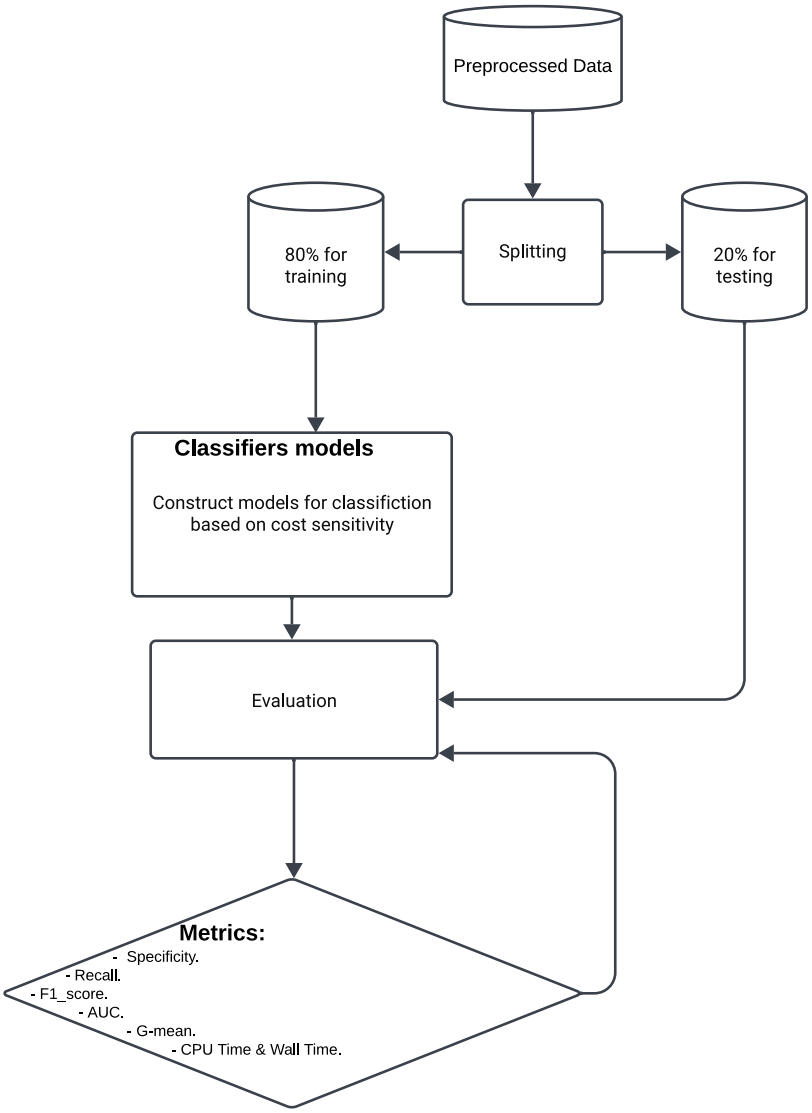

Figure 2. A flowchart of a cost-sensitive classification model for imbalanced data.

Table 4. Optimal hyper-parameters for models.

Model	**Hyper-parameters Values**
XGB	max_depth=6, scale_pos_weight=$\frac{n_G}{n_B}$, learning_rate=0.1
LGBM	max_depth=7, class_weight='balanced', learning_rate=0.05
GB	n_estimators=100, learning_rate=0.1, init=DT
RF	max_depth=10, class_weight='balanced', n_estimators=200
DT	max_depth=5, class_weight='balanced'
Bagg	estimator=DT, n_estimators=50, bootstrap_features=True
SVML	probability=True,, Kernel=linear, class_weight='balanced'
SVMR	probability=True, Kernel=rbf, class_weight='balanced'
LR	class_weight='balanced', max_iter = 1000, solver='liblinear'

We applied the ML models described in Table 2 to both the training and testing subsets, as illustrated in Fig. 2. This method enables a consistent comparison of models, allowing us to assess their strengths and flaws using performance measurements. Each model executed a training subset to learn the basic patterns before being evaluated on a testing subset to determine its ability to it extended to new data.

After evaluating multiple configurations through grid search cross-validation, we identified the optimal hyper-parameters for each model (Table 4). These values balance performance and computational efficiency while addressing class imbalance through parameters like class_weight and scale_pos_weight.

Following splitting the data for our model setup (Fig. 3), we also divided the training set into two groups: a G_G group with a goal value of "1" and a G_B group with a desired value of "0." The CS-MSD model's training and evaluation came next.

