A Robust Experimental Evaluation of Automated Multi-Label Classification Methods

Alex G. C. de Sá
Computer Science Department
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brazil
alexgcsa@dcc.ufmg.br

Cristiano G. Pimenta
Computer Science Department
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brazil
cgpimenta@dcc.ufmg.br

Gisele L. Pappa
Computer Science Department
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brazil
glpappa@dcc.ufmg.br

Alex A. Freitas
School of Computing
University of Kent
Canterbury, Kent, United Kingdom
A.A.Freitas@kent.ac.uk

ABSTRACT

Automated Machine Learning (AutoML) has emerged to deal with the selection and configuration of algorithms for a given learning task. With the progression of AutoML, several effective methods were introduced, especially for traditional classification and regression problems. Apart from the AutoML success, several issues remain open. One issue, in particular, is the lack of ability of AutoML methods to deal with different types of data. Based on this scenario, this paper approaches AutoML for multi-label classification (MLC) problems. In MLC, each example can be simultaneously associated to several class labels, unlike the standard classification task, where an example is associated to just one class label. In this work, we provide a general comparison of five automated multi-label classification methods – two evolutionary methods, one Bayesian optimization method, one random search and one greedy search – on 14 datasets and three designed search spaces. Overall, we observe that the most prominent method is the one based on a canonical grammar-based genetic programming (GGP) search method, namely Auto-MEKA$_{GGP}$. Auto-MEKA$_{GGP}$ presented the best average results in our comparison and was statistically better than all the other methods in different search spaces and evaluated measures, except when compared to the greedy search method.

CCS CONCEPTS

• **Computing methodologies → Machine learning approaches**; **Machine learning**; **Search methodologies**; *Genetic programming*; *Randomized search*;

KEYWORDS

Automated Machine Learning (AutoML), Multi-Label Classification, Search Methods, Search Spaces

ACM Reference Format:
Alex G. C. de Sá, Cristiano G. Pimenta, Gisele L. Pappa, and Alex A. Freitas. 2020. A Robust Experimental Evaluation of Automated Multi-Label Classification Methods. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3390231

1 INTRODUCTION

We are experiencing the era of data. With its great availability, people in general (e.g., practitioners, data scientists, and researchers) are trying hard to extract useful information encoded on data [22]. This resulted in an ever-growing popularity and the indiscriminate use of machine learning (ML) algorithms by many types of users.

The field of *Automated Machine Learning* (AutoML) [13] has emerged to help this wide and heterogeneous public. This field has the purpose of democratizing ML in a way ML can be used with less difficulties by general audiences. In addition, AutoML also aims to assist experienced data scientists. In both scenarios, the field of AutoML has the scope of recommending learning algorithms (and often their hyper-parameters' settings too) when people face a particular problem that might be (partially or totally) solved with ML. Broadly speaking, AutoML proposes to deal with users' biases by customizing the solutions (in terms of algorithms and configurations) to ML problems following different approaches.

AutoML has been successfully and mainly employed to solve traditional (single-label) classification and regression problems [9]. However, this work is interested in AutoML methods for a different and specific type of data, called *Multi-Label Classification* (MLC) [11, 26, 30]. The goal of MLC is to learn a model that expresses the relationships between a set of predictive features (attributes) describing the examples and a predefined set of class labels. In MLC, each example can be simultaneously associated to one or more class labels. Each class label is represented by a discrete value.

When compared to single-label classification (SLC), MLC can be considered a more challenging task, mainly due to the following reasons. First, an MLC algorithm needs to consider the label correlations (i.e., detecting whether or not they exist) in order to learn a model that produces accurate classification results [30]. Second, given the usual limited number of examples for each class label in the dataset [11], the generalization in MLC is considered harder than SLC, as the MLC algorithm needs more examples to create a good model from such complex data [7]. Third, there is a strain

GECCO '20, July 8–12, 2020, Cancún, Mexico
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7128-5/20/07…$15.00
https://doi.org/10.1145/3377930.3390231

to evaluate MLC classifiers as several metrics follow contrasting aspects to define what is a good MLC prediction [18]. Finally, the learning algorithms applied to solve MLC problems need more computational resources than the ones used to solve SLC [11]. This is mainly due to MLC being a generalization of SLC, so that the algorithms need to look at several labels instead of just one.

We claim that these aforementioned challenges are part of the reason why AutoML for MLC problems (i.e., AutoMLC) has not been sufficiently explored. Taking it into consideration, this work performs an assessment of popular *search methods* for AutoMLC, including evolutionary methods, a Bayesian optimization method and blind-*search methods*. For that, we propose two novel AutoMLC *search methods*. The first is an extension of the work of de Sá *et al.* [4] on Grammar-based Genetic Programming (GGP) [16, 28] for AutoMLC, named Auto-MEKA$_{GGP}$. Our extension adds to the GGP core a speciation approach [1] aiming to improve the diversity of the produced solutions. The second *search method* is a Bayesian optimization (BO) method, namely Sequential Model-based Algorithm Configuration (SMAC) [12] – note that there was no such methods previously proposed in the AutoMLC literature. As both proposed methods are based on the well-known MLC MEKA framework [20], we named these *search methods* as Auto-MEKA$_{spGGP}$ and Auto-MEKA$_{BO}$, respectively.

We compare these two proposed *search methods* with Auto-MEKA$_{GGP}$ [4], a random search (namely, Auto-MEKA$_{RS}$) and a greedy search (namely, Auto-MEKA$_{GS}$) on three designed MLC search spaces (namely, Small, Medium and Large) over 14 benchmarking datasets. Finally, in this work, we use five performance measures for evaluating these methods, due to the additional degree of freedom that the MLC algorithms' setting introduces [14].

The experimental results show that Auto-MEKA$_{GGP}$ mostly presented the best average results and also the best average ranks for several *search spaces* and measures. Besides, Auto-MEKA$_{GGP}$ was the only method to be statistically better than all other evaluated *search methods* in different occasions (i.e., performance measures *versus search spaces*), except when compared to Auto-MEKA$_{GS}$.

Although this is a positive result for the evolutionary methods, we believe that more robust methods – such as Auto-MEKA$_{GGP}$, Auto-MEKA$_{spGGP}$ and Auto-MEKA$_{BO}$– can still improve their predictive performances. With this in mind, we observe that these methods could not satisfactorily trade-off between exploration and exploitation as they were not statistically and simultaneously better than pure-exploration and pure-exploitation methods (i.e., Auto-MEKA$_{RS}$ and Auto-MEKA$_{GS}$, respectively).

