IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 10, 2020, accepted December 14, 2020, date of publication December 21, 2020,
date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3046078

Swarm-Based Machine Learning Algorithm for Building Interpretable Classifiers

DIEM PHAM[1,2], BINH TRAN[1], (Member, IEEE), SU NGUYEN[1], (Member, IEEE), AND DAMMINDA ALAHAKOON[1], (Member, IEEE)

[1]Research Centre for Data Analytics and Cognition, La Trobe University, Melbourne, VIC 3086, Australia
[2]College of Information and Communication Technology, Can Tho University, Can Tho 900100, Vietnam

Corresponding author: Binh Tran (b.tran@latrobe.edu.au)

This work was supported by La Trobe University: FUNDREF 10.13039/501100001215, GRANT #(s) Startup Grant / ASSC-2020-RSU20-TRAN.

ABSTRACT This paper aims to produce classifiers that are not only accurate but also interpretable to decision makers. The classifiers are represented in the form of risk scores, i.e. simple linear classifiers where coefficient vectors are sparse and bounded integer vectors which are then optimised by a novel and scalable discrete particle swarm optimisation algorithm. In contrast to past studies which usually use particle swarm optimisation as a pre-processing step, the proposed algorithm incorporates particle swarm optimisation into the classification process. A penalty-based fitness function and a local search heuristic based on symmetric uncertainty are developed to efficiently identify classifiers with high classification performance and a preferred model size or complexity. Experiments with 10 benchmark datasets show that the proposed swarm-based algorithm is a strong candidate to develop effective linear classifiers. Comparisons with other interpretable machine learning algorithms that produce rule-based and tree-based classifiers also demonstrate the competitiveness of the proposed algorithm. Further analyses also confirm the interpretability of the produced classifiers. Finally, the proposed algorithm shows excellent speed-up via parallelisation, which gives it a great advantage when coping with large scale problems.

INDEX TERMS Particle swarm optimization, risk score prediction, interpretable machine learning, classification.

I. INTRODUCTION

In recent years, artificial intelligence (AI) has been widely adopted in various application domains from medicine and energy to supply chain management. AI-based tools used in these domains have shown to be more productive compared to conventional solutions, and even created better outcomes compared to human decisions in certain cognitive tasks [1]. These achievements have motivated researchers and practitioners to further improve and extend AI technologies in order to solve more complex problems such as self-driving cars, digital marketing, production management, and medical diagnosis [2]–[5]. While the preliminary outcomes are promising, the increasing levels of sophistication and automation of AI applications have raised a number of concerns related to trust, reliability, and fairness. Given that AI or the decisions made by AI will impact increasing numbers of users, it is more important than ever to understand how AI makes a particular decision and why such a decision is made.

Interpretable ML (IML) includes a set of models and algorithms which are interpretable by design. Representatives of this class are logistic regression, decision trees [6], and Bayesian Rule List [7]. These algorithms focus on building ML models which can be easily interpreted by decision makers while achieving a strong prediction accuracy. IML is particularly popular in risk assessment problems in a wide range of critical domains such as medicine [8], [9] and finance [10]. In these problems, users are not only interested in calculating the risk but also required to understand how the risks are determined in order to provide appropriate explanations and recommendations to the parties involved. These requirements are derived from the genuine need of the users to solve their problems, e.g. the doctors from intensive care units must know what factors can increase the risk of mortality to come up with prevention or treatment plans. To address these requirements, risk score models have been widely used in medical domain [8], [9].

A. RISK SCORE MODELS

Risk scores are linear classification models that allow users to make quick predictions through simple arithmetic (e.g. additions and subtractions) even without electronic assistance. Fig. I shows an example of risk scores for assessing bad credit where the score is easily calculated by adding up the

The associate editor coordinating the review of this manuscript and approving it for publication was Li He.

1. Status of existing checking account: >= 200 DM.	-1 point		...
2. No checking account.	-3 points	+	...
3. Savings account/bonds < 100 DM.	1 point	+	...
4. Property real estate.	-1 point	+	...
5. Foreign worker.	-1 point	+	...
	Score	=	

Score	-6	-5	-4	-3	-2	-1	0	1
Risk (%)	2.04	3.54	6.05	10.17	16.60	25.93	38.09	51.96

FIGURE 1. **An example risk score model to assess bad credit.**

points of the corresponding conditions (features) related to a person. This score is then checked against the risk score table underneath to determine the risk of this particular patient having a stroke. Thanks to its simplicity, interpretability and straightforward implementation, risk scores have been widely adopted by practitioners [8], [9]. Another advantage of risk scores is that small integer numbers allow users to conveniently extract decision rules by listing conditions when the score exceeds the threshold.

B. CHALLENGES OF CREATING RISK SCORE MODELS

Developing effective risk scores is not a trivial task, and currently most risk scores are generated in an *ad hoc* manner. Heuristics for building risk scores include manually selecting features, rounding coefficients obtained by logistic regression, or using a panel of experts [8]. Xie *et al.* [11] adopted different machine learning algorithms to determine important features but heuristics are used for rounding coefficients obtained by logistic regression. When risk scores are obtained with heuristics, there is no guarantee that the obtained risk scores are optimal or nearly optimal. To address this problem, Ustun and Rudin have formulated a mixed-integer nonlinear program (MINLP) for the optimal risk score problem (ORSP) [12] and proposed a new solution method RiskSLIM. The goal of ORSP and RiskSLIM is to automatically build a risk score model that is sparse, has small integer coefficients, and performs well in terms of risk calibration and sparsity. The empirical results show that RiskSLIM performs very well as compared to traditional risk score methods. Spangler *et al.* [13] attempted to use extreme gradient boosting (XGBoost) algorithm for building risk scores and achieved good prediction performance. However, interpretability remains a limitation of their algorithm.

Solving ORSP is a key factor for producing effective risk scores. However, as ORSP is an NP-hard problem, it is difficult, if not impossible, to solve this problem when the number of features increases and the number of instances is high. The experiments in [12] show that the running time of RiskSLIM increases dramatically and the optimality gaps are prohibitively high as the number of features or the number of instances increase. To overcome this scalability limitation and to solve ORSP, we investigate the swarm intelligence approach, particularly particle swarm optimisation (PSO). There are several reasons for choosing PSO for solving this challenging problem. First, PSO is an effective and efficient meta-heuristic for a wide range of optimisation problems from designing artificial neural networks [14] to evolving cognitive diagnostic models [15]. Also, PSO is very easy to implement and can be deployed independently without the need of commercial solvers. Second, PSO has been applied to feature selection, an ML task similar to ORSP, and has shown a highly competitive performance [16]. Third, as a popular algorithm in meta-heuristics and evolutionary computation (EC) communities, many advances (e.g. representation, fitness function) have been made to enhance PSO capabilities in order to solve challenging and constrained optimisation problems. Finally, as a population-based optimisation algorithm, PSO can easily take advantage of parallel computing to speed up the learning process. This is an important property which helps PSO cope with big high-dimensional data and provides decision makers with accurate and interpretable models within a short time frame.

C. CONTRIBUTIONS AND STRUCTURE OF THE PAPER

The main contributions of this paper are:

- We propose a new formulation for solving ORSP that is more efficient than RiskSLIM.
- We develop the first PSO-based algorithm called PSORisk to build risk score models which are highly interpretable classifiers.
- A new representation for PSO is proposed to evolve risk score models.
- A new penalty-based fitness function is developed to encourage models with high accuracy and small sizes.
- A new learning mechanism is proposed for PSO to have an effective search towards small and accurate models.
- A new local search heuristic is introduced to help PSO search more efficiently.
- Extensive experiments and comparisons with linear and non-linear IML classification algorithms have shown the competitiveness of PSORisk.
- Analyses of accuracy, interpretability, and scalability are presented to show the efficacy of the proposed algorithm.

The rest of this paper is organised as follows. The next section provides background and related work to PSORisk.

The detailed descriptions of the proposed ORSP formulation and PSORisk are presented in Section III. Section IV shows the experimental design and datasets used to validate the performance PSORisk. The results and analyses are shown in Section V and Section VI. Conclusions and future directions are discussed in Section VII.

II. BACKGROUND

This section presents risk score optimisation and interpretable machine learning related research work. It also introduces the original PSO algorithm.

A. RISK SCORE OPTIMISATION AND RELATED WORK

Risk scores are simple linear classification models used for risk assessment. The goal of risk score models is to facilitate the risk calculation by only applying addition, subtraction, and multiplication operators to a few small numbers [12]. Such models allow users to quickly make a prediction

without extensive training or even a computer. These models clearly have advantages when applied to real-world scenarios. Another attractive point of risk scores is their interpretability. With risk score models, users can easily and quickly identify the key risk factors and quantify their contributions to the risk prediction. This property is very useful given that there is an increasing pressure for AI systems to provide explanations, especially for critical applications (e.g. finance, healthcare).

However, risk score optimisation has a number of challenges. To be applicable, risk scores must be rank-accurate (i.e. high AUC to ensure reliable prediction), risk-calibrated (i.e. add support for probability prediction) and sparse (i.e. with a few selected features). In addition, the risk scores have to satisfy operational constraints such as the model size and domain-knowledge prediction constraints (e.g. if the model uses *Hypertension*, then it should also use $Age \geq 75$). Because of the lack of optimisation tools, these requirements have been handled in an ad-hoc manner, which requires multiple heuristics and experts' experience [8]. These manual interventions may lead to poor prediction performance of the risk score models and fail to realise their true potentials.

A number of recent studies have focused on developing automated and systematic approaches to determining optimal risk scores. Souillard-Mandar *et al.* [8] pointed out that the lack of transparency in ML models may restrict the adoption of these models in clinical use. They developed an algorithm called Supersparse Linear Interpretable Models (SLIM) which aims at building optimal risk scores with integer coefficients (or points as shown in Fig. I) and constraints on the range of coefficients. Their experiments showed that SLIM could produce models that are more robust, more interpretable and more accurate than some widely used scoring systems, and require less computation on the part of the clinicians to compute the result. The full descriptions of SLIM are presented in [9] where several datasets from different application domains are used to prove its effectiveness. A comprehensive comparison in terms of model sizes and classification performance between SLIM and other ML algorithms are also presented to demonstrate the usefulness of SLIM. Ustun *et al.* [17] used SLIM to examine the values of medical information on the prediction of sleep apnea. Another study [18] investigated the use of SLIM for recidivism prediction and showed that SLIM model outperforms traditional interpretable ML algorithms.

