Expert Systems With Applications 203 (2022) 117215

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Comprehensive comparative study of multi-label classification methods

Jasmin Bogatinovski [a,b,c], Ljupčo Todorovski [a,d], Sašo Džeroski [a,b], Dragi Kocev [a,b,*]

[a] Department of Knowledge Technologies, Jozef Stefan Institute, Ljubljana, Slovenia
[b] JSI International Postgraduate School, Ljubljana, Slovenia
[c] Department of Distributed Operating Systems, TU Berlin, Berlin, Germany
[d] Faculty of Mathematics and Physics, University of Ljubljana, Ljubljana, Slovenia

ARTICLE INFO

Keywords:
Multi-label classification
Benchmarking machine learning methods
Performance estimation
Evaluation measures

ABSTRACT

Multi-label classification (MLC) has recently attracted increasing interest in the machine learning community. Several studies provide surveys of methods and datasets for MLC, and a few provide empirical comparisons of MLC methods. However, they are limited in the number of methods and datasets considered. This paper provides a comprehensive empirical investigation of a wide range of MLC methods on a wealth of datasets from different domains. More specifically, our study evaluates 26 methods on 42 benchmark datasets using 20 evaluation measures. The evaluation methodology used meets the highest literature standards for designing and conducting large-scale, time-limited experimental studies. First, the methods were selected based on their use in the community to ensure a balanced representation of methods across the MLC taxonomy of methods within the study. Second, the datasets cover a wide range of complexity and application domains. The selected evaluation measures assess the predictive performance and efficiency of the methods. The results of the analysis identify RFPCT, RFDTBR, ECCJ48, EBRJ48, and AdaBoost.MH as the best-performing methods across the spectrum of performance measures. Whenever a new method is introduced, it should be compared with different subsets of MLC methods selected according to relevant (and possibly different) evaluation criteria.

1. Introduction

Predictive modelling is an area in machine learning concerned with developing methods that learn models for predicting the value of a target variable. The target variable is typically a single continuous or discrete variable, corresponding to the two common tasks of regression and classification, respectively. However, in practically relevant problems, more and more often, there are multiple properties of interest, i.e., several target variables. Such practical problems include image annotation with multiple labels (e.g., an image can depict trees and at the same time the sky, grass etc.), predicting gene functions (each gene is typically associated with multiple functions) and drug effects (each drug can affect multiple conditions). The problems with multiple binary variables as targets corresponding to the question if a given example is associated with a subset from a set of predefined labels belong to the widely known task of multi-label classification (MLC) (Herrera et al., 2016; Madjarov et al., 2012; Tsoumakas & Katakis, 2007).

1.1. Practical relevance of MLC

In binary classification, the presence/absence of a single label is predicted. In MLC, the presence/absence of multiple labels is predicted and multiple labels can be assigned simultaneously to a sample. Most often, the MLC task is confused with multi-class classification (MCC). In MCC, there are also multiple classes (labels) that a given example can belong to, but a given example can belong to only one of these multiple classes. In that spirit, the MCC task can be seen as a special case of the MLC task, where exactly one label is relevant for each example. Furthermore, the MLC task is different from the task of multi-target classification (MTC) (Kocev et al., 2013), which is concerned with predicting several targets, each of which can take only one value of several possible classes. MLC can be viewed as a collection of several binary classification tasks, and MTC of several MCC tasks. Finally, another task related to MLC is multi-label ranking. The goal of multi-label ranking is to produce a ranking/ordering of the labels regarding their relevance to a given example (Madjarov et al., 2012).

MLC predicts the set of target attributes (called *labels*) that are relevant for each presented sample. This task arises from practical applications. For example, Xu et al. (2016) introduce three instantiations of the task of predicting the subcellular locations of proteins according to their sequences. The dataset contains protein sequences for humans, viruses and plants. Both GO (Gene Ontology) terms and pseudo amino

* Corresponding author at: Department of Knowledge Technologies, Jozef Stefan Institute, Ljubljana, Slovenia.
E-mail addresses: jasmin.bogatinovski@tu-berlin.de (J. Bogatinovski), ljupco.todorovski@fmf.uni-lj.si (L. Todorovski), saso.dzeroski@ijs.si (S. Džeroski), dragi.kocev@ijs.si (D. Kocev).

https://doi.org/10.1016/j.eswa.2022.117215
Received 2 February 2022; Received in revised form 8 April 2022; Accepted 8 April 2022
Available online 21 April 2022
0957-4174/© 2022 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