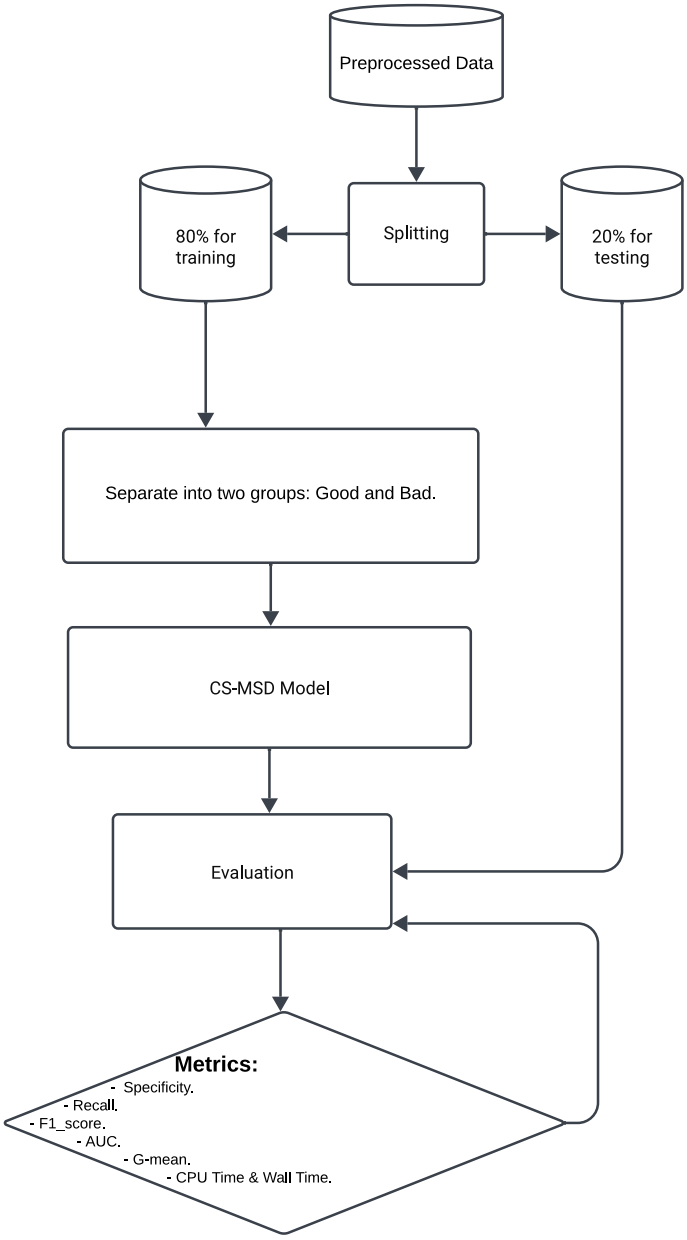

Figure 3. The flowchart representing our approach CS-MSD.

The Gurobi solver configuration presented in Table 5 represents a carefully tuned setup for solving the CS-MSD model. This configuration balances computational efficiency with solution quality while addressing the specific challenges of the optimization problem.

Table 5. Gurobi solver parameters for CS-MSD

Parameter	Value	Type	Description
OutputFlag	False	Boolean	Suppresses solver output messages (quiet mode).
NonConvex	2	Integer	Enables non-convex quadratic constraint handling (required for bilinear terms).
TimeLimit	300 (s)	Integer	Maximum runtime before stopping.
MIPGap	0.01 (1%)	Double	Relative optimality gap tolerance for MIP models.
Threads	4	Integer	Number of CPU threads for parallel computation.
NumericFocus	1	Integer	Reduces numerical instability risks (1=moderate focus).

5.2. Results and discussion

In this research, we compared the performance of our model, CS-MSD, with several well-known models across six different datasets. We employed key metrics, such as specificity, recall, F1_score, AUC, G-mean and computing time, to evaluate the model's effectiveness in accurately detecting both positive and negative cases in each dataset. The Table 6 compares the performance of various classifiers models, including XGB, Bagg, GB, LGBM, LR, SVM, RF, DT and CS-MSD, across six datasets (Wine, BCW, Yeast, Auction, Obesity, P. Student). Metrics such as specificity, recall, F1-score, AUC, G-mean, CPU time and wall time are used to evaluate the models' effectiveness in handling classification tasks.

Table 6. Results comparison.

