




Improved ACO Rank-Based Algorithm for Use in Selecting Features for Classification Models

Roberto Alexandre Delamora^{1,2}^a, Bruno Nazário Coelho³^b and Jodelson Aguilar Sabino⁴^c

¹Graduate Program in Instrumentation, Control and Automation of Mining Process,
Federal University of Ouro Preto, Instituto Tecnológico Vale, Ouro Preto, Brazil

²Vale S.A., Nova Lima, Brazil

³Graduate Program in Instrumentation, Control and Automation of Mining Process,
Federal University of Ouro Preto, Instituto Tecnológico Vale, Ouro Preto, Brazil

⁴Artificial Intelligence Center, Vale S.A., Vitória, Espírito Santo, Brazil

Keywords: Wrapper-Filter Method, Ant Colony Optimization, Metaheuristic, Machine-Learning, Feature Selection, Dimensionality Reduction.

Abstract: Attribute selection is a process by which the best subset of attributes in a given dataset is searched. In a world where decisions are increasingly based on data, it is essential to develop tools that allow this selection of attributes to be more efficiently performed, aiming to improve the final performance of the models. Ant colony optimization (ACO) is a well-known metaheuristic algorithm with several applications and recent versions developed for feature selection (FS). In this work, we propose an improvement in the general construction of ACO, with improvements and adjustments for subset evaluation in the original Rank-based version by Bullnheimer et al. to increase overall efficiency. The proposed approach was evaluated on several real-life datasets taken from the UCI machine-learning repository, using various classifier models. The experimental results were compared with the recently published WFACOFS method by Ghosh et al., which shows that our method outperforms WFACOFS in most cases.

1 INTRODUCTION

The opportunity that data innovation offers the world is virtually unprecedented. Innovative machine-learning tools are already revolutionizing our lives in incredible ways. Now, these tools are helping people to uncover the hidden answers with the growing abundance of data resources. These transformative new technologies are converting data into new products, solutions, and innovations that promise to significantly change people's lives and relationships with the world.


From an economic point of view and on a conservative estimate, economists estimate that if more effective use of data generated small gains, making sectors of activity only 1% more efficient, this would add almost US\$15 trillion to global GDP by 2030. (BSA The Software Alliance, 2015).


Inserted within the context of Industry 4.0, large volumes of data in different formats have been generated, captured, and stored, representing an excellent opportunity to transform them into information that adds value to the business (Ayres et al., 2020).


The big question in this context is not just about having more data, as this will happen naturally, but knowing which data should be used to achieve the expected objective in the best way and minimizing the expenditure of time and resources, material and financial.

While its value proposition is undeniable, to live up to its promise, data needs to meet some basic parameters of usability and quality. Not all data is helpful for all tasks, i.e., the data needs to match the tasks for which it is intended to be used (Sharda et al., 2019).

Machine-learning algorithms are pretty efficient at discovering patterns in large volumes of data, but contradictorily, they are also greatly affected by biases and relationships contained in that data. Redundant attributes impair the machine-learning algorithm per-

^a <https://orcid.org/0000-0003-2609-8862>

^b <https://orcid.org/0000-0002-2809-7778>

^c <https://orcid.org/0000-0003-1690-7849>

formance, both in terms of speed due to the dimensionality of the data, and the success rate, since the presence of redundant information can confuse the algorithm instead of helping it to find a correct model for knowledge. (Witten et al., 2005). Furthermore, keeping irrelevant attributes in a dataset can result in overfitting leading to a loss of generalizability and performance.

The feature selection process primarily focuses on removing redundant or uninformative predictors from the model (Kuhn et al., 2013). A valuable way of thinking about the feature selection problem is a search in solution space. The search space is discrete and consists of all possible combinations of selectable features in the dataset. The objective is to navigate the solution space and find the best match or a match good enough to improve performance relative to using all features (Brownlee, 2014).

Thus, in a world where decisions are increasingly based on data, it is essential to develop tools that allow the selection of features to be more efficiently performed, aiming to improve the final performance of the models, removing those features from the analysis context. That are not significant, or that harm the final result.

The present work focuses on the development of improvements in the computational model of feature selection based on the ACO - Ant Colony Optimization (Stützle et al., 1999) class of algorithms, more specifically on the metaheuristic AS_{rank} - Ant System with elitist strategy and ranking (Bullnheimer et al., 1997). This metaheuristic was developed in 1997 as an evolution of the original AS (Ant System) (Dorigo et al., 1991) model and is still widely used today as a basis for new models adapted to specific needs and applications.

Although ACO (Stützle et al., 1999) and AS_{rank} (Bullnheimer et al., 1997) were initially developed to solve the Traveling Salesman Problem (TSP), they can be customized to fit the FS domain.

Using as reference the WFACOFS - Wrapper-Filter ACO Feature Selection (Ghosh et al., 2019), a FS algorithm of the Wrapper-filter type, and considering as an evaluation function the accuracy measure obtained through the classification of subsets of selected attributes, this work proposes adjustments to the AS_{rank} seeking performance improvements, not only in the accuracy values but also in reducing the dimensionality of the datasets.

Currently, the WFACOFS (Ghosh et al., 2019) method is one of the ones that has presented better results in the studied bases. Develop an evolution from the already widely known algorithm AS_{rank} (Bullnheimer et al., 1997), implementing adjustments

to perform the selection of features and use correlation statistics as a reference of distances in order to obtain an improved algorithm that can match the WFACOFS or even to partially or totally surpass it, would be an outstanding contribution to the theme.

In this way, the present work seeks to contribute to the search for new solutions to the issue of feature selection by proposing a new approach built from a metaheuristic created initially to search for better routes and which presents very desirable characteristics in the proposed problem.