The results also show that there is a high correlation between the size (and definition) of the search space and the effectiveness of AutoMLC methods to select and configure algorithms. When looking at the predictive accuracy of the AutoMLC methods, we have an indication that as the size of an AutoMLC's *search space* decreases, pure-exploration and/or pure-exploitation AutoMLC *search methods* tend to have similar results to robust AutoMLC methods (such as the ones presented in this work).

The remainder of this paper is organized as follows. Section 2 introduces MLC and Section 3 reviews related work on AutoMLC. Section 4 details AutoMLC in terms of the proposed *search spaces* and evaluated *search methods* that are included in the experimental comparison, while Section 5 presents and discusses the obtained results. Finally, Section 6 draws some conclusions and discusses directions for future work.

2 MULTI-LABEL CLASSIFICATION

There is a great number of works on traditional single-label classification (SLC) for machine learning (ML) [29]. In SLC, each example is defined by a tuple (X, y), where $X = \{x_1, ..., x_d\}$ is a d-dimensional vector representing the feature space (i.e., the categorical and/or numerical characteristics of that example) and y is the class value, where $y \in L$, a set of disjoint class labels. In SLC, each example is strictly associated to a single class label.

Nevertheless, there is an increasing number of applications that require associating an example to more than one class label [10], such as medical diagnosis and protein function prediction. This classification scenario is better known as *Multi-Label Classification* (MLC). According to [25], each example in MLC is represented by a tuple (X, Y), where X is the $d-$dimensional feature vector, and $Y \subseteq L$ is a set of non-disjoint class labels. Hence, we would like to find an MLC model h: $X \rightarrow 2^{|L|}$ such that h maximizes a quality criterion λ.

The literature divides MLC algorithms into three categories [10]: problem transformation (PT), algorithm adaptation (AA) and ensemble or meta-algorithms methods (Meta-MLC). Whereas PT methods transform the multi-label problem into one or more single-label classification problems, AA methods simply extend single-label classification algorithms so they can directly handle multi-label data. Finally, Meta-MLC methods act on top of PT or AA multi-label classifiers, aiming to combine the results of MLC algorithms and produce models with more robust predictive performances.

Among the great number of MLC algorithms [11, 26, 30], it is important to mention three methods that transform an MLC problem into one or many SLC problems: Label Powerset (LP), Binary Relevance (BR) and Classifier Chain (CC). LP creates a single class for each unique set of labels that is associated with at least one example in a multi-label training set. BR, in turn, learns $|L|$ independent binary classifiers, one for each label in the labelset L. Finally, CC changes the BR method by chaining the binary classifiers. In this case, the feature space of each link in the chain is increased with the classification outputs of all previous links.

3 RELATED WORK ON AUTOMLC

Most AutoML methods in the literature were designed to solve the conventional single-label classification and regression tasks [9], and can not handle multi-label data. As far as we know, there are only a few works related to *Automated Multi-Label Classification* (AutoMLC).

[3] developed a meta-model (i.e., a 20-Nearest Neighbors classifier) for selecting one out of 11 multi-label classification algorithms, taking into account 30 characterizing measures and 36 meta-datasets. Nevertheless, as it is a preliminary work, it only selects the MLC algorithm, not setting the algorithm's hyper-parameters.

Evolutionary Multi-Label Ensemble (EME) [17] encompasses the problem of selecting MLC algorithms to compose MLC ensembles. The main idea of EME stands on the simplicity of each ensemble's multi-label classifier, which is focused on a small subset of the labels, but still considering the relationships among them and avoiding the high complexity of the output space. Nevertheless, EME takes into account only one type of model to compose the ensembles (i.e.,

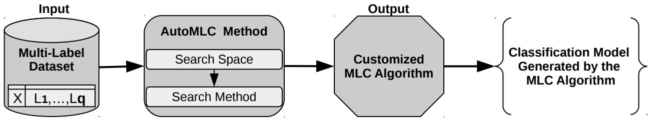


Figure 1: The general AutoMLC framework to select and configure MLC algorithms.

the model produced by label powerset), so it is not sufficient to deal with all types of MLC problems.

Furthermore, [27] proposed an extension to a canonical hierarchical planing method (i.e., ML-Plan) to the MLC context. They named this method as ML^2-Plan (Multi-Label ML-Plan). Basically, ML^2-Plan is implemented as a global best-first search over the graph induced by the planning problem at hand.

Finally, [4, 5] proposed two AutoML methods for MLC problems: GA-Auto-MLC and Auto-MEKA$_{GGP}$. Whereas GA-Auto-MLC employs a real-coded genetic algorithm [8] to perform its search, Auto-MEKA$_{GGP}$ uses a grammar-based genetic programming algorithm [16, 28]. Auto-MEKA$_{GGP}$ is a robust enhancement of GA-Auto-MLC to handle huge and, consequently, complex MLC *search spaces*. Because of that, we decided to only include Auto-MEKA$_{GGP}$ into our experiments. We will further discuss it in Section 4.

4 AUTOML METHODS FOR MULTI-LABEL CLASSIFICATION

This section introduces a generic AutoMLC framework followed by all AutoMLC methods evaluated in this paper. As illustrated in Figure 1, the AutoMLC method receives as input a specific multi-label dataset (with the feature space X and the class labels L_1 to L_q). Structurally, the evaluated AutoMLC methods have two main components: the *search space* and the *search method*.

The *search space* consists of the main building blocks (e.g., the prediction threshold values, the hyper-parameters and the algorithms at the SLC level) from previously designed MLC algorithms. To explore this *search space*, the AutoMLC method uses a *search method*, which finds appropriate MLC algorithms to the dataset at hand. However, the performance of the *search method* depends on what is specified in the *search space*.

Once the search is over, the AutoMLC method outputs an MLC algorithm tailored to the input dataset based on that *search space*. This MLC algorithm is specifically selected and (hyper-)parameterized to this dataset, although it could be applied to any multi-label dataset. In the end, the customized MLC algorithm returns an MLC model, and consequently, its classification results.

The evaluation presented in this paper considers three *search spaces*, namely Small, Medium and Large, which differ from each other in terms of complexity. These three *search spaces* are explored using five *search methods*: Auto-MEKA$_{GGP}$, Auto-MEKA$_{SpGGP}$, Auto-MEKA$_{BO}$, Auto-MEKA$_{RS}$ and Auto-MEKA$_{GS}$, as detailed in Section 4.2.

4.1 Search Spaces

To design the *search spaces* for the AutoMLC methods being evaluated, we first performed a deep study about multi-label classification in the MEKA software. We analyzed in detail all the algorithms and their hyper-parameters, the constraints associated with different hyper-parameter settings, the hierarchical nature of operations performed by problem transformation methods and meta-algorithms, among other issues.

Based on that, we designed three *search spaces*[1]: Small, Medium and Large. The reason behind this threefold modeling is basically because we want to test different levels of *search space* complexity.