	Metrics	Datasets					
		Wine	BCW	Yeast	Auction	Obesity	P.Student
XGB	**specificity**	0,766	0,924	0,609	0,921	**0,923**	0,711
	Recall	0,941	**0,929**	0,747	0,914	0,924	0,852
	F1_score	0,948	**0,979**	**0,821**	0,977	0,982	0,898
	AUC	0,962	0,973	0,817	0,979	0,977	0,892
	G-mean	0,883	**0,944**	**0,706**	0,947	**0,952**	0,809
	CPU time	0.359 s	0.156 s	0.266 s	0.359 s	0.391 s	0.875 s
	Wall time	0.371 s	0.182 s	0.276 s	0.361 s	0.419 s	0.882 s
Bagg	**specificity**	0,766	0,881	0,587	0,903	0,847	0,665
	Recall	0,942	**0,929**	0,694	0,934	**0,937**	0,882
	F1_score	0,948	0,973	0,784	0,986	**0,983**	0,903
	AUC	0,973	0,974	0,796	**0,982**	**0,979**	0,905
	G-mean	0,884	0,934	0,669	0,949	0,922	0,797
	CPU time	1,61 s	2,73 s	2,45 s	1,91 s	2 s	2,38 s
	Wall time	1,75 s	2,83 s	2,74 s	1,95 s	2,55 s	10 s

Continued on next page

Table 6 – continued from previous page

	Metrics	Datasets					
		Wine	BCW	Yeast	Auction	Obesity	P.Student
GB	**specificity**	0,766	0,882	0,587	0,903	0,847	0,665
	Recall	0,942	**0,929**	0,694	0,934	**0,937**	0,883
	F1_score	0,948	0,973	0,7884	0,986	**0,983**	0,903
	AUC	0,973	0,974	0,796	**0,982**	**0,979**	0,905
	G-mean	0,882	0,934	0,669	0,949	0,922	0,797
	CPU time	0.375 s	0.578 s	0.859 s	0.484 s	0.359 s	2,27 s
	Wall time	2,65 s	8,66 s	21,6 s	8,84 s	10,9 s	51,6 s
LGBM	**specificity**	0,766	0,862	0,642	**0,923**	0,865	0,711
	Recall	0,942	**0,929**	0,694	0,931	0,926	0,812
	F1_score	0,948	0,968	0,794	0,986	0,979	0,876
	AUC	0,967	0,974	0,808	0,981	0,977	0,885
	G-mean	0,882	0,924	0,698	**0,956**	0,926	0,791
	CPU time	0.062 s	**0.046** s	**0.062** s	**0.109** s	0.141 s	0.125 s
	Wall time	0.071 s	**0.057** s	**0.066** s	**0.206** s	0.224 s	0.198 s
LR	**specificity**	0,766	0,883	0,036	0,627	0,921	0,753
	Recall	0,897	0,918	**0,926**	0,592	0,894	0,819
	F1_score	0,926	0,968	0,811	0,755	0,966	**0,887**
	AUC	0,957	0,977	0,671	0,732	0,967	**0,909**
	G-mean	0,862	0,929	0,569	0,638	0,937	0,816
	CPU time	0.172 s	0.188 s	0.172 s	0.219 s	0.219 s	0.453 s
	Wall time	0.280 s	0.266 s	0.198 s	0.267 s	0.316 s	1,59 s
SVML	**specificity**	**0,923**	0,883	0,742	0,627	0,902	0,785
	Recall	0,942	0,918	0,525	0,641	0,891	0,796
	F1_score	**0,992**	0,968	0,693	0,791	0,964	0,882
	AUC	**0,982**	**0,978**	0,727	0,786	0,968	0,905
	G-mean	0,963	0,929	0,653	0,664	0,926	0,824
	CPU time	0.125 s	0.062 s	0.219 s	1,39 s	0.188 s	76 s
	Wall time	0.198 s	0.068 s	0.232 s	1,42 s	0.194 s	79 s
SVMR	**specificity**	0,661	0,883	0,753	0,592	0,921	0,671
	Recall	0,811	0,918	0,568	0,581	0,853	**0,899**
	F1_score	0,879	0,968	0,729	0,746	0,943	0,788
	AUC	0,943	0,975	0,774	0,726	0,966	0,556
	G-mean	0,818	0,929	0,683	0,617	0,914	0,786
	CPU time	**0.031** s	0.078 s	0.422 s	1,03 s	0.266 s	7,19 s
	Wall time	**0.037** s	0.089 s	0.435 s	1,14 s	0.284 s	7,23 s
RF	**specificity**	0,766	0,911	0,587	0,679	0,884	0,668
	Recall	0,941	**0,929**	0,742	**0,941**	0,929	0,874
	F1_score	0,948	**0,979**	0,814	0,971	0,982	**0,911**
	AUC	0,981	0,972	**0,819**	0,978	**0,979**	0,905
	G-mean	**0,991**	**0,944**	0,691	0,831	0,936	0,796
	CPU time	3,88 s	8,08 s	5,14 s	5,14 s	5,53 s	10,9 s
	Wall time	4,78 s	8,16 s	5,92 s	5,22 s	5,64 s	12,9 s