2 LITERATURE STUDY

2.1 Feature Selection

As datasets become complex and voluminous, the FS is necessary to refine the information by restricting only relevant and useful features to the process. Consequently, there is also a reduction in computational effort and time due to the reduction in the dimensionality of the data (Ayres et al., 2020).

Given a set of features of dimension n , the FS process aims to find a minimum subset of features of dimension m ($m < n$) adequate to represent the original set. It is a widely used technique and stands out in Data Mining (García et al., 2015).

The literature describes three approaches to the FS processes, Filter, Wrapper, and Embedded (Dong and Liu, 2018), each with different selection strategies.

Filter methods work on the intrinsic properties of data and do not require a learning algorithm. It tends to make them very fast, but as FS is done without consultation of a learning algorithm, the accuracy of FS using filter methods is generally less than wrapper methods. Wrapper methods require a learning algorithm that leads to a higher accuracy and computation time. A compromise between these two methods is embedded methods which are built using a combination of filter and wrapper methods. These techniques balance the two classes of methods and try to incorporate learning algorithms and intrinsic data properties in a method. There may be an acceptable trade-off between computation time and accuracy or even a lower computation cost with no accuracy degradation. Therefore, the general trend has moved to the design of embedded systems (Ghosh et al., 2019).

2.2 Algorithm ACO

The Ant Colony Algorithm (ACO) is a metaheuristic for combinatorial optimization that was created to solve computational problems that involve finding

paths in graphs and is based on probability and population search methods.

It represents the simulation of the behavior of a set of agents (ants) belonging to a colony in the food search, cooperating to optimize the path to be followed between the colony and the food source, using indirect communication.

Computationally, the ACO metaheuristic is a constructive search method in which a population of agents (artificial ants) cooperatively construct candidate solutions for a given problem. The construction is probabilistic, guided by the heuristic of the problem and by a shared memory between the agents, containing the experience of previous iterations. This memory consists of an artificial pheromone trail based on assigning weights to the features of the candidate solutions (Gaspar-Cunha et al., 2012).

This idea of ACO was first implemented by (Dorigo et al., 1996), and they named it AS - Ant System. Since then, many modifications to ACO have taken place over the years (Ghosh et al., 2019).

Ants are considered stochastic procedures and construct the subsets of features iteratively, using both heuristic information and the amount of pheromone accumulated in the trails. The stochastic component brings a complete solution to space exploration and creates a greater variety of subsets than a greedy heuristic. The ant search strategy is reminiscent of reinforcement learning (Dorigo and Stützle, 2019).

The process is characterized by a positive feedback loop, where the probability of an ant choosing a path increases with the number of ants that previously chose the same path (Dorigo et al., 1991).

The AS has very desirable characteristics, according to (Dorigo et al., 1996):

- **It is versatile** as it can be applied to similar versions of the same problem;
- **It is robust**, as it can be applied with only minimal changes to other combinatorial optimization problems;
- It is a **population-based approach**. It is interesting because it allows the exploitation of positive feedback as a search engine.

These desirable properties are counterbalanced because, for some applications, AS can be overcome by more specialized algorithms. It is a problem also shared by other popular methods like Simulated Annealing and Tabu Search (Dorigo et al., 1996).

Several improvements were proposed and tested in the TSP from this first model. All these improved versions of AS have in common a stronger exploration of the best solutions found to drive the ant search process. They differ mainly in some aspects of the search

control (Stützle et al., 1999).

Recently authors have suggested an unsupervised FS algorithm based on ACO. In this method, when ants construct solutions, they use a similarity matrix to select the next feature based on the similarity between the last selected feature and the feature to be selected next. After constructing the solutions, pheromones are updated only based on the frequency of the selection of features (Ghosh et al., 2019).

While FS is a crucial application domain for ACO, several works have also focused on other domains. Even in economics, ACO is used to predict a financial crisis (Uthayakumar et al., 2020). It goes a long way in ascertaining the popularity and applicability of ACO.

2.3 WFACOFS

The hybrid-type WFACOFS (Ghosh et al., 2019) algorithm was implemented to combine the best advantages of the Filter and Wrapper-type methods. For its development, the UFSACO (Tabakhi et al., 2014) and TFSACO (Aghdam et al., 2009) algorithms were considered as a basis, proposing successful techniques to overcome the deficiencies observed in each one.

WFACOFS introduced new concepts, such as the normalization of pheromone values, to prevent the FS process from becoming biased and to improve the exploration of the solution space. Pheromone updating is done globally and locally in the standard ACO and the predecessor algorithms on which WFACOFS was built, but the pheromone value is not delimited. Thus, a feature chosen more often acquires a high pheromone value leading to its selection multiple times.

Another critical contribution of WFACOFS is that the algorithm works with the proposal of carrying out the pheromone deposit in the node and not in the path (edge). It also established the calculation of cosine symmetry between the features for setting up the distance matrix, a parameter required by the ACO.

3 PRESENT WORK

The basis for our proposed method is described in Sect. 3.1 while our proposed method is detailed in Sect. 3.2.

3.1 Basis of Proposed Method

The present work focuses on the development of theoretical research and practical experiments using a

modified version of the algorithm Rank-based System (AS_{rank}) (Bullnheimer et al., 1997) to perform the FS in datasets and considers the WFACOFS algorithm (Ghosh et al., 2019) as reference for the comparison of results.

The algorithm AS_{rank} , developed by (Bullnheimer et al., 1997), was chosen to be the basis for this work because it has favorable characteristics compared to previous versions of AS: (i) excellent performance in solving problems and (ii) speed in converging to reasonable solutions.

In order to provide maximum comparability between the scenario presented by WFACOFS and that presented by our method, the same datasets were considered in studies involving both models.