In RiskSLIM [12], Ustun *et al.* further extended the SLIM algorithm for risk assessment by optimising the logistic loss function shown in Equation (1), rather than the simple accuracy. Given a dataset $\mathbb{D} = \{(\boldsymbol{x}_i, y_i)\}_i^N$ where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{id})^\top \in \mathbb{R}^d$ is the feature vector and $y_i \in \{-1, +1\}$ is the label, RiskSLIM [12] formulates the ORSP as follows:

$$\begin{aligned} &\min_{\boldsymbol{\lambda}} \; l(\boldsymbol{\lambda}') + C_0||\boldsymbol{\lambda}'||_0 \\ &s.t. \; \boldsymbol{\lambda}' \in \mathcal{L} \end{aligned} \tag{1}$$

where $\boldsymbol{\lambda}' \in \mathbb{R}^{d+1}$ is a coefficient vector to be optimised, $l(\boldsymbol{\lambda}') = \frac{1}{N}\sum_{i=1}^{N} \log(1 + \exp(-\langle \boldsymbol{\lambda}', y_i \boldsymbol{x}_i' \rangle))$ is the logistic loss function to achieve high AUC and risk calibration, $\boldsymbol{x}_i' = (1, x_{i1}, \ldots, x_{id})^\top$, and L0-*norm* $||\boldsymbol{\lambda}'||_0 = \sum_{j=0}^{d} 1[\lambda_j' \neq 0]$ is for sparsity. The parameter C_0 is to control the trade-off between loss function and sparsity. $\mathcal{L}$ represents the feasible domain of the risk scores, e.g. $\mathcal{L} = \{-5, \ldots, 5\}^d$. The predicted risk for an example i for a positive class is:

$$\Pr(y_i = +1|\boldsymbol{x}_i') = \frac{1}{(1 + \exp(-\langle \boldsymbol{\lambda}', \boldsymbol{x}_i' \rangle))} \tag{2}$$

In [12], Ustun *et al.* also provided the proof of optimality and proposed a cutting plane algorithm with a commercial optimisation solver (CPLEX) to help RiskSLIM cope with large datasets. Although the results are promising, the scalability issues of their proposed algorithm can be observed when the number of instances and the number of features increase.

Recently, Spangler *et al.* [13] applied extreme gradient boosting (XGBoost) algorithm to risk scores. This approach obtained good prediction performance. However, its interpretability remains limited.

B. RELATED WORK IN INTERPRETABLE MACHINE LEARNING

Interpretability in ML has recently gained an increasing amount of attention with the creation of many methods to provide interpretable models. This section briefly reviews representative ML approaches to interpretability. Readers are referred to more comprehensive surveys such as [19] with a new taxonomy for interpretable ML, or [20] with a new framework for users to evaluate interpretable methods based on their predictive accuracy, descriptive accuracy and relevancy.

Two main approaches to IML are self-explanatory models and post hoc analysis [19], [20]. Representatives of the former are linear models, decision trees, and rule-based models. Linear regression and logistic regression [6] rely on the weighted summation of feature inputs to provide the prediction. Because of the linear relationship, it is fairly easy to interpret the learned models when the number of features is small. As the number of features increase, regularised versions of these techniques are needed to restrict the complexity of the final model. Lasso is one of the most popular techniques to train sparse linear models, i.e., performing feature selection to improve generalisation and interpretability of the trained models. Recently, more advanced techniques aiming at controlling the complexity and enhancing the interpretability at feature-level were also proposed. For example, as introduced above, RiskSLIM [12] was developed for the optimised risk score problem in which the number of features and the coefficients of features are explicitly constrained.

Decision tree (DT) [6] is another popular interpretable ML technique. DT can capture the interactions between features better as compared to linear models. DT is reasonably easy to interpret and has a number of useful ways to measure feature

importance or to deal with missing data. However, DT often has trouble with linear relationships or continuous variables. In those cases, DT can be unstable or produce prediction which are not intuitive.

If-Then rules are one of the most acceptable solutions if the learned rules provide a reasonable accuracy. Most rule learning algorithms depend on some heuristics to extract the frequent patterns or effective rules. Bayesian Rule List (BRL) [7] is a recently proposed algorithm that shows promising results. In the first stage, BRL uses a frequent set mining technique to identify patterns with strong support from the dataset. Then, BRL uses the selected patterns to build the If-Then rules based on Bayesian statistics. Apart from their interpretability, If-Then rules can provide fast prediction and easy implementation. However, similar to DT, learned rules also have difficulty dealing with linear relationships. Tsetlin machine [21], a recently proposed algorithm to solve complex pattern recognition problems with propositional formulas, is also a potential approach to building interpretable classifiers. To handle classification tasks, Tsetlin machine produces a set of conjunctive clauses and performs classification via majority voting. In Tsetlin, learning is done through a reinforcement mechanism to strengthen frequent patterns and to improve the discrimination power of the produced patterns.

Since black-box ML models usually have a high predictive performance, a recent model-agnostic approach to IML called explainers have been proposed providing some level of model interpretation. This approach is also known as post hoc analysis in [20]. Example methods are local interpretable model-agnostic explanations (LIME) [22] and SHapley Additive exPlanations (SHAP) [23], which explain complex models such as XGBoost and deep neural network (DNN). Following this approach, SkopeRules [24] combines reinforcement learning and graph search to generate IF-THEN rules for black-box ML algorithms.

C. PARTICLE SWARM OPTIMISATION

PSORisk is developed based on Particle Swarm Optimisation (PSO) [25] which is a population or swarm based algorithm. Each individual or particle in a swarm represents a candidate solution in its *position*. By exchanging the best solution/position each particle has found so far (***pbest***), particles know the global best solution (***gbest***) and move towards these fruitful areas by adjusting their *velocity* which determines their flying direction and magnitude.

Given d as the problem dimension or the number of features, a particle's velocity and position are d-dimension vectors of numerical values. The position and velocity of particle k are updated using Equations (3) and (4), respectively.

$$p_{kj}^{t+1} = p_{kj}^{t} + v_{kj}^{t+1} \tag{3}$$

$$v_{kj}^{t+1} = w * v_{kj}^{t} + c_1 * r_1 * (pbest_{kj}^{t} - p_{kj}^{t}) + c_2 * r_2 * (gbest_{j}^{t} - p_{kj}^{t}) \tag{4}$$

where p_{kj}^{t} and v_{kj}^{t} are the position and velocity of particle k in dimension j at time t, respectively. w is the inertia weight representing the moving momentum of the particles. $pbest_{kj}$ and $gbest_j$ are $\boldsymbol{pbest}_k$ and ***gbest*** position in dimension j. c_1 and c_2 are acceleration constants. r_1 and r_2 are random values in [0, 1] anew at time t.

Mimicking the social behaviours of bird flocking, PSO is well-known with its global searchability. Compared to other population-based algorithms, PSO requires less computation time and has a faster convergence speed. It has been successfully applied to many optimisation problems such as feature selection [26], rule induction [27] and neural network architecture design [28]. However, PSO has never been used in solving risk score problems.

In summary, risk score is a simple, transparent, and highly interpretable model that is widely adopted by practitioners. Determining optimal risk scores is a promising approach to achieve interpretability for machine learning. However, previous studies have shown that automatically building risk score models is a challenge. Therefore, this paper proposes a novel PSO approach to solving the ORSP problem.

III. THE PROPOSED APPROACH

The main motivation for developing PSORisk is to provide not only accurate but also scalable and highly parallelisable algorithm to solve ORSP. To achieve this goal, we propose a new formulation for ORSP and a new PSO-based algorithm to solve this formulation.

A. A NEW ORSP FORMULATION

To overcome the scalability issues of RiskSLIM, we proposed a new formulation for ORSP as follows.

$$\min_{\boldsymbol{\lambda}} \; l_c(\boldsymbol{\lambda}) + Cg(\boldsymbol{\lambda}) \quad s.t. \; \boldsymbol{\lambda} \in \mathcal{L} \tag{5}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^d$ is a coefficient vector to be optimised, $l_c(\boldsymbol{\lambda}) = \frac{1}{N}\sum_{i=1}^{N} \log\left(1 + \exp\left(-y_i\left(\alpha_0 + \alpha_1 \langle \boldsymbol{\lambda}, \boldsymbol{x}_i\rangle\right)\right)\right)$ is the logistic loss function, $g(\boldsymbol{\lambda})$ is the regularisation term for sparsity, and C is a parameter to control the trade-off between loss and sparsity. $\mathcal{L}$ is the feasible domain of the risk scores.

The main difference between $l(\boldsymbol{\lambda}')$ in Eq. (1) and $l_c(\boldsymbol{\lambda})$ in Eq. (5) is the role of the coefficient vector. In the RiskSLIM formulation, the total score $\langle \boldsymbol{\lambda}', \boldsymbol{x}_i' \rangle$ is directly used to calculate the probability. Meanwhile, in PSORisk, *the total score* $\langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle$ is a new feature constructed based on a linear combination of original features with rounded coefficients. In this case, the predicted risk is calculated as:

$$\Pr(y_i = +1|\boldsymbol{x}_i) = \frac{1}{1 + \exp\left(-\left(\alpha_0 + \alpha_1 \langle \boldsymbol{\lambda}, \boldsymbol{x}_i\rangle\right)\right)} \tag{6}$$

With the above formulation, the ORSP can be broken into two phases: (1) selecting features and determining the coefficient vector $\boldsymbol{\lambda}$, and (2) calibrating the probability. The first phase is handled by a new PSO-based algorithm to determine selected features and their corresponding small integer coefficients to construct the total score $\langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle$. The second phase is handled by logistic regression (LR) to provide calibrated probabilities via determining the best suit coefficients α_0

Par. position	3.5	-1.2	4.7	-0.2	-2.6	0.4	-0.1	4.1	-0.3	2.4
Solution	4	-1	5	0	-3	0	0	4	0	2

FIGURE 2. PSO representation for ORSP.

and α_1. By breaking the solving process into two phases, we can significantly reduce the computational complexity encountered by RiskSLIM [12]. The remainder of this section describes the new PSO-based algorithm in detail.

