Les Cahiers du GERAD ISSN: 0711–2440

An efficient hybrid algorithm to solve credit scoring problem

M. Barhdadi, B. Benyacoub, M. Ouzineb, I. El Hallaoui

G–2023–33

August 2023

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

An efficient hybrid algorithm to solve credit scoring problem

Mohamed Barhdadi [a]

Badreddine Benyacoub [a]

Mohamed Ouzineb [a]

Issmail El Hallaoui [b]

[a] *Institut National de Statistique et d'Economie Appliquée, B.P.:6217 Rabat-Instituts, Madinat Al Irfane, Rabat, Morocco*

[b] *Département de mathématiques et de génie industriel, Polytechnique Montreal & GERAD, Montréal (Qc), Canada, H3T 1J4*

mh.barhdadi@gmail.com
benyacoubb@gmail.com
m.ouzineb@insea.ac.ma
issmail.el-hallaoui@polymtl.ca

August 2023
Les Cahiers du GERAD
G–2023–33

Abstract : This paper develops an efficient hybrid algorithm to solve the credit scoring problem. We use statistical mathematical programming to develop new classification models for the discriminant analysis problems. The novelty of our approach can be summarized by the following: the combination of two objective functions, the first minimizing the sum of misclassified points' distances using linear programming and the second minimizing the number of misclassified points using an efficient variable neighborhood search heuristic based on jackknife resampling technique to improve the classification performance. Our proposed scoring systems are often just as accurate as the most powerful black-box machine learning models, but transparent and highly interpretable. The obtained results prove the effectiveness of the proposed approach on real benchmark datasets.

Keywords : Classification, discriminant analysis, credit scoring, linear programming, variable neighborhood search, jackknife resampling technique

1 Introduction

Credit scoring models have played a vital role in credit granting by lenders and financial institutions. Nowadays, our society is becoming a perfect consumer society, pushing individuals to seek alternative financing solutions to cover their various expenses. For instance, loans, which present a possibly ideal solution, still pose a risk to the lender because the applicant may fail to meet payment deadlines. To mitigate this risk, credit scoring, which represents the operation of collecting and synthesizing data through a combination of statistical, operations research and Machine Learning (ML) techniques to distinguish between "good" and "bad" loans, is deployed. This process is made possible through the rapid storage and treatment of borrowers' information.

Credit scoring has been developed to explore the relationship between the dependent variable that describes the risk of a consumer defaulting on a loan and the independent variables that characterize the consumer [34] (e.g. age, number of previous loans, income, housing, etc.). In the literature, there exists many scoring systems that use different classifications techniques. Linear discriminant analysis was first introduced by Fisher in [8] and is a popular statistical technique in credit scoring [5]. Logistic regression (LR) is among the most commonly used statistical techniques in credit scoring [20] and is still widely used in practice [17]. K-nearest neighbor is also one of the prospective non-parametric statistical methods applied to credit scoring. Henley et al. proposed in [14] an improved k-nearest neighbor *knn* classifier and compared it with other statistical models. Recently, *knn* has been used in credit scoring to build reliable classifiers [3]. Since its introduction by Breiman et al. [4], decision tree (DT) has become a reference non-parametric method for the credit scoring problem. This interesting method is still applied to build models for credit scoring [3, 18].

In the last decades, many effective credit scorecards have been developed via means of machine learning techniques [6, 20]. Most of such developments include artificial neural networks [5, 16, 27], support vector machines and least-squares support vector machines (LS-SVMs) [15, 20], and hybrid models [38, 39]. ML techniques are not explainable and are considered as black boxes [24, 33]. Due to banking regulation authorities, financial institutions in some countries are obliged to provide clear explanations when a loan application is rejected [35]. Most ML algorithms, including deep learning and ensemble models, are so complex that they cannot be directly interpreted by humans and suffer from this limitation. This is exactly the opposite of the proposed approach and the statistical models commonly used in credit scoring, such as linear discriminant analysis and logistic regression, in which humans can refer to model coefficients to interpret the model and its predictions [13].

In addition, classifiers based on optimization techniques have been studied in the literature [29]. It has been shown that the application of optimization techniques helps to improve the performance of machine learning methods [12]. Recently, many deep architectures have been adapted and shown effectiveness in solving classification problems and have been applied especially to credit scoring [19]. The linear classification problem consists of separating two sets of points in real space $\mathbf{R}^n$. The primary objective proposed in the literature is to find the hyperplane that minimizes the sum of the distances from all misclassified points to itself. A given hyperplane can easily be determined by Linear Programming (LP) [2, 23]. When the two sets intersect and the overlapping points exist, perfect linear separation is not possible. Consequently, the dataset can not be perfectly separated by the linear classification model due to the inability of the objective of LP to be met. In an alternative approach presented in [22], a discriminating hyperplane can be found by minimizing the number of misclassified points. In this paper, we consider both objectives: (i) minimizing the number of misclassified points and (ii) minimizing the sum of misclassification distances.

The purpose of the present paper is to develop an hybrid approach using the following two steps:

1. First, linear programming is used to find a performing solution to minimize the sum of misclassification distances;

2. Second, an efficient Variable Neighborhood Search (VNS) heuristic combined with Jackknife Resampling Method is used to improve the solution obtained in the first step based on a well balanced combination of the two objectives defined earlier.

The first step is easy to solve and gives a good approximation that can be used as a warm-start for the second step. VNS is a metaheuristic proposed by Mladenovic and Hansen [26], which considers multiple neighborhood structures in the search for an optimal (or near-optimal) solution. VNS has been successfully applied to solve many hard combinatorial optimization problems such as the 0-1 Quadratic knapsack problem [36], the Pickup and Delivery Problem with Time Windows [32], the vehicle routing problem with time windows [28] and the financial derivative problem [1], etc.

Jackknife resampling method has been successfully used to develop a statistical discriminant mathematical programming model[40]. The jackknife method was firstly proposed in [25, 30]. Since then, the jackknife method has been the focus of much research, for instance in [7, 31]. Recently, several papers, including [37] have used this procedure for assessing or calibrating predictive accuracy. The jackknife procedure applied to discriminant models allows us to generate pseudo-parameters from pseudodata by resampling the original observations and obtaining an accurate estimate of re-trained model parameters.

This paper is organized as follows. Section 2 presents the credit scoring problem. Section 3 describes the ingredients of VNS and presents a hybrid variant of VNS with JRM. Computational experiments are reported in Section 4. Finally, conclusions are drawn in Section 5.

2 The credit scoring problem

2.1 General description of the problem and assumptions

Credit agencies must quickly assess the level of risk associated with granting a new loan to a customer. To do this, they rely on past credit records at their disposal and use classification models that can distinguish between good and bad loans. By construction, there are two well-defined groups of loans G_1 and G_2 (good and bad loans); hence, the credit scoring problem is a two-group classification problem. Assuming a log of past records (input data) is known, each member of the two groups is characterized by a set of attributes (information) $X = (X_1, X_2, \ldots, X_p)$ (age, account, job, income, residency etc.). This information is recorded for different customers. The aim of this work is to provide the best possible discriminant function using the input data by measuring p, the number of discriminatory variables or attributes.

Given a sample E of n customers consisting of n_G good customers and n_B bad ones (possessing good and bad loans respectively); i.e., $n = n_G + n_B$ and $E = G_1 \cup G_2$. Denote by $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ the p attributes of customer i. The classification model used by the credit institution should use these attributes to score the customer. To do this, a weight w_j, $(j = 1, 2, 3, ..., p)$ is assigned to each attribute to define a weight vector solution $W = (w_1, w_2, \ldots, w_p)$, and a threshold c is calculated to separate the good customers from the bad ones. If $\sum_{j=1}^{p} w_j x_{ij} \geq c$ then customer i is classified good, otherwise customer i is classified bad.

2.2 Linear programming approach

The Mathematical Programming (MP) approach has been first applied in classification by Magasarian in [21]. Freed and Glover in [10] presented an evaluation of the linear programming approach (LP) in discriminant analysis. A classification approach has been proposed based on the idea of reducing the misclassification through the minimization of the overlaps / the maximization of the distance between the two groups [9].

