# Enhancing Quantification through Meta-Learning

Guilherme B. Gomes[1], Willian Zalewski[2], and André G. Maletzke[1]

[1] Western Paraná State University, Foz do Iguaçu, Paraná, BR
{guilherme.gomes,andre.maletzke}@unioeste.br
[2] Federal University for Latin American Integration, Foz do Iguaçu, Paraná, BR
willian.zalewski@unila.edu.br

**Abstract.** We advocate that no single quantifier consistently outperforms all others across every possible scenario. We also argue that experimental evaluation in quantification using the Artificial-Prevalence Protocol is significantly more costly than in classification. Although the community has made strides in reducing the number of algorithms to be tested in classification scenarios, this challenge in quantification remains unexplored. To address this issue, we introduce a method that recommends quantifiers for each dataset, leveraging the concept of meta-learning. By analyzing the intrinsic characteristics of datasets through meta-features, our method predicts the most suitable quantification algorithm likely to yield optimal results. Our proposal automates the selection process, providing data-driven recommendations that enhance the efficiency and effectiveness of quantification tasks. We achieved a recommendation accuracy of 83%, meaning that our system successfully identified the optimal quantifier for 83 out of 100 datasets. Furthermore, our architecture enables us to build an ensemble of quantifiers using, for instance, the recommended Top-$k$ quantifiers. Our ensembles lead to superior quantification results compared to other state-of-the-art quantifiers, such as D$y$S, SORD, and MS.

**Keywords:** Meta-Features · Recommendation · Learning to quantify

## 1 Introduction

Meta-learning (MtL)-based recommendation systems can be a viable solution to automatically select data-driven algorithms using knowledge extracted from previous tasks [25]. Meta-learning systems indicate which algorithm should be utilized to achieve the best possible results for each task, according to its particularities [6]. However, for recommendations to be made, this system needs to acquire experience, for example, from (*i*) model evaluations, which involve recommending hyperparameter values, configuration search spaces, and optimization approaches for analogous tasks; (*ii*) previously successful models using transfer learning; and (*iii*) exploring task properties to recommend algorithms based on data characterization and learning performance [25].

Exploring meta-learning for building recommendation systems has been investigated over the years. For example, Ali & Smith [4] used accuracy and complexity measures to build a classifier recommendation system. In [1], the authors also used meta-learning to recommend image segmentation algorithms, and in [24], meta-learning was used to recommend clustering algorithms. Meta-learning as a tool for selecting machine learning algorithms has shown great promise in various machine learning tasks [21,12,23].

Recently, a novel supervised task known as quantification has garnered significant interest from the machine learning community. The task aims to accurately determine the prevalence of each class within an unlabeled dataset. A key distinction between quantification and classification is that the class distribution in quantification is not fixed. Otherwise, it could easily predict the class proportions in the test set based on the training set. Several quantification algorithms have been proposed in the last decade. However, compared to the classification task, quantification has a significantly smaller number of methods. In contrast, quantification experiments are more expensive than classification, requiring several test sets with different class distributions. Consequently, determining the most suitable method for a new problem can be expensive and time consuming. Previous studies have dedicated efforts to evaluate the performance of several quantifiers under different conditions [20,16,27,14]

We advocate that no single quantifier consistently outperforms all others across every possible scenario. This variability in performance requires a tailored approach to selecting the most appropriate quantifier for each dataset. We also argue that experimental evaluation in quantification is massively more costly than classification. Although the community has delivered proposals to reduce the number of algorithms to be experimented within classification scenarios, this challenge in quantification remains unexplored.

Investigating a meta-learning strategy to develop recommendation systems for quantification algorithms is still uncharted territory. To address this challenge, we suggest quantifiers for each dataset using meta-learning principles. By examining the inherent properties of datasets through meta-features, our method forecasts the most appropriate quantification algorithm expected to deliver optimal outcomes. This approach automates the selection procedure, offering data-driven recommendations that improve the efficiency and effectiveness of quantification activities.

The structure of this article is as follows: Section 2 describes the basic concepts and differences between the classification and quantification tasks as well as the algorithm recommendation task formalization, including meta-learning concepts. Section 3 reviews the literature on related works. Sections 4 and 5 present our meta-learning architecture for recommending quantifiers and the experimental setup of this paper, respectively. Section 6 presents the empirical results and discussion. Finally, Section 7 concludes this work and present directions for future work.

## 2 Background

In machine learning, classification assigns one or more classes to a set of individual data items. To achieve this, a learner $\boldsymbol{h}$ is used to generalize a hypothesis from a training set $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of length $n$, in which $x_i \in \mathcal{X} = \{a_{i1}, \ldots, a_{im}\}$ represents a vector with $m$ attributes in the feature space $\mathcal{X}$ and $y_i \in \mathcal{Y} = \{c_1, \ldots, c_w\}$ represents a label in the $w$-label space $\mathcal{Y}$. The hypothesis aims to cover the available training examples and future unseen examples. In this paper, we focus on binary problems, *i.e.*, the label space $\mathcal{Y}$ is restricted to $\mathcal{Y} = \{\oplus, \ominus\}$. Therefore, a classifier is a function that maps an instance of $\mathcal{X}$ to a subset of $\mathcal{Y}$ as follows $\boldsymbol{h} : \mathcal{X} \rightarrow \{\oplus, \ominus\}$.

Many classification algorithms generate scores as an intermediate step in deciding which class is assigned to an instance [9]. In binary classification, it suffices to consider the score for only one of the classes, such as the score for the positive class. A scorer $\boldsymbol{s}(x_i)$ is a function that maps each instance $x_i$ to a value correlating to $\mathrm{P}(y_i = \oplus | x_i)$ as the following equation $\boldsymbol{s} : x_i \rightarrow \mathbb{R}$ [17]. Therefore, a classifier is subsequently achieved by applying a threshold to the score values, categorizing them as positive or negative.

