AIMS Mathematics, 10(7): 16762–16795.
DOI:10.3934/math.2025753
Received: 22 May 2025
Revised: 08 July 2025
Accepted: 16 July 2025
Published: 25 July 2025

https://www.aimspress.com/journal/Math

Research article

Meta-feature-based data preprocessing for machine learning through correlation structures and cost-aware benchmarking: A case study in cybersecurity

Noemí DeCastro-García[1,2,*] **and David Escudero García**[2]

[1] Department of Mathematics, Universidad de León, Campus de Vegazana s/n, 24007, León, Spain
[2] Research Institute of Applied Sciences in Cybersecurity,Universidad de León, Campus de Vegazana s/n, 24007, León, Spain

* **Correspondence:** Email: ncasg@unileon.es.

Abstract: One of the current challenges in applying machine learning is optimizing the constructed models to achieve the best possible performance. This article proposes a novel approach to determine which data preprocessing strategy may be most beneficial for enhancing the evaluation fit metrics, based on studying the correlation between the dataset's meta-features and the performance response variables. Additionally, the meta-features were categorized in terms of modification cost and control over the model's fit to determine which strategies appeared to be the most optimal. Also, we studied if these transformation can improve the results obtained by several training and testing proportions. The study was conducted with 42 different configuration datasets derived from the multiclass classification problem of IP address maliciousness. Three machine learning algorithms and tools were evaluated: Autosklearn, Gaussian Mixture Models, and Extreme Gradient Boosting; all of them have been studied on the same problem in previous works. Also, we applied five different types of data transformations, such as scaling, dimensionality reduction techniques, and quantile transformation. The results show that meta-feature correlation analysis significantly improves machine learning performance by guiding data preprocessing and transformation strategies, sometimes even surpassing the best original performance.

Keywords: machine learning; artificial intelligence; meta-learning; meta-feature; preprocessing data; cybersecurity
Mathematics Subject Classification: 68P01, 68T01

1. Introduction

Machine learning (ML) has become an essential tool in numerous fields because it analyzes large volumes of data, identifies complex patterns, and makes accurate predictions. Currently, it is also crucial in any discipline where personalized recommendation systems, process automation, or enhanced decision-making are required, such as cybersecurity or business intelligence [1, 2].

One of the objectives of current research related to ML is to develop and apply techniques that enhance the performance of models created for specific problems or create automated tools for solving the multi-cash problem [3, 4]. With this aim, there are methods such as feature selection, hyper-parameter optimization, cross-validation, regularization, and data preprocessing. This work focuses on the latter approach. Data transformations are powerful techniques for improving the fit and effectiveness of ML models, provided they are applied carefully. Not all transformations enhance performance if not used correctly. Additionally, this preprocessing adds complexity to the pipeline, as it requires knowledge of the data to select and apply the most appropriate transformations.

The goal of this research is to determine whether the use of meta-learning (MtL) offers a practical approach to optimizing the data preprocessing and transformation process, leading to a significant performance gain in ML models. The hypothesis is that a meta-feature analysis can enhance ML performance by guiding data preprocessing and transformations.

MtL is a subfield of ML that focuses on developing models capable of learning new patterns through the knowledge extracted by applying ML algorithms to a specific problem. Meta-features play a fundamental role throughout the previous process, as they characterize the datasets and have applications without the need to execute the entire MtL pipeline [5, 6]. This work is framed on the study of a novel approach in the use of meta-features: identifying which meta-features of the dataset have more correlation with the response variables would allow to deduce which characteristics of the original dataset are modifiable, to what extent, and at what cost, in order to gain accuracy. In this way, it could be determined in which cases the maximum accuracy of a dataset for use with ML techniques has been achieved, or which preprocessing techniques are most optimal.

In this empirical study, an analysis has been carried out on a cybersecurity dataset in which each IP address is to be classified by its level of maliciousness [7, 8]. The work is based on the computation of 84 meta-features (statistical, general, information theory-based, model-based, and others) following [9, 10]. This has been carried out for each of the 42 datasets obtained by combining the feature sets and partition percentages of the original dataset. The study of their correlation with the response variables of the models has been carried out for accuracy, but it has also included four other response variables not usually included in previous existing studies but fundamental for unbalanced datasets. These are Mathews Correlation Coefficient (MCC), F_1-score, Sensitivity, and Specificity. The ML models applied are Autosklearn, Gaussian Mixture Models (GMM), and Extreme Gradient Boosting (XGB), previously tested algorithms with the dataset of the study. In addition, we have applied different transformations to the data to determine whether meta-features provide useful information with regards to preprocessing. The five transformations we have applied are principal component analysis (PCA), selection of the most informative features with mutual information (SelectKBest), feature scaling with min-max normalization (Scaler), a quantile transformation to map distribution of features (Quantile transform), and adding a duplicate of the original features multiplied by -1 to increase the correlations (Corr increase). The analyses have been conducted through descriptive statistics, inferential statistics,

and correlation computation.

The results show that studying the correlation of meta-features with performance variables, as well as categorizing them by cost and controlling the impact of their modification, is an effective strategy for optimizing the application of data preprocessing techniques and selecting training and testing proportions.

This work is organized as follows: In Section 2, we develop the related work. In Section 3, we include all the details about the experimentation carried out. In Section 4, we discuss the obtained results. Finally, the conclusions and references are given.

2. Related work

Performing data transformation in the preprocessing data science's pipeline is a common and effective practice for improving the fit of a ML model. These transformations can enhance model capacity, minimize noise, and provide better model stability. The most common transformations in the data curation phase involve data normalization, nonlinear transformations, reducing the impact of the outliers, or completing missing values. On the other hand, there are transformations aimed at enriching models through the addition of new variables or data augmentation, encoding of categorical variables, or applying dimensionality reduction techniques. Another point to study is the selection of training and testing data proportions because it significantly impacts model performance. Although it has been demonstrated that the 70%–30% or 80%–20% split is effective in many cases, we need to explore situations where this ideal scenario is not present and examine whether modifying the data can compensate for the hypothesis that fewer training data would result in worse outcomes. Whatever the decision regarding the chosen transformation, characterizing the dataset is essential for optimizing the preprocessing stage to improve the performance of ML models. This targeted approach enables ML algorithms to enhance their generalization ability and maximize model performance.

In this work, we propose the use of MtL to guide the data preprocessing step. Meta-learning's ability to effectively summarize and characterize information facilitates a better understanding of how and why ML models make certain decisions [5]. In this way, using MtL allows us to leverage this knowledge to improve the performance and adaptability of ML models by learning from prior experience. If we want to work in an environment where MtL is applied, the first step is to build a meta-dataset. In this meta-dataset, each meta-example is a dataset in which an ML model has been used to solve a specific problem. The meta-features of the metadata would be attributes of each meta-example, such as characterizations of the datasets acting as examples or the applied ML model. One possible meta-target is the performance of ML models on each meta-example. This performance can be measured using different metrics depending on the type of algorithm used or the nature of the problem being addressed. Once the meta-dataset is constructed, ML algorithms are applied to build meta-models for various applications. Among the applications of MtL, we highlight its intuitive use as a recommendation system for the multi-cash problem, allowing evaluations of which ML algorithms are most suitable for each problem [11–14]. In this regard, the most commonly used algorithm at the meta-level is K-Nearest Neighbors, grouping meta-data under the assumption that the performance of algorithms will be similar for datasets with similar meta-features [15]. But also, MtL can be used to determine model hyper-parameters [16], identify the most significant feature selection algorithm [15, 17], gather datasets with similar meta-features that may be useful in transfer learning applications, or estimate

expected model performance in a particular problem [18, 19].

The effectiveness of MtL largely depends on the type and quality of the meta-features. The first challenge in this topic started with the lack of standardization of the description of the meta-features, which would facilitate comparison and reproducibility of scientific studies. One of the first works in which a characterization of metadata through the analysis of meta-features was given was [20]. Additionally, the meta-features were organized into categories. In [21], the authors proposed a systematic framework that decomposes a meta-feature into three components: meta-function, object, and post-processing. This framework allows for the automatic generation of meta-features by applying various combinations of functions, such as entropy, mutual information, and correlation to data objects, followed by post-processing techniques. In [22], the use of clustering measures of meta-features was proposed to reduce the computational cost to calculate them while maintaining or improving performance. In [9], a meta-feature selection strategy based on unsupervised correlation analysis was proposed to get a reduced subset of meta-features that are optimal in computational cost without losing accuracy. Recently, in [10], the development of a systematic taxonomy of the meta-features was given, presenting a comprehensive list of different meta-features used in the literature, categorizing them, and discussing some aspects of their usage. The presented taxonomy distinguishes between six categories of meta-features:

1) General, which are commonly used and simple to extract: essentially dataset metadata, such as class frequencies and number of features.
2) Statistical meta-features include statistical measures of the data: mean of each feature, their standard deviation, kurtosis, etc.
3) Information-theoretic meta-features are mainly related to Shannon's entropy, both individual and joint for each feature and classes.
4 Model-based meta-features measure the complexity of a decision tree model applied to the data, for example, the number of nodes in the tree for a particular feature.
5) Landmarking meta-features measure the performance of simple ML models such as Naive Bayes or k-Nearest Neighbor.
6) The final group includes other meta-features present in the literature that may have too high a computational cost but may be useful in certain problems. This includes three families: Clustering, Complexity, and Miscellaneous.

The described systematization has been implemented in a software package in Python and R in [23, 24].