As will be proven in the following subsection, it is likely or even possible in practical cases, to perfectly separate the good customers from the bad ones based solely on a LP approach. We generally

tolerate eventual errors (violations) by introducing a positive value a_i such that $\sum_{j=1}^{p} w_j x_{ij} \geq c - a_i$ if customer i is a good one and $\sum_{j=1}^{p} w_j x_{ij} \leq c + a_i$ if customer i is a bad one. The objective will then be to find the values of the weight vector w and the value of the separator c for which $\sum_{i=0}^{n} a_i$ is minimal. Thus, the final model is:

$$\text{Min } f_1(W) = \sum_{i=1}^{n} a_i \tag{1}$$

s. t. :

$$\sum_{j=1}^{p} w_j x_{ij} \geq c - a_i \qquad \forall i \in G_1 \tag{2}$$

$$\sum_{j=1}^{p} w_j x_{ij} \leq c + a_i \qquad \forall i \in G_2 \tag{3}$$

$$\sum_{j=1}^{p} (n_B \sum_{\substack{i=1 \\ i \in G_1}}^{n_G} x_{ij} - n_G \sum_{\substack{i=1 \\ i \in G_2}}^{n_B} x_{ij}) w_j = 1 \tag{4}$$

$$a_i \geq 0 \qquad \forall i, c \in \mathbb{R} \text{ and } w_j \in \mathbb{R} \forall j. \tag{5}$$

The objective function f_1 aims to minimize the distance between the misclassified customers and the hyperplane $W^T x = c$ representing the discriminant function. Constraints (2) and (3) ensure that each customer is either in group 1 (good loans) or group 2 (bad loans). Constraint (4) is a normalization needed to avoid the trivial solution ($w_j = 0 \forall j, c = 0$). Finally, constraints (5) define variables domains.

2.3 LP limits

To explain the shortcomings of the LP approach Figure 1 is introduced. Figure 1 presents a two-dimensional dataset with 30 customers $(I_1; ...; I_{30})$ partitioned into two groups (represented by two circles in the figure), each containing 15 customers. The area containing the points $(I_2, I_3, I_4, I_6, I_8, I_{13}, I_{16}, I_{17}, I_{19}, I_{21}, I_{22})$ represents the overlap of the two sets, and as seen in Figure 1, it is difficult to separate these two groups with a single hyperplane. In Figure 1, "$Hyperplane1$" presents the hyperplane provided by the LP method which minimizes the sum of the distances calculated for the eight points $(I_2, I_4, I_6, I_8, I_{16}, I_{17}, I_{19}, I_{21})$ which are misclassified by "$hyperplane1$". However, it is possible to construct "$hyperplane2$" as another separator for the two groups with only 4 misclassified points (I_3, I_4, I_6, I_{16}) whose sum of their distances is not minimal. Therefore, "$hyperplane2$" is not an optimal solution for the LP model but it minimizes the number of misclassified points.

From Figure 1, the following can be inferred:

- Minimizing the sum of Misclassified Points Distances (SMD) is not the only criterion that should be considered in the classification. The number of misclassified points is another important criterion.
- The discriminant function based on LP model that provides the minimum SMD is not necessarily the best classifier, (i.e. having the minimal number of misclassified points).
- Two separating hyperplanes can sometimes yield two classifiers with the same number of misclassified points with significantly different SMD values.

For the reasons mentioned above, another popular criterion to evaluate the performance of a classifier based on the Number of Misclassified Points (NMP) is introduced. For a given W, the number of misclassified points corresponds to the cases incorrectly classified using the hyperplane.

Hence, the objective of the credit scoring problem considered in this paper is a combination of two functions ($f = f_1 + \alpha f_2$, $\alpha \in \mathbf{R}^+$): (i) f_1 minimizes the sum of misclassified points' distances and (ii) f_2

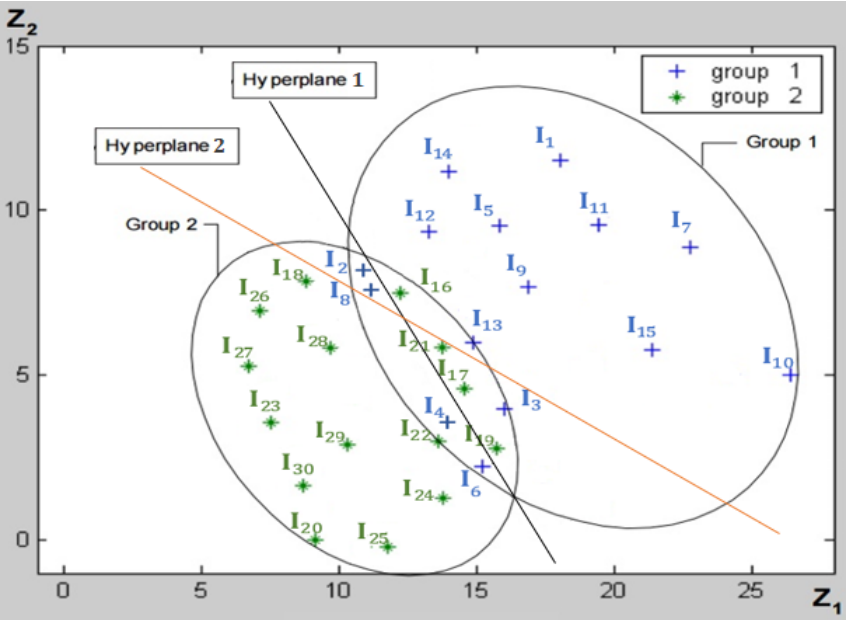

Figure 1: Different separating hyperplanes that best separate the data (in two groups)

minimizes the number of misclassified points. f_1 is depicted by Equation (1) in the mathematical model, on the other hand f_2 is calculated using the training set. To calculate the number of misclassified points for a given solution W, the following procedure is followed: (i) predict Y_c, the class of all clients used, (ii) compare Y_c with $DataY$, the real class already known for each client, and finally (iii) calculate the number of misclassified points. The pseudocode to calculate the function f_2 is provided in Algorithm 1.

Algorithm 1

1: calculate Y_c using W;
2: compare Y_c with $DataY$;
3: return the number of misclassified points

Given the nature of f_2, we must use a good methaheuristics like VNS to minimize f.

3 VNS using Jackknife Resampling Method

The algorithm developed in this paper is an efficient general variable neighborhood search hybridized with JRM. In this section, the neighborhood structures are presented, followed by an explanation of the classical components of VNS, namely the Variable Neighborhood Descent (VND) and the Shaking functions, then, the deployment of jackknife resampling method used in our approach, and finally, the introduction of the hybridization of VNS and JRM.

3.1 Neighborhood structures

The three neighborhood structures considered in this framework are denoted by N_1, N_2 and N_3 respectively. Let $W = (w_1, w_2, ..., w_p, c)$ be a given vector. The neighborhood N_1 is based on a swap operation which strips a value r, called a regenerator, from a component of W and adds it to another component of the same vector and repeats this for all of the first p components. Formally, $N_1(W) = \{(w_1, ..., w_i + r, ..., w_j - r, ..., w_p, c);\ 1 \leq i < j \leq p\}$. Figure 2 illustrates the construction of the neighborhood N_1.

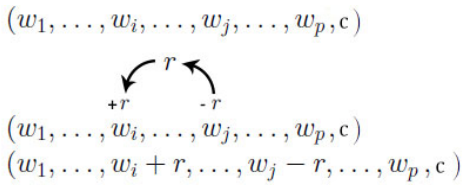

Figure 2: Neighborhood N_1.

Example 3.1 : let $W = (1, 2, 3, 4, 5)$ and $r = 1$,
$N_1(W) = \{$ $(0, 3, 3, 4, 5)$, $(0, 2, 4, 4, 5)$, $(0, 2, 3, 5, 5)$, $(0, 2, 3, 4, 6)$,
$(2, 1, 3, 4, 5)$, $(1, 1, 4, 4, 5)$, $(1, 1, 3, 5, 5)$, $(1, 1, 3, 4, 6)$,
$(2, 2, 2, 4, 5)$, $(1, 3, 2, 4, 5)$, $(1, 2, 2, 5, 5)$, $(1, 2, 2, 4, 6)$,
$(2, 2, 3, 3, 5)$, $(1, 3, 3, 3, 5)$, $(1, 2, 4, 3, 5)$, $(1, 2, 3, 3, 6)$,
$(2, 2, 3, 4, 4)$, $(1, 3, 3, 4, 4)$, $(1, 2, 4, 4, 4)$, $(1, 2, 3, 5, 4)$ $\}$

The neighborhood N_2 is obtained by adding a random regenerator r to the components of W. Formally, $N_2(W) = \{(w_1, \ldots, w_i + r, \ldots, w_p, c); \forall i\}$. Figure 3 illustrates the construction of neighborhood N_2.