Over the years, algorithms to build data-driven models have been proposed, resulting in a large number of options. For instance, both $\boldsymbol{h}$ and $\boldsymbol{s}$ are models induced by some machine learning algorithm based on a dataset. Selecting the most appropriate method to fit $\boldsymbol{h}$ and $\boldsymbol{s}$ for a dataset can be laborious. This fact has inaugurated a new research field that aims to recommend algorithms based on data features. One approach is to develop an automated system for recommending algorithms, which relies on the relationship between algorithm performance and dataset characteristics to suggest suitable algorithms directly.

Let $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_\mathcal{M}\}$ represents a set of $\mathcal{M}$ datasets across various domains and $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_\mathcal{N}\}$ a set of $\mathcal{N}$ candidate algorithms for proper induction of $\boldsymbol{h}$ and $\boldsymbol{s}$ along with a model quality measure $\eta$. An algorithm recommendation for a given $\mathcal{S}_d$ can be defined as follows:

$$\text{Recommender}(\mathcal{S}_d) = \mathop{\mathbf{arg\,max}}_{\mathcal{A}_r \in \mathcal{A}} \eta(\mathcal{S}_d, \mathcal{A}_r)$$

where $\mathbf{arg\,max}_{\mathcal{A}_r \in \mathcal{A}}$ selects the algorithm $\mathcal{A}_r$ that maximizes the quality measure $\eta$ for the dataset $\mathcal{S}_d$.

The selection of quality measures is intrinsically related to the specific nature of the problem and the task at hand. For instance, accuracy is typically employed as a quality metric in classification tasks, while Mean Absolute Error (MAE) is conventionally utilized in regression tasks.

## 2.1 Quantification

Classification aims at assigning one or more classes to each instance from a distribution. Nevertheless, in many cases, the primary objective is to estimate the proportion of each class in the test set rather than to label individual data points. In such applications, predictions of samples matter more than individual instances. This distinction highlights the difference between classification and quantification.

A quantifier is a predictive model $\boldsymbol{q}$ induced from a dataset to predict the class distribution of an unlabeled set, defined according to the following equation:

$$\boldsymbol{q} : 2^{\mathcal{X}} \rightarrow [0, 1]$$

where, $2^{\mathcal{X}}$ represents the power set of $\mathcal{X}$ and $\boldsymbol{q}$ outputs a single number in the interval $[0, 1]$ that correlates to the positive class prevalence.

An important distinction between quantification and classification lies in the non-stationarity of class distributions found in quantification problems. Otherwise, it would be trivial to predict the class proportions in the test set based on the distribution in the training set. Consequently, the most straightforward quantifier, called Classify and Count (CC), which involves classifying each instance and then counting how many of them belong to each class does not match in this context, suffering from the systemic error introduced when the class distribution varies. Numerous techniques have been suggested by the quantification community to address this issue. One of the first is the Adjusted Classify and Count (ACC), which corrects the bias in CC by adopting a correction factor based on the model's true positive rate (*tpr*) and false positive rate (*fpr*).

ACC provides perfect quantification results regardless of the classifier accuracy when precises *tpr* and *fpr* are provided. However, data scarcity and class imbalance make estimating these statistics challenging. The symbiosis between classification and quantification extends far beyond CC and ACC. The quantification community has proposed numerous methods, most of which rely on binary classifiers that produce a score representing the confidence in a positive classification as an intermediate step for quantification. González et al. (2017) [13] provide a comprehensive survey of quantification methods, structuring them into a taxonomy of three groups:

*i.* *Classify, Count, and Correct:* methods that classify each instance and subsequently count the examples that belong to each class. This group also includes methods that apply a correction factor to these counts, including techniques that account for classification error;
*ii.* *Adaptation of classification algorithms:* approaches that modify the mechanics of classification algorithms to transform them into quantifiers;
*iii.* *Distribution matching:* methods that model the training data distribution, typically represented by $P(x|y)$, varying $P(y)$, and then seek parameters that best match the test data distributions.

Table 1 briefly presents some of the most known and utilized quantifiers in the literature. The last column provides the corresponding reference for each method. All methods, except those in the group *ii*, require a preliminary step of learning a classifier capable of predicting a score for each unlabeled test instance. A score is a numerical value that correlates with the posterior probability of a particular class, *i.e.*, $P(\oplus|x)$ for binary problems. Various machine learning algorithms can be used to obtain a scorer, removing the decision threshold step. However, selecting the best scorer does not guarantee the best quantifier, as the premise $P_{training}(y) = P_{test}(y)$ cannot be held in quantification scenarios.

Selecting the best quantifier is highly dependent on the specific characteristics of each dataset, requiring performing extensive experiments using, for instance, the Artificial-Prevalence Protocol (APP) [10]. APP is the most commonly

Table 1: Quantifiers evaluated.

| Taxonomy Group | Quantifier | Acronym | Reference |
|---|---|---|---|
| *i* | Classify and Count | CC | [10] |
|  | Adjust Classify and Count | ACC |  |
|  | Probabilistic Classify | PCC | [5] |
|  | Probabilistic Adjust Classify and Count | PACC |  |
|  | Threshold Selection Method |  | [11] |
|  | Set the decision threshold where $(1 - tpr) = fpr$ | X |  |
|  | Set the decision threshold where $tpr - fpr$ is maximized | MAX |  |
|  | Estimate $tpr$ and $fpr$ for several thresholds, returning the median of them | MS |  |
| *ii* | Quantification Trees | QT | [22] |
| *iii* | Expectation Maximization Quantification | EMQ | [26] |
|  | Distribution matching with Hellinger Distance | HDy | [15] |
|  | Mixture Model Framework | D$y$S | [19] |
|  | Sample Mean Matching | SMM | [16] |
|  | Sample-ORD Method | SORD | [19] |

employed experimental framework for evaluating and contrasting quantification techniques. For binary problems, APP generates several test sets, sub-sampling examples randomly from $\oplus$ or $\ominus$ with predetermined class distributions. Commonly, test sets are generated varying the class distribution across a wide range of possibilities, such as $p = P(\oplus) \in \{0, .01, .02, \ldots, .99, 1\}$. APP entails a probabilistic decision, making it susceptible to random variability. Consequently, many researchers opt to replicate this experiment to minimize variance. This procedure leads to an assessment over a large number of test sets [17].