Despite the advances in the systematic extraction and categorization of meta-features, existing studies exhibit several limitations that constrain the practical and theoretical impact of MtL in data preprocessing optimization. First, while many works have focused on the development and classification of meta-features, there remains a lack of empirical studies that directly link specific meta-feature subsets to the effectiveness of concrete data transformations. That is, although taxonomies are available, there is limited evidence on how these meta-features influence the performance of different preprocessing strategies across diverse tasks and data distributions, particularly in imbalanced and multiclass settings. Second, most prior research assumes ideal experimental conditions, such as balanced datasets or standard training/testing splits. This fails to capture scenarios where data scarcity, noise, or imbalance are prominent, which are precisely the conditions under which preprocessing decisions are most critical. Lastly, computational cost remains an underexplored dimension in the

evaluation of meta-feature utility. While some efforts such as meta-feature selection through correlation analysis or clustering have attempted to address this, comprehensive trade-off analyses between cost and predictive value are still scarce. These shortcomings highlight the need for more robust experimental designs that incorporate diverse data conditions, evaluate meta-feature impact across a wider range of transformations, and systematically analyze the interaction between meta-feature types and transformation strategies. We present a study on which combinations of training and testing yield the most and the best correlated meta-features with the response variables. This will also allow us to examine the types of meta-features and, consequently, decide on the type of transformation that may be most effective in each case.

3. Materials and methods

3.1. Datasets

We have used an unbalanced set of 99,720 events associated with IP addresses provided by the CERT of the Spanish National Cybersecurity Institute (INCIBE), dated May, 2021. The anonymized dataset is available online at `https://github.com/amunc/IP_datasets`. This dataset has been studied previously [7, 8], allowing the determination of the most optimal set of features for the multi-class classification problem, as well as the ML algorithms that yielded the best results. The events are divided in 4 levels of maliciousness. The label distributions for the base data set is shown in Table 1.

Table 1. Distribution of labels of maliciousness.

Maliciousness	Frequency	Percentage (%)
1	8402	8.425
3	24943	25.013
6	54437	54.59
9	11938	11.971

We describe the features used and extracted for the study and then proceed to describe how the meta-datasets have been created. The first set of features is denoted as $\mathcal{F}_B$ and includes the following information: source IP address, timestamp of the event, and geolocation (country of the IP address and its latitude and longitude). Since not all of these features are numerical, we have applied some transformations to be able to use them. Following the work in [25], we added features related to the distribution of maliciousness in the networks present by the definition of a network as a CIDR/24 network. Then, the f_c feature is added to all events for each c class as defined in Eq (3.1). This set of features will be denoted by $\mathcal{F}_N$.

$$f_c = \frac{\text{IPs in the same network belonging to class c}}{\text{IP addresses in the network}}. \tag{3.1}$$

We also apply the work in [26] by creating features according to the frequency relative to the maliciousness class. The value of the feature f_k for the event i is called f_{ki}. The normalized frequency of the value f_{ki} relative to the maliciousness class c is defined in Eq (3.2).

$$NF(f_{ki}) = \frac{\text{Events of class } c \text{ with } f_k \text{ equal to } f_{ki}}{\text{Events of class } c}. \tag{3.2}$$

Then, we select the country code and the fractioned IP address and compute their normalized frequencies for each class. This set of features will be denoted as $\mathcal{F}_F$. The values of $\mathcal{F}_F$ and $\mathcal{F}_N$ have been calculated using only training data.

A summary of the set of features is included in Table 2.

Table 2. Feature sets.

Notation	Description	Cardinality
$\mathcal{F}_B$	Geolocation and temporal information	8
$\mathcal{F}_N$	Proportion of IPs in the same network with a certain class	4
$\mathcal{F}_F$	Normalized frequency of IP and country for each class	20

In this work, we have applied a data processing method over the original dataset to obtain new datasets from existing ones. With the described feature sets, six datasets have been created, considering the features individually, in pairs, and all together ($F_B, F_F, F_N, F_B \cup F_F, F_B \cup F_N, F_N \cup F_F, F_B \cup F_F \cup F_N$). In this way, although we know what is the best feature combination and ML algorithm by [7, 8], we can determine whether by applying preprocessing data transformation, the consequent variation in the value of the meta-features can improve the performance even for feature sets and ML algorithms that did not yield the best results in previous research. We also aimed to examine whether this approach can compensate for the typically poorer fits obtained with highly imbalanced training and testing distributions. Therefore, for each of the described datasets, which were created by feature clustering, six different partitions were made, combining various proportions of training and testing combinations (ttc). Table 3 shows the distribution of partitions for each group after splitting each dataset into three parts, according to the event date. We have created three groups, each corresponding to one of the three 10-day intervals of the month and tested each combination of train-test groups.

Table 3. Distribution by training and testing combinations.

ttc	Instances train (%)	Instances test (%)
0	16200 (38.66%)	25696 (61.34%)
1	16200 (21.88%)	57827 (78.11%)
2	25696 (61.34%)	16200 (38.66%)
3	25696 (30.76%)	57827 (69.23%)
4	57827 (78.11%)	16200 (21.88%)
5	57827 (69.23%)	25696 (30.76%)

3.2. Meta-features

One of the objectives of this work is to determine whether meta-features can be used to guide data preprocessing that improves the performance of ML models. Then, these have been calculated for each

of the 42 datasets obtained by combining the feature sets and ttc described in the previous section. 84 meta-features have been calculated following [10] using the meta-feature Extractor (MFE) developed in [23,24]. A brief description of the meta-features that are computed is included in the Supplementary material section.

Additionally, each meta-feature has been manually categorized based on the cost of modification and the possible control of the impact of the modification in the value of the meta-feature. Both categories are divided into three levels. We assign a 1 in the cost when it is considered easy to modify. Conversely, a 3 is assigned when the process of modifying the value of the meta-feature is considered difficult or costly; see Figure 1.

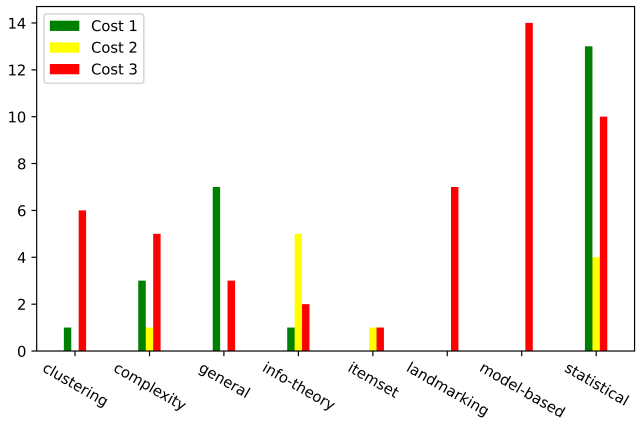

Figure 1. Distribution of the meta-features by the cost of the modification.

On the other hand, we assign 1 if the control over the impact of the modification is low. Conversely, a 3 is assigned if the impact of the modification can be practically predefined; see Figure 2. For example, with the mean of a feature, scaling can be applied to set the new value within the desired data range, which is a 3. Modifying the Calinski-Harabasz index (a clustering metric) would require removing or transforming data, which would affect the clustering itself and consequently the index value, so it is not easy to establish a value (or range of values) for the meta-feature; this is a 1.

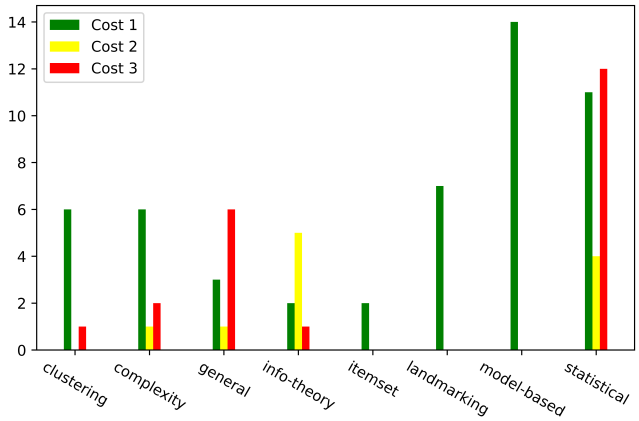

Figure 2. Distribution of the meta-features by the impact of the modification.

The categorization of the cost and the impact of the meta-features is included in the Supplementary material section.

3.3. Meta-dataset

The No Free Lunch Theorem states that it is necessary to experiment with and validate different algorithms to find the most suitable for the problem at hand [27]. Also, regarding the study of how to measure the importance of the meta-features in the performance of the ML algorithms, in [28], the impact of meta-features on ML model performance was studied, concluding that the effect is not uniform (either negative or positive), or equally significant depending on the algorithms used. On the other hand, although there is a growing interest in deep learning approaches to MtL, recent surveys and technical reviews highlight important limitations in terms of data requirements, computational cost, and training stability [29–31]. Furthermore, much of the recent literature shows that many deep MtL methods remain experimental and face challenges regarding observability, scalability, and applicability in constrained scenarios. It is also worth noting that deep learning in MtL is typically explored as a meta-learner architecture itself, rather than being used to generate or characterize instances within a meta-dataset. Given the limited size of our dataset and the nature of the task, we consider that adopting deep MtL architectures would not offer a clear performance or interpretability advantage, and could introduce unnecessary overhead. For the above reasons, we have constructed several ML models comparing well-established and interpretable methods: Gaussian Mixture Models (GMM) [32], Extreme Gradient Boosting (XGB) [33], and models generated using the Autosklearn library [3]. This choice was based on the fact that these algorithms have already been studied in other works related to the dataset used, and they have presented different performance results. In this way, we know which of them performs best in terms of performance and computational cost. This allows us to assess whether they can be improved and provides a robust baseline for comparison.