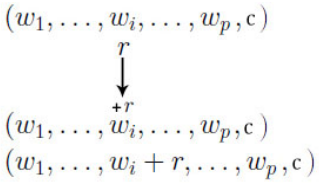

Figure 3: Neighborhood N_2.

Example 3.2: again, let $W = (1, 2, 3, 4, 5)$ and $r = 1$,
$N_2(W) = \{(2, 2, 3, 4, 5), (1, 3, 3, 4, 5), (1, 2, 4, 4, 5), (1, 2, 3, 5, 5), (1, 2, 3, 4, 6)\}$

The neighborhood N_3 is practically identical to N_1, only in this case the regenerator r is stripped from a component of W, and half of it is added to two other components respectively. Consequently, $N_3(W) = \{(w_1, \ldots, w_k + r/2, \ldots, w_i - r, \ldots, w_j + r/2, \ldots, w_p, c), \forall\ 1 \leq i \neq j \neq k \leq p\}$. Figure 4 illustrates the construction of N_3.

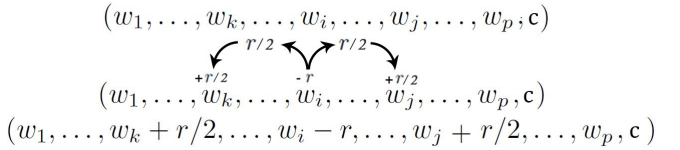

Figure 4: Neighborhood N_3.

Example 3.3: let $W = (2, 2, 3, 4, 5)$ and $r = 2$,
$N_3(W) = \{$ $(0, 3, 4, 4, 5)$, $(0, 3, 3, 5, 5)$, $(0, 3, 3, 4, 6)$, $(0, 2, 4, 5, 5)$, $(0, 2, 4, 4, 6)$, $(0, 2, 3, 5, 6)$,
$(3, 0, 4, 4, 5)$, $(3, 0, 3, 5, 5)$, $(3, 0, 3, 4, 6)$, $(2, 0, 4, 5, 5)$, $(2, 0, 4, 4, 6)$, $(2, 0, 3, 5, 6)$,
$(3, 3, 1, 4, 5)$, $(3, 2, 1, 5, 5)$, $(3, 2, 1, 4, 6)$, $(2, 3, 1, 5, 5)$, $(2, 3, 1, 4, 6)$, $(2, 2, 1, 5, 6)$,
$(3, 3, 3, 2, 5)$, $(3, 2, 4, 2, 5)$, $(3, 2, 3, 2, 6)$, $(2, 3, 4, 2, 5)$, $(2, 3, 3, 2, 6)$, $(2, 2, 4, 2, 6)$,
$(3, 3, 3, 4, 3)$, $(3, 2, 4, 4, 3)$, $(3, 2, 3, 5, 3)$, $(2, 3, 4, 4, 3)$, $(2, 3, 3, 5, 3)$, $(2, 2, 4, 5, 3)$ $\}$

3.2 VND procedure

3.2.1 LocalSearch function

The $LocalSearch()$ function introduced in this paper improves the current solution by performing the following steps. First, using one of the already defined neighborhood structures $N_j(j = 1, 2, 3)$, it creates the set of neighbors of the current solution $\widehat{W}$. Second, it chooses a better solution from the one already found using one of two local search methods: 1) the $first\ improvement$ method, which compares the elements of $\widehat{W}$ with W and returns the solution once it finds one better than W, or 2) the *best improvement* method, which compares the elements of $\widehat{W}$ with each other in order to find the best element W' of this set and return it's value. The $LocalSearch()$ function takes $N_j(j = 1, 2, 3)$ as an input to create $\widehat{W}$ then it returns W', the best solution found. Next, the algorithm compares W with W' and updates W's value if W' is found better than W. The $LocalSearch()$ function finds a better solution among the neighbors compared to the current solution, thereby improving itself. The pseudocode of the $LocalSearch()$ function is provided in Algorithm 2.

Algorithm 2

$\widehat{W} \leftarrow Neighberhood\ (N_j(W))$
$W' \leftarrow argming(w)_{w \in \widehat{W}}$
return W'

3.2.2 VND function

The VND function takes a current solution W as input and improves it. The next step is to use the constructed neighbourhood structures $N_j(j = 1, \ldots, n)$, (i.e. N_1, N_2 and N_3). As long as the $VND()$ function is able to improve W, j takes the value 1 in order to use the neighbourhood structure N_1 (the iterations will continue as long as j is below the number of structures defined in the beginning). The function $LocalSearch()$ is then used to create, firstly, the sets of neighbors of W using the N_1 structure. Secondly, it finds the best solution W' for this set (in this case, we adopt the *best improvement* method in the $LocalSearch()$ while ignoring the $first\ improvement$). If W' is better than W, then W is updated to W' and j gets the the value 1 to restart as a new iteration with N_1 as neighborhood's structure. If else, j is updated to $j+1$ to use the next neighborhood's structure. If there is no possible improvement, the algorithmic process is terminated. The pseudocode of the $VND(W)$ function is presented in Algorithm 3.

Algorithm 3

```
while there is a possible improvement do
    j ← 1
    while j <= 3 do
        W' ← LocalSearch (N_j(W))
        if W' is better than W then
            W ← W'
            j ← 1
        else
            j ← j + 1
        end if
    end while
end while
return W
```

3.3 Shaking() function

The $Shaking()$ function escapes the valleys of local optima by diversifying the search in other regions. The $Shaking(k, W_{best})$ function allows to move "far" away from the incumbent W_{best}. It generates a

new solution after applying m moves. Each move is generated randomly by using the k^{th} neighborhood of W. The value of m is tuned and made constant for all tests.

The pseudocode of the *Shaking*() function provided in Algorithm 4 is presented.

Algorithm 4

1: $i \leftarrow 1$, $W \leftarrow W_{best}$
2: **while** $i <= m$ **do**
3: $W \leftarrow$ randomly select one vector from $N_k(W)$
4: $i \leftarrow i + 1$
5: **end while**
6: return W

3.4 Jackknife procedure

Jackknife procedures have been successfully used to identify statistically significant parameter estimates in the cases where the underlying sampling distribution are either unknown or have no analytical solution. In [40], Ziari et al. developed a statistical discriminant model based on the jackknife procedure, and proved it to be the most promising among the resampling estimation techniques examined. In essence, the jackknife approach consists of partitioning the total sample of size n into t subsets $(k = 1, 2, \ldots, t)$, each with an equal size of d elements $(n = t \times d)$. At each iteration, a given subset is deleted from the whole training data and the effect of estimating the parameter W_k from the rest of the sample is calculated using the formulas given in [40]. The principle result of the jackknife method is to determine the weight vector W_k in each iteration (corresponding to the case of deleting the k^{th} subset).

The final vector will be the average of d resulting values of W_k from all iterations. The jackknife delete-d procedure used to provide estimates of VND coefficient is as follows:

1. Estimate the first jackknife parameter W_1 where $W_1 = (w_1^1, w_2^1, ..., w_p^1, c^1)$ (where p is the number of characteristics, which is the linear discriminant function coefficient using VND approach from $(n - d)$ sized remaining observation set.
2. Omit the second d observation set from the full sample and estimate the second jackknife parameter W_2 where $W_2 = (w_1^2, w_2^2, ..., w_p^2, c^2)$ is the linear discriminant function coefficient from $(n - d)$ sized remaining observation set.
3. Alternately omit the following d observations and estimate the linear discriminant function coefficient W_k, where W_k is the jackknife parameter estimated after deleting k^{th} d observation set from full sample. Thus, t delete-d jackknife sample are obtained consist of t vectors estimated parameter $W_1, W_2, \ldots, W_t$
4. Calculate the jackknife parameter corresponding to linear discriminant function coefficient estimate $W^* = (w_1^*, w_2^*, ..., w_p^*, c^*)$ where each coefficient is given by Efron in [7] as follow :

$$w_s^* = \sum_i^t \frac{w_s^i}{t}, s = 1, 2, \ldots, p \quad and \quad c^* = \sum_i^t \frac{c^i}{t},$$

3.5 A hybrid VNS/JRM

In this section, we present a hybridization of VNS and the JRM (VNS/JRM) for the linear discriminant analysis. JRM is used to study the effect of a particular subset of the population (sample) on a parameter estimate (W in our case). The JRM approach consists of: 1) partitioning the total sample into t subsets of equal size, where t is tuned by experimentation, 2) deleting a subset from the population, 3) estimating the effect on the parameter (we use VND in our case to compute W using the remaining population), and 4) repeating the above steps for every subset and averaging the results (effects).