Meta-learning emerges as a valuable tool, leveraging prior knowledge and experience from multiple learning tasks to guide quantifier selection. By analyzing the properties of datasets and their interactions with various quantifiers, meta-learning can identify patterns and correlations that are not immediately apparent. This enables the development of recommendation systems that predict the most suitable quantifiers or a group of quantifiers for a given dataset. According to Garcia et al. (2018) [12], meta-learning can help differentiate the performance of a set of machine learning methods, aiding in the selection of the best method for a given problem.

## 2.2 Meta-learning

Meta-learning aims to automate selecting, tuning, and combining machine learning algorithms to improve overall performance across a wide range of tasks [3]. To achieve this, past experiences are utilized to learn from the learning process itself, a concept known as learning to learn.

Meta-learning-based recommendation systems feature automatic technique selection driven by data, utilizing knowledge extracted from previous tasks [6]. According to Rivolli et al. (2022) [25], MtL incorporates the following components: the problem space ($\mathcal{S}$), the feature space ($\mathcal{F}$), the machine learning algorithm space ($\mathcal{A}$), the performance space ($\mathcal{E}$), and the machine learning algorithm used for MtL. Therefore, through a data-driven process, the performance ($\mathcal{E}$) of a set of algorithms ($\mathcal{A}$) on various datasets ($\mathcal{S}$) is associated with characteristics of these datasets ($\mathcal{F}$), represented by meta-features. Thus, a machine learning model is induced from the meta-data, represented by $\mathcal{F}$, recommending the most suitable machine learning algorithm from $\mathcal{A}$ for a new dataset.

Meta-features must be tailored to the problem at hand, being able to characterize the problem aiming to induce an effective recommendation model. Existing an assorted strategies for constructing meta-features. Rivolli et al. (2022) [25] propose organizing these strategies into six groups as follows:

- **Simple:** measures that are easily determined and generally do not require high computational effort. Commonly referred to as general metrics, including the number of dataset instances and the number of attributes.
- **Statistical:** these measures capture statistical properties from a dataset, such as mean, standard deviation, correlation, skewness, and kurtosis.
- **Information Theory:** these measures explore information theory concepts to describe a dataset. These metrics rely on entropy, quantifying the information content and complexity within the data.
- **Model-Based:** these measures are extracted from a fitted model learned on the training data. Although these measures can be extracted from different sorts of models, typically, they are derived from decision tree models, including the count of leaves, nodes, and the structure of the decision tree.
- **Landmarking:** these measures are based on the performance of fast and simple learning systems and algorithms to characterize a dataset. These algorithms should exhibit different inductive biases and be able to capture relevant information at a low computational cost.
- **Others:** represent measures that do not fit into any of the previous groups, generally including domain-related concepts and time-related measures.

Constructing a recommender system with meta-features involves multiple stages. Firstly, meta-features are selected and extracted from a diverse set of datasets, creating a meta-dataset where each instance represents a dataset. After that, a performance metric for each recommendable algorithm is estimated and included in the meta-table. A meta-learner is then trained on this meta-dataset to predict the performance of different algorithms based on the meta-features. Finally, when a new dataset arrives, the meta-features are extracted and input into the trained meta-learner, predicting the best-performing algorithm.

## 3 Related Works

Meta-learning has been effectively used to develop recommender systems across various domains. Wang et al. [30] and Zhang et al. (2019) [31] created systems that recommend feature selection and imbalance learning methods, respectively, by associating dataset meta-features with algorithm performance in a meta-table. Similarly, das Dôres et al. (2016) [8] proposed a framework for recommending software fault prediction algorithms, while Garcia et al. (2018) [12] focused on recommending classifiers by predicting their accuracy using data complexity measures. Zhu et al. (2018) [32] introduced a novel approach by applying link prediction in a network of datasets and classifiers to recommend appropriate classifiers. These studies demonstrate the broad applicability of meta-learning in building recommendation systems, with various frameworks proposed for different tasks [24,1,4].

In the last decade, quantification methods have been proposed to address various challenges in machine learning tasks. This rapid development has led to a diverse landscape of techniques, each with unique strengths and weaknesses tailored to specific types of data and problem scenarios. While this variety enriches the field, it also introduces a significant challenge: selecting the most suitable quantification method for a given problem has become increasingly complex.

To address this challenge, we propose a novel framework for recommending quantifiers based on meta-features. To the best of our knowledge, the task of recommending quantifiers has not been explored previously. Our approach leverages the wealth of existing quantification methods and automates the selection process by using meta-features to characterize datasets. These meta-features capture essential properties of the data, enabling the system to predict which quantifiers are likely to perform best.

## 4 Architecture for Recommending Quantifiers

In this section, we introduce a novel method that recommends quantifiers for each dataset by leveraging the concept of meta-learning. By analyzing the intrinsic characteristics of datasets through meta-features, our approach predicts the most suitable quantification algorithm likely to yield optimal results. This proposal automates the selection process, providing data-driven recommendations that enhance the efficiency and effectiveness of quantification tasks.