We select hyper-parameter configurations through Random Search [34] with 25 iterations: For each model and dataset, we evaluate 25 different randomly selected hyper-parameter configurations and select the one with the highest accuracy to make predictions on the testing subset. The evaluation of the hyper-parameter configurations will be carried out by holding out a fraction (20%) of the training dataset.

Table 4 shows the optimized hyper-parameters in both models and their possible values.

Table 4. Model hyperparameters.

Model	Hyperparameter	Range
GMM	Number of components	[20, 400]
XGBoost	n_estimators	[10, 500]
	max_depth	[1, 10]
	learning_rate	[0.0001, 0.3]
	subsample	[0.5, 1]
	colsample_bytree	[0.5, 1]
Autosklearn	All available, removed Kernel PCA and Extra trees preprocessing.	Default range

The response variable commonly used in MtL studies is accuracy. In this work, we have also

included four additional response variables that are not typically used in all MtL studies but are essential for imbalanced datasets. This will provide a more comprehensive and less biased perspective of the models' capabilities. We will use the Accuracy, F_1-score, Matthew's correlation coefficient (MCC) [35], Sensibility, and Specificity. Although some of these variables are correlated, we chose to conduct the study with all of them because, in studies related to cybersecurity, it is crucial to include metrics that allow us to assess the models' ability to predict false positives and false negatives, as the consequences of these can drastically affect the impact of an erroneous prediction and the proposed action policies.

3.4. Research questions

Below are the research questions:

1) Is the value of the meta-features statistically significantly correlated with the performance variables of ML models?
   Suppose the answer to the question is affirmative. In that case, the initial hypothesis regarding the possibility of conducting a study on meta-features to guide a data transformation process that improves model performance requires addressing several questions.
2) Which datasets, based on ttc, have the most significant potential for improvement?
   a) In which ttc do we find the highest percentage of meta-features statistically significantly correlated with the five response variables?
   b) In which ttc is the highest average absolute correlation coefficient observed among the meta-features that are statistically significantly correlated with the five response variables?
3) Are the conclusions obtained from the previous questions consistent across different ML models?
4) What is the type of meta-features that are statistically significantly correlated with the performance variables of ML models in terms of impact and cost?
5) Which data preprocessing techniques will have the most significant impact based on the answers to the previous questions?
6) Has the performance of the ML models been enhanced by the transformations implemented based on the study of meta-features?

3.5. Methodology

1) The initial analysis involved applying ML models to the created datasets, storaging the performance metrics, and calculating the meta-features of each dataset, resulting in the creation of metadata.
2) Once this process was completed, the next step was to calculate the correlation coefficient between the value of the meta-features and each model performance variable using Pearson correlation coefficients. Subsequently, we determined which of these meta-features had a statistically significant correlation with all response variables simultaneously, using inferential analysis with a significance level of $\alpha = 0.05$.
3) The following analysis involved calculating the percentage frequency of meta-features that were statistically significantly correlated for each of the datasets created by ttc. Additionally, the mean absolute correlation coefficient was computed for this selected subset of meta-features, and a frequency study on the type of meta-features correlated the impact and cost.

4) Then, the data preprocessing transformations that may be efficient for improving model performance were identified.
5) Once these transformations were applied, the new performance metrics were computed.
6) Moreover, to determine whether the selected preprocessing transformations yielded a statistically significant improvement in performance, inferential tests were conducted. Performance gains were computed for each metric, across all preprocessing techniques, models, and task configurations. A one-sided one-sample t-test was applied when the data followed a normal distribution, and its non-parametric counterpart, the one-sided Wilcoxon signed-rank test, was used otherwise. Normality was assessed using the Shapiro–Wilk test. The significance level was $\alpha = 0.05$.

In addition to optimizing predictive performance, a central concern in modern ML is ensuring the safety of models in accordance with ethical and regulatory standards. To this end, we integrated a part of the SAFE framework [36, 37]. This framework provides a comprehensive statistical approach to assess the safety of AI applications, aligned with current regulatory principles such as the European Union's AI Act. The authors developed a set of statistical metrics based on a Rank Graduation (RG) divergence measure, which enables the evaluation of four key dimensions of a safe AI system: Sustainability (Rank Graduation Robustness, RGR), Accuracy (Rank Graduation Accuracy, RGA), Fairness (RGA-Parity), and Explainability (Rank Graduation Explainability, RGE). In this study, the SAFE metrics were applied both before and after the most effective preprocessing techniques, as identified by metafeature-based correlation analyses. Consequently, it provided a clear comparative basis for evaluating whether preprocessing contributes to the responsible improvement of AI systems. In the analyses, the calculation of SAFE metrics was limited to Rank Graduation Robustness (RGR), as it reflects a key operational requirement of our autonomous system for detecting malicious IP addresses: robustness to input perturbations. In the cybersecurity domain, small variations in traffic attributes, whether due to noise, evasion attempts, or incomplete data, can be exploited by adversaries. As a result, it is essential to empirically assess the model's resilience to such perturbations, making RGR a relevant and necessary metric. Other SAFE dimensions were not computed for justified reasons. Rank Graduation Accuracy (RGA), while designed to measure changes in model ranking, is not applicable here, as the study does not involve inter-model or cross-dataset ranking. Instead, performance was evaluated using standard absolute metrics such as accuracy, F_1-score, MCC, sensitivity, and specificity, which are sufficient to assess the improvements brought by each preprocessing technique individually. Consequently, RGA does not add interpretive value in this context. Similarly, RGA-Parity was excluded because, although the dataset includes geolocation data, there is no risk of structural discrimination: maliciousness labels are not linked to protected attributes, nor are they used to make decisions affecting individuals or social groups. Lastly, Rank Graduation Explainability (RGE) was omitted, as the model operates as an autonomous detector in a fully automated blocking environment, where interpretability is neither required for business purposes nor mandated by regulatory standards.

All experiments have been carried using Python 3.10.

3.6. Techniques of modification

To check whether modifying the values of the meta-features according to their correlation with the response variables may improve their values, we have applied certain transformations on the data to modify the values of the meta-features for each model in the combination of training and testing with the highest significant correlation. Table 5 enumerates the transformations applied in the experiments. We describe the applied techniques below.

Table 5. Proposed transformations.

Name	Description
Scaler [38]	Scales range of features to [-1.1, 1]
PCA [39]	Applies PCA to features and preserves all components
SelectKBest [40]	Keeps 50% of features with highest mutual information
Corr increase	Duplicates features and multiplies them by -1
Quantile transform [41]	Applies a quantile transformation to features to approximate a uniform distribution

1) Scaler scales each value x in a feature X^i to a range defined by $[lower, upper]$ using the formulas in Eq (3.3).

$$std(x) = \frac{x^i - min(X^i)}{max(X^i) - min(X^i)}$$
$$x_{scaled} = std(x) * (upper - lower) + lower. \tag{3.3}$$

2) PCA projects the original data D to a new space D' with a orthogonal basis (the principal components, A) that maximize variance in the data. The new data X' are defined in Eq (3.4).

$$D' = AD. \tag{3.4}$$

A corresponds to the eigenvectors of the covariance matrix of D.

3) SelectKBest computes the mutual information I between each feature X^i and the labels Y, defined in Eq (3.5). H denotes the entropy.

$$I(X^i; Y) = H(X^i) - H(X^i|Y). \tag{3.5}$$

The preprocessing will remove 50% of the features with the lowest values of $I(X^i; Y)$.

4) Corr increase: For each feature X^i, it adds another feature $X^{i'}$ such that $X^{i'} = -X^i$.

5) Quantile transform attempts to map the distribution of a feature X^i to a new feature X^i_r that follows a random uniform distribution. The mapping is carried out using Eq (3.6). F denotes the cumulative distribution function of X^i, and G denotes the quantile function of the target uniform distribution. The quantile function is defined in Eq (3.7). It maps a probability $p \in [0, 1]$ of a random variable (in our case, the feature X^i) to a value v such that $P(X^i \leq v) = x$.

$$X^i_r = G^{-1}(F(X^i)). \tag{3.6}$$
$$G(p)_{X^i} = v. \tag{3.7}$$

It must be noted that we have only taken into account meta-features with a modification cost (see Section 3.2) of 1 or 2, since a cost of 3 would imply a transformation too costly and/or complex to be feasible in practice.

4. Results and discussion

4.1. Performance metrics

The values of the obtained response variables (average) with each ML model are included in Tables 6–8.