B. PSORisk REPRESENTATION AND SYSTEM OVERVIEW

As discussed, PSO is used to select features and determine the coefficient for each feature. Therefore, a candidate solution which will be encoded in the position of a particle is a coefficient vector assigned to selected features in the dataset. In addition, to make the model more interpretable and easily applied, a risk score solution should use a limited number of features and the coefficients assigned to these features need to be small integers.

To cope with these requirements, the position of a particle k is a d-dimension vector $\boldsymbol{p}_k = [p_{k1}, \ldots, p_{kd}]$ of real numbers given d as the number of features of the dataset. Each dimension corresponding to a feature encodes a coefficient which is bounded to $[-B, \ldots, B]$ where B is a small positive integer number defined by users. To obtain the coefficient vector $\boldsymbol{\lambda}_k$, PSORisk simply rounds the position values to the nearest integer numbers. Although there are a number of techniques proposed in the literature [29] for discrete solutions, we choose the rounding technique because of their simplicity and straightforward implementation. A feature is considered as selected if its rounded coefficient is not zero. This representation decides how each feature contributes to the total score $\langle \boldsymbol{\lambda}_k, \boldsymbol{x}_i \rangle$, positively or negatively.

Fig. 2 shows an example of a particle position with $d = 10$ and $B = 5$. The position values corresponding to all features are rounded to form the solution, i.e. coefficient vector $\boldsymbol{\lambda}_k = [4, -1, 5, 0, -3, 0, 0, 4, 0, 2]$. The shaded features are not selected because their scores are zeros. With this decoding scheme, we ensure that the coefficient vector always satisfies the small integer constraints in ORSP, which is very expensive to handle with RiskSLIM.

Fig. 3 presents the overview of PSORisk, which starts by randomly initialising the positions of all particles in the swarm. The evolutionary process is a loop in which particles are evaluated (using a new fitness function proposed in Section III-C), refined (using a new local search proposed in Section III-E) and updated (using the proposed learning mechanism in Section III-D). When the terminating condition is met, the best feasible solution is returned and used to construct the total score by linearly combining the selected features with their rounded coefficients as described in Section III-A. The total score is then fed into a logistic regression (LR) algorithm to build the classifier or risk score model.

C. FITNESS FUNCTION

The goal of PSORisk is to obtain an accurate model that is easy to interpret. The risk score model is more difficult

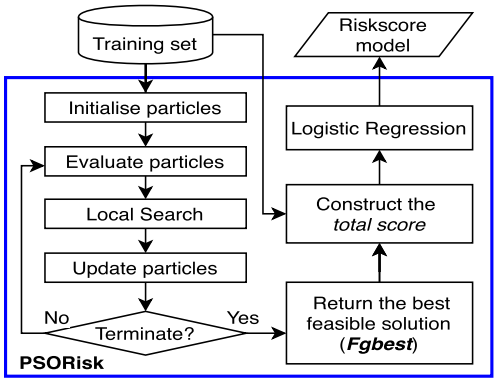

FIGURE 3. Overview of PSORisk.

to interpret if more features are included, similar to other ML algorithms. Therefore, the number of features should be below a certain threshold, i.e. model size M that is pre-defined by the users based on the practical requirements. However, if this constraint is hard coded in the particle representation, PSO may not conduct a smooth search in its fitness landscape for better solutions. Therefore, during the evolutionary process, a particle is allowed to have an arbitrary number of selected features, i.e. it can be either a feasible solution or an infeasible solution. A solution is considered *feasible* if and only if the number of selected features is smaller than or equal to M.

To guide particles moving towards feasible solutions, PSORisk uses a measure called *Oversize* as a penalty to degrade the fitness of those particles that select more features than required. As shown in Equation (7), $Oversize(\boldsymbol{\lambda}_k)$ is a positive number showing how many extra features are selected.

$$OverSize(\boldsymbol{\lambda}_k) = \begin{cases} 0, & \text{if } |FS(\boldsymbol{\lambda}_k)| < M \\ |FS(\boldsymbol{\lambda}_k)| - M, & \text{otherwise} \end{cases} \quad (7)$$

where $FS(\boldsymbol{\lambda}_k)$ is the set of active features determined by the score vector $\boldsymbol{\lambda}_k$ obtained from the above decoding step. In the example presented in Fig. 2, $FS(\boldsymbol{\lambda}_k) = \{1, 2, 3, 5, 8, 10\}$ and $|FS(\boldsymbol{\lambda}_k)| = 6$. If the model size $M = 5$, the penalty $Oversize(\boldsymbol{\lambda}_k) = 1$.

To evaluate the classification performance of a candidate solution $\boldsymbol{\lambda}_k$, PSORisk applies it to the training set to construct the total score as a linear combination of selected features weighted by the corresponding coefficients as described in Section III-A. The total score is combined with the class labels to form a new training set $\mathbb{D}' = \{(\langle \boldsymbol{\lambda}_k, \boldsymbol{x}_i \rangle, y_i)\}_{i=1}^{N}$. LR uses $\mathbb{D}'$ to build the classifier and determine the optimal coefficients α_0 and α_1 as shown in equation (6). A 5-fold cross-validation (CV) is applied to calculate the average AUC, which is used to calculate the fitness of a particle. Although LR is run to evaluate each particle, this evaluation process is inexpensive since $\mathbb{D}'$ has only one feature. Equation (8) shows how PSORisk calculates the fitness of particle k.

$$Fitness(\boldsymbol{p}_k) = \text{AUC}(\boldsymbol{\lambda}_k) - Cg(\boldsymbol{\lambda}_k) \quad (8)$$

where AUC($\boldsymbol{\lambda}_k$) is the AUC obtained by the trained LR model and $g(\boldsymbol{\lambda}_k)$ is the regularisation term based on the model size M.

To heavily penalise solutions that seriously violate the model size constraints, we set $g(\boldsymbol{\lambda}_k) = Oversize(\boldsymbol{\lambda}_k)^2$. As shown in Equation (8), a better solution is the one with higher fitness. Note that PSORisk directly uses AUC to measure the classification performance instead of the logistic loss function as AUC is a preferable metric for assessing the quality of classifiers. The logistic loss function is only used by LR algorithm to optimise α_0 and α_1 in Equation (6).

D. A NEW PSO LEARNING MECHANISM

After evaluation, each particle will update its best position (***pbest***) which is the fittest position it has explored so far. Then all particles communicate with each other using the fully connected topology to learn the global best solution (***gbest***), i.e., the best solution found by the whole swarm.

Since the fitness function shown in Equation (8) comprises two components to maximise the accuracy and minimise the model size, moving towards ***gbest*** does not guarantee to find feasible solutions. To cope with the infeasibility, PSORisk also maintains a global best feasible solution (***Fgbest***) which is the feasible particle with the best AUC.

To guide particles to move smoothly towards the best solution in general (***gbest***) and the best feasible solution (***Fgbest***), ***Fgbest*** is introduced into PSORisk's velocity update as shown in Equation (9). In this way, PSORisk simultaneously improves the quality of feasible solutions and the feasibility of infeasible solutions. To keep the updating equation compact, we also remove the inertia term in this formula after conducting some pilot experiments which suggested that this term is not useful for PSORisk.

As shown in Equation (9), PSORisk velocity updating formula incorporates information from not only particle's self-cognition (***pbest***$_k$) but also from knowledge of the whole swarm (***gbest*** and ***Fgbest***).

$$\begin{aligned} v_{kj}^{t+1} &= c_p * r_1 * (pbest_{kj}^t - p_{kj}^t) \\ &\quad + c_g * r_2 * (gbest_j^t - p_{kj}^t) \\ &\quad + c_f * r_3 * (Fgbest_j^t - p_{kj}^t) \end{aligned} \tag{9}$$

where p_{kj}^{t+1} and v_{kj}^{t+1} are the position and velocity of particle k at dimension j and time $t+1$. c_p, c_g, c_f are acceleration constants, and r_1, r_2, and r_3 are random values which are reset anew in each time step. It should be noted that ***gbest*** can be the same as ***Fgbest*** if ***gbest*** is also feasible, which makes PSORisk behave similarly to the traditional PSO. After the velocity is calculated, PSORisk uses the traditional Equation (3) for updating positions.

E. A NEW LOCAL SEARCH HEURISTIC

To increase the chance of reaching and improving $Fgbest$, a new local search is proposed in PSORisk to refine infeasible solutions to become feasible and more accurate ones. This helps PSORisk quickly identify good and feasible solutions to improve ***Fgbest***.

Algorithm 1 New Local Search Heuristic

Input : current population
Output: refined particles and feasible global best

begin
  $IP \leftarrow \{k : |FS(\boldsymbol{pbest}_k)| > M\}_{k=1}^{Popsize}$;
  $\mathcal{N} \leftarrow$ select $r_{\text{IF}}\% \times Popsize$ in IP with the highest AUCs ;
  for $\boldsymbol{pbest}_k \in \mathcal{N}$ **do**
    nb $\leftarrow$ remove lowest SU features from $\boldsymbol{pbest}_k$ until a feasible solution is found;
    Evaluate ***nb*** using equation (8);
    if ***nb*** *is better than* $\boldsymbol{pbest}_k$ **then**
      $\boldsymbol{p}_k \leftarrow$ ***nb***;
      $\boldsymbol{pbest}_k \leftarrow$ ***nb***;
    end
    if ***nb*** *is better than* ***Fgbest*** **then**
      Fgbest $\leftarrow$ ***nb***;
    end
  end
  Return refined particles $k \in \mathcal{N}$ and ***Fgbest***;
end

Although local search can be applied on all infeasible particles, applying it too intensively may reduce the diversity of the population, which may have negative impacts on the final performance. Therefore, in each iteration, the proposed local search is only applied to an $r_{\text{IF}}\%$ of infeasible ***pbest*** that have the highest AUC.