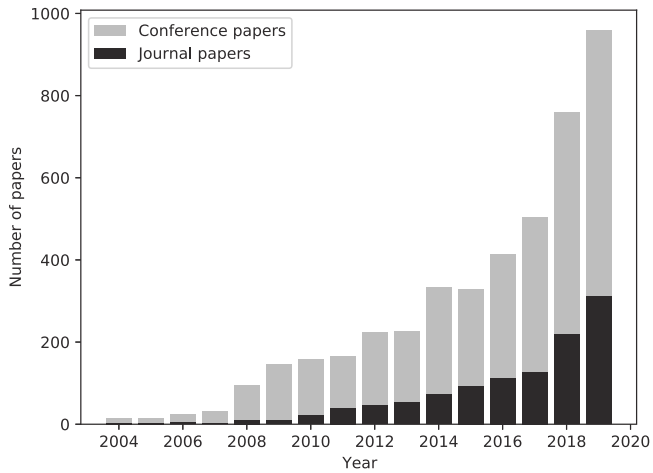


Fig. 1. A summary of the number of papers from the SCOPUS database (https://www.scopus.com/) related to the topic of MLC. The vertical axis represents the number of conference and journal papers related to the topic of MLC. An almost exponential curve of progress can be observed. The absence of large experimental studies with rigorous extensive experimental empirical comparison amplifies the importance of performing a comprehensive study on MLC methods to provide a survey of the landscape of methods.

acid compositions are used to describe the protein sequences. The goal is to predict the relevant sub-cellular locations for each of the proteins. There is a total of six subcellular locations for viruses, 12 subcellular locations for plants, and 14 subcellular locations for humans. Briggs et al. (2012) address the prediction of the type of birds whose songs are present in a given recording using audio signal processing. The set of labels consists of 19 species of birds. Several birds can simultaneously be recorded on a given recording. The text document topic classification is also a MLC problem, as many of the documents refer to more than one topic at the same time. For example, Katakis et al. (2008) present a dataset from BibSonomy entries, annotated with several tags. Next, Boutell et al. (2004) present an image dataset containing images annotated with several labels (beach, sunset, fall foliage, field, urban and mountain) with multiple labels present in a single sample simultaneously. Although the main focus of MLC applications is in text, biology and multimedia, the potential for using MLC in other domains is constantly increasing (medicine Grady & Funka-Lea, 2004; Ratnarajah & Qiu, 2014, environmental modeling Blockeel et al., 1999 social sciences Schulz et al., 2016, commerce Wang et al., 2020 etc.). Liu et al. (2020) give an extensive summary of the various emerging trends and subareas of MLC: extreme multi-label classification, multi-label learning with limited supervision, deep multi-label learning, online multi-label learning, statistical multi-label learning, and rule-based multi-label learning.

1.2. Motivation and related work

Fig. 1 shows the increasing interest in the task of MLC from the machine learning community. The increasing trend indicates the appearance of novel MLC problems and methods. Given the large pool of problems, multi-label methods and datasets, it is not easy for a novice and even an experienced practitioner to select the most suitable method for their problem. Moreover, it is not clear what benchmarking baselines should be used when proposing a novel method. Therefore, landscaping the existing methods and problems is a necessity for the further advancement of this research area.

There are several previous attempts at addressing this issue. However, they have a limited scope concerning the methods and/or the datasets used in the evaluation. Some of these studies require special emphasis because they have helped shape the field by providing a theoretical and empirical discussion on the properties of the various MLC methods. We discuss these studies in chronological order of their literature appearance.

Madjarov et al. (2012) provide the first comprehensive empirical study for the task of MLC. They give a comprehensive analysis of 12 MLC methods on 11 benchmarking problems and 16 evaluation criteria. As a systematic review, its conclusion can guide the practitioners tackling MLC tasks, about relevant method selection. However, from the current perspective, given the wealth of newly proposed methods and problems/datasets, it is outdated in terms of the inclusion of problems and methods that have been introduced in the last decade.

The second study provides details on the MLC tasks (Gibaja & Ventura, 2015). It introduces a concise organization of the methods, evaluation criteria, MLC specific data preprocessing techniques and the different MLC problems. However, it lacks a comprehensive empirical evaluation of the methods across different datasets. The third study (Zhang & Zhou, 2014) provides an in-depth theoretical treatise of eight MLC methods, together with their pseudo-codes and a discussion of how the methods deal with the specifics of the MLC task. The drawbacks of the previous three studies are addressed (to some extent) in Herrera et al. (2016). As a book on MLC, it gives an extensive overview of existing methods through an experimental comparison. However, it lacks comparative experimental rigour as some of the previous works, e.g., Madjarov et al. (2012).

Furthermore, the most recent study (Moyano et al., 2018) provides a similar experimental setup as in Madjarov et al. (2012) with an extension towards the analysis of ensembles of MLC methods. It argues that ensemble learning methods are superior in terms of performance to other, single-model learning approaches. While this is true in many cases, the computational time one requires for building an ensemble is larger than the time for building a single model. Given the complexity of the MLC task, this may arise as a limitation in practical applications. Zhang et al. (2018) focus on providing an overview of a specific type of MLC methods, referred to as binary relevance, but do not assess their predictive performance. In a similar limited context, Rivolli et al. (2020) present an empirical study of seven different base learners used in ensembles on 20 datasets.

