Explainable Random Forests: A Set-Based Particle Swarm Optimization Approach

1st Anje Erasmus
Department of Industrial Engineering
Stellenbosch University
Stellenbosch, South Africa
24123986@sun.ac.za

2nd Andries Engelbrecht
Department of Industrial Engineering and Division of Computer Science
Stellenbosch University
Stellenbosch, South Africa
engel@sun.ac.za

***Abstract*—By extracting human-understandable rules from black-box ensemble models, it is possible to reconcile their strong predictive performance with the interpretability required for user trust. Random forests are highly accurate models, yet classify as black-box models. This study proposes a novel rule extraction technique from random forests using a set-based particle swarm optimization framework. The method constructs a universal rule set from induced random forests, then employs set-based particle swarm optimization in a separate-and-conquer approach to optimize a rule subset. This optimization is guided by a fitness function that effectively balances accuracy with interpretability criteria. Empirically validated across ten diverse datasets, the findings highlight the potential of SBPSO for rule extraction.**

***Index Terms*—set-based particle swarm optimization, random forest, rule extraction**

I. Introduction

Machine learning algorithms are generally divided into two main categories: supervised learning and unsupervised learning [1]. Supervised learning models are trained on labeled data. The goal is to learn a mapping between inputs and outputs so the model can make predictions on new, unseen data. Supervised learning is applied to tasks such as classification and regression. A widely used supervised learning method is the random forest, which belongs to the family of ensemble methods [2]. At its core, a random forest model is built from many decision trees, each of which makes predictions by splitting data into smaller subsets based on feature values [3].

Aggregation of the outputs of multiple decision trees reduces the risk of overfitting, and generalization is improved compared to individual decision trees. Random forest models have gained popularity due to their robustness, flexibility, and strong performance. However, random forest models are regarded as black-box models due to their lack of inherent explainability and transparency, as the ensemble structure obscures the global decision-making process, even though individual trees may be interpretable in isolation. Post-hoc methods are used to improve the explainability of random forests by analyzing the model post-training without the structure of the model being altered. This is done through rule extraction, feature ranking, surrogate models, and visualization techniques [4].

This paper proposes a new approach to rule extraction from random forest models using a set-based particle swarm optimization (SBPSO) algorithm in a separate-and-conquer approach. Before the SBPSO algorithm can be applied, conditions need to be extracted from the random forest model to form the universe for the SBPSO algorithm. The SBPSO rule extraction (SBPSO-RE) algorithm is evaluated on a number of classification datasets, and its performance is compared to existing rule extraction algorithms.

Section II discusses background of random forest models. Section III reviews existing random forest rule extraction methods. Section IV discusses SBPSO and highlights the set-operators used. The proposed SBPSO for random forest rule extraction is discussed in Section V. The empirical process is detailed in Section VI and the results are discussed in Section VII. Finally, Section VIII concludes this paper.

The implementation of the proposed SBPSO-RE framework is publicly available at GitHub and archived with DOI: 10.5281/zenodo.19479292.

II. Random forest models

Random forests, introduced by Breiman (2001) [2], are ensemble learning algorithms that aggregate the predictions of multiple decision trees. Each tree is trained on a bootstrap sample of the dataset, and at each node, a random subset of features is considered for splitting. This randomization reduces correlation between trees, thereby resulting in a enhanced generalization and reduced variance. A majority vote determines the predicted class and a internal estimate of generalization performance is obtained using the out-of-bag error, computed from data instances not included in the bootstrap samples [5].

Random forests offer several advantages, including high accuracy and robustness to noise. However, these strengths come at the cost of interpretability [3]. The ensemble nature of the model obscures the decision-making process, making it difficult to trace how input features contribute to predictions. This limits the adoption of random forests when model accountability is essential [6]. To address this limitation, studies have explored methods to further enhance model explainability. These include model-agnostic techniques such as local interpretable model-agnostic explanations (LIME) [7]. Other methods include model-specific approaches like rule extraction. The present work builds on this line of research by

developing a rule extraction framework designed to improve the interpretability of random forests.