Table 6. Response variables (average) in Autosklearn.

ttc	Autosklearn				
	MCC	F_1-score	Accuracy	Sensitivity	Specificity
0	.2612	.5278	.5278	.5278	.8469
1	.1878	.5109	.5109	.5109	.8413
2	.3292	.5875	.5875	.5875	.8668
3	.1832	.4792	.4792	.2989	.5709
4	.2989	.5709	.5709	.5709	.8624
5	.2408	.5157	.8429	.5157	.5157

Table 7. Response variables (average) in GMM.

ttc	GMM				
	MCC	F_1-score	Accuracy	Sensitivity	Specificity
0	.2352	.4549	.4549	.4549	.8183
1	.1541	.4485	.4485	.4485	.8161
2	.3950	.6192	.6192	.6192	.8730
3	.2016	.4813	.4813	.4813	.8271
4	.4251	.6525	.6525	.6525	.8841
5	.3121	.5571	.5571	.5571	.8523

Table 8. Response variables (average) in XGB.

ttc	XGB				
	MCC	F_1-score	Accuracy	Sensitivity	Specificity
0	.3898	.6057	.6057	.6057	.8685
1	.2742	.5784	.5784	.5784	.8594
2	.5032	.7085	.7085	.7085	.9028
3	.2693	.5455	.5455	.5455	.8485
4	.4865	.7123	.7123	.7123	.9041
5	.3683	.6161	.6161	.6161	.8720

Autosklearn demonstrates significant improvement potential. However, such improvement must be substantial given the high computational cost compared to the other algorithms studied. Given the results, XGB has achieved the best results. Then, we need to examine if there is room for improvement

or if XGB has reached its maximum applicability with this dataset. On the other hand, GMM shows a significant potential for improvement. Moreover, as intuitively expected, the combination ttc=4 yields the best performance. It is important to note that there is a performance difference between the partitions ttc=5 and ttc=2, although their ttc does not significantly differ from ttc=4. Another point is if the results of ttc=0, 1, and 3 can be improved despite having a smaller proportion of training instances.

4.2. Decision about the modifications of the meta-features

We will study which combination has the highest potential in order to increase the performance of the ML algorithms. With this aim, we propose the following questions:

a) Which ttc has the most meta-features with a significant correlation to the response variables?
b) Which ttc has the meta-features with the highest significant absolute correlation coefficient?

We have included the results related to the correlations of the value of the meta-features with all response variables for Autosklearn in Table 9.

Table 9. Correlations with Autosklearn.

ttc	Autosklearn	
	% Significative Meta-features	Average of absolute correlation's coefficient
0	0	0
1	0	0
2	0	0
3	0	0
4	25%	0.7961
5	0%	0

In the case of Autosklearn, the decision on which ttc can be modified by data preprocessing to improve model performance is clear. There is only one combination that aligns with the one that achieved the best results, where we can find meta-features that are significantly correlated with all response variables. The types of statistically significant meta-features in model Autosklearn are included in Figure 3. The cost and the impact of the modification of the statistically significant meta-features in Autosklearn are included in Figure 4.

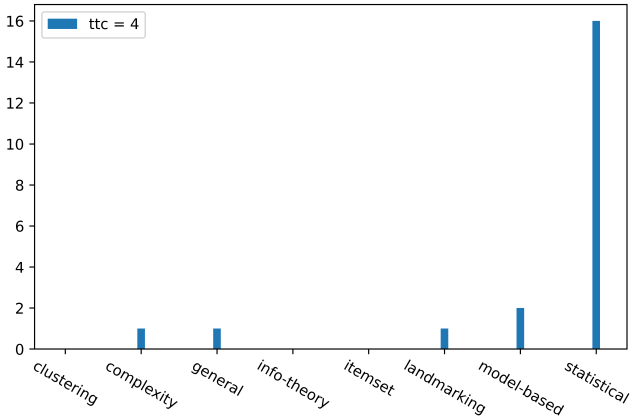

Figure 3. Type of statistically significant meta-features in model Autosklearn.

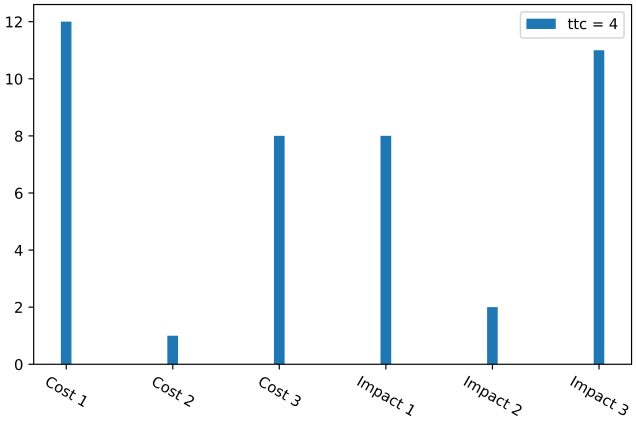

Figure 4. Cost and impact of modification of the statistically significant meta-feature in model Autosklearn.

In Table 10, we list which meta-features we intend to modify for Autosklearn and the average cost and impact of them in each transformation. The transformations that could potentially yield the most significant improvement are those where the difference between impact and cost is the largest. In this case, Scaler, Quantile Transform, and SelectKBest stand out. Therefore, based on the categorization of impact and cost and the correlations obtained, the initial hypothesis is that one of these three transformations is likely to result in the most significant gains.

Table 10. Transformations and meta-features intended to be modified for Autosklearn.

ttc	Transformation	Meta-features	Average cost	Average impact
	Scaler	g_mean, iq_range, mad, max, mean , median, min, range, sd, t_mean, var	1	3
4	PCA	cov, max, mean, median, min, range, t_mean	1.14	2.85
	SelectKBest	cov, g_mean, iq_range, mad, max, mean, median, min, range, sd, t_mean, var	1.08	2.91
	Corr increase	cov, max, mean, median, min, t_mean	1.16	2.83
	Quantile transform	cov, g_mean, iq_range, mad, max, mean, median, min, range, sd, t_mean, var	1.08	2.91

We have included the results related to the correlations of the value of the meta-features with all response variables for GMM in Table 11.

Table 11. Correlations with GMM.

ttc	GMM	
	% Significative Meta-features	Average of Absolute correlation's coefficient
0	9.52%	0.9386
1	8.33%	0.9451
2	7.14%	0.8176
3	7.14%	0.8295
4	3.57%	0.8717
5	20.23%	0.8482

In the case of models created with GMM, several combinations show potential for performance improvement. For instance, ttc=5 has a high percentage of meta-features that are statistically correlated with the response variables. Additionally, combinations ttc= 0 and ttc= 1 have around 10% correlated meta-features but present the highest average of absolute correlation coefficients. Although fewer in number, their adjustment could lead to more significant performance gains. None of these combinations initially achieved the best results, making them a good case for exploring whether preprocessing will surpass the highest performance obtained.

The types of statistically significant meta-features in model GMM are shown in Figure 5.

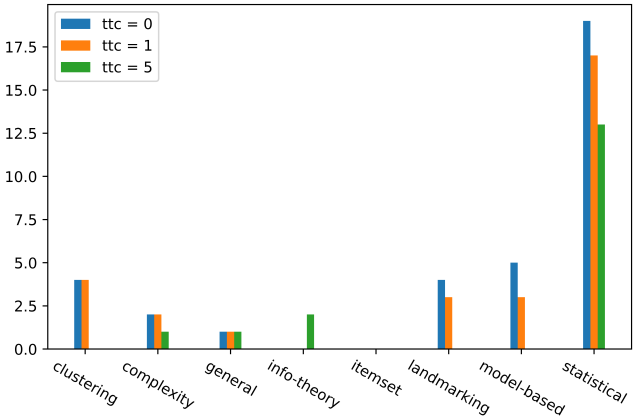

Figure 5. Types of statistically significant meta-features in model GMM.

The cost and the impact of the modification of the statistically significant meta-features in model GMM are shown in Figure 6.

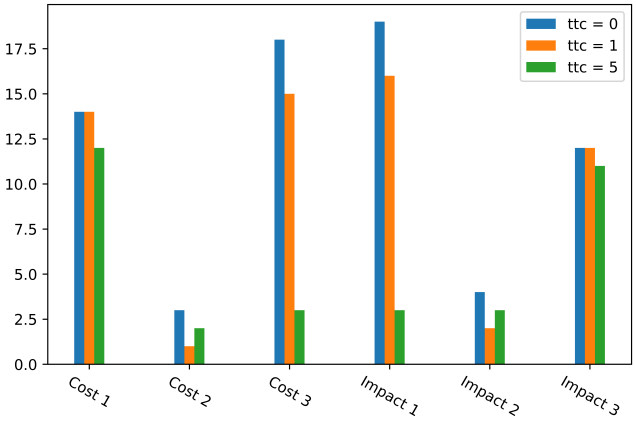

Figure 6. Cost and impact of modification of the statistically significant meta-features in model GMM.

Summarizing, we have several meta-features with low modification costs, whose modifications have a predictable impact on their value. In Table 12, we list which significant correlated meta-features are modified by each proposed transformation.