Specifically, this local search aims to generate a feasible and better neighbouring solution than the given ***pbest***. This is done by removing a number of features to make it fit the model size. For the local search to obtain better solutions, it is important to choose the right features to remove. A simple heuristic is designed to remove the most irrelevant features from the set of active features. Symmetric uncertainty (SU) [30] is used to measure feature relevance. SU is a normalised version of information gain (IG) [31]. It can be used to measure the correlation between two features. A feature is irrelevant to the problem if it weakly correlates to the class; in other words, it has a small SU value with the class label. Equation (10) shows how SU between two features X and Y is calculated:

$$SU(X, Y) = \frac{IG(X|Y)}{H(X) + H(Y)} \tag{10}$$

$$IG(X|Y) = H(X) - H(X|Y) \tag{11}$$

where $H(X)$ is the entropy of X and $H(X|Y)$ is the conditional entropy of X given Y. The value of $SU(X, Y)$ is in the range [0, 1]. The higher the value of SU, the higher the correlation between variables X and Y.

The pseudo-code of this local search is shown in Algorithm 1. First, the top $r_{\text{IF}}\%$ infeasible ***pbest***s will be selected from the population. For each selected $\boldsymbol{pbest}_k$, a new neighbouring solution (***nb***) is created by removing the lowest

Algorithm 2 PSORisk

Input : Training data
Output: Best found solution

```
1  begin
2      Initialise PSO population;
3      for iter = 1 to max_iterations do
4          for k = 1 to Popsize do
5              Calculate Fitness(p_k) using equation (8);
6              if Fitness(p_k) > Fitness(pbest_k) then
7                  Update pbest_k;
8              end
9              if Fitness(p_k) > Fitness(gbest) then
10                 Update gbest);
11             end
12             if p_k is feasible &
               Fitness(p_k) > Fitness(Fgbest) then
13                 Update Fgbest;
14             end
15         end
16         Apply local search in Algorithm 1 ;
17         for k = 1 to Popsize do
18             Update velocity and position of particle k
               using equation (9) and (3);
19         end
20     end
21     TotalScore ← Calculate total score from training set
       and Fgbest (See Section III-A) ;
22     Model ← Train LR using TotalScore;
23     Return Model;
24 end
```

SU features from pbest to reduce the number of selected features to M (Line 5). In other words, $\boldsymbol{nb}$ is a feasible solution. $\boldsymbol{nb}$ is then evaluated (Line 6) based on the fitness evaluation described in Section III-C. If it is better than $\boldsymbol{pbest}_k$ in terms of AUC, both the current position $\boldsymbol{p}_k$ and $\boldsymbol{pbest}_k$ are updated (Lines 7-10). In addition, $\boldsymbol{Fgbest}$ is also updated if the refined solution is better (Lines 11-13).

F. PSORisk OVERALL ALGORITHM

Algorithm 2 shows the pseudo code of PSORisk. Given a training dataset, it returns the best risk score model evolved after a number of *max_iterations*. Firstly, a swarm with *Popsize* particles is initialised. The particles' positions are randomly initialised within the pre-defined boundary B; however, restricted to a small percentage of selected features (i.e. non-zero dimensions in $\boldsymbol{\lambda}_k$) in order to explore the space that has more feasible solutions. Specifically, the expected number of selected features in the initialised particles are twice the model size M. Lines 3-20 shows the main loop of PSORisk with the three main steps, evaluation (Lines 5-15), local search (Line 16) and update particles (Lines 17-19). The best feasible solution $\boldsymbol{Fgbest}$ is then used to calculate the total score for each instance in the training set. LR is then applied on the new training set to build the risk score table (similar to the example in Fig. I) to return as the final solution.

IV. EXPERIMENT DESIGN

This session presents the datasets, baseline methods and parameter setting in experiments. All the experiments are conducted by means of software simulations.

A. DATASETS

Ten binary classification datasets with varying difficulties ranging from tens to hundreds of features are used to test the performance of PSORisk. As shown in Table 1, these datasets also have a wide range of sizes with hundreds to tens of thousands of instances. These datasets are widely used in the risk prediction literature. The adult, bank, mushroom, and spambase datasets are obtained from RiskSLIM's repository [12]. For other datasets, there are some continuous features such as monthly income in the hribm dataset or the customer age in the german dataset. We apply k-bins discretiser and one-hot encoding [42] with $k = 4$ into these datasets. This is an optional step to further simplify the risk score calculation. The same transformed data is applied to all other compared methods in our experiments.

B. EXPERIMENT CONFIGURATION

1) COMPARED METHODS

To test if PSORisk has obtained our primary goal which is creating interpretable and accurate classifiers, we compare PSORisk results with six well-known IML algorithms which were introduced in Section II. The first two methods are logistic regression (LR) and linear support vector classifier (LinearSVC), which are representatives of linear classification algorithms. The remaining four algorithms are decision trees (DT) [6], Bayesian Rule List (BRL) [7], SkopeRules (SK) [24], and Tsetlin [21], which represent the non-linear counterparts.

PSORisk is compared with the six IML methods in terms of classification performance (AUC) and model size, which represent model accuracy and interpretability, respectively. In PSORisk, the model size M is a pre-defined ORSP constraint as introduced in Section III-C. Similar to PSORisk, linear classifiers such as LR and LinearSVC take the number of selected features [9] as the model size. To generate models with different model sizes for LR and LinearSVC, we apply different regularisation parameters. For the remaining four rule-based classifiers, the comparisons are not straightforward as they are fundamentally different from linear classifiers. With DT, the model size can be measured as the number of leaves in the final trees (equivalent to the number of rules if we flatten the decision tree) ***or the total number of nodes***. Different maximum numbers of leaves (ML) are set for DT to have different tree sizes. For SK, BRL, and Tsetlin, the model size is the number of rules in the returned model. In SK, different maximum depths (MD) are used to generate models with different numbers of rules. In Tsetlin, the number of rules is set to 10 to keep the model interpretable. Note that as the model sizes are measured by different approaches,

TABLE 1. Datasets used in the experiments.

Dataset	N	d	Conditions for $y_i = 1$	$Pr(y_i = 1)$	Reference
Cervical Cancer (cervical)	858	69	diagnosis is malignant	6.40%	[32]
Australian Credit Approval (australian)	690	56	credit is approved	44.50%	[33]
German Credit Risk (german)	1,000	77	customer has bad credit	30.00%	[34]
Human Resource-IBM (hribm)	1,470	121	employee attrites	16.10%	[35]
Spambase (spambase)	4,601	57	e-mail is spam	39.40%	[36]
Customer Churn (churn)	7,032	49	customer churns	26.60%	[37]
Mushroom (mushroom)	8,124	113	mushroom is poisonous	48.20%	[38]
Intensive Care Unit (icu)	11,773	53	patient is dead	10.70%	[39]
Adult (adult)	32,561	36	person in 1994 US census earns over $50,000	24.10%	[40]
Bank (bank)	41,188	57	person opens bank account after marketing call	11.30%	[41]

TABLE 2. Parameter setting.

Parameters	#Settings
Population size	100
Maximum iterations	50
c_p, c_g, c_f	1.2
Maximum velocity	2.0
Selection ratio for local search $r_{\text{IF}}\%$	25%
C (oversize weight)	0.001
B (coefficient boundary)	5
M (model size)	5 (default), 7, 10

the results are mainly used for qualitative evaluations to compare the complexity and interpretability of the trained classifiers.

2) PARAMETER SETTING

Table 2 describes parameter settings for PSORisk. The population size is set to 100, which is larger than the usual size set in common PSO applications. Since each particle or candidate solution allows to select a very small number of features (i.e. model size is 5), PSORisk can quickly lose its diversity. Therefore, a large population is used to cover more feasible solutions with the aim of increasing its diversity and avoid premature convergence. Section V-C will investigate the influence of this parameter.

Since PSORisk velocity updating mechanism has three terms with three acceleration constants c_p, c_g, and c_f, we chose a smaller value of 1.2 for each constant instead of using the commonly-used value of 1.49618. The experimental results comparing the two values showed that 1.2 obtained a better AUC. The remaining parameters are either set as popularly used value or experimentally chosen. For example, preliminary runs show that the global best solution is almost unchanged from the 45th iteration, so 50 was chosen as the maximal iteration.

3) EXPERIMENT DESIGN

To create a training and test set for each dataset, we use 10-fold CV to split the data. Each method is run on the training set comprising 9 folds and tested on the remaining fold. Since PSORisk is a stochastic algorithm, 30 independent runs with different seeds are executed on each dataset. Therefore, each dataset will have 300 results (30 runs x 10-fold CV), whose average will be reported and compared with other methods. Wilcoxon statistical test with a significance level of 0.05 is used in the comparisons to confirm if the difference between two methods is significant.

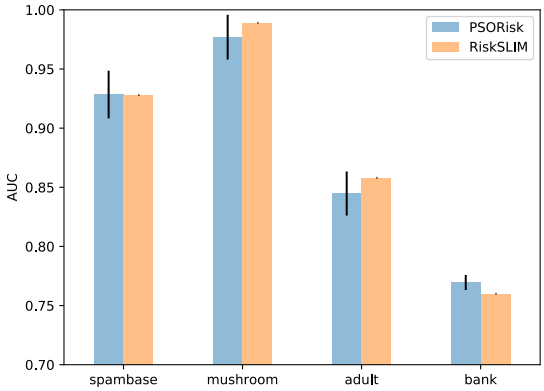

FIGURE 4. 10-fold CV AUC and model size obtained by PSORisk and linear classifiers.

Experiments were run on PC with the following configurations: CPU Intel Core i5-9400 @ 2.9GHz, RAM 8GB, operating system Windows 10, python 3.6. All the experiments are run on a single CPU core except the one in Section VI-B where the fitness evaluations component of PSORisk is run parallel on multiple CPU cores to examine its efficiency.