Initially, we extract meta-features from various datasets to capture their characteristics. Then, we estimate the performance of several quantifiers on these datasets using APP. After that, the extracted meta-features and performance metrics are unified to build the meta-table that serves, in the next step, to fit a recommender model. Finally, when a new and unseen dataset arrives, we extract its meta-features, providing them to the recommender that predicts the most appropriate quantifier. Figure 1 shows the architecture of our proposal.

Inspired by the literature on meta-learning recommendation, our proposal involves the following steps: (1) meta-features extraction, (2) meta-target estimation, (3) meta-learner induction, and (4) quantifier recommendation.

**Step 1 - Meta-features extraction:** for each dataset, meta-features are derived, resulting in a meta-instance within the meta-table. These meta-features encapsulate diverse attributes of the datasets, including statistical properties, information-theoretic metrics, and model-based characteristics. Consequently, each meta-instance in the meta-table offers a thorough summary of a dataset's principal features, aiding the recommender system in determining the most appropriate quantification algorithms for new datasets based

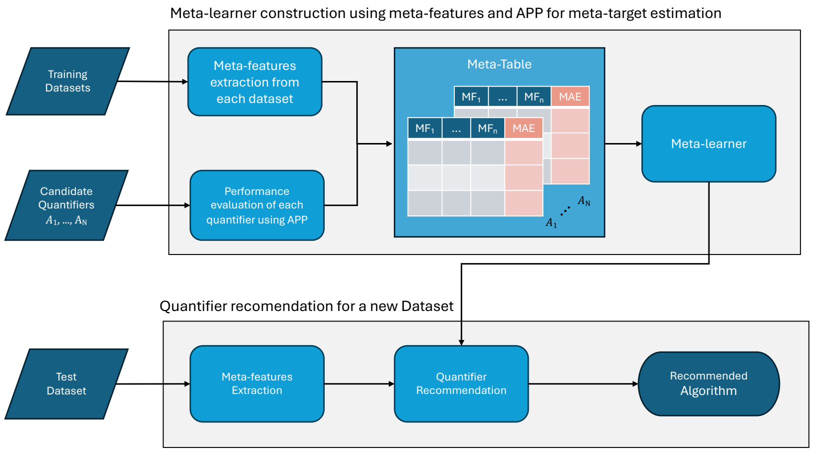

Meta-learner construction using meta-features and APP for meta-target estimation
Training Datasets
Candidate Quantifiers $A_1, ..., A_N$
Meta-features extraction from each dataset
Performance evaluation of each quantifier using APP
Meta-Table
MF₁ ... MFₙ MAE
$A_1$ $A_N$
Meta-learner
Quantifier recomendation for a new Dataset
Test Dataset
Meta-features Extraction
Quantifier Recommendation
Recommended Algorithm

Fig. 1: Architecture for quantifiers recommendation.

on their meta-features. Figure 2 shows the intuition of the meta-feature extraction process.

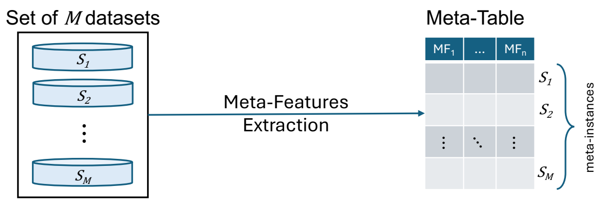


Fig. 2: Meta-features extraction process.

Both quantification and classification are supervised tasks that can explore diverse types of meta-features, *i.e.*, supervised and nonsupervised meta-features. In our architecture, we use the following groups of meta-features: simple, statistical, information-theoretic, model-based, and landmarking. The number of meta-features extracted varies across datasets. To standardize the meta-dataset, since instances cannot have different numbers of attributes, the features values are aggregated by the mean value for each feature, ensuring all feature sets have the same number of elements. Additionally, the range of the meta-features can vary significantly between datasets. To address this, the resulting meta-table can be normalized via min-max scaling.

**Step 2 - Meta-target estimation:** to build a recommendation model, the target attribute (or class) must be added to the meta-table, indicating, for example, which algorithm is most suitable for each dataset. For classification problems, the meta-target represents the performance of the classification algorithm such as accuracy or F-score [25]. In the context of quantification, classification metrics are unsuitable. In contrast, in quantification, the performance is based on the ability of minimizing the difference between true $(P(c_i))$ and predicted $(\hat{P}(c_i))$ class distribution for a set of classes $\mathcal{Y}$. Various measures for quantifier evaluation have been adapted from other contexts, such as Mean Absolute Error and Kullback-Leibler Divergence [28]. We use MAE as the meta-target in our architecture. Figure 3 illustrates the process of meta-target estimation.

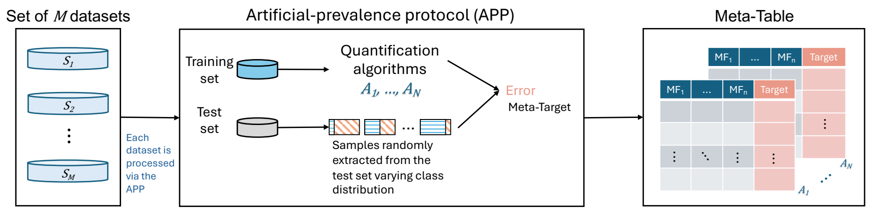

Set of $M$ datasets
$S_1$
$S_2$
$S_M$
Each dataset is processed via the APP
Artificial-prevalence protocol (APP)
Training set
Test set
Quantification algorithms
$A_1, ..., A_N$
Samples randomly extracted from the test set varying class distribution
Error
Meta-Target
Meta-Table
MF₁
...
MFₙ
Target
$A_N$
$A_1$

Fig. 3: Meta-target estimation using Artificial-Prevalence Protocol.