3.2 Proposed Method

For simplicity, we use the alias $ACOFS_{rank}$ - ACO Feature Selection Rank-Based System for our proposed method. A flowchart of the entire work is given in Figure 1.

The application of the $ACOFS_{rank}$ algorithm to perform the FS requires the prior evaluation of these features. This work assumes that the dataset has already undergone the initial data transformation and treatment processes to make it consistent and suitable for use in machine-learning algorithms. Anomaly situations, lack of data, errors in description, and data imbalance, among other problems usually found in the original datasets would already be solved or, at least, mitigated to a large extent.

The statistical correlation metric is used to build the model's matrix. Correlation is not commonly used in this application, but it was considered to bring a new perspective to the method's operation. The Spearman Correlation (Spearman, 1904) was used to calculate the correlation for its straightforward interpretation and explanation.

As the correlation value can assume values between $[0, 1]$ and the value 0 is not desired because it can generate division by zero errors during the algorithm's execution, a re-scaling operation is applied to the matrix so that all values are in the range of $[1, 10]$. This operation follows the calculation presented in Equation 1.

$$cor_adjusted = \frac{9 * (value - cor_min)}{cor_max - cor_min} + 1 \quad (1)$$

wherein:

$cor_adjusted$ correlation after re-scaling process
 $value$ original correlation value
 cor_min minimum value on matrix
 cor_max maximum value on matrix

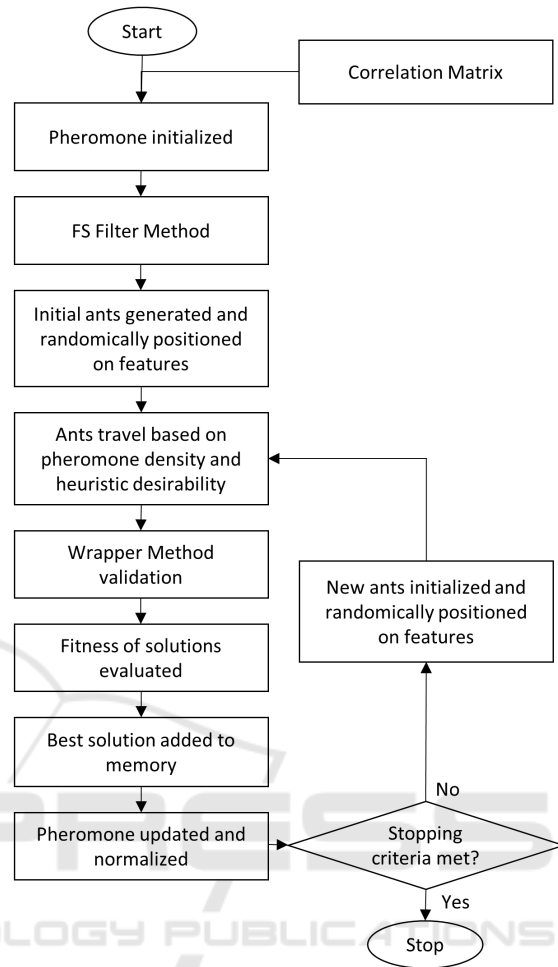


Figure 1: Flowchart for proposed $ACOFS_{rank}$.

After the initialization of the pheromone matrix, the Filter method is applied in the first step of FS. Filter-based FS methods use statistical measures to score the dependence between input features that can be filtered to choose the most relevant resources (Brownlee, 2019).

The main objective to be achieved in this step is to produce a minority reduction in the total amount of features, eliminating those that, in a more statistically evident way, do not add value or have a low influence on the response feature.

The practical implementation of the Filter method was carried out using the ANOVA (Analysis Of Variance) statistical test, taking into account the F-score coefficient as a validation metric. ANOVA is a statistical method to verify if there are significant differences between the means of groups of data, being possible to infer if the features are dependent on each other (Santos, 2021).

Also, as presented by (Gajawada, 2019), the vari-

ance of an independent feature determines how much it impacts the response feature. If the variance is low, this characteristic has no significant impact on the response and vice versa.

Algorithm 1: $ACOFS_{rank}$	
1:	Início
2:	Parameters initialized
3:	Initial ants generated
4:	Pheromone matrix initialized
5:	FS Filter method applied
6:	repeat till stopping criteria met
7:	repeat for each ant
8:	Build solution
9:	Wrapper method validation
10:	till the end
11:	Best solution identified
12:	Besto solution stored in memory
13:	Pheromone updated and normalized
14:	till the end
15:	End

ANOVA calculates the variance ratio between groups divided by variance within groups as described in Equation 2. Thus, the greater the variance between the groups, the more different the two features will be and the greater the F-score (Santos, 2021).

$$F - score = \frac{\text{Variance among groups}}{\text{Variance within groups}} \quad (2)$$

F-score is a univariate feature selection method, which means it scores each feature (x_1, x_2, x_3, \dots) individually without considering that one feature may present better results if combined with others. The higher the F-score, the more likely this feature is to be more discriminating (Chen and Lin, 2006).

ANOVA uses verification by the F-test table to validate if there is any significant difference between the groups of values that make up a feature. If there is no significant difference between the groups, it can be assumed that, statistically, all variances are equal. This feature must then be removed from the (Gajwada, 2019) template.

Once ANOVA is applied to compare each independent feature with the response feature, a F-score coefficient associated with each is obtained. This coefficient represents the influence of the feature's behavior on the response feature's behavior or, in other words, it represents the share of explainability of the behavior of the response feature that is attributed to this feature under analysis.

In this work, the features that, in the accumulated sum of the individual F-scores, represent 95% of the explainability of the response feature will be maintained. This way, removing unimportant features from a statistical point of view is carried out without significant loss of information. A process example is

described in Figure 2, which shows a graph with the F-Score values for each feature on the *Wine* dataset and its accumulated value. The dotted line indicates the 95% cutoff threshold.