V. RESULTS

To show the effectiveness of PSORisk in building classifiers with a high classification performance and a desirable model size, this section compares the AUC and size of the models obtained by PSORisk with linear classifiers (Section V-A) and non-linear or rule-based classifiers (Section V-B). It also analyses the effect of population size (Section V-C) and model size (Section V-D) on PSORisk performance.

A. COMPARISON WITH LINEAR CLASSIFIERS

Fig. 4 shows the average test AUC obtained by PSORisk in 30 runs versus the AUC obtained by LR, and LinearSVC with different model sizes. An ideal classifier should be the one with the highest AUC and the smallest model size (i.e. using the smallest number of features). This means that the closer the result to the top left corner, the better the classifier. The figure showed that PSORisk obtained a closer result to the ideal classifier than LR and LinearSVC on eight out of ten datasets with exceptionally higher AUC on adult and spambase. On the remaining two datasets, PSORisk

obtained a slightly worse result on german with model size of 5, and a similar result when its model size increases to 10. On hribm, PSORisk has a worse performance than LR while obtaining a significantly higher AUC than LinearSVC with small model sizes.

The results on majority of datasets show that LR and LinearSVC require more features to achieve a similar classification performance as PSORisk. This demonstrates the effectiveness of PSORisk in selecting a smaller while more relevant subset of features. Given that PSORisk applies much stricter constraints on coefficients (an integer value in the range of $[-B..B]$) and model size than LR and LinearSVC, the results also indicate that PSORisk effectively selects features and optimises the corresponding coefficients. These constraints also make PSORisk models more interpretable than those generated by LR and LinearSVC. Section VI presents a closer look at PSORisk models.

Since the ten datasets are presented in the ascending order of the dataset sizes, Fig. 4 also shows an interesting pattern that the larger the dataset, the better the performance of PSORisk. This reveals the potential of PSORisk in big data applications.

Figure 5 compares the results of PSORisk with the reported results of RiskSLIM [12] on four common datasets, namely adult, bank, mushroom and spambase. PSORisk was run on these four datasets with 5-fold cross validation and in 10 minutes to have the same experiment setting as RiskSLIM. The minimum and maximum AUC obtained by PSORisk are also plotted using the black bar. As can be seen from the figure, PSORisk can find slightly more accurate solution than RiskSLIM, and its average AUC is very close to RiskSLIM.

In general, the results demonstrate the effectiveness of PSORisk as a classification algorithm. Even when we ignore the interpretability, PSORisk still shows good performances across most of the datasets with very effective feature selection abilities. It can therefore be considered as a good addition to the family of linear classification algorithms.

B. COMPARISON WITH RULE-BASED CLASSIFIERS

Table 3 and 4 compare the results of PSORisk with $M = 5$ versus DT, SK, BRL, and Tsetlin. Column **T** presents the results of the Wilcoxon statistical test comparing two methods. A '+' means the result of the method in the corresponding row is significantly better than the average performance of PSORisk and vice versa. This means that the more '−', the better the performance of PSORisk in terms of accuracy. The last column, **MS**, presents the number of total nodes in the created tree by DT and the number of rules in the generated model by SK and BRL, and Tsetlin.

As can be seen from Tables 3 and 4, PSORisk achieved a significantly better average AUC than DT with all the model sizes on almost all datasets. Only on australian, DT obtained a significantly better AUC of 0.9073 with 9 nodes. However, the best PSORisk model of size 5 on this dataset still has a better AUC of 0.9099.

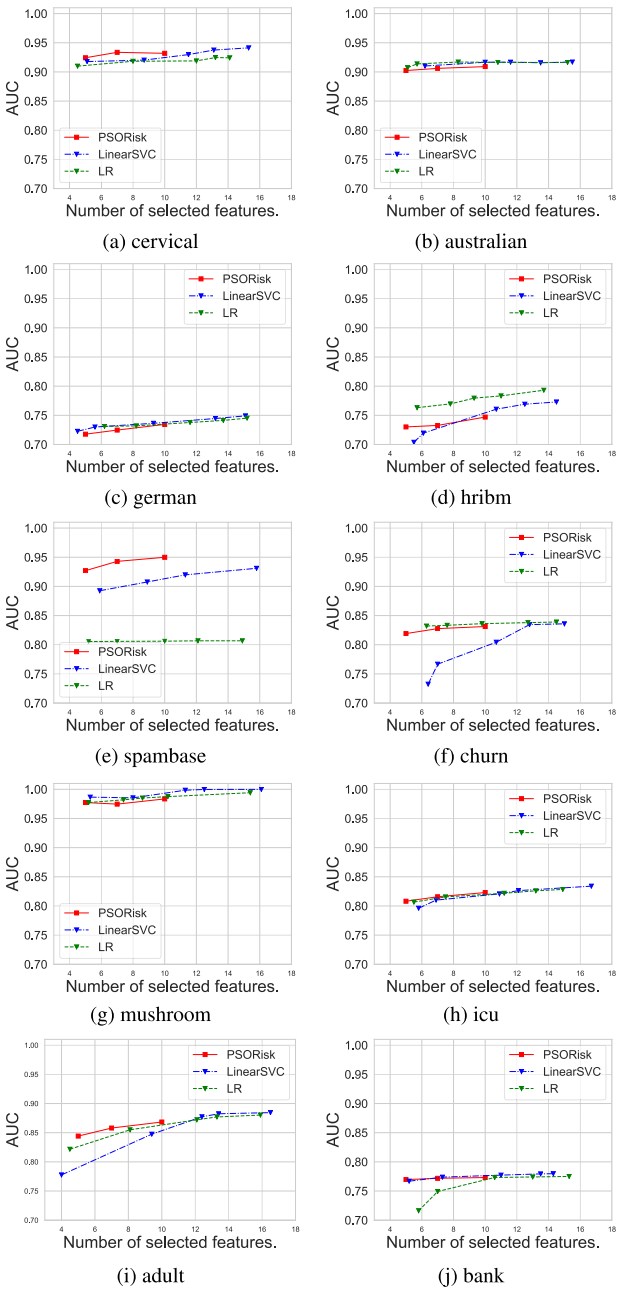

FIGURE 5. 5-fold CV AUC obtained PSORisk and RiskSLIM.

PSORisk outperformed SK with both max depths of 2 and 3 on five datasets, namely hribm, churn, icu, adult and bank with up to 0.3155 difference in average AUC on the adult dataset. For the remaining five datasets, SK with one of the max depths achieved a better AUC than PSORisk with up to 0.0143 difference in the average AUC. However, the average number of rules in these SK models can be quite large, up to 45 rules in german. We also note that SK model sizes change dramatically when the max depth (MD) changes from 2 to 3.

Among all the compared methods, Tsetlin obtained the worst performance on all datasets with the worst AUC of 0.5 on spambase. The best result obtained by Tsetlin on

TABLE 3. 10-fold CV AUC and model size/complexity (MS) obtained by PSORisk and rule-based classifiers.

Dataset	Method	Best	Avg±Std	T	MS
cervical	DT (ML = 3)	0.9159		–	5
	DT (ML = 4)	0.9155		–	7
	DT (ML = 5)	0.9166		–	9
	SK (MD = 2)	0.9170		–	30.4
	SK (MD = 3)	0.9350		+	127.8
	BRL	0.9186		–	3.6
	Tsetlin	0.8129		–	10
	PSORisk	0.9442	0.9243 ± 0.0120		5
australian	DT (ML = 3)	0.9050		+	5
	DT (ML = 4)	0.9049		+	7
	DT (ML = 5)	0.9073		+	9
	SK (MD = 2)	0.9109		+	3
	SK (MD = 3)	0.8886		–	40.5
	BRL	0.9040		=	4.4
	Tsetlin	0.7963		–	10
	PSORisk	0.9099	0.9024 ± 0.0042		5
german	DT (ML = 3)	0.7112		–	5
	DT (ML = 4)	0.7130		–	7
	DT (ML = 5)	0.7140		–	9
	SK (MD = 2)	0.6784		–	4.9
	SK (MD = 3)	0.7232		+	45.3
	BRL	0.7073		–	4.8
	Tsetlin	0.5898		–	10
	PSORisk	0.7399	0.7177 ± 0.0118		5
hribm	DT (ML = 3)	0.6594		–	5
	DT (ML = 4)	0.6786		–	7
	DT (ML = 5)	0.6825		–	9
	SK (MD = 2)	0.6457		–	1.5
	SK (MD = 3)	0.6796		–	24.2
	BRL	0.7105		–	5.7
	Tsetlin	0.5315		–	10
	PSORisk	0.7632	0.7301 ±0.0150		5
spambase	DT (ML = 3)	0.8219		–	5
	DT (ML = 4)	0.8722		–	7
	DT (ML = 5)	0.8804		–	9
	SK (MD = 2)	0.9118		–	53.9
	SK (MD = 3)	0.9387		+	18.0
	BRL	0.9672		+	22.1
	Tsetlin	0.5000		–	10
	PSORisk	0.9415	0.9270 ± 0.0110		5

TABLE 4. 10-fold CV AUC and model size/complexity (MS) obtained by PSORisk and rule-based classifiers. (cont).