Table 12. Transformations and meta-features intended to be modified for GMM.

ttc	Transformation	Meta-features	Average cost	Average impact
0	Scaler	cov, g_mean, iq_range, mad, max, mean, median, min, range, sd, skewness, t_mean, var	1.15	2.84
	Quantile transform	cov, nr_norm, g_mean, iq_range, mad, max, mean, median, min, range, sd, skewness, t_mean, var	1.14	2.71
	PCA	cov, iq_range, nr_norm, mad, max, mean, median, min, range, sd, skewness, t_mean, var	1.15	2.69
	Corr increase	cor, cov, g_mean, iq_range, mad, max, mean, median, min, range, sd, skewness, t_mean, var	1.21	2.78
	SelectKBest	cor, cov, g_mean, iq_range, mad, max, mean, median, min, nr_norm, range, sd, skewness, t_mean, t2, var	1.18	2.62
1	Scaler	iq_range, mad, max, mean,median, min, nr_norm, range, sd, skewness, t_mean, var	1.08	2.75
	Quantile transform	g_mean, iq_range, mad, max, mean, median, min, nr_norm, range, sd, skewness, t_mean, var	1.07	2.76
	PCA	iq_range, mad, max, mean, median, min, nr_norm, range, sd, skewness, t_mean, var	1.08	2.75
	Corr increase	g_mean, iq_range, mad, max, mean, median, min, range, sd, skewness, t_mean, var	1.08	2.91
	SelectKBest	g_mean, iq_range, mad, max, mean, median, min, nr_norm, range, sd, skewness, t2, t_mean, var	1.07	2.71
5	Scaler	cov, iq_range, mad, max, mean, median, min, range, sd, t_mean, var	1.09	2.90
	Quantile transform	cov, g_mean, iq_range, mad, max, mean, median, min, range, sd, t_mean, var	1.08	2.91
	PCA	cov, g_mean, max, mean, median, min, range, t_mean	1.125	2.875
	Corr increase	cov, max, mean, median, min, t_mean	1.16	2.83
	SelectKBest	cov, g_mean, iq_range, mad, max, mean, median, min, mut_inf, range, sd, t_mean, var	1.15	2.84

For ttc=0, the initial hypothesis is that the transformations most likely to yield better results are Scaler, Quantile Transform, and Corr Increase. In the case of ttc=1, data points to Corr Increase and Quantile Transform. Finally, for ttc=5, the difference between impact and cost suggests that the transformations expected to achieve the best results are Quantile Transform and Scaler.

The results related to the correlations of the value of the meta-features with all response variables for XGB are included in Table 13.

Table 13. Correlations with XGB.

ttc		XGB
	% Significative Meta-features	Average of absolute correlation's coefficient
0	0	0
1	0	0
2	3.57%	0.8009
3	1.19%	0.8188
4	4.76%	0.8185
5	5.9552 %	0.778

For models created with XGB, we can identify the ttc=5 combination as having the highest percentage of meta-features significantly correlated with the response variables. However, ttc=3 exhibits the highest mean absolute correlations. On the other hand, ttc=4 balances both aspects, with performance close to the mentioned combinations. Therefore, it seems appropriate to apply data transformations for all three cases. Additionally, ttc=4 predominantly achieves the best results in the study, making it interesting to explore whether further improvements can be achieved

The types of statistically significant meta-features in model XGB is shown in Figure 7.

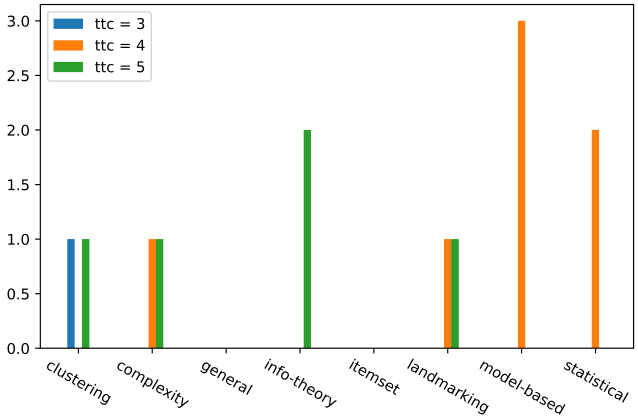

Figure 7. Types of statistically significant meta-features in model XGB.

The cost and impact of the modification of the statistically significant meta-features in model XGB is shown in Figure 8.

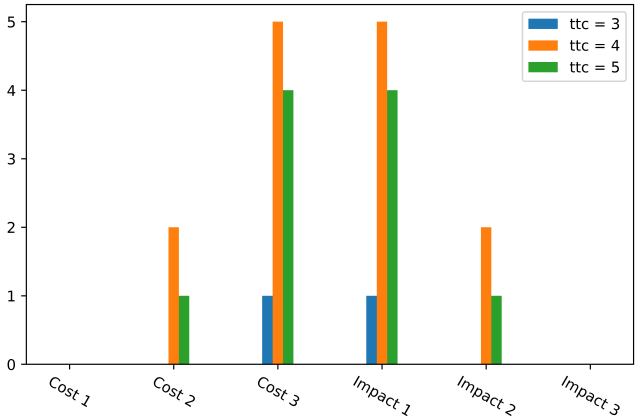

Figure 8. Cost and impact of modification of the statistically significant meta-features in model XGB.

In Table 14, we list which meta-features we intend to modify for XGB with each applied transformation.

Table 14. Transformations and meta-features intended to be modified for XGB.

ttc	Transformation	Meta-features	Average cost	Average impact
3	Scaler Quantile transform PCA Increase corr SelectKBest	All significant have cost 3	3	1
4	Scaler	cor, nr_cor_attr	2	2
	Quantile transform	cor, nr_cor_attr	2	2
	PCA	cor, nr_cor_attr	2	2
	Increase corr	cor, nr_cor_attr	2	2
	SelectKBest	cor, nr_cor_attr	2	2
5	Scaler	mut_inf	2	2
	Quantile transform	mut_inf	2	2
	PCA	mut_inf	2	2
	Increase corr	mut_inf	2	2
	SelectKBest	mut_inf	2	2

In this case, differences between cost and impact are observed only for ttc=3. Therefore, this would be the only combination where, a priori, we might expect any gain. However, the modification cost is high, and there are no differences between preprocessing techniques. This may provide insight into the hypothesis regarding XGB: improving the original performance with this algorithm is either very costly or unlikely to be significant. It raises the question of whether XGB has reached its performance limit.

For the reasons stated, we will work with data transformations aimed at modifying the value of

the meta-features for the combinations ttc=4 with Autosklearn, tt=0,1, and 5 for GMM, and ttc=3,4,5 for XGB. Quantile Transform will be the clear choice if preprocessing optimization is required, as it consistently shows strong potential for improvement across all the cases studied. However, in this study, we will apply all transformations to examine whether the proposed hypotheses hold true.

4.3. Enhancing performance metrics

This section will examine whether the transformations applied during the preprocessing stage could help maintain or improve the original performance. We will assess whether there has been an improvement and whether the gain has surpassed the best original performance, regardless of the ttc that achieved it. Additionally, we will investigate whether the hypotheses regarding the most optimal type of transformation based on the meta-features are confirmed.

In the case of Autosklearn, as shown in Table 15, Quantile Transform is the only method that improves the original results. It increases the performance originally obtained with ttc=4 and surpasses the best results that had been achieved with ttc=2. Thus, one of the transformations hypothesized to be optimal has indeed yielded the best results.

Table 15. New response variables (average) in Autosklearn.

ttc	Autosklearn				
	Transformation	MCC	F_1-score	Accuracy	Sensitivity	Specificity
4	Scaler	.2512	.5876	.5876	.5876	.8625
	Select K-best	.2570	.51	.51	.51	.8366
	PCA	.1996	.4969	.4969	.4969	.8323
	Corr.increase	.2793	.5579	.5579	.5579	.8526
	Quantile transf	.3617	.5970	.5970	.5970	.8656

In the case of GMM (Table 16), for ttc=0, all preprocessing techniques improve the performance of the original models. Moreover, the Scaler and Quantile transformations, two of the methods previously identified as most optimal, surpass the best original performance of GMM, which had been achieved at ttc=4. This is significant because ttc=0 represents a combination with a low training proportion. Nevertheless, the transformations enhanced model performance to the extent that they even outperform the best results from the entire original experimentation, previously achieved with XGB at ttc=2. For ttc=1, all preprocessing techniques also improve the performance originally achieved by this partition for GMM. Quantile Transform once again proves to be the technique that provides the greatest improvement, and it had been anticipated as one of the optimal choices. However, it is important to note that in this case, the maximum performance obtained in ttc= 4 with the original models is not surpassed, either for GMM or XGB. Finally, for ttc=5, three preprocessing techniques improved the original performance, with two of them being those proposed as most optimal based on the correlation of meta-features with the response variables. Quantile Transform once again provided the greatest improvement. However, as in previous cases, the maximum performance of the original models could not be surpassed.