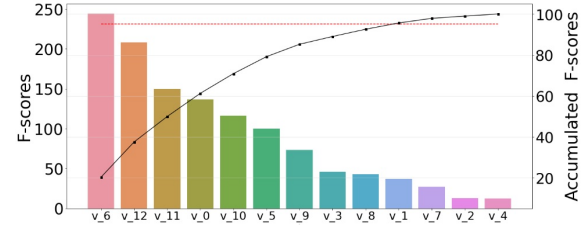


Figure 2: F-score analysis using ANOVA.

After selecting the features in the Filter method step, a new dataset is generated with only the remaining features, which follow the algorithm's flow of analysis and processing.

The ants are randomly placed on these remaining features. According to (Bullheimer et al., 1997), ACO achieves better results when the number of ants equals the number of features, and each ant starts its journey in a different feature. This setting was also used in $ACOFS_{rank}$.

The definition of using an ant for each variable has its pros and cons. One benefit is that it increases the search space, ensuring that more subsets of variables are analyzed. On the other hand, it increases complexity and computational cost, especially in datasets with a large number of features.

From there, each ant traverses a number of features that is also randomly defined. In this way, solutions with different amounts of features are built and analyzed, allowing the model to autonomously explore and discover reasonable solutions with reduced numbers of features.

Starting from a different initial feature, each ant chooses the next feature to be visited, considering a probability that is a function of the correlation value between the features and the amount of residual pheromone present on the edge that connects to this feature. The probability calculation is defined by Equation 3.

$$p_{ij}^k = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta}{\sum_{h \in \Omega} ([\tau_{ih}(t)]^\alpha \cdot [\eta_{ih}(t)]^\beta)} \quad (3)$$

$$\eta_{ij} = \frac{1}{d_{ij}}$$

wherein:

τ_{ij} intensity of the pheromone present in the edge between features i and j

- α influence of the pheromone
- β influence of the correlation between the features i and j
- d_{ij} correlation between features i and j
- η_{ij} visibility between features i and j
- Ω list of features not yet visited by the ant

As in AS_{rank} (Bullnheimer et al., 1997), a memory of the features already visited is kept in each ant, thus preventing repetitions of features in the same stretch. Likewise, the pheromone is only updated at the end of the construction phase.

The best global track is always used to upgrade pheromone levels, which characterizes an elitist strategy. Also, only some of the best ants from the current iteration can add pheromones. The amount of pheromone an ant can deposit is defined according to the ranking index r . Only the best $(\omega - 1)$ ants from each iteration can deposit pheromone. The best global solution is given the weight ω . The r th best ant of the iteration contributes to the pheromone update with a weight given by $\max\{0, \omega - r\}$ (Stützle et al., 1999).

The way the pheromone is updated in Equations 4 and 5 allows the success of the previous iterations to be reflected in future generations. The constant ϕ defines the balance between the importance of accuracy and the number of features used in the solution.

After updating the pheromone matrix, the solutions found (a subset of features) are ordered according to the level of pheromone present. The best solution among them is chosen and stored. It is up to the Wrapper method to define the values of the statistical metric associated with each solution.

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \sum_{r=1}^{\omega} (\omega - r) \cdot \Delta\tau_{ij}^r(t) + \omega \cdot \gamma_{best} \quad (4)$$

$$\Delta\tau_{ij}^r(t) = \begin{cases} \phi \cdot \gamma(G) + \frac{(1-\phi) \cdot (n-|G|)}{n} & , \text{if } i \in G \\ 0 & , \text{otherwise} \end{cases} \quad (5)$$

wherein:

- n number of ants
- r ranking of ants
- $\Delta\tau_{ij}^r$ increase of pheromone by r th ant
- ω number of elitists ants
- ϕ balance of accuracy
- G a subgroup of selected features
- $\gamma(G)$ accuracy for selected features defined by r th ant
- γ_{best} accuracy of the best ant

A new cycle or iteration is started with new ants being built, and the pheromone matrix is maintained.

In this way, the pheromones matrix works as a solutions' memory mapped in the previous cycle and identifies those considered the best.

The values of α and β provide the necessary balance between exploitation and exploration. Using ρ_i (pheromone on the i th feature) provides the scope of including previous success in decision-making.

Even though it is a stochastic process, or even because of it, the undesired situation of having solutions that are precisely the same as those previously generated by other ants may occur. In these cases, the duplicated solution is discarded, and a new option is then constructed, following the same precepts of randomness in the definition of each solution. This process aims to increase the exploration of the solution space, preventing the algorithm from remaining stuck in a particular search region.

There may also be cases in which solutions with different features but in the same amount have equal accuracy values. To mitigate the problem mentioned above, after executing the Wrapper method, a function to validate the fitness of the solutions was implemented in the algorithm. For its implementation, the statistical measure F-measure (also known as F1-score) is used, which is the harmonic mean between precision and recall and can be interpreted as a measure of the reliability of accuracy. A high value on this measure means that the accuracy is relevant (Silva, 2018). Equation 6 provides the objective function in determining the fitness of a subset of features G .

$$fit = w_1 \cdot \gamma(G) + F1(G) + w_2 \cdot e^{-\frac{r}{n}} \quad (6)$$

wherein:

- w_1 weight to the accuracy
- w_2 weight to the ratio of unselected features to the feature dimension
- $\gamma(G)$ accuracy of the selected group
- $F1(G)$ F-measure of the selected subgroup
- n total number of features
- r number of features of the solution created by the ant

4 EXPERIMENTAL RESULTS

To evaluate the performance of $ACOFS_{rank}$ concerning to $WFAOFS$, the same datasets used in $WFAOFS$ were selected, which were still available. These datasets are available through the UCI (Dua and Graff, 2017) repository and are frequently cited in the literature for evaluating machine-learning models. This chapter presents the results achieved.