For the *search space* Small, for instance, we have five MLC algorithms, where four of them can be combined with other five SLC algorithms, as they are from the PT category. The only algorithm that can not be combined with SLC algorithms is ML-BPNN, which belongs to the AA category. Therefore, the *search space* Small consists of 10 learning algorithms – five MLC algorithms and five SLC algorithms, which gives a set of 21 combinations of learning algorithms, where the AA category counts as one combination.

In contrast, the *search space* Medium has 30 learning algorithms – 15 MLC algorithms and 15 SLC algorithms, which produces 211 combinations of algorithms. Finally, the *search space* Large has a total of 54 learning algorithms – 26 MLC algorithms and 28 SLC algorithms, which produces 16,568 possible combinations of learning algorithms.

Although the main difference between the Small and Medium *search spaces* is the number of learning algorithms, note that, when comparing Medium to Large, we have a change on the structure of the *search space*. This happens because we only added meta-algorithms at the MLC and SLC levels into Large. Hence, we have more levels in the multi-label hierarchy to consider. For example, when a *search method* is selecting a new MLC algorithm in this *search space*, it must decide whether to include or exclude meta-algorithms at the MLC and SLC levels. As a result, this *search space* is hierarchically more complex than the other two (i.e., Small and Medium).

Taking into account the number of learning algorithms, the number of hyper-parameters, and the constraints in the choices of algorithms' components and (hyper)-parameters in MEKA, we estimated the size of the three *search spaces*[2]. In total, the *search space* Small has $[(5.070\times10^7)+(8.078\times10^8\times m)+(2.535\times10^{10}\times q)]$ possible MLC algorithm configurations (i.e., a given set of learning algorithms with their respective hyper-parameters), where m is the number of features (or attributes) and q is the number of labels of the dataset. The *search space* Medium, on the other hand, is estimated as having $[(2.545 \times 10^{16}) + (8.078 \times 10^8 \times m) + (1.151 \times 10^{12} \times q)]$ possible MLC algorithm configurations. Finally, the *search space* Large is estimated to have approximately $[(6.555 \times 10^{29}) + (1.443 \times 10^{14} \times m) + (3.042 \times 10^{22} \times q) + (1.291 \times 10^{27} \times \sqrt{q})]$ possible MLC algorithm configurations.

4.2 Search Methods

This section details the five *search methods* used in our comparison. They all follow the same methodology.

Each *search method* starts its own iterative process by generating, evaluating and looking for MLC algorithms configurations. To perform the evaluation, the *search methods* use the average of four well-known measures [10, 25]: Exact Match (EM), Hamming Loss (HL), F_1 Macro averaged by label (FM) and Ranking Loss (RL),

[1] For more details about the MLC and SLC algorithms and meta-algorithms that compose the *search spaces*, see the supplementary material.

[2] In these estimations, the real-valued hyper-parameters have always taken 100 different discrete values.

as indicated in Equation 1. The *search method* keeps iterating while a maximum time budget is not reached. At the end, the best MLC algorithm configuration in accordance to the quality criteria is returned and assessed in the test set.

$$Fitness = \frac{EM + (1 - HL) + FM + (1 - RL)}{4} \qquad (1)$$

Regarding the MLC evaluation measures, EM is a very strict metric, as it only takes the value one when the predicted label set is an exact match to the true label set for an example, and takes the value zero otherwise. HL, in turn, calculates how many example-label pairs are misclassified. Furthermore, FM is the harmonic mean between precision and recall, and its average is first calculated for each label and, after that, across all the labels. Finally, RL measures the number of times that irrelevant labels are ranked higher than relevant labels, i.e., it penalizes the label pairs that are reversely ordered in the ranking for a given example. All four metrics are within the $[0, 1]$ interval. However, whereas the EM and FM measures should be maximized, the HL and RL measures should be minimized. Hence, HL and RL are subtracted from one in Equation 1 to make the search maximize the fitness function.

4.2.1 Auto-MEKA$_{GGP}$. This method was proposed in [4], and relies on a Grammar-based Genetic Programming (GGP) approach [16, 28], which has the advantage of hierarchically exploring the nature of the AutoMLC problem. In this case, the grammar encompasses the *search space* of MLC algorithms and hyper-parameter settings.

In Auto-MEKA$_{\text{GGP}}$, each individual expresses an MLC algorithm configuration, and is represented by a derivation tree generated from the grammar. Individuals are first generated by choosing at random a valid production rule, and then mapping it into an MLC algorithm (with a specific hyper-parameter setting).

Figure 2 details the whole mapping process followed by the evaluation process for each GGP individual. In the example of Figure 2, ellipsoid nodes are the grammar's non-terminals, whereas the rectangles are the terminals.

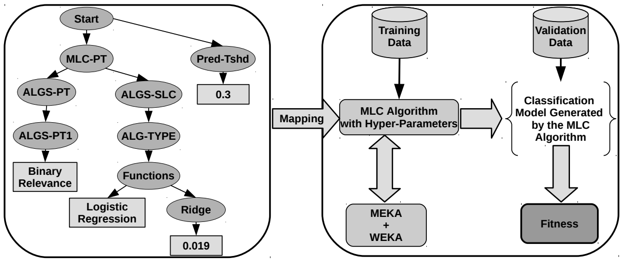


Figure 2: Individual's evaluation process in Auto-MEKA$_{GGP}$.

The mapping process takes the terminals from the tree and constructs a valid MLC algorithm. The mapping in Figure 2 will produce the following MLC algorithm: a Binary Relevance method combined with a Logistic Regression algorithm (with the hyper-parameter ridge set to 0.019), using a threshold of 0.3 to classify the MLC data.

Next, individuals have their fitness calculated as previously explained, and undergo tournament selection. The GGP operators (i.e., Whigham's crossover and mutation [28]) are applied to the selected individuals to create a new population. These operators also respect the grammar constraints, ensuring that the produced individuals represent valid solutions.

4.2.2 Auto-MEKA$_{spGGP}$. This novel AutoMLC method enhances the search mechanisms of Auto-MEKA$_{GGP}$ by adding a speciation process [1] into its *search method*. The general idea is to use Grammar-based Genetic Programming with Speciation (spGGP) to improve the trade-off between exploration and exploitation of the search for MLC algorithms and hyper-parameter settings. Because the proposed *search spaces* have an exponential size and a complex hierarchical nature, it may be crucial to use this approach to deal with these aspects. A species is a set of individuals that resemble each other more inherently than the individuals in another species [1]. In speciation-based evolutionary computation, the objective is to emphatically restrict mating to those among like individuals from the population. In this case, likeness among individuals is identified if they have similar genotypes or phenotypes.