Table 16. New response variables (average) in GMM.

ttc	GMM				
	Transformation	MCC	F_1-score	Accuracy	Sensitivity	Specificity
0	Scaler	.4674	.6803	.6803	.6803	.8934
	Select K-best	.3698	.5804	.5804	.5804	.8601
	PCA	.3969	.5927	.5927	.5927	.8642
	Corr.increase	.2898	.4944	.4944	.4944	.8314
	Quantile transf	.5305	.7160	.7160	.7160	.9053
1	Scaler	.3905	.6434	.6434	.6434	.8811
	Select K-best	.2972	.5431	.5431	.5431	.8477
	PCA	.388	.6114	.6114	.6114	.8704
	Corr.increase	.2685	.4794	.4794	.4794	.8209
	Quantile transf	.4205	.6809	.6809	.6809	.8938
5	Scaler	.3278	.5952	.5952	.5952	.8650
	Select K-best	.2942	.5623	.5623	.5623	.8541
	PCA	.3296	.5961	.5961	.5961	.8653
	Corr.increase	.1777	.2756	.2756	2756	.9490
	Quantile transf	.3770	.6184	.6184	.6184	.8728

Table 17. New response variables (average) in XGB.

ttc	XGB				
	Transformation	MCC	F_1-score	Accuracy	Sensitivity	Specificity
3	Scaler	.3703	.62	.62	.62	.8733
	Select K-best	.3163	.5888	.5888	.5888	.8629
	PCA	.3296	.5961	.5961	.5961	.8653
	Corr.increase	.2846	.5855	.5855	.5855	.8618
	Quantile transf	.3823	.6206	.6206	..6206	.8735
4	Scaler	.4174	.6608	.6608	.6608	.8869
	Select K-best	.3883	.6464	.6464	.6464	.8821
	PCA	.2479	.4591	.4591	.4591	.8197
	Corr.increase	.3754	.6492	.6492	.6492	.8830
	Quantile transf	.4520	.6919	.6919	.6919	.8973
5	Scaler	.3714	.6210	.6210	.6210	.8736
	Select K-best	.3119	.5879	.5879	.5879	.8626
	PCA	.3031	.5617	.5617	.5617	.8539
	Corr.increase	.3327	.6126	.6126	.6126	.8708
	Quantile transf	.4083	.6364	.6364	.6364	.8788

In contrast, for XGB (Table 17), the results are not as positive. Although for ttc=3, all transformations outperform the performance achieved by this partition in the original models, none surpass the adjustment level obtained with ttc=2, which was the highest achieved for the IP address

classification problem. Based on the meta-features, the combination ttc=3 was initially identified as the one with the greatest potential for improvement in XGB. This is confirmed for ttc=4, where none of the transformations surpass the results achieved originally. For ttc=5, only Quantile Transform and Scaler have improved performance, but neither of them exceeds the maximum performance level achieved with XGB. Nonetheless, Quantile Transform remains the technique that provides the greatest improvement.

The conclusions drawn are not stable across the analyzed ML models. GMM demonstrates the highest improvement rates, surpassing the best results achieved initially by XGB. This indicates that the meta-features have been effective in guiding the preprocessing techniques. On the other hand, XGB, which initially presented the best performance, has shown the slightest improvement, raising the question of whether it has reached its performance limit. This conclusion aligns with the results obtained from the study of the correlation of meta-features. Furthermore, the hypothesis that Quantile Transform was the most optimal preprocessing technique, based on the study of correlations and the categorization of meta-features by cost and impact, has been confirmed.

On the other hand, the greatest gains for both Autosklearn and GMM have been observed in MCC. The response variable with the least improvement was specificity. For XGB, the largest loss was also in MCC, while the smallest loss was in specificity.

4.4. Statistically significative improvements

The statistical testing conducted on the gain distributions reveals important distinctions between preprocessing techniques in terms of their ability to consistently improve classification performance (Table 18).

Among the five evaluated techniques, Quantile transformation clearly stands out as the most effective. It shows statistically significant improvements across all five performance metrics. This suggests that the technique robustly enhances both the overall classification ability and the balance between classes, even under varying models and task types. Standard scaling (Scaler) also demonstrates promising results, with significant gains detected in F_1-score and Sensitivity. Although the p-values for MCC and Accuracy were marginally above the 0.05 threshold, they may still indicate potentially meaningful trends, especially when considered in conjunction with the moderate gain magnitudes observed in Table 19. In contrast, the remaining techniques, PCA, SelectKBest, and Correlation-based increase, did not achieve statistical significance in any metric.

Table 18. Hypothesis testing for significative gain for each technique and metric. A one-sided t-test is used when normality is met, and a one-sided Wilcoxon test is used otherwise. Normality is assessed using the Shapiro-Wilk test. Significant improvements are shown in bold.

Preprocessing technique	Performance Variable	Statistic	p-value	Test
Scaler	MCC	1.692	0.058	t-test
	F_1-score	1.827	**0.045**	t-test
	Accuracy	1.742	0.053	t-test
	Sensitivity	1.827	**0.045**	t-test
	Specificity	1.532	0.072	t-test
PCA	MCC	-0.491	1.000	t-test
	F_1-score	12.000	0.656	Wilcoxon
	Accuracy	12.000	0.656	Wilcoxon
	Sensitivity	12.000	0.656	Wilcoxon
	Specificity	9.000	0.832	Wilcoxon
SelectKBest	MCC	18.000	0.812	Wilcoxon
	F_1-score	20.000	0.594	Wilcoxon
	Accuracy	21.000	0.500	Wilcoxon
	Sensitivity	20.000	0.594	Wilcoxon
	Specificity	20.000	0.594	Wilcoxon
Corr increase	MCC	-0.500	1.000	t-test
	F_1-score	12.000	0.656	Wilcoxon
	Accuracy	12.000	0.656	Wilcoxon
	Sensitivity	12.000	0.656	Wilcoxon
	Specificity	18.000	0.289	Wilcoxon
Quantile transf	MCC	2.822	**0.006**	t-test
	F_1-score	2.469	**0.012**	t-test
	Accuracy	2.422	**0.014**	t-test
	Sensitivity	2.469	**0.012**	t-test
	Specificity	2.374	**0.016**	t-test

Table 19. Average of gain (Δ=postprocessing (value)-preprocessing (value)) in each performance metric.

Model	ttc	Technique	MCC	F_1-score	Accuracy	Sensitivity	Specificity
Autosklearn	4	Scaler	-0.0477	.0167	.0167	.0167	.0001
	4	PCA	-0.0419	-0.0609	-0.0609	-0.0609	-0.0258
	4	SelectKBest	-0.0993	-0.0740	-0.0740	-0.0740	-0.0301
	4	Corr increase	-0.0196	-0.0130	-0.0130	-0.0130	-0.0098
	4	Quantile transf	.0628	.0261	.0261	.0261	.0032
GMM	0	Scaler	.2322	.2254	.2254	.2254	.0751
	0	PCA	.1346	.1255	.1255	.1255	.0418
	0	SelectKBest	.1617	.1378	.1378	.1378	.0459
	0	Corr increase	.0546	.0395	.0395	.0395	.0131
	0	Quantile transf	.2953	.2611	.2611	.2611	.0870
	1	Scaler	.2364	.1949	.1949	.1949	.0650
	1	PCA	.1431	.0946	.0946	.0946	.0316
	1	SelectKBest	.2339	.1629	.1629	.1629	.0543
	1	Corr increase	.1144	.0309	.0309	.0309	.0048
	1	Quantile transf	.2664	.2324	.2324	.2324	.0777
	5	Scaler	.0157	.0381	.0381	.0381	.0127
	5	PCA	-0.0179	.0052	.0052	.0052	.0018
	5	SelectKBest	.0175	.0390	.0390	.0390	.0130
	5	Corr increase	-0.1344	-0.2815	-0.2815	-0.2815	.0967
	5	Quantile transf	.0649	.0613	.0613	.0613	.0205
XGB	3	Scaler	.1010	.0745	.0745	.0745	.0248
	3	PCA	.0470	.0433	.0433	.0433	.0144
	3	SelectKBest	.0603	.0506	.0506	.0506	.0168
	3	Corr increase	.0153	.0400	.0400	.0400	.0133
	3	Quantile transf	.1130	.0751	.0751	.0751	.0250
	4	Scaler	-0.0691	-0.0515	-0.0515	-0.0515	-0.0172
	4	PCA	-0.0982	-0.0659	-0.0659	-0.0659	-0.0220
	4	SelectKBest	-0.2386	-0.2532	-0.2532	-0.2532	-0.0844
	4	Corr increase	-0.1111	-0.0631	-0.0631	-0.0631	-0.0211
	4	Quantile transf	-0.0345	-0.0204	-0.0204	-0.0204	-0.0068
	5	Scaler	.0031	.0049	.0049	.0049	.0016
	5	PCA	-0.0564	-0.0282	-0.0282	-0.0282	-0.0094
	5	SelectKBest	-0.0652	-0.0544	-0.0544	-0.0544	-0.0181
	5	Corr increase	-0.0356	-0.0035	-0.0035	-0.0035	-0.0012
	5	Quantile transf	.0400	.0203	.0203	.0203	.0068

4.5. Safe artificial intelligence

We present in Table 20 the average RGR values before and after preprocessing, offering insight into the impact of different data transformation techniques on model stability. A higher RGR (lower

divergence) implies that the model preserves the relative ordering of its output probabilities when the data are mildly distorted. While some pipelines, particularly those involving AutoSklearn, exhibited a decline in robustness, other configurations achieved clear improvements. For example, GMM-based pipelines consistently showed increased RGR values after the application of scaling or Quantile transformation, suggesting enhanced resilience to input perturbations. Similarly, several configurations with XGB (ttc=3) maintained or slightly improved RGR. These trends indicate that the choice of preprocessing method has a non-trivial and model-dependent effect on robustness. Interestingly, in certain cases where RGR decreased, this was accompanied by modest gains in predictive performance metrics (Table 19), pointing to a potential trade-off between accuracy and robustness.

Table 20. Average of RGR (preprocessing Vs post-processing).