3.5.1 LocalMinima function

The $LocalMinima()$ function is a hybridization of the $jackknife$ procedure that statistically distorts the solution, and the VND function that looks for the local minimums given an initial solution. Applying this hybridization ensures the convergence towards a certain local minimum, as it is capable of creating t data samples at each iteration of the $jackknife$ function, to which the VND function finds a local minimum so that we obtain t local minimums. The function returns the mean of these local minimums. The pseudocode of the $LocalMinima()$ function is presented in Algorithm 5.

Algorithm 5

1: $k \leftarrow 1$
2: Partition the population into t subsets of equal size.
3: **while** $k \leq t$ **do**
4: Remove the k^{th} subset from the population
5: $W_k \leftarrow$ VND(W)
6: $k \leftarrow k + 1$
7: **end while**
8: return $W^* = \sum_{k=1}^{t} \frac{W_k}{t}$

3.5.2 General algorithm

The initial solution W^0 is generated by CPLEX using f_1 (i.e. $\alpha = 0$) for the VNS/JRM model, then the neighborhood structures $Nj(j = 1; \ldots; n)$ are defined, which, in our case, are $N1, N2, N3$. While the algorithm is able to find an improvement for W_{best} (using the objective function f), k gets the values from 1 to K_{max} (where K_{max} is the number of neighborhoods) for shaking the current solution using the $Shaking()$ function. Afterwards, the $LocalMinima()$ function is used to find the local minimum W'. Then, we check if W' is better than W_{best}. If the condition is met, W_{best} is updated to W' and k is set to the initial value (i.e. $k = 1$), else increment k (i.e. $k = k + 1$) and restart the $Shacking()$ function in order to move away from the local minimum solutions. Finally, if the $Shaking()$ function runs K_{max} times without any improvements, it is assumed that there isn't any improvement possible, thus the algorithm stops. The pseudocode of the proposed solution is presented in Algorithm 6.

Algorithm 6

1: initialize W
2: $W_{best} := W$
3: define a set of neighborhood structures N_j, $(j = 1, 2, 3)$ of the current solution
4: **while** stopping condition is not met **do**
5: $k \leftarrow 1$
6: **while** $k <= K_{max}$ **do**
7: $W \leftarrow$ Shaking (k,W_{best})
8: $j \leftarrow 1$
9: $W^{'} \leftarrow$ LocalMinima (W)
10: **if** W' is better than W_{best} **then**
11: $W_{best} \leftarrow W'$
12: $k \leftarrow 1$
13: **else**
14: $k \leftarrow k + 1$
15: **end if**
16: **end while**
17: **end while**
18: return W_{best}

To summarize, as demonstrated in Figure 5, the VNS/JRM algorithm contains two main phases: (i) the construction of an initial solution W^0 by solving the linear program of Section 2.2 using CPLEX, (ii) the improvement of W_{best} ($W_{best} = W^0$ in the beginning of phase 2 by combining VNS and JRM using the objective f and tuning the parameter α. Each time the algorithm is able to find an improvement of W_{best}, it returns back to $k = 1$. The algorithm stops if there is no improvement.

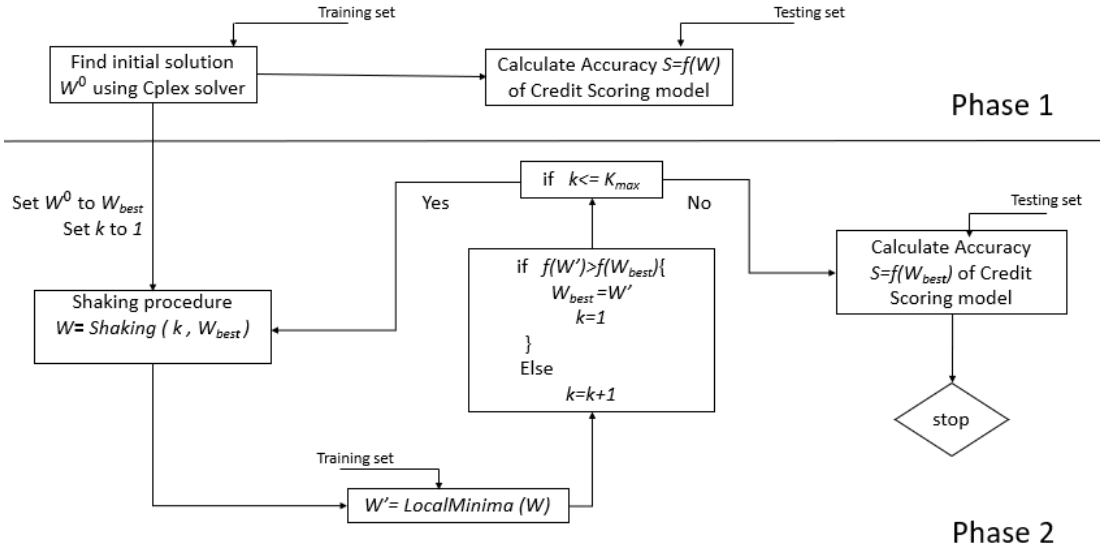

Figure 5: Block diagram of building VNS model using JRM

4 Numerical experiments

In the first subsection, we briefly describe all the datasets used in the computational results. To evaluate the performance of the VNS/JRM classifier with CPLEX, a cross-validation (CV) training-testing method (stratified 10-fold) was adopted. The same training set that is used by CPLEX in phase 1, is also used by VNS/JRM to improve the current solution in phase 2. The VNS/JRM classifier is then validated using the same testing set to assess its performance and compare it to the solution obtained by CPLEX and to state of the art classifiers.

4.1 Datasets description

Three real datasets (the Australian, German, and Taiwanese datasets which are derived from the UCI Machine Learning Repository and most used in the literature [13]) are used to evaluate the VNS/JRM classifier. Table 1 summarizes the main characteristics of the datasets. The number of numerical features is given by N_{num} while the number of categorical features is given by N_{cat}.

Table 1: Datasets characteristics

DATASET	Sample size	No. features	N_{num}	N_{cat}
Australian	690	14	8	6
German	1000	20	13	7
Taiwanese	30000	23	14	9

All the benchmark datasets that have been used to test the algorithm can be found at :

- https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)
- https://archive.ics.uci.edu/ml/datasets/statlog+(australian+credit+approval)
- https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

4.2 Performance evaluation metrics

Different evaluation metrics have been used in the literature to evaluate the performance of classification techniques. They include accuracy, sensitivity (or recall), specificity, precision, F1 score, G-mean,

and area under the curve (AUC). In this study, all the aforementioned evaluation metrics, which are most used in credit scoring [16], are considered and can be calculated based on a confusion matrix.

Accuracy (ACC) is a criterion for measuring the classification accuracy of a model, which can be calculated by the following equation:

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

Precision is the proportion of all predictions that are correctly predicted as positive are true. It can be calculated by the following equation:

$$Precision = \frac{TP}{TP \ + \ FP}$$

Sensitivity or Recall shows how many positive examples in the dataset are predicted correctly and can be calculated by the following equation:

$$Recall = \frac{TP}{TP \ + \ FN}$$

Specificity shows how many negatives examples in the data set are predicted correctly and can be calculated by the following equation:

$$Recall = \frac{TN}{TN \ + \ FP}$$

F1-measure is a comprehensive evaluation metric based on precision and recall metrics, it is calculated using the following expression:

$$F1 - measure = \frac{2(Precision * Recall)}{Precision + Recall}$$

G-mean is an evaluation method constructed with sensitivity and specificity. It can be interpreted as the geometric mean of both measures, which is calculated by the following equation:

$$\text{G-mean} = \sqrt{sensitivity * specificity}$$

Where FP (False Positive) is the number of customers mistakenly classified as “good”, and who are actually “bad”, TN (True Negative) is the number of customers classified as “bad” and who are indeed “bad” (well-classified), FN (False Negative) is the number customers mistakenly classified as “bad” and who are actually “good”, and TP (True Positive) is the number of customers classified as “good”, and who are indeed “good” (well-classified).