In this work, we defined a set of species based on the types of MLC hyper-parameters (i.e., categorical, discrete or continuous) and their interactions. It is worth noting that, whilst the categorical hyper-parameters take Boolean and string-based values (e.g., an algorithm name or a procedure name in an algorithm), the discrete hyper-parameters only take integer values. Therefore, our speciation-based method focuses not exclusively on the choice of the learning algorithms but primarily on the different types of hyper-parameters, where the choice of the learning algorithms is set as a special case of a categorical hyper-parameter.

In general, we would like to understand if there is a dependence between the final AutoMLC predictive performance and the types (and the interactions) of hyper-parameters for a given dataset. For instance, if we would like to recommend MLC algorithms for two datasets with different characteristics, understanding only the categorical hyper-parameters for the first dataset may be more beneficial than understanding discrete and/or continuous hyper-parameters. This could be the opposite for the second dataset.

In this context, we design eight species. Different species specialize on optimizing different combinations of hyper-parameter types and their interactions. All species have instances of all learning algorithms at both MLC and SLC levels based on the defined *search space*, but they vary on the types of hyper-parameters that are left with their default values during evolution and cannot be updated. In the descriptions below, the settings of the hyper-parameters we refer to can be changed by the evolutionary process, while all others are set to their default values. The species may vary:

(1) **Learning algorithms:** Only the categorical hyper-parameters referring to the names of the (traditional and meta) learning algorithms at the MLC and SLC levels can be combined and evolved.
(2) **Learning algorithms and common categorical hyper-parameters:** Together with the categorical hyper-parameters indicating the names of the learning algorithms (species 1), this species also allows the combination and evolution of common categorical hyper-parameters (e.g., the names of a metric). In addition, this species also encompasses Boolean hyper-parameters.
(3) **Learning algorithms and discrete hyper-parameters:** This species considers, alongside with the categorical hyper-parameters that indicate the names of the learning algorithms, the discrete (integer) hyper-parameters.

(4) **Learning algorithms and continuous hyper-parameters:** This species allows the modification and combination of the continuous hyper-parameters of species 1.
(5) **Learning algorithms and the combination of common categorical and discrete hyper-parameters:** In this species, we evolve the individuals considering the learning algorithms themselves (species 1) together with common categorical and discrete hyper-parameters.
(6) **Learning algorithms and the combination of common categorical and continuous hyper-parameters:** In this case, we make the evolutionary process consider the hyper-parameters representing the learning algorithms, the common categorical hyper-parameters and the continuous hyper-parameters.
(7) **Learning algorithms and the combination of discrete and continuous hyper-parameters:** This species allows the combination of the names of the learning algorithms with discrete and continuous hyper-parameters.
(8) **All types of hyper-parameters:** This species is more general, and all types of hyper-parameters (categorical referring to the names of the learning algorithms, common categorical, discrete, continuous hyper-parameters) are considered to be explored/exploited.

The first step of Auto-MEKA$_{\text{spGGP}}$'s evolutionary process is the initialization procedure, where we generate for each species a population of individuals, which are represented by trees and built based on a specific grammar for that species.

Auto-MEKA$_{\text{spGGP}}$ also differs from Auto-MEKA$_{\text{GGP}}$ in the crossover operator, which can be performed for both intra-species and inter-species individuals. By interchangeably using both types of crossover operations, we have more chances to test unknown regions of the *search space* (exploration) when using the inter-species crossover, while a more local search over the different types of hyper-parameters is performed in each species by the intra-species crossover (exploitation).

It is worth noting that we decided to design the mutation operator as a local operator in each specie. By doing that, Whigham's mutation uses the *grow* method on the individual's derivation tree but ensures that the MLC grammar of the current species is applied over the *grow* method.

4.2.3 *Auto-MEKA$_{BO}$.*

This proposed Bayesian Optimization (BO) AutoMLC method employs the Sequential Model-based Algorithm Configuration (SMAC) [12] as a procedure to search for suitable MLC configurations. In our generic framework, Auto-MEKA$_{BO}$ can be categorized as a sophisticated extension of Auto-WEKA [24].

Hence, given a dataset and a *search space*, Auto-MEKA$_{\text{BO}}$ uses a performance model (in our case, a Random Forest) to robustly select the MLC configurations. This model is initialized with a default MLC algorithm with default hyper-parameter settings. In the case of Auto-MEKA$_{\text{BO}}$, we initialize the model with different algorithms as the *search spaces* allow different types of learning algorithms. For the *search space* Small, we run and include into the model the results of the classifier chain (CC) algorithm using the naïve Bayes (NB) classification algorithm at the single-label base level.

As the *search space* Medium is similar to Small in terms of the hierarchical levels, we keep the CC algorithm at the multi-label level. However, we have tried to improve the single-label classification level by using a strong algorithm, i.e., we use a more sophisticated Bayesian network classifier (BNC) algorithm instead of a simple NB classification algorithm. Hence, at this level, the K2 algorithm is employed.

Finally, for the *search space* Large, we define as the initial configuration to the model the random subspace meta-algorithm for multi-label classification (RSML), using the Bayesian classifier chain (BCC) algorithm at the multi-label base level, the locally weighted learning (LWL) algorithm at the single-label meta level, and the BNC K2 algorithm at the single-label base level. Except for RSML, which is a robust meta-algorithm, the other levels were chosen in an arbitrary fashion, although they are also considered strong algorithms in the machine learning literature.

After this initialization step, we choose the next configuration from the MLC *search space* in the configuration files, relying on this performance model. To do that, the SMAC method is used to select a better configuration. Next, this MLC configuration is evaluated in the MEKA framework and then compared with the best MLC configuration found so far. If the current configuration has a better score than the current best configuration, it is saved and set as the new best configuration. Otherwise, the process continues by verifying if the time budget was reached. If this criterion is met, Auto-MEKA$_{\text{BO}}$ returns the best configuration found up to now. Otherwise, the last evaluated MLC configuration is added to the performance model with its corresponding quality value, updating it. The process continues by following these same steps until the time budget expires.

4.2.4 *Auto-MEKA$_{RS}$.*

The AutoMLC random *search method* (RS) iterates over the predefined MLC *search space* at random. First, it creates p MLC algorithm configurations, evaluates them and saves the best configuration in terms of the proposed quality measure (see Equation 1) into a list. Next, it creates other p new MLC algorithm configurations, evaluates these configurations and saves the best at this iteration into the same list. RS keeps doing this procedure until the total time budget is reached. At the end, it returns the best MLC algorithm configuration from the list based on the quality measure.