Model	ttc	Technique	RGR (Before)	RGR (After)
Autosklearn	4	Scaler	.6187	.4763
	4	Quantile transf	.6187	.3744
GMM	0	Scaler	.4128	.5592
	0	Quantile transf	.4128	.4275
	1	Scaler	.3966	.4573
	1	Quantile transf	.3966	.4379
	5	Scaler	.5904	.5382
	5	Quantile transf	.5904	.4320
XGB	3	Scaler	.3133	.3204
	3	Quantile transf	.3133	.3374
	4	Scaler	.3310	.2624
	4	Quantile transf	.3310	.3018
	5	Scaler	.41	.3084
	5	Quantile transf	.41	.3164

5. Conclusions

The study of meta-features is a relevant area in ML because these meta-characteristics are used to describe and characterize the datasets that are used. In this paper, we have carried out a study evaluating how a meta-feature selection strategy based on correlation and modification costs and impacts can address unfavorable training and testing distributions. As a result, we have proposed a systematic evaluation framework for preprocessing techniques grounded in meta-feature correlation analysis. It integrates a cost-impact perspective and performance benchmarking across multiple metrics and transformations. Key contributions include:

1) An analysis of how preprocessing affects meta-feature correlations and classification performance.
2) A meta-learning approach to benchmark five preprocessing strategies without assuming linear effects.
3) The integration of a cost-impact framework linking preprocessing choices to resource considerations.
4) Statistical validation through significance testing and mixed-effects models.

5) A real-world application to cybersecurity, demonstrating the framework's practical relevance. A meta-feature approach can help overcome common ML limitations such as the existence of limited data, labels, and tuning constraints.

Future work will address whether the use of meta-features is beneficial across different datasets and ML algorithms such as deep neural and convolutional networks. Additionally, it would be relevant to systematize the process of categorizing meta-features by cost and impact and explore their relationship with the performance gains they can provide. Another interesting line would be to study the relationship between the conclusions and the degrees of conceptual drift in the datasets.

Author contributions

Noemí DeCastro-García: conceptualization, formal analysis, funding acquisition, investigation, methodology, project administration, resources, supervision, writing-original draft preparation, writing-review & editing; David Escudero García: data curation, formal analysis, investigation, methodology, software, validation, visualization, writing-original draft preparation, writing-review & editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have used Artificial Intelligence (AI) tools in the creation of this article. During the preparation of this work the authors used ChatGPT in order to verify that the abstract and highlights accurately reflected the content of the article. It was also used to review the grammar and the references. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Data availability statement

The dataset is available online in: https://github.com/amunc/IP_datasets.

For reproducibility, all the paper experiments implementation is public on this repository: https://github.com/amunc/Optimizing-meta-feature-computation-for-enhanced-fit-capacity-of-machine-learning-models-experiments/tree/main.

Acknowledgments

This research is part of the Strategic Project Data Science for an Artificial Intelligence Model in Cybersecurity (C073/23), resulting from the collaboration agreement between The Spanish National Cybersecurity Institute (INCIBE) and the University of León. This initiative is carried out within the framework of the Recovery, Transformation, and Resilience Plan, funded by the European Union (Next Generation).

Conflict of interest

The authors declare no potential conflict of interests.

References

1. M. Guven, Leveraging deep learning and image conversion of executable files for effective malware detection: A static malware analysis approach, *AIMS Mathematics*, **9** (2024), 15223–15245. http://doi.org/10.3934/math.2024739
2. M. Aljebreen, H. A. Mengash, K. Mahmood, A. A. Alhashmi, A. S. Salama, Enhancing cybersecurity in cloud-assisted Internet of Things environments: A unified approach using evolutionary algorithms and ensemble learning, *AIMS Mathematics*, **9** (2024), 15796–15818. http://doi.org/10.3934/math.2024763
3. M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, In: *Proceedings of the 29th international conference on neural information processing systems*, Cambridge: MIT Press, 2015, 2755–2763.
4. N. DeCastro-García, Castañeda, Á. L. Muñoz Castañeda, M. Fernández-Rodríguez, RADSSo: An automated tool for the multi-CASH machine learning problem, In: *Hybrid artificial intelligent systems, HAIS 2020*, Cham: Springer, 2020, 183–194. https://doi.org/10.1007/978-3-030-61705-9_16
5. C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, In: *Proceedings of the 34th international conference on machine learning (ICML)*, Cambridge: MIT Press, 2017, 1126–1135.
6. D. Kedziora, T.-D. Nguyen, K. Musial, B. Gabrys, On taking advantage of opportunistic meta-knowledge to reduce configuration spaces for automated machine learning, *Expert Syst. Appl.*, **239** (2024), 122359. http://doi.org/10.1016/j.eswa.2023.122359
7. M. V. Carriegos, N. DeCastro-García, D. Escudero, Towards supercomputing categorizing the maliciousness upon cybersecurity blacklists with concept drift, *Comput. Math. Methods*, **2023** (2023), 5780357. https://doi.org/10.1155/2023/5780357
8. N. DeCastro-García, D. E. García, M. V. Carriegos, A mathematical analysis about the geo-temporal characterization of the multi-class maliciousness of an IP address, *Wireless Netw.*, **30** (2024), 5033–5048. https://doi.org/10.1007/s11276-022-03215-2
9. A. Rivolli, L. P. F. Garcia, A. C. Lorena, A. C. P. L. F. de Carvalho, A study of the correlation of metafeatures used for metalearning, In: *Advances in computational intelligence*, Cham: Springer, 2021, 471–483. https://doi.org/10.1007/978-3-030-85030-2_39
10. A. Rivolli, L. P. F. Garcia, C. Soares, J. Vanschoren, A. C. P. L. F. de Carvalho, Meta-features for meta-learning, *Knowl.-Based Syst.*, **240** (2022), 108101. https://doi.org/10.1016/j.knosys.2021.108101
11. Y. Jeong, S. Lee, J. Lee, W. Choi, Data-efficient surrogate modeling using meta-learning and physics-informed deep learning approaches, *Expert Syst. Appl.*, **250** (2024), 123758. https://doi.org/10.1016/j.eswa.2024.123758
12. B. A. Pimentel, A. C. P. L. F. de Carvalho, A new data characterization for selecting clustering algorithms using meta-learning, *Inform. Sciences*, **477** (2019), 203–219. https://doi.org/10.1016/j.ins.2018.10.043

13. Y. H. Peng, P. A. Flach, C. Soares, P. Brazdil, Improved dataset characterisation for meta-learning, In: *Discovery Science*, Berlin: Springer, 2002, 141–152. https://doi.org/10.1007/3-540-36182-0_14

14. S.-N. Vo, T.-T. Vo, B. Le, Interpretable extractive text summarization with meta-learning and BI-LSTM: A study of meta learning and explainability techniques, *Expert Syst. Appl.*, **245** (2024), 123045. http://doi.org/10.1016/j.eswa.2023.123045

15. A. Filchenkov, A. Pendryak, Datasets meta-feature description for recommending feature selection algorithm, *Proceedings Of 2015 Artificial Intelligence And Natural Language And Information Extraction, Social Media And Web Search FRUCT Conference (AINL-ISMW FRUCT)*, Petersburg, Russia, 2015, 11–18. http://doi.org/10.1109/AINL-ISMW-FRUCT.2015.7382962

16. T. L. Marinho, D. C. do Nascimento, B. A. Pimentel, Optimization on selecting XGBoost hyperparameters using meta-learning, *Expert Syst.*, **41** (2024), e13611. http://doi.org/10.1111/exsy.13611

17. S. Shilbayeh, S. Vadera, Feature selection in meta learning framework, *2014 Science And Information Conference*, London, UK, 2014, 269–275. http://doi.org/10.1109/SAI.2014.6918200

18. A. Raghu, J. Lorraine, S. Kornblith, M. McDermott, D. Duvenaud, Meta-learning to improve pre-training, In: *Proceedings of the 35th international conference on neural information processing systems*, New Youk: Curran Associates Inc., 2021, 23231–23244.

19. D. Carneiro, M. Guimarães, M. Carvalho, P. Novais, Using meta-learning to predict performance metrics in machine learning problems, *Expert Syst.*, **40** (2023), e12900. http://doi.org/10.1111/exsy.12900

20. C. Castiello, G. Castellano, A. M. Fanelli, Meta-data: characterization of input features for meta-learning, In: *Modeling decisions for artificial intelligence*, Berlin: Springer, 2005, 457–468. https://doi.org/10.1007/11526018_45

21. F. Pinto, C. Soares, J. Mendes-Moreira, Towards automatic generation of metafeatures, In: *Advances in knowledge discovery and data mining*, Cham: Springer, 2016, 215–226. https://doi.org/10.1007/978-3-319-31753-3_18

22. L. P. F. Garcia, F. Campelo, G. N. Ramos, A. Rivolli, A. C. P. L. F. de Carvalho, Evaluating clustering meta-features for classifier recommendation, In: *Intelligent systems*, Cham: Springer, 2021, 453–467. https://doi.org/10.1007/978-3-030-91702-9_30

23. E. Alcobaça, F. Siqueira, A. Rivolli, L. P. F. Garcia, J. T. Oliva, A. C. P. L. F. de Carvalho, MFE: Towards reproducible meta-feature extraction, *J. Mach. Learn. Res.*, **21** (2020), 1–5.