AUC refers to the area under the receiver operating characteristic curve (ROC), which is a comprehensive indicator reflecting the continuous variables of sensitivity and specificity. The range of AUC is generally between 0.5 and 1. If a classifier has an AUC value equal to 1, it means that two groups of cases can be completely separated by the classifier.

4.3 Computational results and discussion

4.3.1 Experimental evaluation of VNS/JRM

The main objective of VNS/JRM procedure is to optimize the model performance. Due to the fact that real-world datasets are usually linear inseparable, it is hard for the LP approach to obtain the best classifier. To highlight the limits of the minimizing process, an extended empirical study of VNS process using jackknife procedure is employed.

A : Evaluation of VNS/JRM using the function f_1 as objective. In order to show numerically the limits of the LP approach defined in Section 2.2, the VNS/JRM started with an initial solution different than that of CPLEX (randomly generated) and the objective based on minimizing the sum of misclassified points distances (SMD) is applied (i.e minimizing the function f_1). In addition to minimizing SMD, we are interested in identifying the impact of the proposed algorithm on the average number of the well classified points which is defined by Accuracy. For this purpose, we evaluate the performance of the corresponding scoring model in terms of SMD and we compute the Accuracy simultaneously. The application of the jackknife procedure allows us to generate a number of Local Minimums.

For a given iteration of the algorithm, the SMD and the Accuracy of the developed models by LocalMinima procedure for the Australian, the German and the Taiwanese datasets are illustrated in Figures 6, 7 and 8 respectively. Since the number of iterations generated by the algorithm for the Taiwanese dataset is very large, we have added a curve which zooms in on a part of the Figure 8. For each figure, five descents can be distinguished from the curve of SMD. However, the curve of Accuracy shows fluctuating results and represents a different trend compared to the curve of the SMD. It can be inferred from Figures 6, 7 and 8 that the SMD values decrease to reach a local minimum several times while Accuracy presents a volatile curve and unstable behavior. From the above remarks, it can be deduced that there is no consistency between a SMD minimal and an Accuracy maximal.

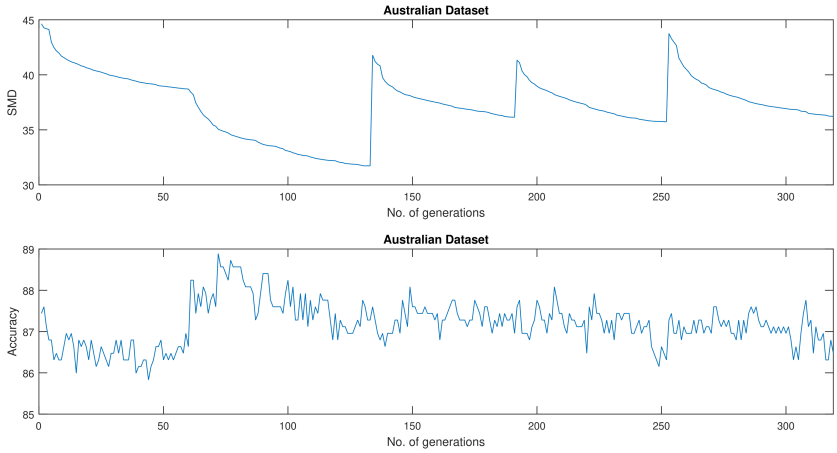

Figure 6: SMD and Accuracy of VNS/JRM for Australian dataset

B : Evaluation of VNS/JRM using the function $f_1 + \alpha f_2$ as objective. The number of misclassified points (NMP) denoted by the function f_2 will be considered in the evaluation of the proposed method. In this case, we study the combination of f_1 (SMD) and the f_2 (NMP) as defined in sect. 2.3. Different tests were done to verify the new objective's impact to the results considering the three datasets. For each iteration, the new fitness function $f_1 + \alpha f_2$ noted by (SMD & NMP) is computed for a given value of the parameter α and the Accuracy of the developed scoring model is calculated. Many tests have been done to determine the optimal value of α and the results have shown that the best value for α are 100, 1000 and 100 for the Australian, the German and the Taiwanese datasets respectively.

Figures 9, 10 and 11 depict the SMD & NMP and Accuracy curves of each dataset. From these figures, we can see that the combination of two objectives SMD and NMP has a significant impact on the results of the VNS/JRM approach. It is observed that the Accuracy of the developed models increases along with the minimizing process for all the datasets considered. Hence, the numerical results show the proposed combination method to be effective for improving the accuracy for credit scoring datasets.

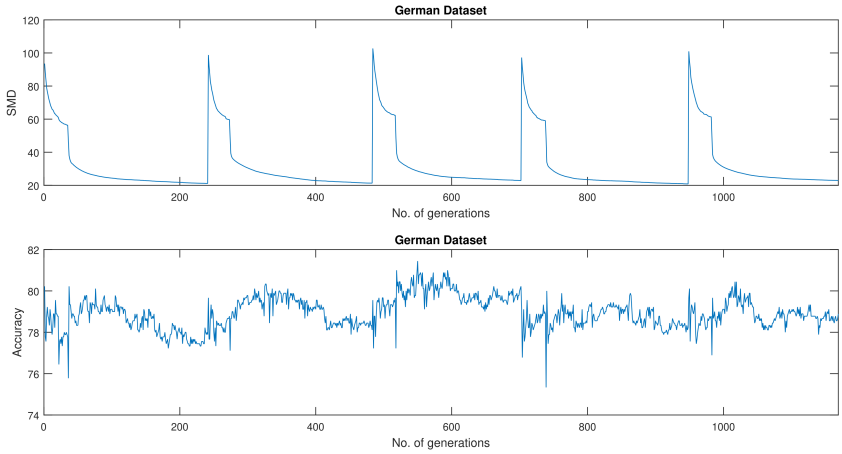

Figure 7: SMD and Accuracy of VNS/JRM for German dataset

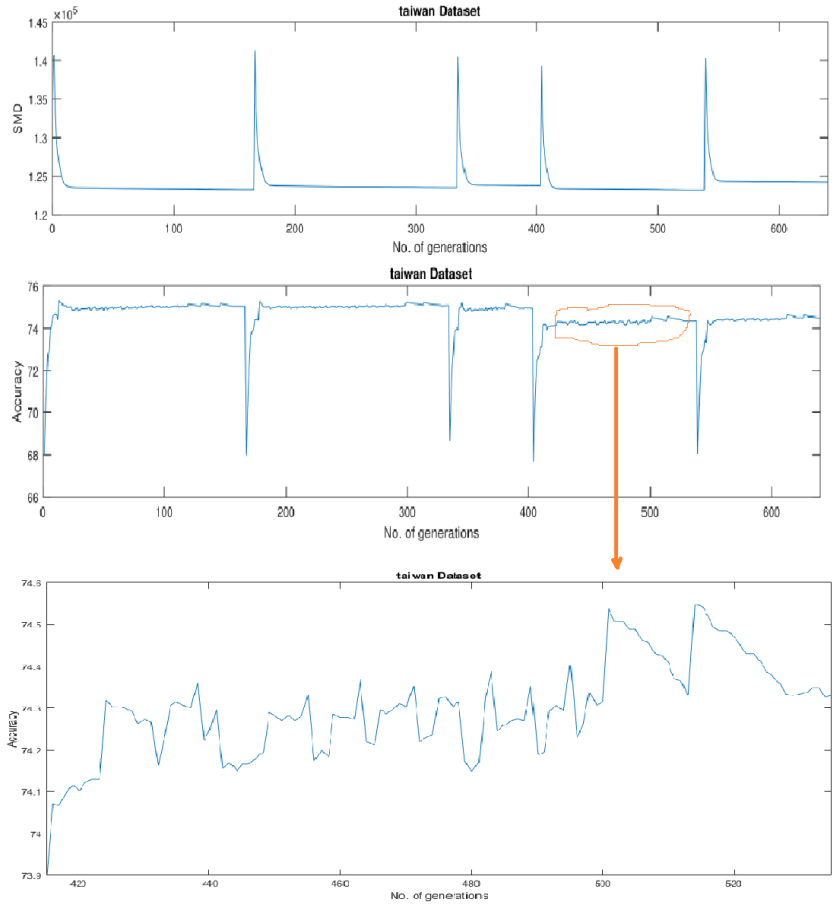

Figure 8: SMD and Accuracy of VNS/JRM for Taiwanese dataset

C : Comparing results of CPLEX and VNS/JRM. Table 2 presents the results of CPLEX and VNS/JRM over five performance metrics considering the three datasets. Firstly, the optimal solution was obtained by minimizing only f_1 which is the CPLEX results. Secondly, the outcome can be improved using VNS/JRM which minimizes $f_1 + \alpha f_2$, the fitness function for the three datasets. The