1.3. Objectives

A shared property of the previous studies is the focus on a smaller part of the landscape of methods and problems. However, given the plethora of problems and methods introduced in recent years, a comparative analysis on a larger scale is highly desired. **The main aim of this work is to fill this gap by performing an extensive study of MLC in terms of both methods and problems/datasets**. Simultaneously, it aims to identify the strengths and limitations of existing methods beyond predictive performance and efficiency by studying how the different methods deal with the specific MLC task challenges across the vast set of problems.

1.4. Contributions

By performing the extensive experimental study, a landscape map of the MLC will be obtained: It will **include the performance of 26 MLC methods evaluated on 42 benchmark datasets using 18 performance evaluation measures**. Next, it will reveal the best-performing methods per method group and evaluation measure. Hence, it will identify the most suitable baselines that need to be used when proposing a novel MLC method. Furthermore, it will outline the strengths and weaknesses of the MLC methods concerning one another. Moreover, it will highlight the used MLC methods in terms of their potential for addressing several MLC specific properties (e.g., label dependencies and high-dimensional label spaces).

This study is the most comprehensive experimental work for the task of MLC performed thus far. In a nutshell, it identifies a subset of five methods that should be used in baseline comparisons:

RFPCT (Random Forest of Predictive Clustering Trees) (Kocev et al., 2013), RFDTBR (Binary Relevance with Random Forest of Decision Trees) (Tsoumakas & Katakis, 2007), ECCJ48 (Ensemble of Classifier Chains built with J48) (Read, 2010), EBRJ48 (Ensemble of Binary Relevance built with J48) (Read, 2010) and AdaBoost (Schapire & Singer, 1999). These methods show the best predictive performance on average across the different problems. The first two methods are computationally much more efficient compared to the others. Detailed results from this study, alongside the descriptions of the methods, datasets, and evaluation measures are available in the study repository accessible at http://mlc.ijs.si/.

1.5. Organization of the paper

The remainder of the paper is organized as follows. In Section 2, we formally define the task of MLC task and review/ describe the available MLC datasets and problems. Section 3 organizes the MLC methods into a taxonomy of methods, describes the methods in detail and discusses how these methods address specific properties of MLC, such as handling label-dependence and high-dimensional label spaces. Section 4 outlines the design of the experimental study by describing the experimental methodology and setup, the hyper-parameter instantiations, the evaluation measures, and the statistical analysis of the obtained results. Section 5 discusses the statistical analysis results from different viewpoints. Finally, Section 6 concludes and presents the main outcomes of the study, together with further guidelines for benchmarking MLC methods.

2. The task of multi-label classification

This section begins by describing the benchmark datasets considered in the study. Next, it defines the task of MLC. Finally, it overviews the methods considered in the study.

2.1. Multi-label classification datasets

Most of the datasets considered in this study come from application areas such as biology, text and multimedia. The datasets from biology, in general, include proteins representation as descriptive variables and as targets either gene function prediction or sub-cellular localization. The datasets from the text domain most often represent the problem of topic classification for news documents. However, other textual datasets predict targets such as cardiovascular condition states from medical reports, or recommendation tags from reviews. The datasets from the multimedia domain can be split into two major categories: datasets concerned with the classification of images (most often scenes in a given image) and datasets concerned with the classification of audio content (e.g., genre, emotions).

Some datasets come from other domains such as medicine and chemistry. The datasets from the medical domain represent the classification of diseases based on symptoms or state of the patient based on vital measurements such as blood pressure. The dataset from the domain in chemistry is concerned with the classification of chemical concentration in observed subjects. The diversity of the available MLC datasets witnesses the vast application potential of the MLC task. The detailed statistics of the datasets used in this study can be found in Table A.1 in Appendix A, available online.

The MLC datasets are described with five basic meta-features (i.e., features describing the dataset properties): number of instances/ examples, number of features, number of labels, label cardinality and label density. The distribution of the datasets across these meta-features is depicted in Fig. 2. The *number of training examples* ranges from 174 to 17190. The wide range of the sample number enables testing the strengths and weaknesses of the MLC methods from the perspective of input data richness measured by the instance number.