Evaluating a quantifier requires providing a sample of instances and varying the class distribution instead of a set of instances, as in the classification setup. We use the Artificial-Prevalence Procotol, which splits the dataset into training and test sets and then extracts several batches varying the class distribution from the test partition.

**Step 3 - Meta-learner induction:** from the meta-tables built in the previous step, we learn several regressors, one for each meta-table. These regressors serve as recommenders for an unknown dataset. Figure 4 illustrates the procedure that leads to the creation of a collection of meta-learners (regressors).

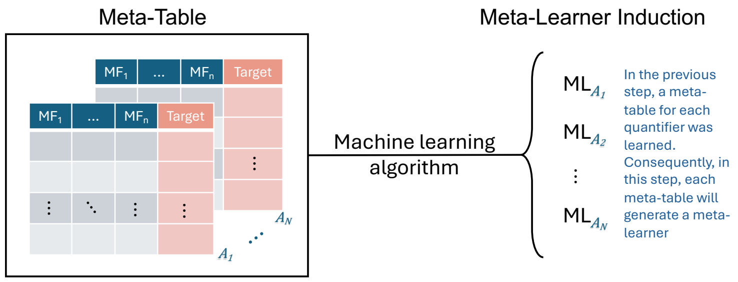
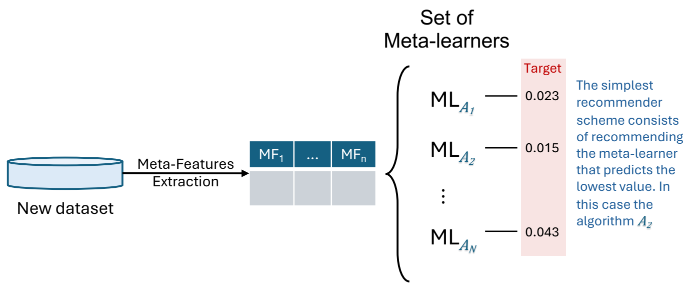

Meta-Table
MF₁
...
MFₙ
Target
$A_N$
$A_1$
Meta-Learner Induction
Machine learning algorithm
$ML_{A_1}$
$ML_{A_2}$
$ML_{A_N}$
In the previous step, a meta-table for each quantifier was learned. Consequently, in this step, each meta-table will generate a meta-learner

Fig. 4: Meta-learner induction step.

**Step 4 - Quantifier recommendation:** once the meta-learners are trained, the system can recommend suitable quantification algorithms for new datasets.

When a new dataset is presented, its meta-features are extracted and used as input to the trained meta-learners. Each meta-learner provides a prediction of the expected performance (in terms of MAE) for its corresponding quantifier. Figure 5 shows the recommendation process using meta-learners to predict the best quantifier for a new dataset. In order to enhance the



Set of Meta-learners
Target
$ML_{A_1}$
0.023
$ML_{A_2}$
0.015
$ML_{A_N}$
0.043
MF$_1$
...
MF$_n$
Meta-Features Extraction
New dataset
The simplest recommender scheme consists of recommending the meta-learner that predicts the lowest value. In this case the algorithm $A_2$

Fig. 5: Recommendation process based on a set of meta-learners.

recommendation's robustness, we propose forming an ensemble of quantifiers by selecting the Top-$k$ recommended methods. The ensemble can be constructed using various strategies, such as averaging the predictions of the selected quantifiers or assigning weights to each quantifier based on their predicted errors[3], where the quantifier with the lowest error receives the highest weight. In our proposal, the following strategies are explored:

– **Top-1:** the recommender selects the best quantifier for a new dataset, choosing the method that produces the lower error. This quantifier is then employed to perform the quantification task, leveraging its suitability as determined by the meta-features of the dataset.
– **Top-$k$:** we adopt an ensemble strategy to select the $k$ methods whose predicted errors were the lowest. Next, we merge the Top-$k$ methods using the median of their predictions, with the goal of leveraging the advantages of various quantifiers.
– **Top-$k$+W:** we further refine the ensemble approach by weighting each of the $k$ selected quantifiers based on their errors. Quantifiers with the lowest errors predicted are given higher weights. Let $q_i$ represents the $i$-th quantifier, and $e_i$ the predicted error by the recommender. The weight $w_i$ for each quantifier is inversely proportional to its error. The weights $w_i$ are calculated as follows: $w_i = \frac{1/e_i}{\sum_{j=1}^{k}(1/e_j)}$ The weighted ensemble quantifier is expressed as:

$$\text{Top-}k + \text{W} = \sum_{i=1}^{k} w_i \cdot \hat{p}_{q_i}(\oplus)$$

[3] Note that the meta-learner aims to predict the MAE for a new dataset. Consequently, meta-learners with the lowest MAE can be combined.

where $\hat{p}_{q_i}(\oplus)$ represents the predicted distribution of the positive class by $q_i$ among the $k$ quantifiers with the smallest errors. This equation guarantees that quantifiers with smaller errors have a greater influence on the final quantification outcome.

## 5 Experiments

This section details the experiments conducted with multiple established quantification algorithms and a wide range of meta-features. To reduce the presence of bias, we collected a total of 100 datasets of binary classification problems from different domains in public data repositories [18,29]. Each dataset was chosen not only for its diversity but also to ensure it contained enough instances to apply the APP with a batch size of 100 instances. We prepared each dataset by transforming incompatible attributes for machine learning through one-hot encoding for categorical attributes and removing attributes with missing values. Dataset descriptions and codes are available in the paper repository[4].

The meta-features were extracted using the Python package Meta-Feature Extractor (MFE) [2]. We extracted all the default meta-features from the MFE package, resulting in a total of 111 meta-features categorized into the following groups: Simple, Statistical, Information-Theoretic, Model-Based, and Landmarking. Mean and standard deviation were used as summary functions to aggregate the different numbers of meta-features. Min-max normalization was applied to the resulting meta-features.