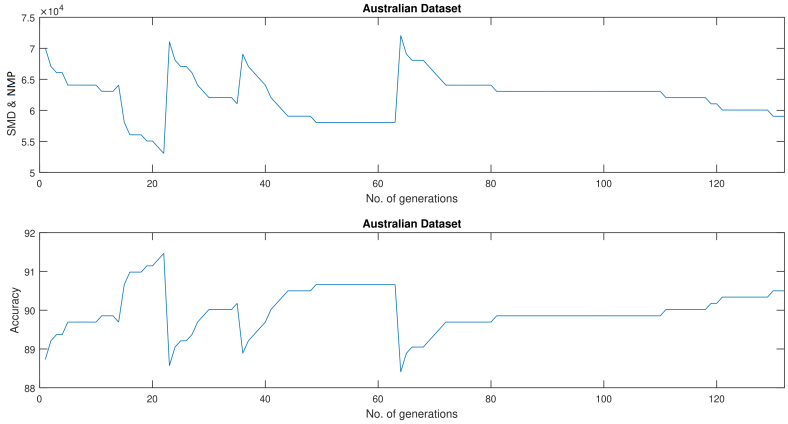

Figure 9: SMD & NMP and Accuracy of models in many generations for Australian dataset

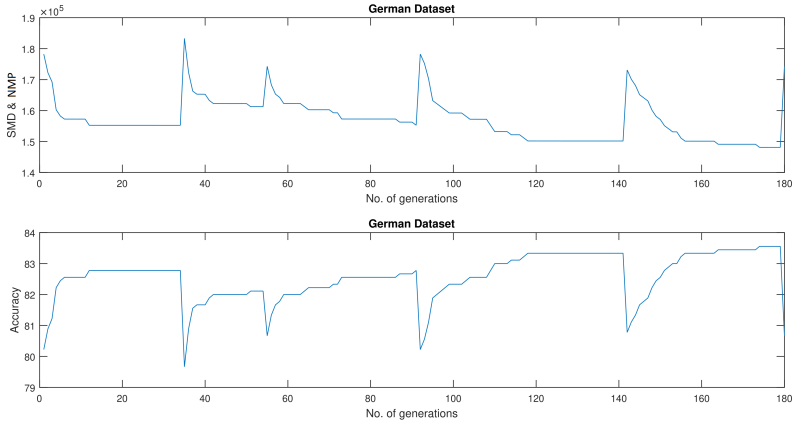

Figure 10: SMD & NMP and Accuracy of models in many generations for German dataset

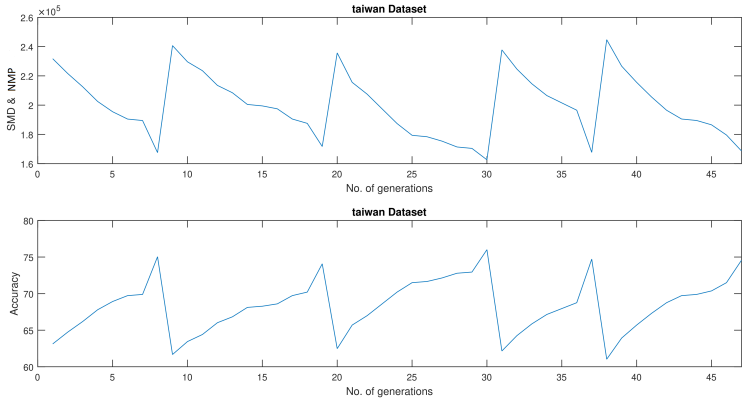

Figure 11: SMD & NMP and Accuracy of models in many generations for Taiwanese dataset

VNS/JRM with CPLEX algorithm is more effective than CPLEX alone because it could reach the best solutions for all the cases, as shows in Table 2. This comparison indicates that although minimizing the f_1 on its own is an efficient method to obtain a good solution, the outcome can be improved not only by considering local optimal solutions, but also by reducing the number of misclassified cases. Through this comparison, the VNS/JRM approach which considers $f_1+\alpha f_2$ as fitness function is found to be a viable alternative to the linear programming problem in credit scoring.

Table 2: Comparison of Cplex and VNS/JRM over five performance metrics

Datasets.	CPLEX Solver				
	Accuracy	Precision	Recall	F1-measure	AUC
Australian	85.65	87.57	86.42	84.01	92.27
German	73.60	81.32	80.86	81.09	76.12
Taiwanese	76.06	84.65	84.61	84.63	70.87
Datasets.	**VNS/JRM**				
	Accuracy	Precision	Recall	F1-measure	AUC
Australian	87.53	84.64	87.95	86.26	91.61
German	76.40	79.67	89.00	84.08	75.82
Taiwanese	81.50	83.10	95.73	88.97	65.01

4.3.2 Finding the optimal number of moves m for shaking function

The *Shaking*() function is an essential element in the VNS/JRM approach as it is used to avoid the local minima traps. It requires an input diversification parameter m which represents the maximum moves in each shaking. In order to investigate the effects of the proposed algorithm, we record the Accuracy, Specificity, Sensitivity and G-Mean of the feasible solutions obtained by varying the value of parameter m. To ensure the representation of different regions of the search space different values for the parameter m are set, ranging from 10 to 150.

Figures 12 and 13 show the Accuracy, Specificity and Sensitivity for each iteration in the tenfold cross-validation procedure of the experiments. From Figure 12, the accuracy improves with an increase in the number of m moves applied to a solution to jump from a local optimum. Good solutions are obtained when m= 110 for the Australian dataset, and m=60 for the German dataset, while for the Taiwanese dataset it is when m=150 and as m exceeds that value, the computation time of VNS/JRM is more than 10 minutes with a slight variation of the performance of the model. It is worth mentioning that the search strategy proposed in this paper is effective in improving the accuracy of data classification.

The results of Figure 13 indicate that specificity and sensitivity fluctuate and tend to increase slightly by varying the value of m. Moreover, the sensitivity and the specificity can be improved with increasing values of m as shown earlier for the accuracy. The results shown in figures 12 and 13 reveal that the number of moves in the shaking function is important for the VNS/JRM approach.

The metric G-mean is used to measure the balance between the classification performances on both the majority and minority classes. Best values of G-mean obtained across various m for the Australian, the German and the Taiwanese datasets were 87.62, 75.34, and 69.14 respectively. It can be concluded that even if the Taiwanese dataset is the most imbalanced, the VNS/JRM model performs better in this situation and yields better performance. The Australian dataset having the highest G-mean followed by the German dataset confirms the existing balance between the two classes.

4.3.3 Comparing with other classifiers focusing on credit scoring

In this subsection, we compare the performance of VNS/JRM with that of the state of art methods in machine learning, such as Linear Discriminant Analysis (LDA), Logistic Regression (LR), K-nearest neighbor(knn), Decision tree (DT), Neural-Network Analysis (ANN), Support Vector Machines (SVM) for both linear (Lin SVM) and RBF kernels (RBF SVM), and Least Square Support Vector Machines (LS-SVM) for RBF kernels (RBF LS-SVM) for credit scoring. (LDA, LR, DT, knn are performed by a manual search for setting parameters [11] and a grid search has been applied to the parameter tuning of LinSVM, RBFSVM, RBFLS-SVM and ANN [13]). For each dateset, the results of the evaluation measures obtained from each learning algorithm are found by performing ten experiments and is averaged after a 10-fold cross-validation. The best results obtained are marked in bold font.