Continued on next page

Table 6 – continued from previous page

	Metrics	Datasets					
		Wine	BCW	Yeast	Auction	Obesity	P.Student
DT	**specificity**	0,689	0,861	**0,764**	0,886	0,847	0,707
	Recall	0,941	0,918	0,534	**0,941**	0,921	0,811
	F1_score	0,929	0,962	0,705	**0,987**	0,978	0,874
	AUC	0,865	0,936	0,746	0,963	0,937	0,856
	G-mean	0,837	0,919	0,668	0,943	0,916	0,788
	CPU time	0.234 s	0.312 s	0.203 s	0.156 s	0.219 s	0.438 s
	Wall time	0.311 s	0.397 s	0.324 s	0.305 s	0.373 s	4,69 s
CS-MSD	**specificity**	0,751	**0,959**	0,613	0,566	**1**	**0,811**
	Recall	**0,958**	0,913	0,507	0,731	0,856	0,861
	F1_score	0,921	0,944	0,603	0,814	0,922	0,882
	AUC	0,854	0,936	0,751	0,849	0,928	0,835
	G-mean	0,848	0,936	0,658	0,843	0,925	**0,835**
	CPU time	0.052 s	0.048 s	0.064 s	0.135 s	**0.121** s	**0.123** s
	Wall time	0.054 s	0.059 s	0.069 s	0.272 s	**0.209** s	**0.174** s

Models like XGB and GB consistently deliver high recall and AUC scores across datasets, indicating their strong capability in correctly identifying positive instances, particularly in imbalanced datasets like BCW and Obesity. However, these models tend to have longer CPU and wall times, making them less computationally efficient compared to faster models like LR and LGBM. SVM and RF also show good overall classification performance, particularly in terms of specificity and F1-score, but suffer from relatively longer computational times for more complex datasets.

When it comes to reaching the right equilibrium between the efficiency and performance over various datasets, the CS-MSD model presents significant strengths. With best results in many datasets and the excellent classification of negative class, it stands out especially in terms of specificity. Also, it shows how well it can correctly recognize positive observations by achieving the highest recall performance in wine dataset. This approach is effective for real-life scenarios since it provides competitive speed, generally ranking against the fastest in both CPU and wall time.

Other models like XGB, RF and SVML generally rule in metrics like F1score and AUC, showing better classification performance under many dataset, even though CS-MSD outperforms on certain data. A few models, for example, typically provide high area-under-curve outcomes or high harmonic mean precision-recall balances.

A significant advantage of LGBM is its speed; it can handle some datasets far faster than others, which may be useful in situations where time is of the pure factor. For the P.Student dataset, the analysis reveals that LGBM offers competitive scores with a specificity of 0.711 and a recall of 0.812, while LR stands out with a good specificity and a solid AUC of 0.909. SVML presents a balanced performance across metrics. Conversely, SVMR shows the lowest specificity but maintains a high recall, indicating a tendency to capture positive instances at the cost of increased false positives. Finally, RF ranks highly with a specificity of 0.668, a strong recall and the highest F1_score (0.911), while DT demonstrates reasonable predictive capabilities. CS-MSD performs well, outperforming other models in specificity and G-mean in particular while maintaining notably faster execution times.

CS-MSD can be a good choice because of its combination of computing speed and efficiency in essential real-time fields, even though it is not definitely the most effective option in each field. When the objective is to optimize for speed and particular scenarios or to maximize particular metrics of performance will ultimately figure out which approach is appropriate.