4.2.5 *Auto-MEKA$_{GS}$.*

The AutoMLC greedy *search method* (GS) starts by generating an initial random solution (i.e., an MLC algorithm configuration), which is set as the current best. From this solution, we generate p others by performing local changes into its representation. We use the aforementioned grammar-based representation for both random and greedy searches. Thus, from the grammar, we generate a derivation tree and employ Whigham's mutation [16, 28] to perform local operations in the respective tree. We evaluate these solutions (see Equation 1) and check if one of them has a better quality score than the current best MLC configuration. If so, we update the best configuration with the best score. Otherwise, we maintain the best MLC configuration. Next, from the current best configuration, we continue looking at its neighbors to create, evaluate and possibly find new better solutions. This search process remains while the final time budget is not reached. At the end, the best found MLC algorithm configuration is returned based on the proposed quality measure.

5 EXPERIMENTAL ANALYSIS

This section presents the experimental results of the AutoMLC methods discussed in the previous section. The experiments involve a set of 14 datasets selected from the KDIS (Knowledge and

Discovery Systems) repository[3], as described in Table 1. These datasets were chosen based on their differences in application domain, the number of instances (n), the number of features (m), the number of labels (q), the label cardinality – the average number of labels associated with each example in the dataset (Card.), the label density – the average number of labels associated with each example divided by the number of labels (Dens.), and the label diversity – the percentage of labelsets in the dataset divided by the number of possible labelsets (Div.).

Table 1: Datasets used in the experiments.

Datasets	Acronym	n	m	q	Card.	Dens.	Div.
Bibtex	BTX	7395	1836	159	2.402	0.015	0.386
Birds	BRD	645	260	19	1.014	0.053	0.206
CAL500	CAL	502	68	174	26.044	0.150	1.000
CHD_49	CHD	555	49	6	2.580	0.430	0.531
Enron	ENR	1702	1001	53	3.378	0.064	0.442
Flags	FLG	194	19	7	3.392	0.485	0.422
Genbase	GBS	662	1186	27	1.252	0.046	0.048
GpositivePseAAC	GPP	519	440	4	1.008	0.252	0.438
Medical	MED	978	1449	45	1.245	0.028	0.096
PlantPseAAC	PPA	978	440	12	1.079	0.090	0.033
Scene	SCN	2407	294	6	1.074	0.179	0.234
VirusPseAAC	VPA	207	440	6	1.217	0.203	0.266
Water-quality	WQT	1060	16	14	5.073	0.362	0.778
Yeast	YST	2417	103	14	4.237	0.303	0.082

The two evolutionary *search methods* (i.e., Auto-MEKA$_{GGP}$ and Auto-MEKA$_{spGGP}$) were run with 80 individuals evolved considering a time budget of five hours, tournament selection of size two, elitism of one individual, and crossover and mutation probabilities of 0.8 and 0.2, respectively. For these two methods, the learning and validation sets are also resampled from the training set every five generations in order to avoid overfitting. Additionally, we use time and memory budgets for each MLC algorithm (generated by the *search methods*) of three minutes and 2GB of RAM, respectively. If any MLC algorithm reaches these budgets, it is assigned the lowest fitness, i.e., zero. Furthermore, the following convergence criterion is considered: at each iteration, we check if the best individual has remained the same for over five generations and the *search method* has run for at least 20 generations. If this happens, we restart the evolutionary process with another pseudo-random seed.

In the case of Auto-MEKA$_{spGGP}$, as we have eight species, we specify 10 individuals for each species. We define Auto-MEKA$_{spGGP}$'s convergence criteria for each species individually. We set the intra-species and inter-species crossover probabilities for Auto-MEKA$_{spGGP}$ as 0.5 and 0.5, respectively. On the other hand, Auto-MEKA$_{BO}$ has kept only the time and memory budgets – i.e., five hours of run for its respective *search method*, and three minutes and 2GB of time and memory budgets for each produced MLC algorithm, respectively. As in the evolutionary methods, the MLC algorithms that reach time and memory budgets are assigned a fitness of zero.

In order to be fair in the comparisons with the evolutionary methods, we set the value of p equal to 80 for both Auto-MEKA$_{RS}$ and Auto-MEKA$_{GS}$, i.e., the methods that implement random search and greedy search, respectively.

All experiments were run using a stratified five-fold cross-validation procedure [21]. This section shows three measures considered when evaluating the results in terms of classification quality: hamming loss (HL), ranking loss (RL), and the general measure we defined as the fitness/quality criteria (see Equation 1)[4]. Given the average values – based on the 14 datasets – for all methods on a particular measure, results are evaluated using an adapted Friedman test followed by a Nemenyi post-hoc test with the usual significance level of 0.05 [6].

5.1 Experimental Results

Table 2 shows the average values, the average ranks and the final statistical analysis for HL, RL and fitness measures, respectively.

Table 2: Comparison of the hamming loss (to be minimized), ranking loss (to be minimized) and fitness (to be maximized) obtained by all *search methods* in the test set for the three designed *search spaces* with five hours of execution.

Search Space	Evaluated Result	spGGP	GGP	BO	RS	GS
Hamming Loss (HL)						
Small	**Avg. Values**	0.135	**0.134**	0.208	0.137	0.135
	Avg. Ranks	3.143	**2.250**	3.714	3.107	2.786
	Stat. Comparison	no differences among all methods				
Medium	**Avg. Values**	0.157	0.136	**0.134**	0.139	0.142
	Avg. Ranks	4.036	**2.251**	2.607	3.00	3.107
	Stat. Comparison	{GGP} ≻ {spGGP}, no differences among the others				
Large	**Avg. Values**	0.137	0.134	**0.130**	0.143	0.139
	Avg. Ranks	3.214	2.571	**2.07**	4.00	3.142
	Stat. Comparison	{BO} ≻ {RS}, no differences among the others				
Ranking Loss (RL)						
Small	**Avg. Values**	**0.153**	0.161	0.210	0.167	0.167
	Avg. Ranks	2.321	**2.214**	3.750	3.429	3.286
	Stat. Comparison	no differences among all methods				
Medium	**Avg. Values**	**0.146**	0.150	0.152	0.156	0.148
	Avg. Ranks	**2.679**	2.821	3.250	3.321	2.929
	Stat. Comparison	no differences among all methods				
Large	**Avg. Values**	0.147	**0.135**	0.149	0.157	0.140
	Avg. Ranks	2.821	**2.536**	2.786	4.071	2.786
	Stat. Comparison	no differences among all methods				
Fitness						
Small	**Avg. Values**	0.626	**0.631**	0.590	0.624	0.626
	Avg. Ranks	2.679	**1.964**	3.857	3.535	2.964
	Stat. Comparison	{GGP} ≻ {BO}, no differences among the others				
Medium	**Avg. Values**	0.634	0.639	0.626	0.629	**0.645**
	Avg. Ranks	3.107	**2.643**	2.929	3.429	2.893
	Stat. Comparison	no differences for all methods				
Large	**Avg. Values**	0.632	**0.635**	**0.635**	0.626	0.632
	Avg. Ranks	2.607	**2.429**	2.571	4.286	3.107
	Stat. Comparison	{spGGP, GGP, BO} ≻ {RS}, no differences among the others				