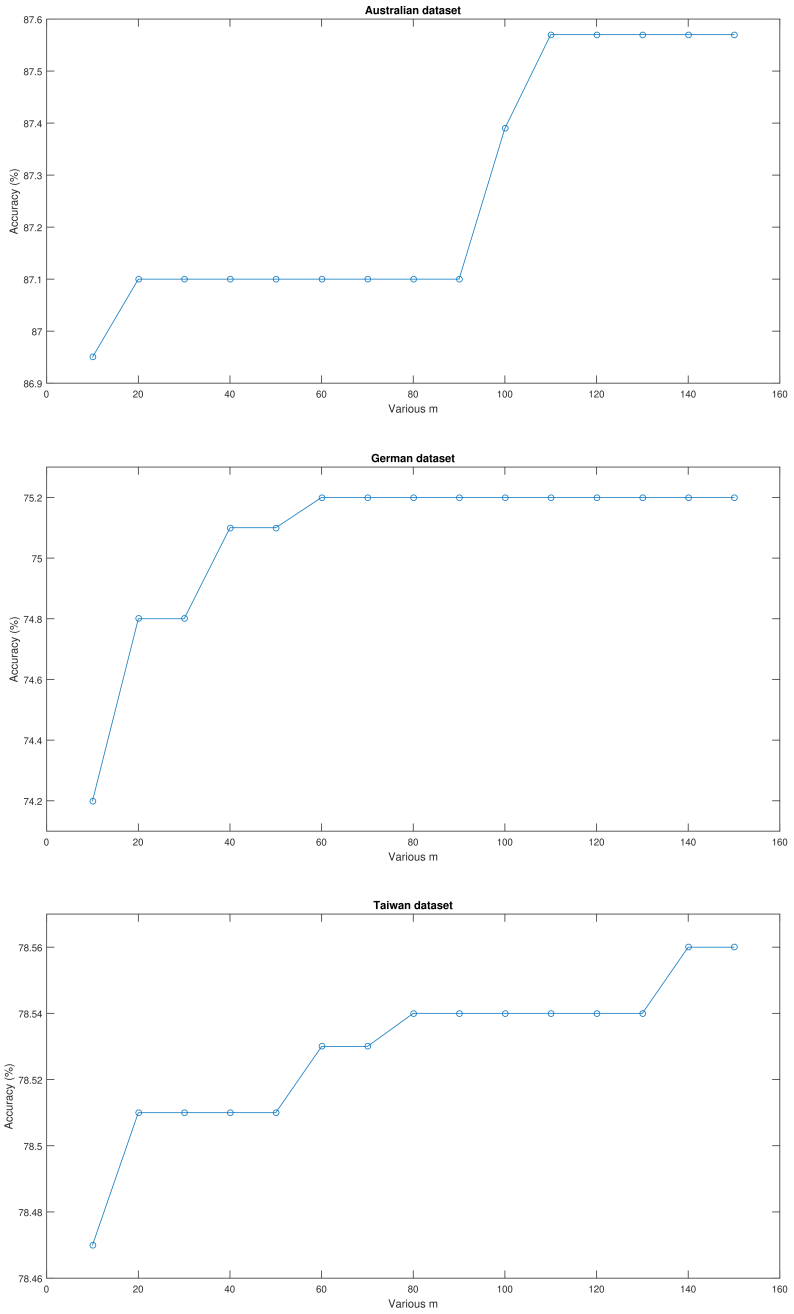

Figure 12: Accuracy of VNS/JRM of each dataset for various m

Table 3 presents the average results of each classifier on the Australian dataset. Compared to all other classifiers, JRM/VNS model obtains the best ranking results three times out of the five. As shown in Table 3, the accuracy, precision and the F1-measure of the VNS/JRM excel all counterparts, and its AUC achieves a average of 91.61% which is a better value compared to other classifiers.

Table 4 shows the average computational results on the German dataset. VNS/JRM achieves the highest AUC of all classifiers and exhibits the second best accuracy and F1-measure with slight difference. Moreover, among the models used in this study, VNS/JRM obtains good values for the remaining measures.

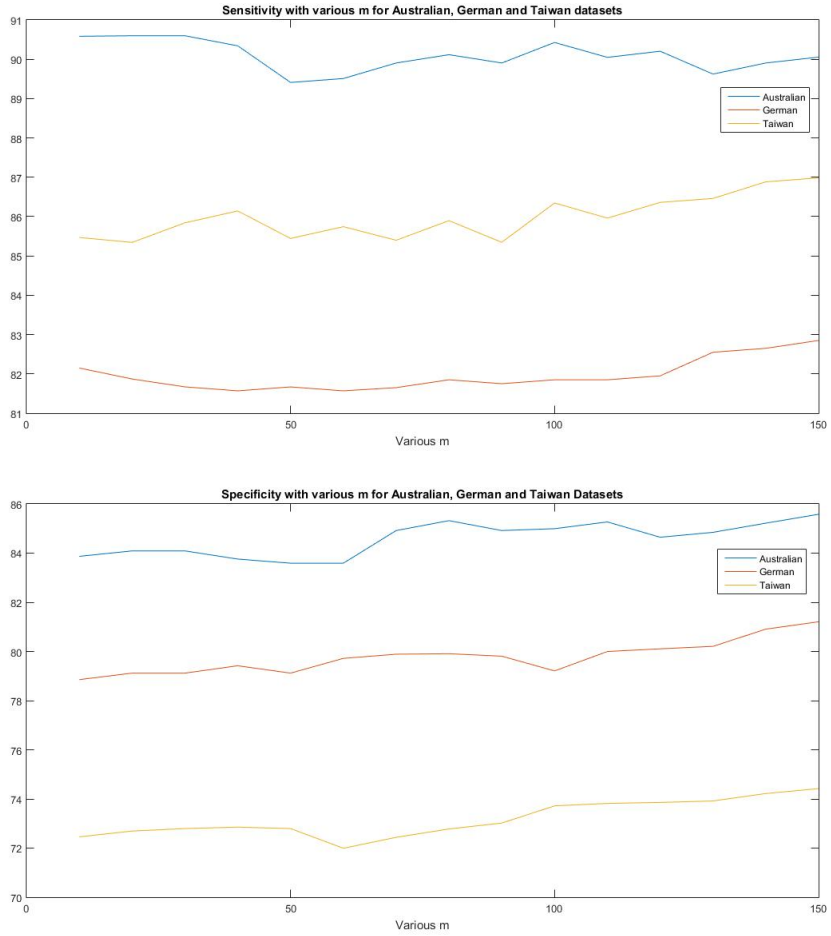

Figure 13: Sensitivity and Specificity of VNS/JRM of each dataset for various m

Table 3: Result of comparing performance of classifiers over Australian dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-measure (%)	AUC
VNS/JRM	**87.53**	**84.64**	87.95	**86.26**	91.61
LDA	85.94	79.99	92.11	85.39z	**92.98**
LR	85.50	82.64	86.48	84.19	92.96
knn	80.14	82.22	71.11	75.79	87.63
DT	81.44	79.81	78.61	78.98	85.36
LinSVM	71.59	69.42	79.51	71.07	86.83
RBFSVM	85.50	78.91	**92.78**	85.08	92.13
RBFLS-SVM	86.52	83.56	87.92	85.36	92.68
ANN	84.63	82.81	83.32	82.68	91.37

Table 5 shows the results of the tenfold cross-validation of the Taiwanese dataset. VNS/JRM performs better than all classifies except ANN in terms of accuracy, F1-measure, and AUC with slight difference. Meanwhile, VNS/JRM achieves acceptable Precision and good Recall values compared to most of its counterparts.

The running time did not exceed 10 minutes for the largest used dataset (Taiwan). The comparisons of the results of the three tables shows that VNS/JRM is able to produce significant results in a reasonable amount of time. Hence, VNS/JRM presents a promising alternative for solving the classification problem.