III. Rule Extraction

The increased adoption of artificial intelligence (AI) has amplified the need for models that are accurate and explainable [5]. This demand for trust through transparency has been the driving force behind the growing field of explainable AI [8].

For this study, explainability is defined as a model's ability to provide human-understandable justifications for its predictions. In random forests, explainability is not inherent due to the complexity of aggregating multiple trees, and is therefore achieved through post-hoc methods that approximate or extract the underlying decision logic.

Rule extraction refers to deriving interpretable rules from the random forest ensemble, describing conditions under which specific predictions were made [9]. Extracted rules are typically expressed in a logical *if–then* format, which provides a transparent and human-readable description of the decision-making process. Each rule consists of an antecedent (*if*), defining feature-based conditions, and a consequent (*then*), specifying the predicted class. Formally, an extracted rule can be represented as

$$\text{IF } (f_1 \text{ operator } t_1) \ \wedge \ \ldots \ \wedge \ (f_k \text{ operator } t_k) \Rightarrow \text{Class} = C$$

where f_i denotes the i^{th} feature, operator represents a relational operator such as $>$, $\leq$, or $=$, and t_i is the corresponding threshold value derived from the internal decision boundaries of the model. Rules may also include metrics such as support, confidence, or accuracy to assess their reliability and applicability.

For random forests, each rule corresponds to a path from the root node to a leaf node within an individual decision tree. Rule extraction aggregates and prunes these paths across multiple trees to eliminate redundancy and simplify overlapping conditions. This results in a compact and interpretable rule-set representation of the ensemble behavior. Users can trace predictions back to specific feature thresholds and conditions to understand why an outcome was produced. By analyzing which features and thresholds appear frequently across the extracted rules, users can also infer global feature importance and gain deeper insight into the model's reasoning process.

Several rule-extraction algorithms have been developed to enhance the interpretability of ensemble models: ***RuleCOSI+*** [10] extracts interpretable rules from tree-based ensembles by combining and simplifying decision paths into a compact ruleset. It aggregates rules across trees and applies pruning and merging strategies to reduce redundancy. ***RuleFit*** [11] generates a large collection of simple decision rules from shallow trees and models these as features in a sparse linear regression framework regularized with LASSO to retain only the most relevant rules. Stable and Interpretable RUlesets ***SIRUS*** [12] focuses on model stability and interpretability by extracting conditional statements from random forests and retaining only the most frequently occurring and reliable rules. The resulting rule set is then aggregated through simple averaging, yielding a compact yet stable predictive model.

IV. Set-based particle swarm optimization

This section introduces SBPSO followed by detailed information on how particle positions and velocities are formulated using set-based operators.

A. Background

Particle swarm optimization (PSO) models a swarm of particles exploring a search space, guided by personal and global best solutions [13]. Inspired by the flight behavior of birds, particles search for optimal solutions by moving through the landscape, with positions updated based on their velocities. These velocities are influenced by prior motion, the particle's personal best position, and the global best position found by the swarm.

The SBPSO algorithm was suggested by Langeveld and Engelbrecht [14] to address certain limitations of the original PSO implementation with the initial goal of addressing the multi-dimensional knapsack problem. SBPSO adapts the classical PSO equations to manipulate particle positions as sets by adding or removing elements in discrete problem spaces.

SBPSO modifies the standard PSO framework by introducing set-specific operations that allow particles to add or remove elements from their current positions. The velocity of the particle is determined by a combination of influences, including its personal best position, the global best identified within its neighborhood, and the specialized addition and removal operators. Together, these mechanisms control how particles explore and update their positions within the discrete solution space.