The results for the HL measure with five hours, in Table 2, showed that Auto-MEKA$_{GGP}$ had the best average values and ranks in the *search space* Small, but it had only the best average rank in the *search space* Medium. In addition, Auto-MEKA$_{BO}$ obtained the best average value and Auto-MEKA$_{GGP}$ was statistically better than Auto-MEKA$_{spGGP}$ in the *search space* Medium. Nonetheless, there is no indication of statistical difference for the other cases of *search space* Medium, neither for any cases of Small. On the other hand, when considering the *search space* Large, Auto-MEKA$_{BO}$ had the best average value and the best average rank. By looking at the statistical results of HL in Table 2, we can observe that only Auto-MEKA$_{BO}$ was able to significantly outperform Auto-MEKA$_{RS}$ for the *search space* Large. In the other cases, we do not see an indication of the *search methods* to trade-off well between exploration and exploitation. Therefore, based on these HL results, we can conclude that the size of the *search space* has a strong influence on the performance of the *search method.*

[3] The datasets are also available at: http://www.uco.es/kdis/mllresources/.

[4] We did not present and analyze the results of exact match and F_1 Macro averaged by label due to their similar predictive performances to HL, RL and/or fitness measures.

In addition, Table 2 summarizes the results for the evaluation measure from a different evaluation context, i.e., RL. Whilst Auto-MEKA$_{spGGP}$ was the *search method* with the best average value in the *search space* Small, Auto-MEKA$_{GGP}$ achieved the best average rank in this scenario. Distinctly, Auto-MEKA$_{spGGP}$ selected and configured MLC algorithms in a way that produced the best average value and rank in the *search space* Medium. For the *search space* Large, Auto-MEKA$_{GGP}$ was the *search method* with the best average value and the best average rank.

The results of RL did not present any evidence of statistical significance. Hence, the *search methods* did not differ from each other. As this metric comes from another context, we would like to understand why it presented such a flat result for all *search spaces* and *search methods*. We believe that the RL measure is very conservative and it is not sensitive to different choices of multi-label classifiers. As it only penalizes reversed pairs of labels into the ranking and it does not take into account the label-pair depth into the ranking to penalize, this can make this measure not good enough to be used in isolation to evaluate MLC algorithms. In future work, it might be interesting to evaluate whether this measure is appropriate to be part of our study or whether we should consider a rank-based measure that takes into account the depth of the ranking.

HL and RL are measures that compose the fitness/quality function. We also analyze the results of the fitness measure to have a better assessment of the results of *search methods*. In the last part of Table 2, we show the results for this measure. With respect to the *search space* Small, Auto-MEKA$_{GGP}$ presented the best average value and also the best average rank. Besides, we found statistical evidence that Auto-MEKA$_{GGP}$ has better results than Auto-MEKA$_{BO}$ in this *search space*. In the *search space* Medium, we observe that Auto-MEKA$_{GGP}$ produced the best average ranking and Auto-MEKA$_{GS}$ presented the best average value. Finally, in the comparison regarding the *search space* Large, Auto-MEKA$_{spGGP}$, Auto-MEKA$_{GGP}$ and Auto-MEKA$_{BO}$ achieved significantly better results than Auto-MEKA$_{RS}$, showing their capabilities to handle enormous *search spaces* – when giving enough time for them to proceed with their searches. Apart from the statistical results, we can observe in Table 2 that Auto-MEKA$_{GGP}$ also reached the best average value and rank within five hours of running. Furthermore, Auto-MEKA$_{BO}$ had even results to Auto-MEKA$_{GGP}$ in terms of the average value.

We can now provide overall conclusions based on the results of this section, specially on the fitness measure. We believe that the size of the *search spaces* explored/exploited by the *search methods* influenced the accuracy of the MLC predictions. We understand that, for smaller *search spaces* (i.e., Small and Medium), the *search methods* find it easier to proceed with their searches and, hence, their results become broadly similar among each other. When we increase the *search space* to Large, only those with robust search mechanisms could deal better with the trade-off between exploration and exploration, making them achieve competitive results.

However, we believe the robust methods (i.e., those based on the evolutionary and Bayesian optimization frameworks) can still improve their final predictive performances as they could not beat the greedy *search method*, a pure-exploitation method. They also face challenges to beat the random *search method* in some of the analyzed cases. Thus, these *search methods* still could not satisfactorily balance between exploration and exploitation. In our perspective, this would occur if they could beat simultaneously Auto-MEKA$_{RS}$ and Auto-MEKA$_{GS}$. For the *search spaces* Small and Medium, this result is more understandable. The smaller the *search space*, the easier it is to perform the search on it. This leads to better results for Auto-MEKA$_{RS}$ and Auto-MEKA$_{GS}$ in such a way that the other *search methods* were not statistically different from both of them. For bigger *search spaces* (i.e., Large), this should be the opposite. As they have robust search mechanisms, they should obtain statistically better results against pure-exploration and pure-exploitation *search methods*.