Table 1 details the properties of these datasets, which were categorized according to the number of features, following the same criteria defined by (Ghosh et al., 2019): Accuracy per classifier with 10, 20, and 30 iterations

1. **Small** \rightarrow features ≤ 10

- Breast Cancer (BC)

2. **Medium** $\rightarrow 10 <$ features ≤ 100

- Wine (WI)
- Ionosphere (IO)
- Soybean Small (SS)
- Hill Valley (HV)

Table 1: Description of the datasets used in the present work.

Dataset	Features	Classes	Samples
Breast Cancer	9	2	699
Wine	13	3	178
Ionosphere	34	2	351
Soybean Small	35	4	47
Hill Valley	100	2	606

Datasets with a high number of features (above 100) require an analysis of their performance in processing time and are generally datasets with dense data characteristics (Ayres, 2021). There are no restrictions on using ACO in this type of dataset, but they were not considered in this work and are part of the proposal for improvements and future work.

The ACOFS_{rank} algorithm was developed in Python language version 3.8.3, using the Spyder IDE v.4. All computational experiments were performed on an Intel Core i5-7200U CPU @ 2.50GHz (2 processing cores) with 16GB of RAM and Windows Home 10 64-bit operating system.

All hyper-parameters used in the algorithm are given in Table 2. The calibration of the values applicable to the hyper-parameters ρ , α , β , ϕ , w_1 and w_2 were done using the IRACE library (López-Ibáñez et al., 2016), which uses the statistical software package R (R, 2015).

Figure 3 presents the values used by IRACE to define the best set of hyper-parameters, and Figure 4 contains the screen with the final results presented by the tool as optimal options. According to the IRACE documentation, the final options presented by the tool are equivalent in terms of algorithm performance and any of them can be chosen. The selection of hyper-parameters adopted in ACOFS_{rank} is highlighted in Figure 4.

The algorithm had been run on all bases and with all classifiers in three blocks, with 10, 20, and 30 iterations.

```
## Parameter file for MAIN
# name switch type values [conditions (using R syntax)]
rho "--rho" c (0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35)
a "--a" c (1.0, 2.0, 2.5, 3.0, 3.5)
b "--b" c (0.5, 1.0, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 8.0)
phi "--phi" c (0.25, 0.50, 0.75, 0.85, 0.90, 0.95, 0.96, 0.97)
w1 "--w1" c (100, 150, 175, 200)
w2 "--w2" c (0.75, 1.0, 1.5, 1.75, 2.0, 2.5)
```

Figure 3: Ranges of hyper-parameters used in IRACE.

```
# 2022-10-21 20:16:51 -03: Elite configurations (first number is the
# rho a b phi w1 w2
364 0.35 2.0 1.0 0.50 100 2.0
313 0.15 2.0 1.0 0.50 150 2.0
339 0.15 3.0 1.0 0.50 175 0.75
432 0.35 2.0 1.0 0.50 150 2.0
# 2022-10-21 20:16:51 -03: Stopped because there is not enough budget
# You may either increase the budget or set 'minNbSurvival' to a low value
# Iteration: 8
# nbIterations: 8
# experimentsUsedSoFar: 2999
# timeUsed: 0
# remainingBudget: 1
# currentBudget: 1
# number of elites: 4
# nbConfigurations: 3
# Best configurations (first number is the configuration ID, listed
# rho a b phi w1 w2
364 0.35 2.0 1.0 0.50 100 2.0
313 0.15 2.0 1.0 0.50 150 2.0
339 0.15 3.0 1.0 0.50 175 0.75
432 0.35 2.0 1.0 0.50 150 2.0
# Best configurations as commandlines (first number is the configuration ID)
364 "--rho 0.35 --a 2.0 --b 1.0 --phi 0.50 --w1 100 --w2 2.0
313 "--rho 0.15 --a 2.0 --b 1.0 --phi 0.50 --w1 150 --w2 2.0
339 "--rho 0.15 --a 3.0 --b 1.0 --phi 0.50 --w1 175 --w2 0.75
432 "--rho 0.35 --a 2.0 --b 1.0 --phi 0.50 --w1 150 --w2 2.0
```

Figure 4: IRACE final results and chosen hyper-parameters set.

Table 3 and Figure 5 show the results achieved in experiments. The number of features selected in each case is described in parentheses, and the best result for each dataset is in bold and underlined.

After obtaining the feature subset through the Filter method, we used different classifiers to evaluate the solutions obtained by each ant in each iteration. The classifiers used are K-Nearest Neighbors - KNN (Luz, 2018) (Brownlee, 2020); MLP (Ferreira, 2019) (Mohanty, 2019) (Moreira, 2018); XGBoost (Chen and Guestrin, 2016) (Brownlee, 2020); and Random Forest (Ho, 1995) (Brownlee, 2020).

The KNN algorithm is a non-parametric, supervised learning classifier which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another (IBM, 2020). This classifier is very popular due to its simplicity and efficiency at the same time.

MLP is a popularly used and efficient classifier. It is a feed-forward artificial neural network consisting of three layers — input, hidden, and output. The layers form a connected graph and are assigned random weights, modified during training using the backpropagation algorithm (Ghosh et al., 2019).

XGBoost is an efficient open-source implementation of the gradient-boosted trees algorithm. Gradient

Table 2: Description of algorithm hyperparameters.