Since we propose an ensemble method, we generated baseline ensembles for a fair comparison. For every dataset, we chose $k$ quantifiers at random and used the median prediction of these $k$ quantifiers as the ensemble output. We tested 1, 3, and 5 as values $k$. To reduce the influence of randomly selecting quantifiers, the baseline ensembles were generated 30 times for each dataset, and the mean result was reported for a fair comparison.

To build the meta-target we followed a structured process to accurately assess the performance of each quantification algorithm on the selected datasets. Using the APP, we split each dataset into training and test sets using stratified sampling without replacement with 70% and 30% proportions, respectively. Then, from the training set we learn a scorer (classifier) using Regularized Logistic Regression (LR), provided by the Scikit-Learn[5] library. For each dataset, we tune the following hyperparameters: **C** in the range $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}\}$, and **class-weight** in *balanced* or *none*. Since some quantifiers require scores, *tpr*, and *fpr*, we estimate them using 10-fold stratified cross-validation on the training set. In our experiments, we include quantifiers that require a scorer/classifier in an intermediate step. The following quantifiers were included: CC, ACC, MAX, PCC, PACC, X, MS, HDy, SMM, SORD, and D*y*S. The hyperparameter settings for HDy and D*y*S were based on [15] and [19], respectively.

[4] `https://github.com/Bachega/lequa2024_workshop`
[5] `https://scikit-learn.org`

In the testing phase, from each test dataset, we create multiple samples composed of 100 instances with different class distributions in each sample. The positive class distribution in each test sample varies from 0% to 100%, in increments of 5%. We repeated each setting ten times to reduce the error variance and averaged the results.

To assess the performance of quantifiers, we use the MAE in two situations: estimating the meta-target of the meta-table in Step 2 and evaluating the final performance of quantifiers. We compare the methods according to the Friedman test with 95% confidence and the Nemenyi post-hoc test.

To evaluate the effectiveness of recommendations, we use the recommendation hit that evaluates the success of a recommendation system by checking if the recommended algorithm is among the top performing ones for a given dataset. It ensures that the recommended algorithm is either the best or performs similarly to the best algorithms, thereby validating the recommender's effectiveness [30]. We employ the Random Forest algorithm as the meta-learner, utilizing the default hyperparameters available in the Scikit-Learn library[5] library.

Suppose $\mathcal{A}_{\text{opt}}$ represents the optimal quantifier algorithm for a dataset $\mathcal{S}_d$, and $\mathcal{A}_{\text{Setopt}}$ denotes the set of quantifier algorithms in which each algorithm has no significant difference from $\mathcal{A}_{\text{opt}}$, including $\mathcal{A}_{\text{opt}}$ itself. The recommendation hit for a dataset $\mathcal{S}_d$ is defined as a binary measure that indicates whether the recommended algorithm $\mathcal{A}_{\text{rec}}$ is in the optimal algorithm set. Therefore, $\text{Hit}(\mathcal{A}_{\text{rec}}, S_d) = 1$ indicates that the recommendation is effective and the recommended quantifier algorithm $\mathcal{A}_{\text{rec}}$ is included in $\mathcal{A}_{\text{Setopt}}$ for $\mathcal{S}_d$. Conversely, $\text{Hit}(\mathcal{A}_{\text{rec}}, S_d) = 0$ means that the recommended quantifier algorithm $\mathcal{A}_{\text{rec}}$ does not belong to $\mathcal{A}_{\text{Setopt}}$. In other words, $\mathcal{A}_{\text{rec}}$ performs significantly worse than the optimal quantifier algorithm $\mathcal{A}_{\text{opt}}$ on $\mathcal{S}_d$, indicating a poor recommendation. The $\mathcal{A}_{\text{Setopt}}$ set for each dataset is defined using a Friedman test followed by the Conover post-hoc test with Holm procedure, as recommended by [30], performed at a significance level of 0.05. To evaluate the recommendation system, we calculate the average Hit Ratio using Leave-One-Out across all datasets $\mathcal{S}$:

$$\text{Hit Ratio}(\mathcal{A}_{\text{rec}}, \mathcal{S}) = \frac{1}{\mathcal{M}} \sum_{d=1}^{\mathcal{M}} \text{Hit}(\mathcal{A}_{\text{rec}}, \mathcal{S}_d)$$

## 6 Results

Our meta-learner achieves a Hit Ratio of 0.83 (83%) when it recommends the best quantifier ($k = 1$), which means that for 83 out of 100 datasets, the recommended quantifier is optimal, according to the Hit Ratio measure. As we consider the recommendation of the Top-3 ($k = 3$) and Top-5 ($k = 5$) scenarios, the Hit Ratios increase to 0.97 (97%) and 1.00 (100%), respectively. These results were expected given the criterion utilized by the Hit Ratio, which is based on a statistical test to define the $k$ best quantifiers. This means that a Hit is recorded not only for the absolute best quantifier but also for those that do not have a statistically significant difference from the best one, ensuring a more comprehensive evaluation of the recommended quantifiers' effectiveness. These results support our main hypothesis that an MtL architecture can effectively recommend high-quality quantifiers according to the characteristics of each dataset.

Inspired by our preliminary findings, our subsequent analysis seeks to demonstrate the improvement in quantification outcomes achieved by either suggesting the optimal quantifier ($k = 1$) or constructing an ensemble composed of the $k$ recommended quantifiers, as outlined in Step 4 of our architecture. Figure 6 summarizes the results to illustrate the overall performance in terms of the average ranking of quantifiers across 100 datasets.