B. Set-Based Operators

Let $P(U)$ denote the power set of U. The SBPSO operators are listed in Table I. For more detail the reader is referred to [14], [15]). The velocity and position are then updated as

$$V_i(t+1) = c_1 r_1 (Y_i(t) \ominus X_i(t)) \oplus c_2 r_2 (\hat{Y}(t) \ominus X_i(t)) \oplus c_3 r_3 \phi^+ A_i(t) \oplus c_4 r_4 \phi^- S_i(t) \quad (7)$$

$$X_i(t+1) = X_i(t) \oplus V_i(t+1) \quad (8)$$

where $c_1, c_2 \in [0, 1]$, $c_3, c_4 \in [0, |U|]$, and $r_k \sim U(0, 1)$. The removal operator ϕ^- may also employ tournament selection, as implemented in this work for rule induction.

V. Rule extraction from a random forest model using set-based particle swarm optimization

This section describes the theoretical foundation of SBPSO-RE. SBPSO-RE builds a universal set of conditions from the internal nodes of the random forest to form the swarm's search space. Each particle represents a candidate rule, and through iterative optimization with a single-objective fitness function, SBPSO-RE extracts interpretable rules that closely reflect the model's original decision logic.

Section V-A discusses how SBPSO-RE will be utilized. Thereafter, Section V-B provides the motivation for using

TABLE I: SBPSO Operators

Operator	Definition	(Eq.)
Velocity addition	$V_1 \oplus V_2 = V_1 \cup V_2$	(1)
Position difference	$X_1 \ominus X_2 = (X_1 \setminus X_2) \cup (X_2 \setminus X_1)$	(2)
Scalar multiplication	$\phi \otimes V =$ random subset of size $\lfloor \phi \lvert V \rvert \rfloor$	(3)
Velocity application	$X \oplus V = V(X)$, elements in V added/removed from X	(4)
Element removal	$\phi^- S =$ random subset of S of size $N_{\phi,S},\ S = X(t) \setminus Y(t) \setminus \hat{Y}(t)$	(5)
Element addition	$\phi^+ A = k$-Tournament selection from $A,\ A = U \setminus (X(t) \cup Y(t) \cup \hat{Y}(t))$	(6)

a meta-heuristic approach. Section V-C focuses on how the universal set is constructed from the random forest models conditions. The representation of individual particles is then discussed in Section V-D and lastly, the objective function is defined in Section V-E.

A. Using SBPSO-RE

To extract rules from a random forest model using SBPSO, a protocol is defined to guide which decision paths in the trees are targeted and how they are aggregated into rules. Each SBPSO instance is used to extract a single rule that approximates a particular class label based on the majority occurrence within the random forest predictions for the dataset. Unlike classical rule extraction methods that typically extract rules sequentially or greedily from a single tree, the SBPSO allows the particle swarm to search across the ensemble of decision paths. The SBPSO runs for a predetermined number of iterations, and the global best position identified by the swarm represents a candidate rule. This rule is then added to the extracted rule set and used to cover the corresponding instances in the dataset. The process repeats iteratively, targeting the current majority class among the uncovered instances, until all instances in the dataset are covered by rules extracted from the random forest.

B. Motivation for a Meta-Heuristic Approach

Rule extraction from random forests using SBPSO leverages the exploratory and exploitative dynamics of the particle swarm to identify high-quality rule antecedents directly from the ensemble. This approach differs from traditional methods, which typically use gain-based, top-down or bottom-up heuristics to build rules iteratively from individual trees.

By using SBPSO, each particle can explore different combinations of feature selectors across multiple decision trees, enabling the construction of complete rules in one step. The velocity update mechanism of SBPSO naturally balances exploration and exploitation, allowing particles to refine candidate rules efficiently while maintaining flexibility in rule length.

C. Construction of the Universal Set from Random Forests

In this study, the universe U is constructed by extracting feature conditions from a trained random forest model, distinguishing rule extraction from rule induction. While induction methods such as repeated incremental pruning to produce error reduction (RIPPER) [16], partial decision trees (PART) [17], and SBPSO-based rule induction [15] derive rules directly from data, rule extraction interprets the decision logic of an existing model.