Fig. 4 illustrates that our model, CS-MSD, has a balanced yet distinctive shape on the radar graph, with wide coverage across each of the five metrics. Despite not perfectly symmetrical like some top models (e.g., RF or XGB), its large shape shows constant, solid performance with no major weaknesses. The model succeeds very well

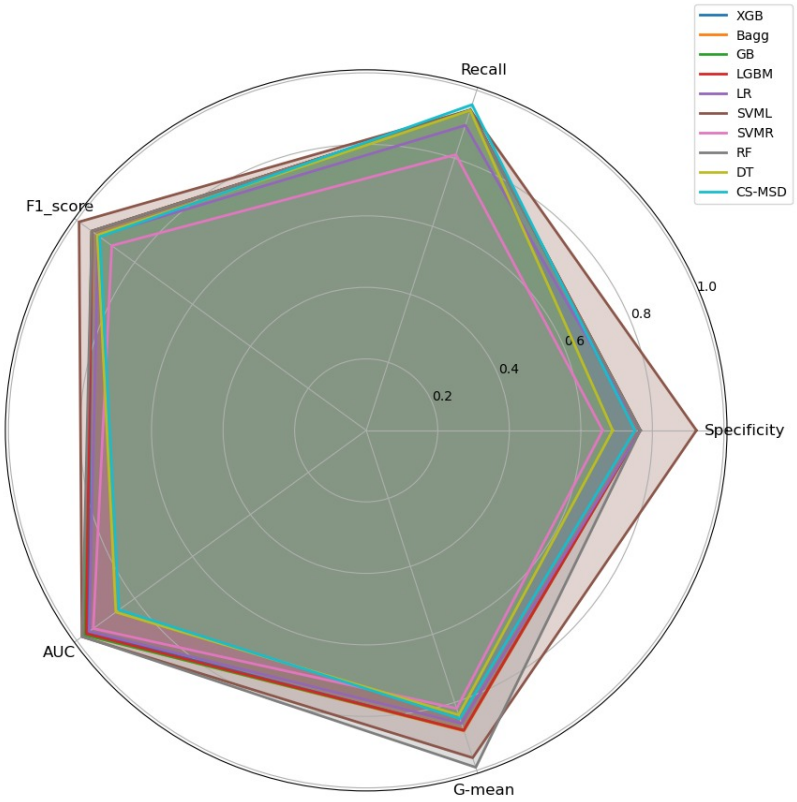

Figure 4. Model performance comparison on Wine dataset.

in Recall, that's reference an important improvement in its competences, while maintaining favorable results in other metrics like F1-score and G-mean. This balanced but not perfect symmetry indicates that CS-MSD aims for regular overall success.

The size and shape of CS-MSD's radar plot are especially useful for recognize because they clearly demonstrate the strategic trade-offs. The vast region contained by its geometric shapes demonstrates its overall capacity, but the little declines in Specificity (0.751) and AUC (0.854) demonstrate where it compromises Recall. CS-MSD's middle-level asymmetry makes it a realistic technique for high-sensitivity applications that require true positives while maintaining stability, unlike other models such as SVMR and RF.

6. Conclusions

This study highlights the vital role of binary classification in dealing with challenges related to imbalanced datasets. Asymmetry in the distribution of classes often results in skewed models that prioritize the majority class over the minority class. To deal with this issue, various approaches, especially cost-sensitive learning techniques, have been explored in the context of MP. This study is focused on cost-sensitive learning methods that employ MP fundamentals to build the CS-MSD model. By integrating cost sensitivity and MP, CS-MSD successfully tackles the complex issues presented by imbalanced datasets across various areas. The empirical examination shows that CS-MSD outperforms other approaches in Recall and F1-score metrics. CS-MSD always ranks excellently in comparison evaluations, showing its reliability. Future investigations will focus on advancing the CS-MSD model across three key areas: scalability, interpretability, and real-world robustness. The framework will be expanded to handle large-scale, high-dimensional datasets, optimizing computational efficiency while rigorously benchmarking

performance against modern scalability solutions. Interpretability will be enhanced through domain-specific feature weight analysis, particularly in critical applications such as medical diagnostics and fraud detection, to ensure actionable insights for practitioners. Additionally, the model's robustness will be rigorously tested on real-world banking and healthcare datasets, with a focus on addressing class imbalance through hybrid techniques like SMOTE-enhanced cost-sensitive learning.

Acknowledgment

The author is quite appreciative of Gurobi Optimization, LLC for their assistance in granting an academic license, which made it possible to use their potent optimization software for this study. For providing the varied datasets necessary for the research and conclusions of this study, the author additionally expresses sincere gratitude to the UCI Machine Learning Repository. These contributions have significantly improved the study's overall rigor and quality while also furthering the research aims. The author would also like to extend heartfelt thanks to the reviewers for their insightful feedback and constructive suggestions, which greatly enhanced the quality and clarity of this work.