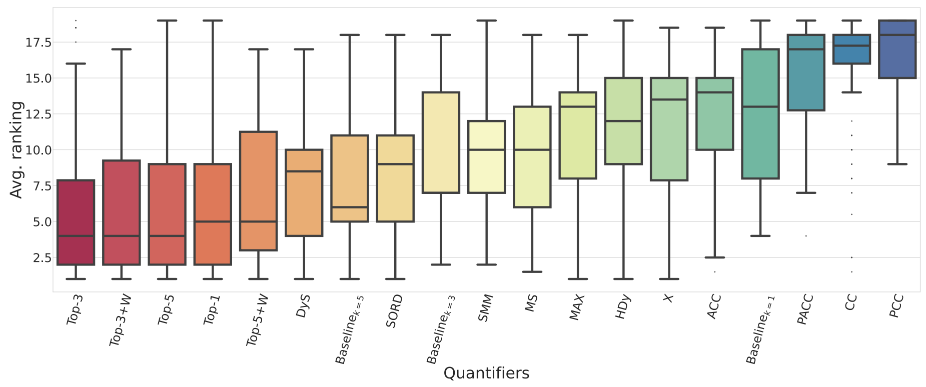


Fig. 6: Aggregation of several rank positions for all quantifiers, including those recommended by our MtL architecture.

To compute the average ranking for each dataset, we first assess the ranking of all quantifiers based on MAE for each dataset and subsequently calculate the mean ranks across all datasets. The quantifier that provides minimum MAE is ranked first. We highlight that the Leave-One-Out cross-validation was employed, ensuring that the dataset under testing is excluded from the training partition of the meta-leaner induction. Consequently, each dataset is tested individually while not contributing to the training process, providing an unbiased evaluation of the recommendation architecture's performance. This rigorous validation technique enhances the reliability of our results by ensuring that the model's predictions are not influenced by prior exposure to the test dataset.

Our MtL architecture improves on the best quantifiers and outperforms them in a large number of datasets from a variety of domains. This achievement corroborates our ancillary hypothesis, demonstrating the robustness and adaptability of our recommendation architecture. Additionally, our proposal helps to select the best $k$ quantifiers using the Hit Ratio criterion, enabling the construction of an ensemble of quantifiers in a data-driven scheme. This approach further enhances the accuracy and reliability of quantification tasks by leveraging the combined strengths of multiple quantifiers tailored to the unique characteristics of each dataset. Figure 7 shows the critical difference diagram for all quantification approaches, including those built by our architecture.

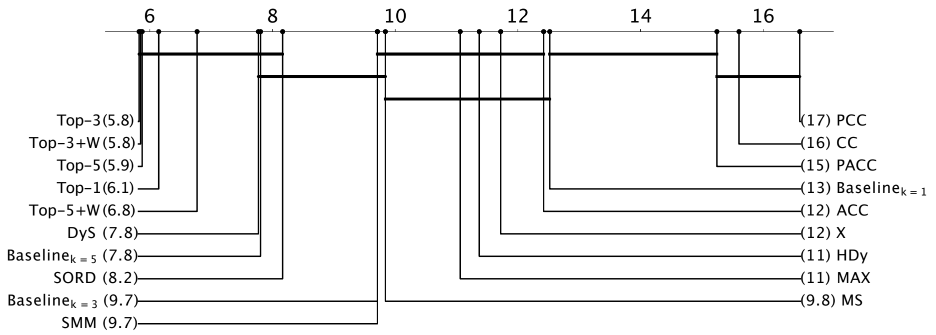


Fig. 7: Friedman's Nemenyi post-hoc test [7] for mean absolute quantification error. Groups of methods that are not significantly different at $p < 0.05$ are connected.

Our architecture identified the optimal quantifier for most datasets (Top-1), leading to enhanced performance compared to choosing a single quantifier (D$y$S). The results reveal that the recommended quantifier significantly outperform most quantifiers, except D$y$S and SORD. Using our architecture to select a quantifier for a given dataset, or applying our proposed ensemble methods, resulted in a statistical better performance than any other existing quantifier, except D$y$S and SORD. Although no statistical difference was observed between our proposals and state-of-the-art quantifiers, choosing a base quantifier tailored by data characteristics through our meta-learning approach proves to be highly effective and stable, being Top-1 ranked consistently better than any other existing methods.

The comparison between ensemble and nonensemble methods might be considered unfair. Thus, we include baseline ensembles that select $k$ quantifiers randomly. The proposed ensembles consistently outperform the baselines. Whether weighted or not, the Top-3 and Top-5 methods demonstrate superior effectiveness compared to the baselines, which confirms that building ensembles through meta-learning leads to better quantification accuracy. Interestingly, the baselines also demonstrate competitive performance, particularly Baseline$_{k=3}$ and Baseline$_{k=5}$, surpassing most base quantifiers. The comparable performance of most base quantifiers can explain this result, as also noted by [27].

Finally, our ensembles consistently outperform the baseline methods and base methods, with the differences being statistically significant in most cases.

## 7 Conclusion

This paper presents the first architecture for quantifier recommendation using meta-learning, significantly improving the efficiency and effectiveness of quantification tasks. By analyzing datasets through meta-features, our approach predicts the most suitable quantification algorithm for each dataset. Our method can identify the best quantifier for 83% of the studied datasets. Furthermore, our ensemble strategies outperform other state-of-the-art quantifiers, highlighting the robustness of our approach.

Our main contributions include introducing a meta-learning-based scheme for quantifier recommendation, validating its effectiveness through extensive experiments, and demonstrating the potential of ensemble strategies to surpass individual quantifiers. Our architecture is in its early stages, with potential for exploring additional ensemble-building strategies to enhance robustness and performance. Future research should map and analyze situations where recommendations fail, providing insights to refine the recommendation mechanism.

**Acknowledgments.** We thank the National Council for Scientific and Technological Development (CNPq) (001440855/2022-5) and the Ministry of Science, Technology and Innovation of Brazil (MCTI) for funding and support.