5.2 Analysis of the Diversity of the Selected Algorithms

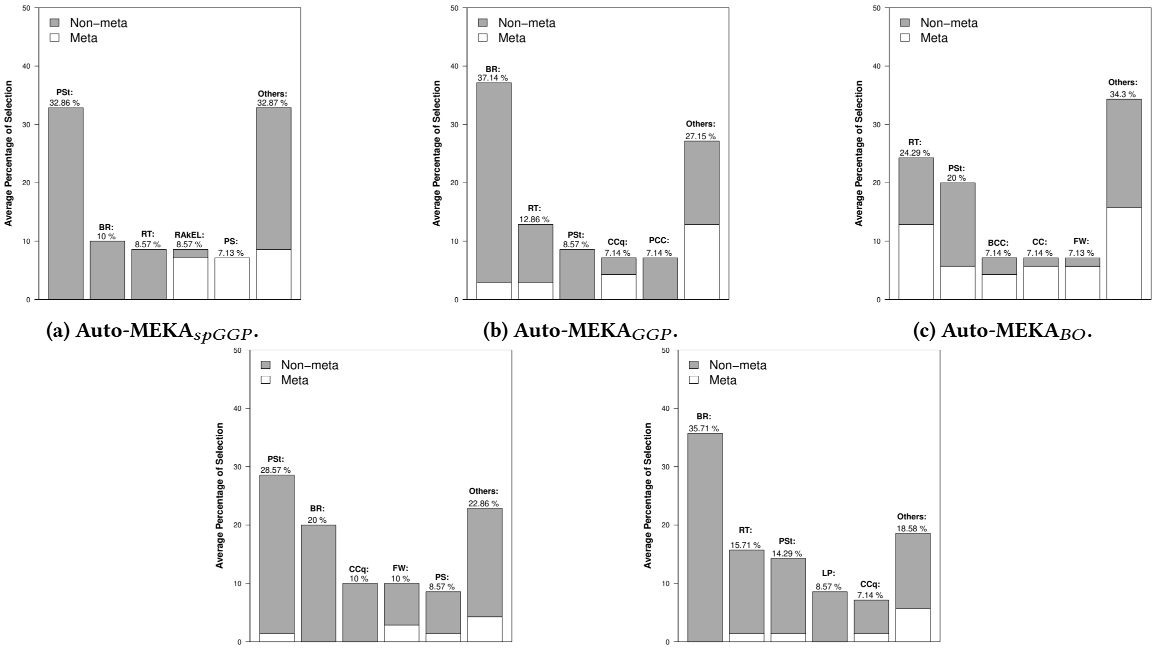
We also analyzed the diversity of the MLC algorithms and meta-algorithms selected by the five AutoMLC *search methods*. We focus only on the selected MLC algorithms and meta-algorithms (which are the "macro-components"), and not on their selected SLC algorithms and hyper-parameter settings (the "micro-components"), to simplify our analysis.

It is important to emphasize that, by analyzing the MLC and SLC algorithms and meta-algorithms selected by all *search methods* in each *search space*, we can better understand the results of Table 2. This would give an idea of how the choice of an MLC algorithm influences the performance of the *search methods*. However, for the sake of simplicity, we perform this analysis only for the *search space* Large.

We present in Figures 3b through 3e the bar plots to analyze the relative frequency of selection of MLC algorithms for the AutoMLC *search methods*. In these figures we have, for each MLC algorithm, a bar representing the average relative frequency of selection of an algorithm type over all the 70 runs: 14 datasets times five independent runs per dataset (five cross-validation folds times one run per fold). We consider two cases: (i) when the traditional MLC algorithm is solely selected; (ii) when the traditional MLC algorithm is selected together with a MLC meta-algorithm. To emphasize these two cases, the bar for each traditional MLC algorithm is divided into two parts, with sizes proportional to the relative frequency of selection as a standalone algorithm (in gray color) and the relative frequency of selection as part of a meta-algorithm (in white color).

Considering this information, BR, PSt and RT were the traditional MLC algorithms most frequently selected by all AutoMLC *search methods* in the *search space* Large. BR was chosen, on average, in 21.43% of all runs for all methods. PSt and RT, in turn, were selected on average in 20.66% and 13.43% of all runs for the five evaluated *search methods*, respectively. Nevertheless, some of these MLC algorithms were not so present in the selections performed by the *search methods*. For instance, BR and RT were not frequently chosen by Auto-MEKA$_{BO}$ and Auto-MEKA$_{RS}$. This partially shows the differences in the selection and configuration of the AutoMLC *search methods*, although most of them had similar algorithms at the top five regarding the ranking of selection.

We can also justify the performance of the methods based on their selection at the MLC meta level. For example, Auto-MEKA$_{GGP}$ achieved the best results for the *search space* Large in terms of the average value and rank based on fitness, which is the measure we use to decide (for all methods) what algorithm is the most appropriate. By looking at Auto-MEKA$_{GGP}$'s selection at the MLC meta level we can understand why this happened. Auto-MEKA$_{GGP}$ and



(a) Auto-MEKA$_{spGGP}$.

(b) Auto-MEKA$_{GGP}$.

(c) Auto-MEKA$_{BO}$.

(d) Auto-MEKA$_{RS}$.

(e) Auto-MEKA$_{GS}$.

Figure 3: Bar plots for the algorithms' selection frequencies at the MLC level over 70 runs.

Auto-MEKA$_{spGGP}$ have chosen these MLC meta-algorithms with a low relative frequency (22.86% for both methods). Therefore, the complexity of the final solution turned them into better options for AutoMLC in the MLC context when contrasted to Auto-MEKA$_{BO}$, which selected meta-algorithms in 50% of the cases. However, their level of selection of MLC meta-algorithms is still high when compared to Auto-MEKA$_{RS}$ and Auto-MEKA$_{GS}$, which selected MLC meta-algorithms in only 10% of the cases. This might be the reason for the competitiveness of Auto-MEKA$_{RS}$ and Auto-MEKA$_{GS}$.

One test that might be interesting is to remove the MLC meta-algorithms from the *search space*, and reexecute the *search methods*. This could show us whether or not the learned model is more likely to overfit on the training set when we select very complex combinations of base and meta-algorithms. We did that in the *search spaces* Small and Medium, but they do not include all traditional MLC algorithms as the *search space* Large does.

6 CONCLUSIONS

This paper presented an overall comparison among five AutoML *search methods* in the context of multi-label classification – i.e., Auto-MEKA$_{GGP}$, Auto-MEKA$_{spGGP}$, Auto-MEKA$_{BO}$, Auto-MEKA$_{RS}$ and Auto-MEKA$_{GS}$. To perform this assessment, the *search methods* were run in 14 MLC datasets with the same execution time budget (i.e., five hours) and in three designed *search spaces*.

The experimental results indicate that Auto-MEKA$_{GGP}$ is so far the best *search method* as it yields the best predictive results. Besides, it is the only method to be statistically better than Auto-MEKA$_{spGGP}$, Auto-MEKA$_{BO}$ and Auto-MEKA$_{RS}$ in different cases.

However, we expected that methods with robust search mechanisms (e.g., Auto-MEKA$_{spGGP}$, Auto-MEKA$_{BO}$ and Auto-MEKA$_{GGP}$) could balance better between exploration and exploitation. This limitation made these methods not being able to produce statistically better results than Auto-MEKA$_{RS}$ and Auto-MEKA$_{GS}$, which are pure-exploration and pure-exploitation methods, respectively.

We also observed that the size of the *search space* is a crucial issue for the AutoML methods' behavior. Thus, as a first future work, we intend to better understand the trade-off between parsimony and sufficiency [2]. In other words, we would like to investigate which algorithms should be included to or excluded from the *search spaces*, in order to keep good (combinations of) learning algorithms.

In addition, as two out of the five measures (i.e., FM and RL) yielded flat results, we need to understand how neutral the *search spaces* are in terms of the chosen MLC performance measures [15, 19]. This would help us to have insights to propose efficient methods or enhancements to these *search spaces*.