Dataset	Method	Best	Avg±Std	T	MS
churn	DT (ML = 3)	0.7786		–	5
	DT (ML = 4)	0.7950		–	7
	DT (ML = 5)	0.8009		–	9
	SK (MD = 2)	0.7529		–	1.7
	SK (MD = 3)	0.6890		–	2.1
	BRL	0.8239		+	11.4
	Tsetlin	0.6796		–	10
	PSORisk	0.8264	0.8191 ± 0.056		5
mushroom	DT (ML = 3)	0.9477		–	5
	DT (ML = 4)	0.9714		–	7
	DT (ML = 5)	0.9816		+	9
	SK (MD = 2)	0.9677		–	2.4
	SK (MD = 3)	0.9916		+	15.2
	BRL	0.9992		+	9.6
	Tsetlin	0.8890		–	10
	PSORisk	0.9916	0.9773 ± 0.0114		5
icu	DT (ML = 3)	0.7533		–	5
	DT (ML = 4)	0.7561		–	7
	DT (ML = 5)	0.7689		–	9
	SK (MD = 2)	0.6394		–	2.8
	SK (MD = 3)	0.6253		–	6.7
	BRL	0.8514		+	16.6
	Tsetlin	0.5596		–	10
	PSORisk	0.8204	0.8081 ±0.0081		5
adult	DT (ML = 3)	0.7766		–	5
	DT (ML = 4)	0.8126		–	7
	DT (ML = 5)	0.8273		–	9
	SK (MD = 2)	0.5287		–	0.4
	SK (MD = 3)	0.7257		–	3.0
	BRL	0.8581		+	23.5
	Tsetlin	0.7358		–	10
	PSORisk	0.8616	0.8442 ± 0.0115		5
bank	DT (ML = 3)	0.6658		–	5
	DT (ML = 4)	0.7522		–	7
	DT (ML = 5)	0.7586		–	9
	SK (MD = 2)	0.5781		–	2
	SK (MD = 3)	0.5839		–	9.4
	BRL	0.7888		+	16.8
	Tsetlin	0.5845		–	10
	PSORisk	0.7755	0.7696 ± 0.0033		5

the ten datasets is on mushroom with AUC of 0.8890, while PSORisk obtained an average AUC of 0.9773 on this dataset. Each Tsetlin model has 10 rules, each of which comprises of multiple predicates. Therefore, its models are usually bigger and more complex than PSORisk.

BRL is the most competitive algorithm among the four rule-based classifiers and able to outperform PSORisk in 6 out of 10 datasets. However, BRL is relatively computationally expensive and tends to generate classifiers with a large number of rules. In all the six cases where BRL obtained up to 0.043 higher AUC than PSORisk, the number of rules in BRL models is up to 18 rules more than PSORisk. With a larger model size such as $M = 7$ or $M = 10$ as shown in Table 5, PSORisk can easily outperform BRL. Given that risk scores can also be flexibly transformed into rules (as shown in Section VI), PSORisk is possibly a more attractive solution in real-world applications.

In general, the results show that PSORisk outperforms the compared methods in 54 out of 70 comparisons, similar on 1 and worse on the remaining 15. Among these 15 lower performing cases, the best model of PSORisk can still have a higher AUC in many cases.

C. INFLUENCE OF POPULATION SIZES

To examine how the population size has influenced the quality of obtained risk scores, we conducted additional experiments with different population sizes ranging from 30 to 90. The results of ten datasets are shown in Figs. 6 and 7. As can be seen from the figures, PSORisk benefits from a larger population size in almost all datasets. Larger populations are especially useful (i.e. significant improvements) when PSORisk deals with datasets that have a large number of features such as mushroom (113 features) and hribm (121 features). It is obvious that selecting a good set of 5 features from such a large number of combinations requires good coverage of the search space. This supports our hypothesis that a large population is needed to maintain the diversity of the population and help PSORisk improve its performance.

D. INFLUENCE OF THE MODEL SIZE

Table 5 compares the best and the average test AUC of PSORisk with model size 5, 7 and 10. The last column shows the results of Wilcoxon tests which confirm if the differences are significant. A '+' means the corresponding model size achieves a significantly better average AUC than the default model size of 5 and vice versa. The results show that

TABLE 5. Classification performance with different model sizes.

Dataset	M	Best	Avg $\pm$ Std	T
cervical	5	0.9442	0.9243 $\pm$ 0.0120	
	7	0.9591	0.9336 $\pm$ 0.0129	+
	10	0.9528	0.9319 $\pm$ 0.0125	+
australian	5	0.9099	0.9024 $\pm$ 0.0042	
	7	0.9195	0.9061 $\pm$ 0.0080	+
	10	0.9208	0.9091 $\pm$ 0.0055	+
german	5	0.7399	0.7177 $\pm$ 0.0118	
	7	0.7415	0.7247 $\pm$ 0.0096	+
	10	0.7501	0.7044 $\pm$ 0.0084	+
hribm	5	0.7632	0.7301 $\pm$ 0.0150	
	7	0.7631	0.7327 $\pm$ 0.0213	=
	10	0.7816	0.7470 $\pm$ 0.0169	+
spambase	5	0.9415	0.9270 $\pm$ 0.0110	
	7	0.9526	0.9429 $\pm$ 0.0058	+
	10	0.9571	0.9499 $\pm$ 0.0042	+
churn	5	0.8264	0.8191 $\pm$ 0.0056	
	7	0.8314	0.8276 $\pm$ 0.0026	+
	10	0.8372	0.8312 $\pm$ 0.0029	+
mushroom	5	0.9916	0.9773 $\pm$ 0.0114	
	7	0.9945	0.9746 $\pm$ 0.0157	=
	10	0.9924	0.9835 $\pm$ 0.0097	+
icu	5	0.8204	0.8081 $\pm$ 0.0081	
	7	0.8305	0.8158 $\pm$ 0.0078	+
	10	0.8337	0.8230 $\pm$ 0.0068	+
adult	5	0.8616	0.8442 $\pm$ 0.0115	
	7	0.8713	0.8580 $\pm$ 0.0104	+
	10	0.8760	0.8683 $\pm$ 0.0040	+
bank	5	0.7755	0.7696 $\pm$ 0.0033	
	7	0.7756	0.7718 $\pm$ 0.0021	+
	10	0.7779	0.7731 $\pm$ 0.0029	+

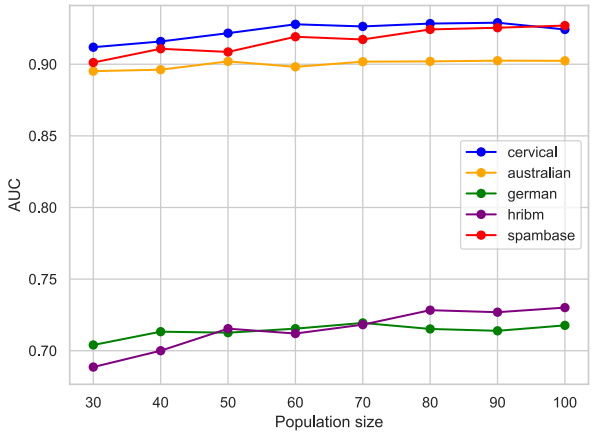

FIGURE 6. Influences of population size on adult, churn, icu, australian and bank datasets.

PSORisk consistently obtained better AUC when the model size increases. This is an important property for risk scores as we do not want to increase the model size without gaining any significant classification performance. The results show the effectiveness of PSORisk when dealing with different model size constraints. It is also noted that PSORisk is always able to find feasible solutions for all the datasets, which confirms the effectiveness of the proposed learning mechanism and local search in evolving feasible risk scores.

We also conducted extra experiments to investigate the impact of the proposed local search on the quality of obtained risk scores. The results show that PSORisk can effectively determine good solutions for all datasets when the model size $M = 5$ and there are no significant differences between PSORisk with and without the local search. However, as the model size increases to 7 and 10, PSORisk may fail to identify feasible solutions without the support of the proposed local search.

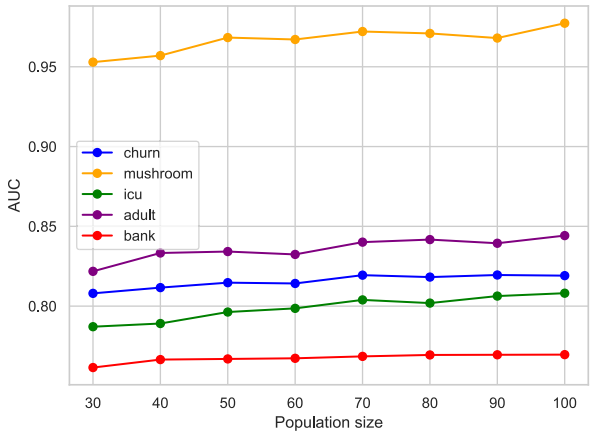

FIGURE 7. Influences of population size on spambase, cervical, german, mushroom and hribm datasets.

1. Working overtime.	2 points	
2. Job position at lowest level.	1 point	+ ...
3. Single.	1 point	+ ...
4. Working environment satisfaction at lowest level.	1 point	+ ...
5. Work-related relationship satisfaction at lowest level.	1 point	+ ...
	Score	=

Score	0	1	2	3	4	5	6
Risk (%)	3.00	6.99	15.46	30.79	51.98	72.48	86.50

FIGURE 8. PSORisk model for hribm dataset (AUC = 0.7461).

VI. FURTHER ANALYSES

The previous section has demonstrated the effectiveness of PSORisk in producing accurate and simple risk scores. In this section, we will further examine the interpretability of the obtained risk scores and PSORisk's scalability.

A. INTERPRETABILITY OF OBTAINED RISK SCORES

To see how interpretable the risk scores obtained by PSORisk are, we present here some of the risk score models generated by PSORisk for hribm and adult with $M = 5$. To make the paper concise, we only compare PSORisk models with BRL in this section as BRL obtained the most competitive performance among the baseline methods. As shown in Tables 3 and 4, PSORisk and BRL rank first on hribm and adult, respectively.

1) MODELS FOR hribm

Fig. 8 shows a PSORisk model evolved on hribm with an AUC of 0.7461. The goal of this model is to predict the risk at which an employee will leave her/his job based on their information such as age, business travel, daily rate, and department in the company. In the presented model, PSORisk selects five features among 121 features of this dataset,

IF Marital Status is Single AND Working overtime
THEN probability of leaving job is 48.4%
ELSE IF Job position is not lowest level
THEN probability of leaving job is 8.0%
ELSE IF NOT Working overtime AND Marital Status is NOT Single
THEN probability of leaving job is 9.3%
ELSE IF Age NOT in 18.0 to 29.83
THEN probability of leaving job is 26.6%
ELSE probability of leaving job is 47.8%

FIGURE 9. **BRL rules for** hribm **dataset (AUC = 0.7140).**

1. Married.	4 points	...
2. Having some capital gains.	3 points	
3. Age range 45 -59.	2 points	
4. Having bachelor or graduate degree.	2 points	+ ...
	Score	=

Score	0	2	3	4	5	6
Risk (%)	3.28	9.50	15.58	24.50	36.33	50.09

Score	7	8	9	10	11
Risk (%)	63.84	75.64	85.52	90.57	94.41

FIGURE 10. **Risk score model for the** adult **dataset (AUC = 0.8384).**

namely working overtime, job level, marital status, environment satisfaction, and relationship satisfaction. Among the five features, working overtime is the most important factor contributing to attrition risk. It has twice as many points as the remaining features. Other features selected in this model are also intuitively and highly relevant to the risk. For example, employees who are at entry level and single have low environment satisfaction, and low manager relationship satisfaction will have a higher risk of attrition.