After training the random forest on the pre-processed dataset, each internal decision node defines a condition,

$$(\texttt{feature}_j, \texttt{operator}, \texttt{threshold}) \tag{9}$$

where $\texttt{feature}_j$ is the j^{th} input feature used for the split, `operator` denotes the direction of the split ($<$ or $\geq$), and `threshold` represents the split value.

All such unique conditions across all trees and nodes are aggregated to form the universal set U, encompassing all conditions that contribute to the decision boundaries learned by the random forest model. Each selector represents a potential antecedent condition that may appear in an extracted rule. The SBPSO algorithm then operates over subsets of U, where each subset corresponds to a potential rule antecedent.

By constructing U from the learned structure of the random forest, the SBPSO algorithm performs rule extraction rather than induction, as the search for optimal rule subsets is constrained to conditions that the random forest has already deemed relevant.

D. Particle Representation

Each particle in the swarm represents an individual rule extracted from the random forest model. Unlike traditional SBPSO-based rule induction, where particles directly encode rule antecedents, this implementation operates on pre-extracted conditions. Each position of the particle therefore corresponds to a subset of these extracted conditions, representing a candidate rule set that explains the decision boundaries of the model.

Formally, let the universal set U denote all unique selectors derived from the decision paths of the random forest. Each element $u_i \in U$ is a tuple, (feature, operator, value), representing a condition used in at least one decision path within the ensemble. A particle's position $X_i(t) \subseteq U$ is thus a subset of these selectors, defining a possible rule antecedent. The velocity defined in standard SBPSO, $V_i(t)$, is a set of operations $\{(+, u_j), (-, u_k)\}$ indicating which selectors should be added or removed during the position update. These operations guide the movement of the particle within the discrete, set-based search space.

Selectors ($u_j \in U$) are further guided by a desirability measure $d(u_j)$ which quantifies the contribution of each condition to the predictive performance. During each update step, selectors with higher desirability scores are more likely

to be added to a particle's position, while less desirable selectors are more likely to be removed. This introduces a guided search mechanism that biases exploration toward more informative and model-consistent conditions, steering particles toward subsets of selectors that are both meaningful and representative.

Each particle also maintains its local best position $Y_i(t)$ and the global best position $\hat{Y}_i(t)$, both subsets of U, representing the best-performing rule antecedents found locally and globally. The update process employs the SBPSO operators to add or remove elements from the current subset, enabling exploration of different condition combinations while preserving interpretability.

E. Objective Function

The evaluation of each particle relies on an objective function designed to balance predictive performance and interpretability. The function combines several sub-objectives into a single fitness score. The objective function is defined as:

$$\begin{aligned} f(D,r) &= \big[\alpha A(D,r) + (1-\alpha)C(D,r)\big]\Big(1 - \beta\frac{L(r)}{L_{\max}}\Big) \\ &- \gamma \max\big(0, C(D,r) - \tau\big) - R(D,r) \end{aligned}$$

where $A(D,r)$ is the accuracy of rule r on dataset D, $C(D,r)$ is the rule coverage, $L(r)$ is the number of conditions in r, with $L_{\max}$ the maximum rule length, β controls the length penalty, promoting interpretability, γ penalises overly general rules when $C(D,r) > \tau$, α balances accuracy and coverage, and $R(D,r)$ is the redundancy penalty for overlapping or conflicting rules.

Each weight $\alpha, \beta, \gamma \in [0,1]$ (including 0 and 1) controls the trade-off between these criteria, ensuring a balanced search between performance and simplicity. By using the random forest as the foundation for the universal set U, the SBPSO process focuses on refining and optimising combinations of already learned conditions.

VI. Methodology

Section VI outlines the empirical process used to evaluate and compare SBPSO-RE to existing rule extraction methods.

A. Dataset Acquisition

Appropriately complex datasets need to be selected for the training and testing of SBPSO-RE. The ten gathered datasets were selected to represent a diverse data landscape. Evaluating the robustness of SBPSO-RE across diverse datasets provides insight into its scalability and generalization capabilities. Each dataset can be categorized as either easy, medium or hard, based on predefined characteristics.