REFERENCES

1. Tomescu, V.-I., Czibula, G., & Niţică, Ş. (2021). A study on using deep autoencoders for imbalanced binary classification. Procedia Computer Science, 192, 119–128. https://doi.org/10.1016/j.procs.2021.08.013.
2. Dalgıç, Ö. O., Wu, H., Safa Erenay, F., Sir, M. Y., Özaltın, O. Y., Crum, B. A., & Pasupathy, K. S. (2021). Mapping of critical events in disease progression through binary classification: Application to amyotrophic lateral sclerosis. Journal of Biomedical Informatics, 123, 103895. https://doi.org/10.1016/j.jbi.2021.103895.
3. Kamal, I. M., & Bae, H. (2022). Semi-supervised binary classification with latent distance learning (arXiv:2211.15153). arXiv. http://arxiv.org/abs/2211.15153.
4. Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). Learning from Imbalanced Data Sets. Springer International Publishing. https://doi.org/10.1007/978-3-319-98074-4.
5. Barella, V. H., Garcia, L. P. F., De Souto, M. C. P., Lorena, A. C., & De Carvalho, A. C. P. L. F. (2021). Assessing the data complexity of imbalanced datasets. Information Sciences, 553, 83–109. https://doi.org/10.1016/j.ins.2020.12.006.
6. Hakimi, R., Benyacoub, B., & Ouzineb, M. (2025). Unveiling the Power of Cost Sensitivity: A Comparative Study of Different Models on Imbalanced Datasets in Machine Learning. Communication and Information Technologies through the Lens of Innovation,113–120. ICATH 2023. Advances in Science, Technology & Innovation. Springer, Cham. https://doi.org/10.1007/978-3-031-74470-9_14.
7. Bach, M. (2022). New Undersampling Method Based on the kNN Approach. Procedia Computer Science, 207, 3403–3412. https://doi.org/10.1016/j.procs.2022.09.399.
8. Bach, M., Werner, A., & Palt, M. (2019). The Proposal of Undersampling Method for Learning from Imbalanced Datasets. Procedia Computer Science, 159, 125–134. https://doi.org/10.1016/j.procs.2019.09.167.
9. Hanifah, F. S., Wijayanto, H., & Kurnia, A. (2015). SMOTE bagging algorithm for imbalanced dataset in logistic regression analysis (case: Credit of bank X). Applied Mathematical Sciences, 9, 6857–6865. https://doi.org/10.12988/ams.2015.58562.
10. Torres, F. R., Carrasco-Ochoa, J. A., & Martínez-Trinidad, J. Fco. (2016). SMOTE-D a Deterministic Version of SMOTE. In Pattern Recognition (Vol. 9703, pp. 177–188). Springer International Publishing. https://doi.org/10.1007/978-3-319-39393-3_18.
11. Bach, M., & Werner, A. (2021). Improvement of Random Undersampling to Avoid Excessive Removal of Points from a Given Area of the Majority Class. In Computational Science – ICCS 2021 (Vol. 12744, pp. 172–186). Springer International Publishing. https://doi.org/10.1007/978-3-030-77967-2_15.
12. Shaikh, S., Changan, L., & Malik, M. R. (2021). An Empirical And Comparatively Research On Under-Sampling & Over- Sampling Defect-Prone Data-Sets Model In Light Of Machine Learning. International Journal of Advanced Networking and Applications, 12(05), 4719–4724. https://doi.org/10.35444/IJANA.2021.12508.
13. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953.
14. Sharma, S., Bellinger, C., Krawczyk, B., Zaiane, O., & Japkowicz, N. (2018). Synthetic Oversampling with the Majority Class: A New Perspective on Handling Extreme Imbalance. 2018 IEEE International Conference on Data Mining (ICDM), 447–456. https://doi.org/10.1109/ICDM.2018.00060.
15. Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). Cost-Sensitive Learning. In Learning from Imbalanced Data Sets (pp. 63–78). Springer International Publishing. https://doi.org/10.1007/978-3-319-98074-4_4.
16. Thai-Nghe, N., Gantner, Z., & Schmidt-Thieme, L. (2010). Cost-sensitive learning methods for imbalanced data. The 2010 International Joint Conference on Neural Networks (IJCNN), 1–8. https://doi.org/10.1109/IJCNN.2010.5596486.
17. Yotsawat, W., Phodong, K., Promrat, T., & Wattuya, P. (2023). Bankruptcy prediction model using cost-sensitive extreme gradient boosting in the context of imbalanced datasets. International Journal of Electrical and Computer Engineering (IJECE), 13(4), 4683. https://doi.org/10.11591/ijece.v13i4.pp4683-4691.