From the PSORisk generated risk score model, simple rules can also be extracted. For example, if we want to identify employees with an attrition risk of 70% or more, i.e., employees who score at least 5 points according to the model in Fig. 8., a simple rule will be to find those who are working overtime and match any three (or more) out of the last 4 features in the model.

Figure 9 shows a BRL model generated for hribm with an AUC of 0.7140. It comprises five rules, each of which is associates with a probability of employee's leaving the job. Among the four selected features, three are in common with PSORisk model, namely working overtime, job level, and marital status. While the PSORisk model also considers satisfaction in working environment and work-related relationship, the BRL model looks at age as an additional feature.

2) adult DATASET

Fig. 10 shows a risk score model evolved by PSORisk for the adult dataset, where the goal is to determine the probability of earning 50,000 USD per annum per capita. Among 36 features of this dataset, PSORisk selects four features including married, having some capital gains, age in range 45-59, and having a bachelor or graduate degree. Among the four features, being married is identified as the most important factor with 4 points, while the age range (from 45 to 59) and having high education share the lowest point, at 2 points. Having some capital gains ranks second, at 3 points. A higher score means that the considered person has a higher probability of gaining a high income. For example, a person who has some capital gains and is married has a score of 7. This means that he or she has at least a 63% chance of earning 50,000 USD per year.

IF Work hours per week less than 40 AND NOT Married
THEN probability of earning 50.000USD annually is 1.7%
ELSE IF without having high school diploma
AND without having any capital gains
THEN probability of earning 50.000USD annually is 6.7%
ELSE IF without having high school diploma
THEN probability of earning 50.000USD annually is 29.2%
ELSE IF having some capital gains AND having no high school diploma
THEN probability of earning 50.000USD annually is 80.7%
ELSE IF having some capital gains AND get married
THEN probability of earning 50.000USD annually is 62.2%
ELSE … *(other 18 rules with the probability of earning 50.000USD annually ranging from 1.6% to 64.9%)*
ELSE probability of earning 50.000USD annually is 89.7%

FIGURE 11. **BRL rules for** adult **dataset (AUC = 0.8621).**

Using the model generated from PSORisk, we can also generate rules that reflect an individual income. For example, in order to have an over 80% chance of earning more than 50,000 USD annually (or more than 8 points according to the generated risk score model), a person must:

- be married and have some capital gains, and either
  - is from 45 to 59 years old, or
  - has a bachelor or graduate degree.

Figure 11 shows a model generated by BRL on adult data. With 24 rules showing the probability of earning 50.000USD per year of a person, this model is much bigger than the risk score model generated by PSORisk. Its interpretability therefore becomes a question, especially for rules that have a low probability such as 1.6% or 1.7%. While BRL models are as transparent as PSORisk ones, the relevance of each feature and relationship between the features in BRL rules cannot be determined directly as in PSORisk because they are sequentially combined in a global IF..ELSE rule.

To further investigate PSORisk explanatory capability, we have conducted extra experiments to compare the interpretability of our risk scores and SHAP [10]. For these experiments, we use extreme gradient boosting (XGBoost) as the black-box classification algorithm and a SHAP explainer is used to show the impact of each feature on the model outputs. Figures 12 and 13 show top 20 impactful features in SHAP interpretation for hribm and adult, respectively. As can be seen from these figures, both SHAP and PSORisk models (in Figures 8 and 10) can consistently determine the most important features that influence the outputs. For examples, married, capital gains, and ages (45 to 59) are important features of both PSORisk and SHAP (XGBoost) for the adult dataset. Similarly, overtime, job level, marital status, environment satisfaction, and relationship satisfaction are important features for the hribm dataset. However, the risk scores are very transparent in the way risk

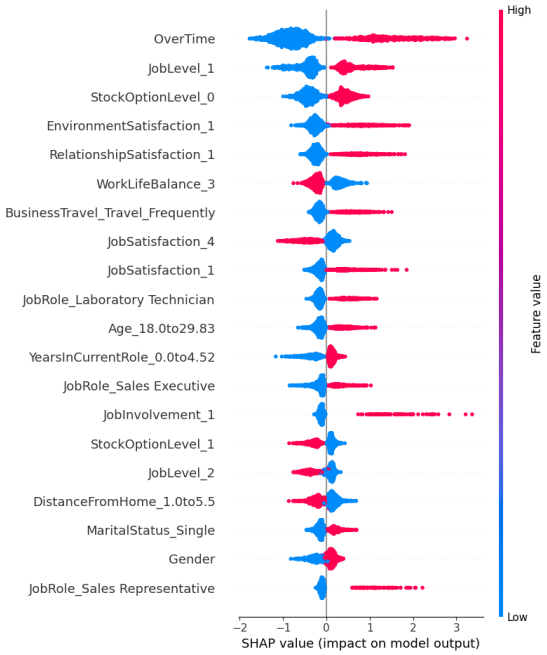

FIGURE 12. SHAP interpretation for XGBoost models on hribm.

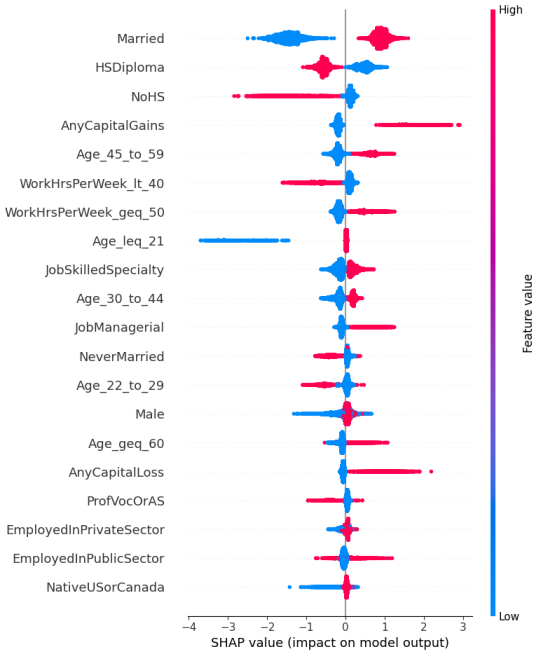

FIGURE 13. SHAP interpretation for XGBoost models on adult.

is determined while the outputs of SHAP (XGBoost) are quite noisy and the relative importance of features is hard to determine.

The two examples demonstrate that PSORisk classifiers are transparent and easier to interpret than those learnt by BRL and SHAP, which are representative IML approaches. PSORisk models can also be transformed into simple decision rules. The simplicity of the models enables anyone to use them even without a calculator.

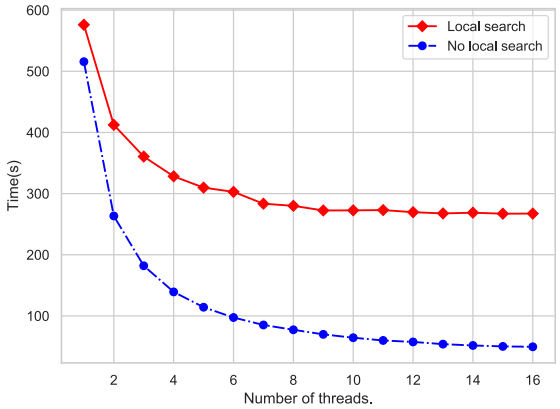

FIGURE 14. Multi-core performance of PSORisk.

B. SCALABILITY OF PSORisk

In this section, we examine the scalability of PSORisk by running a parallelised version of PSORisk that utilises multiple cores for fitness evaluations. Fig. 14 shows the running times of PSORisk with and without local search on bank. All the other parameters are set as in 2.

As can be seen from Fig. 14, the running times of PSORisk without local search reduce dramatically as more cores are utilised, reaching more than 10× reduction when 16 cores are used. PSORisk with local search has a roughly 2× speed-up. This reduction is not as extensive as the PSORisk without local search because the local search heuristic is still serially implemented in this experiment. A number of strategies can be used to enhance the efficiency in this case. The easiest approach is to parallelise the local search heuristics in Algorithm 1. Another sophisticated but more efficient solution is to develop an asynchronous implementation of PSORisk so that PSO learning mechanism and local search can be flexibly applied and enhance the effectiveness of each other.

While RiskSlim [12] is a powerful tool to build optimal risk scores, it cannot scale well for large datasets. Experiments in Ustun and Rudin [12] shows that the optimality gaps of RiskSlim increases rapidly when the the number of instances and the number of features increase. An example of this limitation can be seen in Figure 5 where RiskSlim performs worse than PSORisk when used to process the largest dataset. Also, compared with PSORisk, it is much more challenging to speed up RiskSlim with parallelism due to the complexity of the algorithm. Therefore, PSORisk is a more attractive algorithm to handle the scalability issues of ORSP without deteriorating the classification accuracy.

VII. CONCLUSION AND FUTURE WORK

This paper proposes PSORisk, a scalable and interpretable machine learning algorithm based on particle swarm optimisation. The novelty of PSORisk is a simple yet effective classification algorithm to evolve risk score models. As a first PSO-based method for risk score, PSORisk is comprised of a new PSO representation for risk score, a new learning mechanism, a new fitness function that helps PSORisk effectively

evolve feasible risk score models, and a new local search heuristic to refine solutions discovered by the swarm.