Table II summarizes the number of attributes and instances for each dataset after preprocessing as well as the classification of the dataset according to the criteria.

B. Algorithm Tuning

The SBPSO-RE algorithm was tuned using Bayesian optimization implemented via the optuna framework to maximise the objective function (Section V-E) [18]. A tree-structured parzen estimator sampler was employed to explore the parameter space efficiently [19]. This method models the likelihood of promising parameter configurations and adaptively focuses the search toward regions yielding higher objective function values, offering a more efficient alternative to exhaustive or quasi-random sampling techniques.

Each optimization run sampled 100 candidate configurations from the defined parameter ranges given in Table III, with each configuration evaluated using 30 swarm particles over 300 iterations. The tuning process was conducted independently for each of the datasets. Following optimization, the algorithm was retrained using the corresponding optimal configuration for each dataset.

C. Pruning

Following rule extraction, a pruning phase was conducted to refine the rule set by removing redundant or low-quality rules. The goal of pruning was to improve the interpretability and conciseness of the extracted rule set without compromising predictive accuracy or fidelity.

Each rule was evaluated iteratively within the context of the full rule set. The process began by computing the macro-averaged F1-score of the complete ruleset applied to the test data. During each iteration, one rule at a time was removed, and the resulting change in macro F1-score was assessed. If the removal of a rule maintained or improved performance, the rule was permanently pruned. Rules were also removed if their individual fitness score fell below a predefined threshold. Once pruned, rules were not reconsidered in subsequent iterations. This iterative elimination continued until no further improvement in the overall F1-score was observed.

D. Benchmarking Algorithms

Benchmarking is conducted to evaluate the performance of the proposed method. Three algorithms were selected, and their control parameters were tuned using the same Bayesian optimization framework applied to SBPSO-RE.

RuleCOSI+: maximum rule length, minimum rule support, and number of ensemble trees were optimized. The search ranges were $[2,6]$, $[0.01, 0.2]$, and $[100, 500]$, respectively.

RuleFit: maximum tree depth, number of trees, and L_1 regularisation strength [20] were tuned within $[3,8]$, $[100, 500]$, and $[0.001, 0.1]$, respectively.

SIRUS: the stability threshold (p_0), number of trees, and maximum tree depth were varied. The search ranges were $[0.4, 0.9]$, $[200, 1000]$, and $[3,8]$, respectively, balancing rule stability and model sparsity.

All tuning procedures were conducted independently for each dataset, with every algorithm evaluated over 100 sampled configurations using identical random seeds and evaluation protocols as the proposed SBPSO model to prevent tuning bias. To ensure a fair comparison between RuleFit, RuleCosi+,

TABLE II: Summary of datasets with number of input attributes, instances, number of extracted conditions, and complexity classification.

Dataset	Input Attributes	Instances	Extracted Conditions	Classification
Adult Income	14	48,842	85,968	Medium
Bank Marketing	20	45,207	32,426	Medium
Breast Cancer	30	569	3,880	Easy
Customer Churn	12	505,207	26,196	Medium–Hard
Diabetes	8	768	10,342	Easy–Medium
Forest Cover	54	581,012	300,320	Hard
Loan Approval	12	4,269	10,532	Medium
Mushroom	23	8,124	11,562	Easy–Medium
Telco Churn	21	7,043	42,106	Medium
Wine Quality	11	6,497	9,784	Medium

TABLE III: Control Parameters Tuned for the SBPSO Algorithm

Parameter	Range	Description
c_1	[0.5, 2.5]	Cognitive coefficient (particle self-influence)
c_2	[0.5, 2.5]	Social coefficient (swarm influence)
c_3	[0.5, 2.5]	Desirability weighting coefficient
c_4	[0.5, 2.5]	Penalisation coefficient controlling rule complexity
k	[2, 10]	Maximum number of antecedent conditions per rule
Swarm size	[10, 100]	Number of particles in the swarm
Max iterations	[10, 200]	Maximum optimisation iterations

SIRUS, and SBPSO-RE, all experiments employed 5-fold cross-validation and each method executed 10 independent times per fold. The average performance across runs was reported. Each dataset was partitioned into five folds, with models trained on four and tested on the remaining fold in turn. Reported accuracy and coverage values in Table V represent the mean performance across all folds.