24. A. Rivolli, L. P. F. Garcia, C. Soares, J. Vanschoren, A. C. P. L. F. de Carvalho, Characterizing classification datasets: A study of meta-features for meta-learning, arXiv: 1808.10406, (2019). https://doi.org/10.48550/arXiv.1808.10406

25. G. C. M. Moura, R. Sadre, A. Pras, Taking on internet bad neighborhoods, *2014 IEEE Network Operations And Management Symposium (NOMS)*, Krakow, Poland, 2014, 1–7. https://doi.org/10.1109/NOMS.2014.6838284

26. A. Renjan, K. P. Joshi, S. N. Narayanan, A. Joshi, DAbR: Dynamic attribute-based reputation scoring for malicious IP address detection, *2018 IEEE International Conference On Intelligence And Security Informatics (ISI)*, Miami, FL, USA, 2018, 64–69. https://doi.org/10.1109/ISI.2018.8587342

27. D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE T. Evolut. Comput.*, **1** (1997), 67–82. https://doi.org/10.1109/4235.585893

28. S. Uddin, H. H. Lu, Dataset meta-level and statistical features affect machine learning performance, *Sci. Rep.*, **14** (2024), 1670. https://doi.org/10.1038/s41598-024-51825-x

29. M. Huisman, J. N. van Rijn, A. Plaat, A survey of deep meta-learning, *Artif. Intell. Rev.*, **54** (2021), 4483–4541. https://doi.org/10.1007/s10462-021-10004-4

30. A. Vettoruzzo, M.-R. Bouguelia, J. Vanschoren, T. Rögnvaldsson, K. Santosh, Advances and challenges in meta-learning: A technical review, *IEEE T. Pattern Anal.*, **46** (2024), 4763–4779. https://doi.org/10.1109/TPAMI.2024.3357847

31. C. Raymond, Meta-learning loss functions for deep neural networks, PhD Thesis, Victoria University of Wellington, 2025.

32. D. J. Hand, Mixture models: inference and applications to clustering, *J. R. Stat. Soc. C-Appl.*, **38** (1989), 384–385. https://doi.org/10.2307/2348072

33. T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, New York: Association for Computing Machinery, 2016, 785–794. https://doi.org/10.1145/2939672.2939785

34. J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, *J. Mach. Learn. Res.*, **13** (2012), 281–305.

35. B. W. Matthews, Comparison of the predicted and observed secondary structure of T4 phage lysozyme, *Biochimica Et Biophysica Acta (BBA)-Protein Structure*, **405** (1975), 442–451. https://doi.org/10.1016/0005-2795(75)90109-9

36. P. Giudici, Safe machine learning, *Statistics*, **58**, (2024), 473—477. https://doi.org/10.1080/02331888.2024.2361481

37. G. Babaei, P. Giudici, A statistical package for safe artificial intelligence, *Stat. Methods Appl.*, (2025). https://doi.org/10.1007/s10260-025-00796-y

38. A. Jain, K. Nandakumar, A. Ross, Score normalization in multimodal biometric systems, *Pattern Recogn.*, **38** (2005), 2270–2285. https://doi.org/10.1016/j.patcog.2005.01.012

39. K. Pearson, LIII. On lines and planes of closest fit to systems of points in space, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **2** (1901), 559–572. https://doi.org/10.1080/14786440109462720

40. L. F. Kozachenko, N. Leonenko, Sample estimate of the entropy of a random vector, *Probl. Pered. Inform.*, **23** (1987), 9–16.

41. D. Amaratunga, J. Cabrera, Analysis of data from viral DNA microchips, *J. Am. Stat. Assoc.*, **96** (2001), 1161–1170. https://doi.org/10.1198/016214501753381814

Supplementary

Table 6 shows the meta-features list computed in the study.

Meta-feature	Cost	Impact	Group	Description
leaves	3	1	model-based	Compute the number of leaf nodes in the DT model.
leaves_branch	3	1	model-based	Compute the size of branches in the DT model.
leaves_corrob	3	1	model-based	Compute the leaves corroboration of the DT model.
leaves_homo	3	1	model-based	Compute the DT model Homogeneity for every leaf node.
leaves_per_class	3	1	model-based	Compute the proportion of leaves per class in DT model.
nodes	3	1	model-based	Compute the number of non-leaf nodes in DT model.
nodes_per_attr	3	1	model-based	Compute the ratio of nodes per number of attributes in DT model.
nodes_per_inst	3	1	model-based	Compute the ratio of non-leaf nodes per number of instances in DT model.
nodes_per_level	3	1	model-based	Compute the ratio of number of nodes per tree level in DT model.
nodes_repeated	3	1	model-based	Compute the number of repeated nodes in DT model.
tree_depth	3	1	model-based	Compute the depth of every node in the DT model.
tree_imbalance	3	1	model-based	Compute the tree imbalance for each leaf node.
tree_shape	3	1	model-based	Compute the tree shape for every leaf node.
var_importance	3	1	model-based	Compute the features importance of the DT model for each attribute.
attr_conc	2	2	info-theory	Compute concentration coef. of each pair of distinct attributes.
attr_ent	3	1	info-theory	Compute Shannon's entropy for each predictive attribute.
class_conc	2	2	info-theory	Compute concentration coefficient between each attribute and class.
class_ent	1	3	info-theory	Compute target attribute Shannon's entropy.

eq_num_attr	3	1	info-theory	Compute the number of attributes equivalent for a predictive task.
joint_ent	2	2	info-theory	Compute the joint entropy between each attribute and class.
mut_inf	2	2	info-theory	Compute the mutual information between each attribute and target.
ns_ratio	2	2	info-theory	Compute the noisiness of attributes.
c1	1	3	complexity	Compute the entropy of class proportions.
c2	1	3	complexity	Compute the imbalance ratio.
f1	3	1	complexity	Maximum Fisher's discriminant ratio.
f1v	3	1	complexity	Directional-vector maximum Fisher's discriminant ratio.
f3	3	1	complexity	Compute feature maximum individual efficiency.
f4	3	1	complexity	Compute the collective feature efficiency.
t2	1	2	complexity	Compute the average number of features per dimension.
t3	3	1	complexity	Compute the average number of PCA dimensions per points.
t4	2	1	complexity	Compute the ratio of the PCA dimension to the original dimension.
can_cor	3	1	statistical	Compute canonical correlations of data.
cor	2	2	statistical	Compute the absolute value of the correlation of distinct dataset column pairs.
cov	2	2	statistical	Compute the absolute value of the covariance of distinct dataset attribute pairs.
eigenvalues	3	1	statistical	Compute the eigenvalues of covariance matrix from dataset.
g_mean	1	3	statistical	Compute the geometric mean of each attribute.
gravity	3	1	statistical	Compute the distance between minority and majority classes center of mass.
iq_range	1	3	statistical	Compute the interquartile range (IQR) of each attribute.
kurtosis	3	1	statistical	Compute the kurtosis of each attribute.

lh_trace	3	1	statistical	Compute the Lawley-Hotelling trace.
mad	1	3	statistical	Compute the Median Absolute Deviation (MAD) adjusted by a factor.
max	1	3	statistical	Compute the maximum value from each attribute.
mean	1	3	statistical	Compute the mean value of each attribute.
median	1	3	statistical	Compute the median value from each attribute.
min	1	3	statistical	Compute the minimum value from each attribute.
nr_cor_attr	2	2	statistical	Compute the number of distinct highly correlated pair of attributes.
nr_disc	3	1	statistical	Compute the number of canonical correlation between each attribute and class.
nr_norm	1	1	statistical	Compute the number of attributes normally distributed based on a given method.
nr_outliers	1	3	statistical	Compute the number of attributes with at least one outlier value.
p_trace	3	1	statistical	Compute the Pillai's trace.
range	1	3	statistical	Compute the range (max - min) of each attribute.
roy_root	3	1	statistical	Compute the Roy's largest root.
sd	1	3	statistical	Compute the standard deviation of each attribute.
skewness	2	2	statistical	Compute the skewness for each attribute.
sparsity	3	1	statistical	Compute (possibly normalized) sparsity metric for each attribute.
t_mean	1	3	statistical	Compute the trimmed mean of each attribute.
var	1	3	statistical	Compute the variance of each attribute.
w_lambda	3	1	statistical	Compute the Wilks' Lambda value.
attr_to_inst	1	3	general	Compute the ratio between the number of attributes.
cat_to_num	1	2	general	Compute the ratio between the number of categoric and numeric features.

freq_class	1	3	general	Compute the relative frequency of each distinct class.
inst_to_attr	1	3	general	Compute the ratio between the number of instances and attributes.
nr_attr	1	3	general	Compute the total number of attributes.
nr_bin	3	1	general	Compute the number of binary attributes.
nr_cat	3	1	general	Compute the number of categorical attributes.
nr_class	3	1	general	Compute the number of distinct classes.
nr_inst	1	3	general	Compute the number of instances (rows) in the dataset.
nr_num	1	3	general	Compute the number of numeric features.
best_node	3	1	landmarking	Performance of a the best single decision tree node.
elite_nn	3	1	landmarking	Performance of Elite Nearest Neighbor.
linear_discr	3	1	landmarking	Performance of the Linear Discriminant classifier.
naive_bayes	3	1	landmarking	Performance of the Naive Bayes classifier.
one_nn	3	1	landmarking	Performance of the 1-Nearest Neighbor classifier.
random_node	3	1	landmarking	Performance of the single decision tree node model induced by a random attribute.
worst_node	3	1	landmarking	Performance of the single decision tree node model induced by the worst informative attribute.
one_itemset	2	1	itemset	Compute the one itemset meta-feature.
two_itemset	3	1	itemset	Compute the two itemset meta-feature.
ch	3	1	clustering	Compute the Calinski and Harabasz index.
int	3	1	clustering	Compute the INT index.
nre	1	3	clustering	Compute the normalized relative entropy.
sc	3	1	clustering	Compute the number of clusters with size smaller than a given size.

sil	3	1	clustering	Compute the mean silhouette value.
vdb	3	1	clustering	Compute the Davies and Bouldin Index.
vdu	3	1	clustering	Compute the Dunn Index.

AIMS Press