Furthermore, we expect to test other quality criteria to discover appropriate MLC algorithms configurations for a given dataset of interest. This may include finding other relevant performance measures or to set new weights for the current measures that compose the proposed fitness metric.

Finally, we also plan to include into the proposed *search methods* a bi-level optimization approach [23] to diminish the hardness of the search in huge *search spaces*. Fundamentally, this would mean to select the learning algorithms in the first place and only configure their hyper-parameters in a second step of the search procedure.

ACKNOWLEDGEMENTS

The authors would like to thank FAPEMIG (through the grant no. CEX-PPM-00098-17), MPMG (through the project Analytical Capabilities), CNPq (through the grant no. 310833/2019-1), CAPES, MCTIC/RNP (through the grant no. 51119) and H2020 (through the grant no. 777154) for their partial financial support.

REFERENCES

[1] Thomas Back, David B. Fogel, and Zbigniew Michalewicz (Eds.). 1999. *Evolutionary Computation 2: Advanced Algorithms and Operators* (1st ed.). IOP Publishing Ltd., Bristol, UK.

[2] Wolfgang Banzhaf, Frank D. Francone, Robert E. Keller, and Peter Nordin. 1998. *Genetic Programming: An Introduction.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[3] Lena Chekina, Lior Rokach, and Bracha Shapira. 2011. Meta-learning for Selecting a Multi-label Classification Algorithm. In *Proceedings of the International Conference on Data Mining Workshops (ICDMW'11).* IEEE, New York, NY, USA, 220–227.

[4] Alex G. C. de Sá, Alex A. Freitas, and Gisele L. Pappa. 2018. Automated Selection and Configuration of Multi-Label Classification Algorithms with Grammar-based Genetic Programming. In *Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN'18).* Springer, Cham, Switzerland, 308–320.

[5] Alex G. C. de Sá, Gisele L. Pappa, and Alex A. Freitas. 2017. Towards a Method for Automatically Selecting and Configuring Multi-Label Classification Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO'17).* ACM, New York, NY, USA, 1125–1132.

[6] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine learning Research* 7, 1 (2006), 1–30.

[7] Pedro Domingos. 2012. A Few Useful Things to Know About Machine Learning. *Commun. ACM* 55, 10 (2012), 78–87.

[8] Agoston E. Eiben and James E. Smith. 2003. *Introduction to Evolutionary Computing.* Vol. 53. Springer, Cham, Switzerland.

[9] Radwa Elshawi, Mohamed Maher, and Sherif Sakr. 2019. Automated Machine Learning: State-of-the-Art and Open Challenges. arXiv preprint, arXiv1906.02287.

[10] Eva Gibaja and Sebastián Ventura. 2015. A Tutorial on Multilabel Learning. *Comput. Surveys* 47, 3 (2015), 52:1–52:38.

[11] Francisco Herrera, Francisco Charte, Antonio J. Rivera, and María J. Del Jesus. 2016. *Multilabel Classification: Problem Analysis, Metrics and Techniques* (1st ed.). Springer, Cham, Switzerland.

[12] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential Model-based Optimization for General Algorithm Configuration. In *Proceedings of the International Conference on Learning and Intelligent Optimization (LION'11).* Springer-Verlag, Berlin/Heidelberg, Germany, 507–523.

[13] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren (Eds.). 2019. *Automated Machine Learning: Methods, Systems, Challenges.* Springer, New York, NY, USA. Available at http://automl.org/book.

[14] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. 2012. An Extensive Experimental Comparison of Methods for Multi-Label Learning. *Pattern Recognition* 45, 9 (2012), 3084–3104.

[15] Katherine M. Malan and Andries P. Engelbrecht. 2013. A Survey of Techniques for Characterising Fitness Landscapes and Some Possible Ways Forward. *Information Sciences* 241 (2013), 148–163.

[16] Robert McKay, Nguyen Hoai, Peter Whigham, Yin Shan, and Michael O'Neill. 2010. Grammar-based Genetic Programming: A Survey. *Genetic Programming and Evolvable Machines* 11, 3 (2010), 365–396.

[17] Jose M. Moyano, Eva L. Gibaja, Krzysztof J. Cios, and Sebastián Ventura. 2019. An Evolutionary Approach to Build Ensembles of Multi-Label Classifiers. *Information Fusion* 50 (2019), 168–180.

[18] Rafael B. Pereira, Alexandre Plastino, Bianca Zadrozny, and Luiz H. C. Merschmann. 2018. Correlation analysis of performance measures for multi-label classification. *Information Processing and Management* 54, 3 (2018), 359–369.

[19] Erik Pitzer and Michael Affenzeller. 2012. A Comprehensive Survey on Fitness Landscape Analysis. In *Recent advances in intelligent engineering systems.* Springer-Verlag, Berlin/Heidelberg, Germany, 161–191.

[20] Jesse Read, Peter Reutemann, Bernhard Pfahringer, and Geoff Holmes. 2016. MEKA: A Multi-label/Multi-target Extension to WEKA. *Journal of Machine Learning Research* 17, 21 (2016), 1–5.

[21] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2011. On the Stratification of Multi-Label Data. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD'11).* Springer-Verlag, Berlin/Heidelberg, Germany, 145–158.

[22] Eric Siegel. 2013. *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die* (1st ed.). Wiley Publishing, Hoboken, New Jersey, USA.

[23] El-Ghazali Talbi. 2013. A Taxonomy of Metaheuristics for Bi-Level Optimization. In *Metaheuristics for Bi-Level Optimization.* Springer-Verlag, Berlin/Heidelberg, Germany, 1–39.

[24] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13).* ACM, New York, NY, USA, 847–855.

[25] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. *Mining Multi-label Data.* Springer, Boston, MA, USA, 667–685.

[26] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2011. Random k-Labelsets for Multilabel Classification. *IEEE Transactions on Knowledge and Data Engineering* 23, 7 (2011), 1079–1089.

[27] Marcel Wever, Felix Mohr, Alexander Tornede, and Eyke Hüllermeier. 2019. Automating Multi-Label Classification Extending ML-Plan. In *Proceedings of the ICML Workshop on Automated Machine Learning (AutoML'19).* AutoML.org, Freiburg, Germany, 1–8.

[28] Peter A. Whigham. 1995. Grammatically-based Genetic Programming. In *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications.* NRL TR 95.2, University of Rochester, Rochester, New York, USA, 33–41.

[29] Mohammed J. Zaki and Wagner Meira Jr. 2020. *Data Mining and Analysis: Fundamental Concepts and Algorithms* (2nd ed.). Cambridge University Press, Cambridge, UK.

[30] Min-Ling Zhang and Zhi-Hua Zhou. 2014. A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2014), 1819–1837.