VII. Results

The following section presents the results obtained after applying SBPSO-RE to the datasets discussed in Section VI-A.

Table IV compares the accuracy and number of extracted rules for SBPSO-RE before and after pruning, alongside the performance of the tuned random forest model. Results are reported as mean values with 90% confidence intervals, which indicate the range within which the true performance is expected to lie with 90% confidence.

The results in Table IV show that the SBPSO-RE method initially extracted a large number of rules, particularly for datasets classified as 'hard' in Table II. After pruning, most datasets retained fewer than 31 rules. Pruning substantially reduced model complexity while often maintaining or improving predictive performance. As expected, the post-pruning accuracy of SBPSO-RE is generally lower than that of the random forest model; however, interpretability is significantly improved.

In Table IV a notable drop in accuracy is observed for complex datasets such as Forest Cover, where accuracy decreases from 94.05% (random forest) to 55.89% (SBPSO-RE) after pruning. However, as shown in Table V, all rule extraction methods struggle on this dataset. Despite this, pruning still produces simpler and more interpretable models.

An example from the Mushroom dataset illustrates the difference in rule complexity. The random forest produces rules with multiple conditions, whereas SBPSO-RE yields a simple rule: *IF odor* ≤ 4.5 *THEN Class = 0.*

The SBPSO-RE method achieved competitive predictive accuracies across all datasets while maintaining full coverage. On datasets such as mushroom, breast cancer, and loan approval, it reached near-perfect accuracy, outperforming RuleFit, RuleCosi+ and SIRUS (Table V). Although SBPSO-RE produced more rules on average, its overall performance remained competitive for these datasets.

RuleCosi+ generally extracted the fewest rules, prioritizing simplicity at the expense of accuracy; however, it struggled with multi-class predictions. In contrast, RuleFit tended to produce the largest rule sets but performed similarly or worse than other benchmarking methods. This is evident in the Breast Cancer dataset, where RuleFit generated 109.4 rules while achieving similar accuracy to SBPSO-RE (5.66 rules).

SIRUS and SBPSO-RE demonstrated the most balanced trade-off between accuracy and interpretability. While SIRUS performed well on multi-class datasets such as Wine Quality (58.02% accuracy, 20.26 rules), SBPSO-RE achieved a slightly worse performance with far fewer rules for Wine Quality (50.92% accuracy, 6.46 rules). SBPSO-RE slightly outperformed SIRUS on the Forest Cover dataset in terms of predictive accuracy (55.89% vs. 55.25%), while also producing more compact rule sets (12.89 versus 23.04 for SIRUS). SIRUS, however, produces shorter rules on average when compared to SBPSO-RE (1.17 conditions vs. 2.65 conditions respectively).

Rule lengths were shortest for SIRUS and SBPSO-RE, and longest for RuleCosi+, while coverage was highest for SIRUS and SBPSO-RE (often 100%) and lower for RuleFit and RuleCosi+, indicating partial explanations.

VIII. Conclusion

This study introduced a set-based approach to rule extraction from random forests using the SBPSO algorithm and

TABLE IV: SBPSO-RE and random forest performance (mean ± 90% CI)