18. Freed, N., & Glover, F. (1986). Evaluating alternative linear programming models to solve the two-group discriminant problem. Decision Sciences, 17(2), 151–162. https://doi.org/10.1111/j.1540-5915.1986.tb00218.x.
19. Ziari, H. A., Leatham, D. J., & Ellinger, P. N. (1997). Development of Statistical Discriminant Mathematical Programming Model Via Resampling Estimation Techniques. American Journal of Agricultural Economics, 79(4), 1352–1362. https://doi.org/10.2307/1244291.
20. Kou, G., Peng, Y., Shi, Y., Wise, M., & Xu, W. (2005). Discovering Credit Cardholders' Behavior by Multiple Criteria Linear Programming. Annals of Operations Research, 135(1), 261–274. https://doi.org/10.1007/s10479-005-6245-5.
21. Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, 6(1), 20–29. https://doi.org/10.1145/1007730.1007735.
22. Rodríguez, N., López, D., Fernández, A., García, S., & Herrera, F. (2021). SOUL: Scala Oversampling and Undersampling Library for imbalance classification. SoftwareX, 15, 100767. https://doi.org/10.1016/j.softx.2021.100767.
23. Watthaisong, T., Sunat, K., & Muangkote, N. (2024). Comparative Evaluation of Imbalanced Data Management Techniques for Solving Classification Problems on Imbalanced Datasets. Statistics, Optimization & Information Computing, 12(2), 547–570. https://doi.org/10.19139/soic-2310-5070-1890.
24. Kim, K. (2021). Noise Avoidance SMOTE in Ensemble Learning for Imbalanced Data. IEEE Access, 9, 143250–143265. https://doi.org/10.1109/ACCESS.2021.3120738.
25. Ait Brahim, H., Banouar, O., El-Hadaj, S., & Metrane, A. (2024). A hybrid sampling combining Smote and RF algorithm for cancer chemotherapy protocols Classification. Statistics, Optimization & Information Computing, 12(3), 617–629. https://doi.org/10.19139/soic-2310-5070-1941.
26. Driouich, A., El Ouissari, A., El Moutaouakil, K., & Akharraz, I. (2024). A Metaheuristic for Fuzzy Density Based SVM and Confidence SMOTE for Early Prediction of Diabetes. Statistics, Optimization & Information Computing, 13(4), 1595–1609. https://doi.org/10.19139/soic-2310-5070-1348.
27. Chowdhury, M. M., Ayon, R. S., & Hossain, M. S. (2024). An investigation of machine learning algorithms and data augmentation techniques for diabetes diagnosis using class imbalanced BRFSS dataset. Healthcare Analytics, 5, 100297. https://doi.org/10.1016/j.health.2023.100297.
28. Bach, M., & Werner, A. (2018). Cost-Sensitive Feature Selection for Class Imbalance Problem. In Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology – ISAT 2017 (Vol. 655, pp. 182–194). Springer International Publishing. https://doi.org/10.1007/978-3-319-67220-5_17.
29. Turney, P. D. (1995). Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. Journal of Artificial Intelligence Research, 2, 369–409. https://doi.org/10.1613/jair.120.
30. Benítez-Peña, S., Blanquero, R., Carrizosa, E., & Ramírez-Cobo, P. (2019). Cost-sensitive Feature Selection for Support Vector Machines. Computers & Operations Research, 106, 169–178. https://doi.org/10.1016/j.cor.2018.03.005.
31. Turney, P. D. (2002). Types of Cost in Inductive Concept Learning. CoRR, cs.LG/0212034. http://arxiv.org/abs/cs/0212034.
32. Olowookere, T. A., & Adewale, O. S. (2020). A framework for detecting credit card fraud with cost-sensitive meta-learning ensemble approach. Scientific African, 8, e00464. https://doi.org/10.1016/j.sciaf.2020.e00464.
33. Zhang, J., Wang, X., Zhao, C., Bai, W., Shen, J., Li, Y., Pan, Z., & Duan, Y. (2020). Application of cost-sensitive LSTM in water level prediction for nuclear reactor pressurizer. Nuclear Engineering and Technology, 52(7), 1429–1435. https://doi.org/10.1016/j.net.2019.12.025.
34. Ma, X.-A., & Zhao, X. R. (2019). Cost-sensitive three-way class-specific attribute reduction. International Journal of Approximate Reasoning, 105, 153–174. https://doi.org/10.1016/j.ijar.2018.11.014.
35. Wang, W., & An, A. (2024). Identification of market power abuse in Chinese electricity market based on an improved cost-sensitive support vector machine. International Journal of Electrical Power & Energy Systems, 158, 109907. https://doi.org/10.1016/j.ijepes.2024.109907.
36. Mienye, I. D., & Sun, Y. (2021). Performance analysis of cost-sensitive learning methods with application to imbalanced medical data. Informatics in Medicine Unlocked, 25, 100690. https://doi.org/10.1016/j.imu.2021.100690.
37. Bhargava, Y., Shetty, S. K., & Baths, V. (2024). Subjective Cognitive Decline Prediction on Imbalanced Data Using Data-Resampling and Cost-Sensitive Training Methods. Procedia Computer Science, 235, 1964–1979. https://doi.org/10.1016/j.procs.2024.04.186.
38. Freed, N., & Glover, F. (1981). Simple but powerful goal programming models for discriminant problems. European Journal of Operational Research, 7(1), 44–60. https://doi.org/10.1016/0377-2217(81)90048-5.
39. Glen, J. J. (1999). Integer programming methods for normalisation and variable selection in mathematical programming discriminant analysis models. Journal of the Operational Research Society, 50(10), 1043–1053. https://doi.org/10.1057/palgrave.jors.2600804.
40. Glen, J. J. (2003). An iterative mixed integer programming method for classification accuracy maximizing discriminant analysis. Computers & Operations Research, 30(2), 181–198. https://doi.org/10.1016/S0305-0548(01)00088-0.
41. Glover, F. (1990). Improved Linear Programming Models for Discriminant Analysis. Decision Sciences, 21(4), 771–785. https://doi.org/10.1111/j.1540-5915.1990.tb01249.x.
42. Brownlee, J. (2020). Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning. Machine Learning Mastery. https://books.google.co.ma/books?id=jaXJDwAAQBAJ.
43. Elkan, C. (2001). The Foundations of Cost-Sensitive Learning. Proceedings of the 17th International Joint Conference on Artificial Intelligence, 973–978.
44. Sun, Y., Kamel, M. S., Wong, A. K. C., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition, 40(12), 3358–3378. https://doi.org/10.1016/j.patcog.2007.04.009.
45. Markelle Kelly, Rachel Longjohn, & Kolby Nottingham. The UCI Machine Learning Repository [Dataset]. Retrieved 2 October 2024, from The UCI Machine Learning Repository.
46. Qiu, Y., Zhou, J., Khandelwal, M., Yang, H., Yang, P., & Li, C. (2022). Performance evaluation of hybrid WOA-XGBoost, GWO-XGBoost and BO-XGBoost models to predict blast-induced ground vibration. Engineering with Computers, 38(S5), 4145–4162. https://doi.org/10.1007/s00366-021-01393-9.