The results of PSORisk on ten datasets with varying difficulties show that PSORisk is an innovative application of PSO to solve challenging machine learning tasks. Comparisons with representative linear and non-linear methods in IML showed that PSORisk can produce not only accurate but also highly interpretable models. PSORisk has three key practical advantages: (1) interpretability to provide both transparency and simple risk assessment, (2) flexibility to integrate operational constraints into risk scores, and (3) scalability to cope with large numbers of features and large numbers of instances. Although, to our knowledge this is the first work on PSO for solving optimal risk score problems, PSORisk has shown great promise and an excellent addition to the interpretable machine learning toolbox.

There are a number of directions for future studies. The current representation can be enhanced by co-evolving high-level interpretable features with the risk scores, which can significantly improve the discrimination power of PSORisk. Another aspect to investigate is the position updating mechanism which can improve the efficiency of PSORisk when searching for accurate and feasible risk scores. There have been many techniques proposed in the literature to handle constraints, and it is important to investigate which techniques can be useful for PSORisk. Moreover, a very interesting direction is to extend PSORisk to deal with multi-objective optimisation risk score problems in which both accuracy and model size are optimised simultaneously.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[2] D. E. O'Leary, "Artificial intelligence and big data," *IEEE Intell. Syst.*, vol. 28, no. 2, pp. 96–99, Mar./Apr. 2013.

[3] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, May 2017.

[4] S. Nguyen, M. Zhang, D. Alahakoon, and K. C. Tan, "People-centric evolutionary system for dynamic production scheduling," *IEEE Trans. Cybern.*, early access, Sep. 5, 2019, doi: 10.1109/TCYB.2019.2936001.

[5] E. J. Topol, "High-performance medicine: The convergence of human and artificial intelligence," *Nature Med.*, vol. 25, no. 1, pp. 44–56, Jan. 2019.

[6] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed. Berlin, Germany: Springer, 2009.

[7] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan, "An interpretable stroke prediction model using rules and Bayesian analysis," in *Proc. 17th AAAI Conf. Late-Breaking Develop. Field Artif. Intell. (AAAIWS)*, 2013, pp. 65–67.

[8] W. Souillard-Mandar, R. Davis, C. Rudin, R. Au, D. J. Libon, R. Swenson, C. C. Price, M. Lamar, and D. L. Penney, "Learning classification models of cognitive conditions from subtle behaviors in the digital clock drawing test," *Mach. Learn.*, vol. 102, no. 3, pp. 393–441, Mar. 2016.

[9] B. Ustun and C. Rudin, "Supersparse linear integer models for optimized medical scoring systems," 2015, *arXiv:1502.04269*. [Online]. Available: http://arxiv.org/abs/1502.04269

[10] S. B. van der Zon, W. Duivesteijn, W. van Ipenburg, J. Veldsink, and M. Pechenizkiy, "ICIE 1.0: A novel tool for interactive contextual interaction explanations," in *Proc. ECML PKDD Workshops*, C. Alzate, A. Monreale, L. Bioglio, V. Bitetta, I. Bordino, G. Caldarelli, A. Ferretti, R. Guidotti, F. Gullo, S. Pascolutti, R. G. Pensa, C. Robardet, and T. Squartini, Eds. Cham, Switzerland: Springer, 2019, pp. 81–94.

[11] F. Xie, B. Chakraborty, M. E. H. Ong, B. A. Goldstein, and N. Liu, "Autoscore: A machine learning–based automatic clinical score generator and its application to mortality prediction using electronic health records," *JMIR Med. Inf.*, vol. 8, no. 10, p. e21798, Oct. 2020.

[12] B. Ustun and C. Rudin, "Optimized risk scores," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 1125–1134.

[13] D. Spangler, T. Hermansson, D. Smekal, and H. Blomberg, "A validation of machine learning-based risk scores in the prehospital setting," *PLoS ONE*, vol. 14, no. 12, pp. 1–18, Dec. 2019.

[14] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 997–1006, Apr. 2004.

[15] Y. Lin, Y. Jiang, Y. Gong, Z. Zhan, and J. Zhang, "A discrete multiobjective particle swarm optimizer for automated assembly of parallel cognitive diagnosis tests," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2792–2805, Jun. 2019.

[16] K. Mistry, L. Zhang, S. C. Neoh, C. P. Lim, and B. Fielding, "A micro-GA embedded PSO feature selection approach to intelligent facial emotion recognition," *IEEE Trans. Cybern.*, vol. 47, no. 6, pp. 1496–1509, Jun. 2017.

[17] B. Ustun, M. B. Westover, C. Rudin, and M. T. Bianchi, "Clinical prediction models for sleep apnea: The importance of medical history over symptoms," *J. Clin. Sleep Med.*, vol. 12, no. 2, pp. 161–168, Feb. 2016.

[18] J. Zeng, B. Ustun, and C. Rudin, "Interpretable classification models for recidivism prediction," *J. Roy. Stat. Soc., A, Statist. Soc.*, vol. 180, no. 3, pp. 689–722, Jun. 2017.

[19] J.-X. Mi, A.-D. Li, and L.-F. Zhou, "Review study of interpretation methods for future interpretable machine learning," *IEEE Access*, vol. 8, pp. 191969–191985, 2020.

[20] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Definitions, methods, and applications in interpretable machine learning," *Proc. Nat. Acad. Sci. USA*, vol. 116, no. 44, pp. 22071–22080, 2019.

[21] O.-C. Granmo, "The tsetlin machine—A game theoretic bandit driven approach to optimal pattern recognition with propositional logic," 2018, *arXiv:1804.01508*. [Online]. Available: http://arxiv.org/abs/1804.01508

[22] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should i trust you?': Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2016, pp. 1135–1144. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939778

[23] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, 2017, pp. 4768–4777. [Online]. Available: http://dl.acm.org/citation.cfm?id=3295222.3295230

[24] M. Christoph. (2019). Interpretable machine learning. Lulu. [Online]. Available: https://christophm.github.io/interpretable-ml-book/

[25] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci. (MHS)*, 1995, pp. 39–43.

[26] B. Tran, B. Xue, and M. Zhang, "Variable-length particle swarm optimization for feature selection on high-dimensional classification," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 473–487, Jun. 2019.

[27] Y.-J. Zheng, H.-F. Ling, J.-Y. Xue, and S.-Y. Chen, "Population classification in fire evacuation: A multiobjective particle swarm optimization approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 70–81, Feb. 2014.

[28] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "A particle swarm optimization-based flexible convolutional autoencoder for image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 8, pp. 2295–2309, Aug. 2019.

[29] D. Zouache, A. Moussaoui, and F. B. Abdelaziz, "A cooperative swarm intelligence algorithm for multi-objective discrete optimization with application to the knapsack problem," *Eur. J. Oper. Res.*, vol. 264, no. 1, pp. 74–88, Jan. 2018.

[30] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[31] J. R. Quinlan, *C4. 5: Programs for Machine Learning*. Amsterdam, The Netherlands: Elsevier, 2014.

[32] K. Fernandes, J. S. Cardoso, and J. Fernandes, "Transfer learning with partial observability applied to cervical cancer screening," in *Pattern Recognition and Image Analysis*, L. A. Alexandre, J. S. Sánchez, and J. M. F. Rodrigues, Eds. Cham, Switzerland: Springer, 2017, pp. 243–250.
[33] J. R. Quinlan, "Simplifying decision trees," *Int. J. Man-Mach. Stud.*, vol. 27, no. 3, pp. 221–234, Sep. 1987.
[34] A. Khashman, "Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes," *Expert Syst. Appl.*, vol. 37, no. 9, pp. 6233–6239, 2010.
[35] Kaggle. (2019). *IBM HR Analytics Employee Attrition & Performance*. Accessed: Nov. 11, 2019. [Online]. Available: https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset
[36] L. F. Cranor and B. A. LaMacchia, "Spam!" *Commun. ACM*, vol. 41, no. 8, pp. 74–83, 1998.
[37] Kaggle. (2019). *Telco Customer Churn*. Accessed: Nov. 11, 2019. [Online]. Available: https://www.kaggle.com/blastchar/telco-customer-churn
[38] J. C. Schlimmer, *Concept Acquisition Through Representational Adjustment*. Irvine, CA, USA: Univ. California, Irvine, 1987.
[39] A. E. Johnson, J. Aboab, J. Raffa, T. Pollard, R. Deliberato, L. Celi, and D. Stone, "A comparative analysis of sepsis identification methods in an electronic database," *Crit. Care Med.*, vol. 46, no. 4, pp. 494–499, 2018.
[40] R. Kohavi, "Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining (KDD)*, 1996, pp. 202–207. [Online]. Available: http://dl.acm.org/citation.cfm?id=3001460.3001502
[41] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decis. Support Syst.*, vol. 62, no. 1, pp. 22–31, Jun. 2014.
[42] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," in *Proc. ECML PKDD Workshop, Lang. Data Mining Mach. Learn.*, 2013, pp. 108–122.

DIEM PHAM received the M.Sc. degree in computer science from Cantho University, Vietnam, in 2010. She is currently pursuing the Ph.D. degree with the Business School, La Trobe University, Melbourne, Australia. Her research interests are in data analytics and machine learning.

BINH TRAN (Member, IEEE) received the Ph.D. degree in computer science from the Victoria University of Wellington, New Zealand, in 2018. She is currently a Lecturer of business analytics with the Business School, La Trobe University, Melbourne, Australia. Her research interests are in computational intelligence, data analytics, feature manipulation, and machine learning.

SU NGUYEN (Member, IEEE) received the Ph.D. degree in artificial intelligence and data analytics from the Victoria University of Wellington, New Zealand, in 2013. He is currently a Senior Research Fellow with the Centre for Research in Data Analytics and Cognition, La Trobe University, Australia. His primary research interests include computational intelligence, optimization, data analytics, largescale simulation, and their applications in operations management and social media.

DAMMINDA ALAHAKOON (Member, IEEE) received the Ph.D. degree from Monash University, Australia, in 2002. He is currently a Professor of business analytics with the Business School, La Trobe University, Melbourne, Australia, and the Director of the Research Centre for Data Analytics and Cognition. He has published over 100 research articles in data mining, clustering, neural networks, machine learning, and cognitive systems. He received the Monash Artificial Intelligence Prize from Monash University.