Dataset	Before Pruning		After Pruning		Random Forest	
	Accuracy (%)	No. Rules	Accuracy (%)	No. Rules	Accuracy (%)	No. Trees
Adult Income	71.91 ± 0.74	59.30 ± 3.89	73.29 ± 1.06	12.00 ± 1.31	85.08	693520
Bank Marketing	73.57 ± 0.47	57.04 ± 3.49	74.80 ± 0.95	8.96 ± 0.75	86.34	164415
Breast Cancer	94.22 ± 0.52	17.16 ± 0.34	96.31 ± 0.43	5.66 ± 0.48	98.60	2206
Customer Churn	98.07 ± 0.01	34.2 ± 4.18	98.76 ± 0.01	8 ± 3.28	99.98	67966
Diabetes	74.23 ± 0.73	39.04 ± 2.80	76.81 ± 0.95	8.86 ± 0.96	86.00	12693
Forest Cover	59.25 ± 0.32	112 ± 2.30	55.89 ± 0.62	12.89 ± 1.78	94.05	1210691
Loan Approval	94.90 ± 1.20	74.46 ± 22.57	96.46 ± 1.49	14.15 ± 5.60	97.96	14111
Mushroom	98.89 ± 0.44	22.28 ± 1.07	99.01 ± 0.44	10.54 ± 0.68	100.00	2906
Telco Churn	74.67 ± 0.01	65.04 ± 6.88	75.86 ± 0.01	15.76 ± 2.08	81.36	115764
Wine Quality	53.67 ± 0.01	66.78 ± 4.45	50.92 ± 0.02	6.46 ± 1.04	74.57	10097

TABLE V: Comparison of the pruned RuleFit, RuleCosi+, SIRUS, and SBPSO-RE performance

Dataset	RuleFit				RuleCosi+				SIRUS				SBPSO-RE			
	Rules	Accuracy (%)	Coverage (%)	Avg. Cond.	Rules	Accuracy (%)	Coverage (%)	Avg. Cond.	Rules	Accuracy (%)	Coverage (%)	Avg. Cond.	Rules	Accuracy (%)	Coverage (%)	Avg. Cond.
Adult Income	25	75.94	83.98	2.269	8.4	74.19	73.00	6.66	13.49	74.32	100.00	1.78	12.0	73.29	100.00	2.32
Bank Marketing	20.0	74.78	87.45	1.93	7.72	67.60	77.28	5.96	15.84	70.52	100.00	1.57	8.96	74.80	100.00	1.75
Breast Cancer	101.9	94.74	100.00	1.74	12.4	93.67	98.42	3.38	12.06	93.63	100.00	1.32	5.66	96.31	100.00	1.70
Customer Churn	71.72	99.03	100.00	1.93	7.38	91.53	91.24	3.32	6.48	98.49	100.00	1.69	8	98.76	100.00	1.51
Diabetes	11.5	68.85	80.20	2.19	6.9	73.52	72.82	3.58	7.14	76.10	100.00	1.00	8.86	76.81	100.00	1.77
Forest Cover	24.88	11.70	28.20	3.15	57	24.24	95.23	5.30	23.04	55.25	100.00	1.17	12.89	55.89	100.00	2.65
Loan Approval	30.48	95.79	99.24	1.56	4.50	95.74	96.79	2.33	9.60	92.85	89.72	1.79	14.15	96.46	100.00	1.74
Mushroom	29.18	97.83	100.00	2.47	4.80	95.79	89.44	3.29	14.86	87.63	99.58	1.79	10.54	99.01	100.00	1.44
Telco Churn	17.00	75.87	90.00	1.88	8.20	73.21	74.09	3.86	7.54	74.16	100.00	1.46	15.76	75.99	100.00	1.66
Wine Quality	9.86	33.53	37.20	2.86	79.40	56.83	90.17	6.28	20.26	58.02	99.20	1.11	6.46	50.92	100.00	2.15

demonstrated its effectiveness across multiple classification datasets. The proposed SBPSO algorithm consistently produced accurate and interpretable rule sets. By leveraging a single-objective fitness function, the predictive performance is balanced with rule simplicity. The overall performance of the SBPSO-RE method is comparable to the performance of the benchmarking algorithms, with similar result being obtained to SIRUS. Future work includes performing statistical significance testing to validate the observed differences in performance across datasets and includes comparing the performance of the model to other noted benchmarking algorithms such as OptExplain and MIRCO [21], [22]

REFERENCES

[1] V. Nasteski, "An overview of the supervised machine learning methods," *HORIZONS.B*, vol. 4, pp. 51–62, 12 2017.