Parameter	Description	Value
n	Number of ants	Equal to the number of features
m	Number of elitists ants	30% of the number of features, limited to 15
α	Pheromone influence	2.0
β	Correlation between features influence	1.0
Iterations	Number of iterations	10, 20 and 30
ρ	Pheromone evaporation factor	0.15
ϕ	Balance factor for accuracy	0.50
w_1	Weight parameter for accuracy	150
w_2	Weight parameter for the number of features of the selected subset	2

Table 3: Accuracy per classifier with 10, 20, and 30 iterations.

	Dataset	Wine	Soybean Small	Ionosphere	Breast Cancer	Hill Valley
	Acronyms	WI	SS	IO	BC	HV
	Features / Samples	13 / 178	35 / 47	34 / 351	9 / 699	100 / 606
WFACOFS with KNN	Accuracy % (# Features)	100% (6)	100% (6)	97.35% (7)	99.00% (5)	55.61% (63)
WFACOFS with MLP	Accuracy % (# Features)	100% (4)	100% (4)	98.68% (12)	99.67% (7)	64.52% (66)
10 iterations						
ACOFS_rank	KNN	97.22 % (3)	100 % (3)	97.18 % (3)	98.57 % (9)	63.11 % (8)
	MLP	100 % (7)	100 % (3)	98.59 % (5)	97.86 % (4)	87.70 % (67)
	XGB	100 % (3)	100 % (3)	97.18 % (11)	98.57 % (5)	63.93 % (30)
	Random Forest	100 % (3)	100 % (3)	97.18 % (6)	98.57 % (3)	61.48 % (8)
20 iterations						
ACOFS_rank	KNN	97,22 % (3)	100 % (3)	97,18 % (3)	98,57 % (9)	64,75 % (10)
	MLP	100 % (6)	100 % (3)	98,59 % (4)	97,86 % (4)	90,16 % (68)
	XGB	100 % (3)	100 % (3)	98,59 % (8)	98,57 % (5)	63,93 % (11)
	Random Forest	100 % (3)	100 % (3)	97,18 % (5)	98,57 % (3)	62,30 % (4)
30 iterations						
ACOFS_rank	KNN	97,22 % (3)	100 % (3)	97,18 % (3)	98,57 % (9)	66,39 % (16)
	MLP	100 % (6)	100 % (3)	100 % (8)	97,86 % (4)	89,34 % (23)
	XGB	100 % (3)	100 % (3)	98,59 % (8)	98,57 % (5)	65,57 % (18)
	Random Forest	100 % (3)	100 % (3)	97,18 % (5)	98,57 % (3)	63,11 % (19)

boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models (AWS, 2022).

Random forest is a supervised learning algorithm that can be used both for classification and regression.

It is also the most flexible and easy to use. A for-

est is comprised of trees, and it is said that the more trees it has, the more robust a forest is. Random forest creates decision trees on randomly selected data samples, gets predictions from each tree, and selects the best solution through voting. It also provides a good indicator of the feature’s importance (Naviani, 2018).