Table 4: Result of comparing performance of classifiers over German dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-measure	AUC
VNS/JRM	76.40	79.67	89.00	84.08	**79.82**
LDA	72.60	85.94	72.64	78.65	79.26
LR	75.80	79.44	88.39	83.55	79.07
knn	72.90	73.62	**95.91**	83.14	74.83
DT	70.60	77.71	81.16	79.30	68.18
LinSVM	75.50	78.33	90.11	83.67	79.09
RBFSVM	71.90	**86.04**	71.38	77.88	79.77
RBFLS-SVM	**76.80**	79.91	89.51	**84.32**	78.57
ANN	72.10	77.83	85.06	80.92	74.02

Table 5: Result of comparing performance of classifiers over Taiwanese dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-measure	AUC
VNS/JRM	81.50	83.10	95.73	88.97	75.01
LDA	72.26	**86.93**	75.68	80.90	71.51
LR	80.96	81.73	97.26	88.82	72.07
knn	81.10	82.96	95.25	88.68	73.12
DT	73.79	82.90	83.55	83.22	65.65
LinSVM	77.81	77.81	**100**	87.51	55.12
RBFSVM	77.85	77.87	99.96	87.53	50.45
RBFLS-SVM	80.21	81.14	97.13	88.41	71.44
ANN	**82.03**	84.11	94.81	**89.14**	**76.24**

5 Conclusion

In this paper, we propose an efficient heuristic based on jackknife resampling technique to improve the classification performance obtained by statistical Mathematical Programming for solving the credit scoring problem. The novelty of our approach can be summarized by the following: (i) the combination of two objective functions, one which minimizes the sum of misclassified points' distances and the second which minimizes the number of misclassified points, and (ii) the reduction of the bias of parameters using JRM.

Compared with machine learning methods, our credit scoring approach is more explainable, very easy to interpret and to implement. From the numerical results presented in this paper, the proposed VNS/JRM approach is often as accurate as the best performing black box machine learning models, yielding the best risk default detection and improving the efficiency and effectiveness of the classifier. Combining statistical learning, mathematical programming, and methaheuristics can add value to the existing literature by yielding powerful tools in classification. For future investigation, the approach presented here will be applied to higher dimensional linearly inseparable data. Additionally, it will be applied to quadratic models used in more complicated classification problems.

References

[1] N.I.L. Amaldass, C. Lucas, and N. Mladenovic. Variableneighbourhood search for financial derivative problem. Yugoslav Journal of Operations Research, 29:359—-373, 2019.

[2] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. Optim. Methods Softw., 1(1):23–34, 1992.

[3] A. Bequé and S. Lessmann. Extreme learning machines for credit scoring: An empirical evaluation. Expert Syst. Appl., 86:42–53, 2017.

[4] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA., 1984.

[5] J.N. Crook, D.B. Edelman, and L.C. Thomas. Recent developments in consumer credit risk assessment. Eur J Oper Res, 183(3):1447–1465, 2007.

[6] X. Dastile, T. Celik, and M. Potsane. Statistical and machine learning models in credit scoring: A systematic literature survey. Applied Soft Computing Journal, 91, art. no. 106263, DOI: 10.1016/j.asoc.2020.106263:58–69, 2020.

[7] Gong G. Efron, B. A leisurely look at the bootstrap, the jackknife, and cross-validation. The American Statistician, 37:1–36, 1983.

[8] R. A. Fisher. The use of multiple measurements in taxonomic problems. Annals of Eugenics, 7:179–188, 1936.

[9] N. Freed and F. Glover. Evaluating alternative linear, programming models to solve the two group discriminant problem. Decision Science, 17:151–162, 1986.

[10] N. Freed and F. Glover. Simple but powerful goal programming models for discriminant problems. European Journal of Operational Research, 1981:44–60, 7.

[11] T. Ge-Er, H. Chang-Zheng, and J. Jin, X. Xiao-Yi. Customer credit scoring based on hmm/gmdh hybrid model. Knowl. Inf. Syst., 36(3):731–747, 2013.

[12] M.B. Gorzaczany and F. Rudziski. A multi-objective genetic optimization for fast, fuzzy rule-based credit classification with balanced accuracy and inter- pretability. Appl. Soft Comput., 40:206–220, 2016.

[13] Y. Guo, J. He, L. Xu, and W. Liu. A novel multi-objective particle swarm optimization for comprehensible credit scoring. Soft. Comput., 23:9009–9023, 2019.

[14] W.E. Henley and D.J. Hand. Construction of a k-nearest-neighbour credit-scoring system. IMA JManag Math., 8(4):305–321, 1997.

[15] C.-L. Huang, M.-C. Chen, and C.-J. Wang. Credit scoring with a data mining approach based on support vector machines. Expert Syst. Appl., 2007.

[16] A. Khashman. Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. Expert Systems with Applications, 37:6233–6239, 2010.

[17] N. Kozodoi, S. Lessmann, K. Papakonstantinou, Y. Gatsoulis, and B. Baesens. multi-objective approach for profit-driven feature selection in credit scoring. Decis. Support Syst., 120:106–117, 2019.

[18] T.S. Lee, C.C. Chiu, Y.C. Chou, and C.J. Lu. Mining the customer credit using classification and regression tree and multivariate adaptiveregression splines. Computational Statistics & Data Analysis, 50:1113–1130, 2006.

[19] K. Lei, Y. Xie, S. Zhong, J. Dai, M. Yang, and Y. Shen. Generative adversarial fusion network for class imbalance credit scoring. Neural Computing and Applications, 32 (12), DOI: 10.1007/s00521-019-04335-1:8451–8462, 2020.

[20] S. Lessmann, B. Baesens, H.V. Seow, and L.C. Thomas. Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research. European Journal of Operational Research, 247:124–136, 2015.

[21] O.L Magasarian. Linear and non linear separation of patterns by linear programming. Operations research, 13:444–452, 1965.

[22] O. L. Mangasarian. Misclassification minimization. Journal of Global Optimization, 5:309–323, 1994.

[23] O. L. Mangasarian. Arbitrary-norm separating plane. Oper. Res. Lett., 24(1–2):15–23, 1999.

[24] D. Martens, B. Baesens, T.V. Gestel, and J. Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. Eur J Oper Res, 183:1466–1476, 2007.

[25] Rupert G Miller. The jackknife: a review. Biometrika, 61(1):1–15, 1974.

[26] N. Mladenovic and P.: Hansen. Variable neighborhood search. Computers & Operations Research, 24:1097–1100, 1997.

[27] V. Neagoe, A. Ciotec, and G. Cucu. Deep convolutional neural networks versus multilayer perceptron for financial prediction. In International Conference on Communications, COMM, pages 201–206, 2018.

[28] B. Olli. A reactive variable neighborhood search for the vehicle-routing problem with time windows. INFORMS Journal on Computing, 15:347–368, 2003.

[29] F. Plastria, S. De Bruyne, and E. Carrizosa. Alternating local search based vns for linear classification. Annals of Operations Research, 174:121—-134, 2010.

[30] Maurice H Quenouille. Notes on bias in estimation. Biometrika, 43(3/4):353–360, 1956.

[31] J. Shao and D.S. Tu. The jackknife and bootstrap. Springer-Verlag, New York, 1995.

[32] R. Stefan and P. David. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science, 40:455–472, 2006.

[33] G. Stiglic, P. Kocbek, N. Fijacko, M. Zitnik, K. Verbert, and L. Cilar. Interpretability of machine learning-based prediction models in healthcare. Data Mining Knowl Discov, 10:1–13, 2020.

[34] L.C. Thomas. A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. International Journal of Forecasting, 16:149–172, 2000.

[35] J.M. Tomczak and M. Zieba. Classification restricted boltzmann machine for credit comprehensible scoring model. Expert Syst Appl, 42:1789–1796, 2015.

[36] S. Toumi, M. Cheikh, and B. Jarboui. 0–1 quadratic knapsack problem solved with vns algorithm. Electronic Notes in Discrete Mathematics, 47:269–276, 2015.

[37] H. Yang and Y. Zhao. Smoothed jackknife empirical likelihood for the one-sample difference of quantiles. Comput. Statist. Data Anal., 120:58–69, 2018.

[38] H. Zhang, H. He, and W. Zhang. Classifier selection and clustering with fuzzy assignment in ensemble model for credit scoring. Neurocomputing, 316:210–221, 2018.

[39] B. Zhu, W. Yang, H. Wang, and Y. Yuan. A hybrid deep learning model for consumer credit scoring. In International Conference on Artificial Intelligence and Big Data (ICAIBD), pages 205–208, 2018.

[40] H. A. Ziari, D. J. Leatham, and P. N. Ellinger. Development of statistical discriminant mathematical programming model via resampling estimation techniques. American Journal of Agricultural Economics, 79:1352–1362, 1997.