[2] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] E. Halabaku and E. Bytyçi, "Overfitting in machine learning: A comparative analysis of decision trees and random forests," *Intelligent Automation and Soft Computing*, vol. 39, pp. 1–10, 01 2024.

[4] C. O. Retzlaff, A. Angerschmid, A. Saranti, D. Schneeberger, R. Röttger, H. Müller, and A. Holzinger, "Post-hoc vs ante-hoc explanations: xai design guidelines for data scientists," *Cognitive Systems Research*, vol. 86, 2024.

[5] D. E. Mathew, D. U. Ebem, A. C. Ikegwu, P. E. Ukeoma, and N. F. Dibiaezue, "Recent emerging techniques in explainable artificial intelligence to enhance interpretability and understanding of ai models for humans," *Neural Processing Letters*, vol. 57, pp. 1–32, 2025.

[6] M. A. Naji, S. E. Filali, M. Bouhlal, E. H. Benlahmar, R. A. Abdelouhahid, and Q. Debauche, "Breast cancer prediction and diagnosis through a new approach based on majority voting ensemble classifier," *Procedia Computer Science*, vol. 191, pp. 481–486, 2021.

[7] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," *CoRR*, vol. abs/1602.04938, 2016. [Online]. Available: http://arxiv.org/abs/1602.04938

[8] D. Minh, H. X. Wang, Y. F. Li, and T. N. Nguyen, "Explainable artificial intelligence: A comprehensive review," *Artificial Intelligence Review*, vol. 55, pp. 3503–3568, 2021.

[9] K. Mexis, S. Xenios, and A. Kokosis, "On the systematic development of large-scale kinetics using stability criteria and high-throughput analysis of curated dynamics from genome-scale models," *Computer Aided Chemical Engineering*, vol. 52, pp. 2729–2734, 2023.

[10] J. Obregon and J. H. Jung, "Rulecosi+: Rule extraction for interpreting classification tree ensembles," *Information Fusion*, vol. 89, pp. 355–381, 2023.

[11] J. H. Friedman and B. E. Popescu, "Predictive learning via rule ensembles," *The Annals of Applied Statistics*, vol. 2, pp. 916–954, 2008.

[12] B. Clement, B. Gerard, D. V. Sebastien, and S. Erwan, "Sirus: Stable and interpretable rule set for classification," *Electronic Journal of Statistics*, vol. 15, pp. 427–505, 2021.

[13] J. Kennedy and R. Eberhardt, "Particle Swarm Optimization (PSO)," in *Proceedings of IEEE International Conference on Neural Networks*, 1995, Conference Proceedings, pp. 1942–1948.

[14] J. Langeveld and A. P. Engelbrecht, "A generic set-based particle swarm optimization algorithm," *International Conference on Swarm Intelligence*, 2011.

[15] J.-P. van Zyl and A. P. Engelbrecht, "Set-based particle swarm optimisation: A review," *Mathematics*, vol. 11, no. 13, p. 2980, 2023. [Online]. Available: https://www.mdpi.com/2227-7390/11/13/2980

[16] W. W. Cohen, "Fast Effective Rule Induction," in *Machine Learning: Proceedings of the Twelfth International Conference*, 1995, Conference Proceedings.

[17] E. Frank and I. H. Witten, "Generating Accurate Rule Sets Without Global Optimization," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 144–151.

[18] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 2623–2631.

[19] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011.

[20] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[21] H. Maissae and B. Abdelaziz, "Forest-ore: Mining optimal rule ensemble to interpret random forest models," *Engineering Applications of Artificial Intelligence*, vol. 143, 2025.

[22] G. Zhang, Z. Hou, Y. Huan, and J. Shi, "Extracting optimal explanations for ensemble trees via logical reasoning," *Applied Intelligence*, vol. 53, 2022.