Table 4 presents the percentage number of features

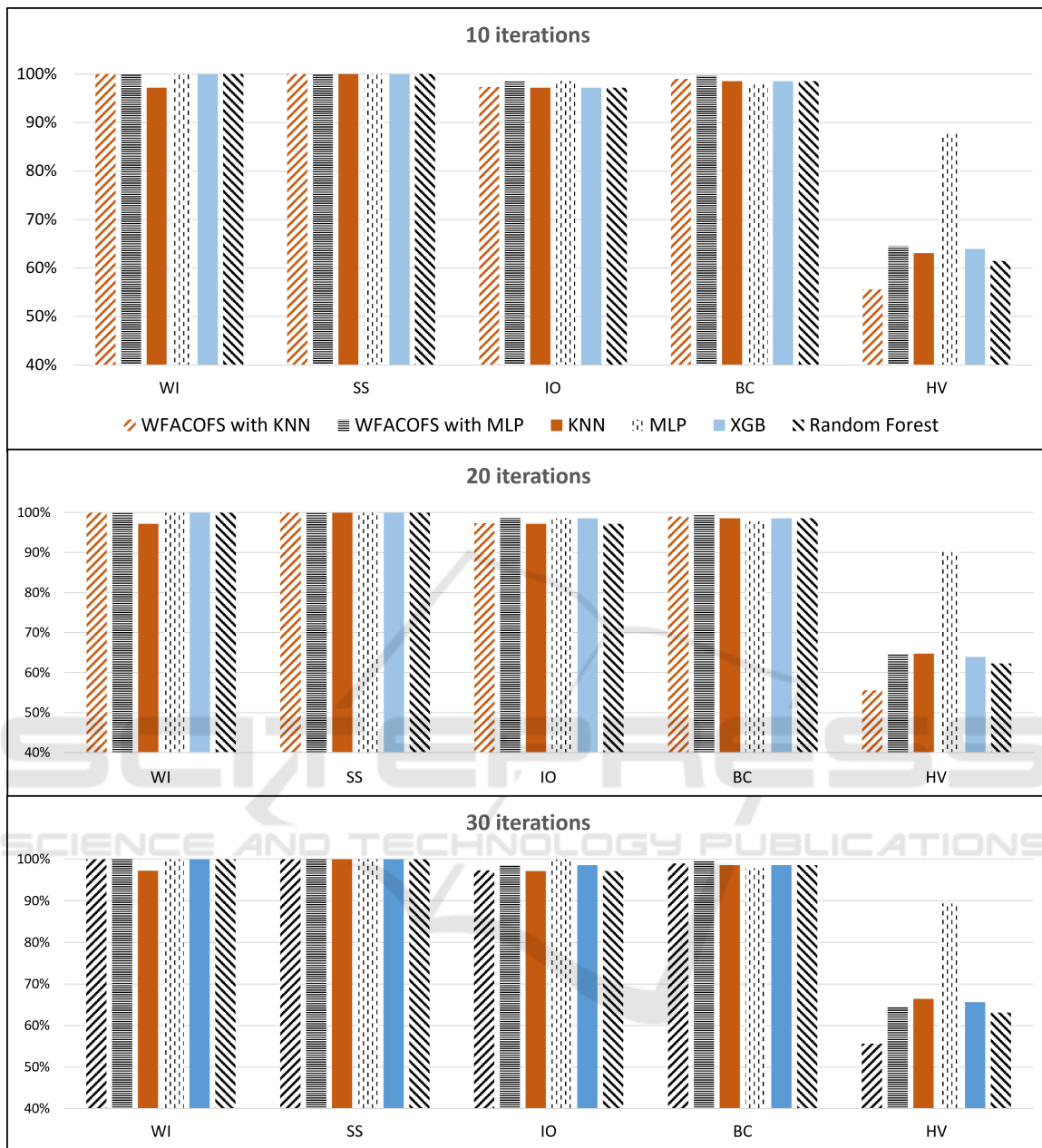


Figure 5: Accuracy per classifier with 10, 20, and 30 iterations.

selected by each classifier in each dataset concerning the total number of features. The values obtained by $ACOFS_{rank}$ are compared to the values defined in WFACOFS. Note that the $ACOFS_{rank}$ achieves a more significant reduction in the dimensionality of the datasets.

Table 5 shows a comparison among the best values of accuracy obtained by the algorithms in each dataset. These results do not consider the type of classifier but only the best accuracy result obtained. The number of features selected in each case is described

in parentheses, and the best result for each dataset is in bold and underlined.

From Tables 3, 4, and 5, it can be observed that the proposed model is comparable to WFACOFS. Thus, we can state that $ACOFS_{rank}$ is a model which applies to FS problems with some important gains. It uses the Filter approach to reduce the computational cost of the system and the power of a Wrapper approach to enhance the classification ability, which makes it an overall robust embedded model. It also uses correlation as the primary statistical metric to fill the dis-

Table 4: No. of features considered for the best accuracy about the total of features.

	WI	SS	IO	BC	HV
WFACOFS with KNN	46%	17%	21%	56%	63%
WFACOFS with MLP	31%	11%	35%	78%	66%
10 iterations					
KNN	23%	9%	9%	100%	8%
MLP	54%	9%	15%	44%	67%
XGB	23%	9%	32%	56%	30%
Random Forest	23%	9%	18%	33%	8%
20 iterations					
KNN	23%	9%	9%	100%	10%
MLP	46%	9%	12%	44%	68%
XGB	23%	9%	24%	56%	11%
Random Forest	23%	9%	15%	33%	4%
30 iterations					
KNN	23%	9%	9%	100%	16%
MLP	46%	9%	24%	44%	23%
XGB	23%	9%	24%	56%	18%
Random Forest	23%	9%	15%	33%	19%

Table 5: Comparison of our proposed approach with WFACOFS algorithm.

	WI	SS	IO	BC	HV
WFACOFS	100% (4)	100% (4)	98,68 % (12)	99,67 % (7)	64,52 % (66)
ACOFsrank	100% (3)	100% (3)	100 % (8)	98,57 % (3)	90,16 % (68)

tances matrix and applies additional validation on the fitness function to select the best solution, even on unbalanced datasets.

5 CONCLUSION AND FUTURE WORKS

In this work, we propose an improvement in the general construction of the well-known AS_{rank} algorithm (Bullnheimer et al., 1997) to obtain an increase in performance with a reduction in the dimensionality of the datasets applying a FS process. The proposed algorithm was compared to WFACOFS (Ghosh et al., 2019), a recently developed embedded algorithm that presents excellent results in the analyzed aspects.

It can be considered that the general objective of creating an improved algorithm using the Rank-based

Ant System (AS_{rank}) metaheuristic was achieved, taking into account that the results obtained by the new proposed $ACOFs_{rank}$ algorithm surpassed in most of the databases those obtained by the reference model WFACOFS. Furthermore, the reduction in dimensionality promoted by $ACOFs_{rank}$ was more significant than that of WFACOFS.

Despite being a more complex solution than other already available, the results demonstrate the potential of $ACOFs_{rank}$ in FS operations in a wide range of datasets. The possibility of using different robust classifiers through parameterization characterizes the good adaptability and flexibility of the algorithm.

As a proposal for future scope, one might consider exploring new ways of measuring heuristic desirability using other filter methods instead of ANOVA. Using other classifiers for the Wrapper method and for evaluating the values in the fitness function is also very interesting and can present promising results. Adapting the algorithm to work with classification and regression algorithms will bring greater flexibility for broader use in projects involving these two strands.

ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ), the Instituto Tecnológico Vale (ITV), the Universidade Federal de Ouro Preto (UFOP) and the Vale S.A..

REFERENCES

- Aghdam, M. H., Ghasem-Aghaee, N., and Basiri, M. E. (2009). Text feature selection using ant colony optimization. *Expert systems with applications*, 36(3):6843–6853.
- AWS (2022). How xgboost works. <https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html>. (accessed on March 16, 2022).
- Ayres, P. F. (2021). Seleção de atributos baseado no algoritmo de otimização por colônia de formigas para processos mineradores. Master's thesis, UFOP - Universidade Federal de Ouro Preto, Ouro Preto. (Mestrado Profissional em Instrumentação, Controle e Automação de Processos de Mineração).
- Ayres, P. F., Sabino, J. A., and Coelho, B. N. (2020). Seleção de variáveis baseado no algoritmo otimização colônia de formigas: Estudo de caso na indústria de mineração. In *Congresso Brasileiro de Automática-CBA*, volume 2.