## References

1. Aguiar, G.J., Mantovani, R.G., Mastelini, S.M., de Carvalho, A.C., Campos, G.F., Junior, S.B.: A meta-learning approach for selecting image segmentation algorithm. Pattern Recognit. Lett. **128**, 480–487 (2019)
2. Alcobaça, E., Siqueira, F., Rivolli, A., Garcia, L.P., Oliva, J.T., De Carvalho, A.C.: Mfe: Towards reproducible meta-feature extraction. J. Mach. Learn. Res. **21**(111), 1–5 (2020)
3. Alexandros, K., Melanie, H.: Model selection via meta-learning: a comparative study. Int. J. Artif. Intell. Tools **10**(04), 525–554 (2001)
4. Ali, S., Smith, K.A.: On learning algorithm selection for classification. Appl. Soft Comput. **6**(2), 119–138 (2006)
5. Bella, A., Ferri, C., Hernández-Orallo, J., Ramirez-Quintana, M.J.: Quantification via probability estimators. In: IEEE ICDM. pp. 737–742. IEEE (2010)
6. Brazdil, P., Carrier, C.G., Soares, C., Vilalta, R.: Metalearning: Applications to data mining. Springer Science & Business Media (2008)
7. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
8. das Dôres, S.N., Alves, L., Ruiz, D.D., Barros, R.C.: A meta-learning framework for algorithm recommendation in software fault prediction. In: SAC – ACM SIGAPP. pp. 1486–1491. ACM, Pisa, Italy (Apr 2016)
9. Flach, P.: Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press, USA (2012)
10. Forman, G.: Counting positives accurately despite inaccurate classification. In: ECML PKDD. pp. 564–575. Springer (2005)
11. Forman, G.: Quantifying trends accurately despite classifier error and class imbalance. In: KDD – ACM SIGKDD. pp. 157–166. ACM (2006)
12. Garcia, L., Lorena, A., Lehmann, J.: Ecol: Complexity measures for classification problems (2018)
13. González, P., Castaño, A., Chawla, N.V., Coz, J.J.D.: A review on quantification learning. ACM Comput. Surv. **50**(5), 74 (2017)

14. González, P., Moreo, A., Sebastiani, F.: Binary quantification and dataset shift: an experimental investigation. Data Min. Knowl. Discovery pp. 1–43 (2024)
15. González-Castro, V., Alaiz-Rodríguez, R., Alegre, E.: Class distribution estimation based on the hellinger distance. Inf. Sci. **218**, 146 – 164 (2013)
16. Hassan, W., Maletzke, A., Batista, G.: Accurately quantifying a billion instances per second. In: IEEE DSAA. pp. 1–10. IEEE, Sydney, Australia (2020)
17. Hassan, W., Maletzke, A., Batista, G.: Pitfalls in quantification assessment. In: International Workshop on Learning to Quantify: Methods and Applications(LQ2021). vol. 3052. CIKM, GoldCoast, Australia (05 Nov 2021)
18. Kelly, M., Longjohn, R., Nottingham, K.: The uci machine learning repository. `https://archive.ics.uci.edu` (2023)
19. Maletzke, A., dos Reis, D., Cherman, E., Batista, G.: Dys: a framework for mixture models in quantification. In: AAAI. Honolulu, United States (2019)
20. Maletzke, A., Hassan, W., Reis, D.d., Batista, G.: The importance of the test set size in quantification assessment. In: IJCAI. pp. 2640–2646. Yokohama, Japan (July 2020)
21. Mantovani, R.G., Rossi, A.L., Alcobaça, E., Vanschoren, J., de Carvalho, A.C.: A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves svm classifiers. Inf. Sci. **501**, 193–221 (2019)
22. Milli, L., Monreale, A., Rossetti, G., Giannotti, F., Pedreschi, D., Sebastiani, F.: Quantification trees. In: IEEE ICDM. pp. 528–536. IEEE, Abu Dhabi, United Arab Emirates (2013)
23. Olmo, J.L., Romero, C., Gibaja, E., Ventura, S.: Improving meta-learning for algorithm selection by using multi-label classification: A case of study with educational data sets. Int. J. Comput. Intell. Syst. **8**(6), 1144–1164 (2015)
24. Pimentel, B.A., De Carvalho, A.C.: A new data characterization for selecting clustering algorithms using meta-learning. Inf. Sci. **477**, 203–219 (2019)
25. Rivolli, A., Garcia, L.P., Soares, C., Vanschoren, J., de Carvalho, A.C.: Meta-features for meta-learning. Knowledge-Based Syst. **240**, 108101 (2022)
26. Saerens, M., Latinne, P., Decaestecker, C.: Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. Neural Comput. **14**(1), 21–41 (2002)
27. Schumacher, T., Strohmaier, M., Lemmerich, F.: A comparative evaluation of quantification methods. arXiv preprint arXiv:2103.03223 (2021)
28. Sebastiani, F.: Evaluation measures for quantification: An axiomatic approach. Inf. Retr. J. **23**(3), 255–288 (2020)
29. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: Openml: networked science in machine learning. SIGKDD Explorations **15**(2), 49–60 (2013). `https://doi.org/10.1145/2641190.2641198`
30. Wang, G., Song, Q., Sun, H., Zhang, X., Xu, B., Zhou, Y.: A feature subset selection algorithm automatic recommendation method. J. Artif. Intell. Res. **47**, 1–34 (2013)
31. Zhang, X., Li, R., Zhang, B., Yang, Y., Guo, J., Ji, X.: An instance-based learning recommendation algorithm of imbalance handling methods. Appl. Math. Comput. **351**, 204–218 (2019)
32. Zhu, X., Yang, X., Ying, C., Wang, G.: A new classification algorithm recommendation method based on link prediction. Knowledge-Based Syst. **159**, 171–185 (2018)