47. Zhou, C., Zou, L., Liu, C., & Song, Z. (2023). FL-Lightgbm prediction method of unbalanced small sample anti-breast cancer drugs. In Y. Zhong (Ed.), Fifth International Conference on Computer Information Science and Artificial Intelligence (CISAI 2022) (p. 44). SPIE. https://doi.org/10.1117/12.2667385.
48. Bai, M., Zheng, Y., & Shen, Y. (2022). Gradient Boosting Survival Tree with Applications in Credit Scoring. Journal of the Operational Research Society, 73(1), 39–55. https://doi.org/10.1080/01605682.2021.1919035.
49. Speiser, J. L., Miller, M. E., Tooze, J., & Ip, E. (2019). A Comparison of Random Forest Variable Selection Methods for Classification Prediction Modeling. Expert Systems with Applications, 134, 93–101. https://doi.org/10.1016/j.eswa.2019.05.028.
50. Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81–106. https://doi.org/10.1007/BF00116251.
51. Abdoli, M., Akbari, M., & Shahrabi, J. (2021). Bagging Supervised Autoencoder Classifier for Credit Scoring (arXiv:2108.07800). arXiv. http://arxiv.org/abs/2108.07800.
52. Mammone, A., Turchi, M., & Cristianini, N. (2009). Support vector machines. WIREs Computational Statistics, 1(3), 283–289. https://doi.org/10.1002/wics.49.
53. Niu, K., Zhang, Z., Liu, Y., & Li, R. (2020). Resampling ensemble model based on data distribution for imbalanced credit risk evaluation in P2P lending. Information Sciences, 536, 120–134. https://doi.org/10.1016/j.ins.2020.05.040.
54. Joseph, V. R. (2022). Optimal ratio for data splitting. Statistical Analysis and Data Mining: The ASA Data Science Journal, 15(4), 531–538. https://doi.org/10.1002/sam.11583.