- Brownlee, J. (2014). Feature selection to improve accuracy and decrease training time. <https://machinelearningmastery.com/feature-selection-to-improve-accuracy-and-decrease-training-time/>. (accessed on September 30, 2022).
- Brownlee, J. (2019). How to choose a feature selection method for machine learning. <https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/>. (accessed on July 22, 2022).
- Brownlee, J. (2020). How to calculate feature importance with python. <https://machinelearningmastery.com/calculate-feature-importance-with-python/>. (accessed on March 2, 2022).
- BSA The Software Alliance, B. (2015). What is the big deal with data? https://data.bsa.org/wp-content/uploads/2015/12/bsadatastudy_en.pdf. (accessed on September 4, 2022).
- Bullnheimer, B., Hartl, R. F., and Strauss, C. (1997). A new rank-based version of the ant system. a computational study.
- Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree-boosting system. In *Proceedings of the 22nd ACM Sigkdd international conference on knowledge discovery and data mining*, pages 785–794.
- Chen, Y.-W. and Lin, C.-J. (2006). Combining svms with various feature selection strategies. In *Feature extraction*, pages 315–324. Springer.
- Dong, G. and Liu, H. (2018). *Feature engineering for machine learning and data analytics*. CRC Press.
- Dorigo, M., Maniezzo, V., and Colomi, A. (1991). Positive feedback as a search strategy.
- Dorigo, M., Maniezzo, V., and Colomi, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41.
- Dorigo, M. and Stützle, T. (2019). Ant colony optimization: overview and recent advances. *Handbook of meta-heuristics*, pages 311–351.
- Dua, D. and Graff, C. (2017). Uci machine learning repository.
- Ferreira, C. A. (2019). Mlp classifier. <https://medium.com/@carlosalbertoff/mlp-classifier-526978d1c638>. (accessed on March 2, 2022).
- Gajawada, S. K. (2019). Anova for feature selection in machine learning. <https://towardsdatascience.com/anova-for-feature-selection-in-machine-learning-d9305e228476>. (accessed on August 3, 2022).
- García, S., Luengo, J., and Herrera, F. (2015). *Data preprocessing in data mining*, volume 72. Springer.
- Gaspar-Cunha, A., Takahashi, R., and Antunes, C. H. (2012). *Manual de computação evolutiva e meta-heurística*. Coimbra University Press.
- Ghosh, M., Guha, R., Ram, S., and Ajith, A. (2019). A wrapper-filter feature selection technique based on ant colony optimization. *Neural Computing & Applications*, 32(12):7839–7857.
- Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.
- IBM (2020). What is the k-nearest neighbors algorithm? <https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point>. (accessed on March 2, 2022).
- Kuhn, M., Johnson, K., et al. (2013). *Applied predictive modeling*, volume 26. Springer.
- López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Bittanti, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.
- Luz, F. (2018). Algoritmo knn para classificação. <https://inferir.com.br/artigos/algoritmo-knn-para-classificacao/>. (accessed on March 2, 2022).
- Mohanty, A. (2019). Multi-layer perceptron (mlp) models on real-world banking data. <https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f>. (accessed on March 2, 2022).
- Moreira, S. (2018). Multi-layer perceptron (mlp) models on real-world banking data. [https://medium.com/ensina-ai/rede-neural-perceptron-multicamadas-f9de8471f1a9#:~:text=Perceptron%20Multicamadas%20\(PMC%20ou%20MLP,sa%20ADda%20de%20sejada%20nas%20camadas%20intermedi%C3%A1rias](https://medium.com/ensina-ai/rede-neural-perceptron-multicamadas-f9de8471f1a9#:~:text=Perceptron%20Multicamadas%20(PMC%20ou%20MLP,sa%20ADda%20de%20sejada%20nas%20camadas%20intermedi%C3%A1rias). (accessed on March 2, 2022).
- Naviani, A. (2018). Understanding random forests classifiers in python tutorial. <https://www.datacamp.com/tutorial/random-forests-classifier-python>. (accessed on March 22, 2022).
- R, C. T. (2015). R: A language and environment for statistical computing. <https://www.R-project.org>. (accessed on September 20, 2022).
- Santos, G. (2021). Estatística para seleção de atributos. <https://medium.com/data-hackers/estat%C3%ADstica-para-sele%C3%A7%C3%A3o-de-atributos-81bdc274dd2c>. (accessed on July 10, 2022).
- Sharda, R., Delen, D., and Turban, E. (2019). *Business Intelligence e Análise de Dados para Gestão do Negócio-4*. Bookman Editora.
- Silva, T. A. (2018). Como implementar as métricas precisão, revocação, acurácia e medida-f. [https://tiago.blog.br/precisao-revocacao-acuracia-e-medida-f#:~:text=Medida%20\(F%20Measure%2C%20F1,medida%20de%20confiabilidade%20da%20acur%C3%A1cia](https://tiago.blog.br/precisao-revocacao-acuracia-e-medida-f#:~:text=Medida%20(F%20Measure%2C%20F1,medida%20de%20confiabilidade%20da%20acur%C3%A1cia). (accessed on September 20, 2022).
- Spearman, C. (1904). The proof and measurement of association between two things. *Amer. Journal of Psychology*, 15(1):72–101.
- Stützle, T., Dorigo, M., et al. (1999). Aco algorithms for the traveling salesman problem. *Evolutionary algorithms in engineering and computer science*, 4:163–183.
- Tabakhi, S., Moradi, P., and Akhlaghian, F. (2014). An unsupervised feature selection algorithm based on ant colony optimization. *Engineering Applications of Artificial Intelligence*, 32:112–123.
- Uthayakumar, J., Metawa, N., Shankar, K., and Lakshmanaprabu, S. (2020). Financial crisis prediction

model using ant colony optimization. *International Journal of Information Management*, 50:538–556.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2005). *Data mining practical machine learning tools and techniques*. volume 2.