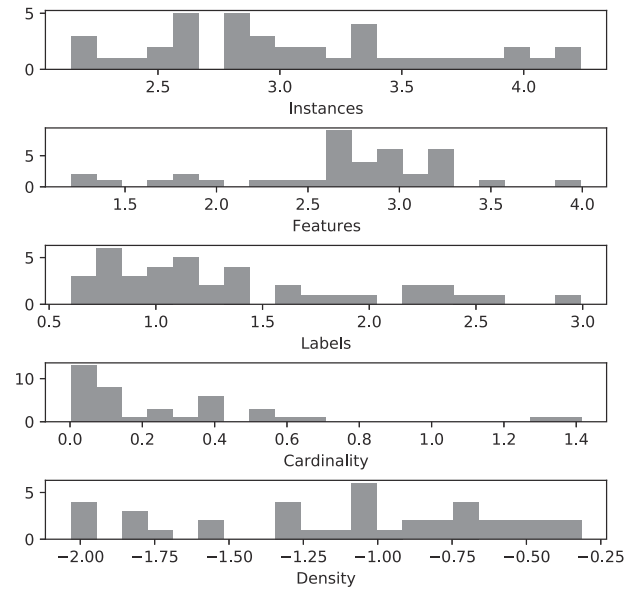


Fig. 2. Distribution of the values of five dataset features on the 42 datasets. The feature values are given on a log scale.

Additionally, the richness in the sample description (*number of features*) ranges from 19 to 9844. Predominantly the number of features range from 300 until 2000. Regarding the type of features, there are few datasets with a mixture of nominal and numeric attributes. In most of the datasets, the features are either solely numeric or solely nominal. The *number of labels* is a unique property of the MLC classification task. The number of labels ranges from 4 to 374, with one dataset being an exception, containing 983 labels. Predominantly the number of labels is in the range from 4 until 53.

Additional discussion about the datasets follows in terms of the meta properties of *label cardinality* and *label density*. Label cardinality is a measure of labels distribution per example. It is defined as the mean number of labels associated with a sample (Tsoumakas & Katakis, 2007). In many of the datasets, this meta-feature is smaller than 1.5. It indicates that these datasets on average have one label associated with their examples. In most cases, there are no more than three labels assigned to an example in a dataset. As exceptions are the datasets `delicious` and `cal500`, which have around 19 and 26 labels assigned for each example on average, respectively. Label density is a measure of the frequency of the labels. It is calculated as the division of the label cardinality by the number of labels. It indicates the frequency of the labels among all the instances. On the figure it is given in log scale.

2.2. Task description

The task of MLC can be viewed as an instantiation of the structure output prediction paradigm (Kocev, 2011; Kocev et al., 2013). The goal is for each example to define two sets of labels — the set of relevant and the set of irrelevant labels. Following Madjarov et al. (2012), the task of MLC is defined as:

Given:

- an example space $\mathcal{X}$ consisting of tuples of values of primitive data types (categorical or numeric), i.e., $\forall \mathbf{x_i} \in \mathcal{X}, \mathbf{x_i} = (x_{i_1}, x_{i_2}, \ldots, x_{i_D})$, where D denotes the number of descriptive attributes,
- a label space $\mathcal{L} = \{\lambda_1, \lambda_2, \ldots, \lambda_Q\}$ which is a set of Q possible labels,
- a set of examples E, where each element is a pair of a tuple from the example space and a subset of the label space, i.e., $E = \{(\mathbf{x_i}, \mathcal{Y}_i) | \mathbf{x_i} \in \mathcal{X}, \mathcal{Y}_i \subseteq \mathcal{L}, 1 \leq i \leq N\}$ and N is the number of examples of E ($N = |E|$), and

- a quality criterion q, which rewards models with high predictive performance and low complexity.

Find: a function h: $\mathcal{X} \rightarrow 2^{\mathcal{L}}$ such that h maximizes q.

2.3. Taxonomy of multi-label classification methods

There are a plethora of MLC methods presented in the literature. In this paper, we follow the taxonomy of the methods as proposed in Tsoumakas and Katakis (2007). The MLC methods are separated into two categories problem transformation and algorithm adaptation. The group of problem transformation methods approaches the problem of MLC by transforming the multi-label dataset into one or multiple datasets. These datasets are then approached with single-target machine learning methods and build one or multiple single-target models. At prediction time, it is required that all models are invoked to predict for the test sample.

Algorithm adaptation methods include some adaptation of the training and prediction phases of the single target methods towards handling multiple labels simultaneously. For example, trees change the heuristic used when creating the splits, neural networks directly handle the MLC task, while Support Vector Machines (SVMs) employ additional threshold techniques. The adaptations aim to provide a mechanism to directly handle the dependency between the labels. Their grouping is based on the type of underlying adapted paradigm. The literature recognizes five defined groups of algorithm adaptation methods according to the performed adaptation: trees, neural networks, support vector machines, instance-based and probabilistic (Herrera et al., 2016). There are additional methods that utilize various approaches from other domains, e.g., genetic programming. However, they lack shared unifying ground and are characterized as unspecified method groups. For more details, one can refer to Herrera et al. (2016).

3. Methods for multi-label classification

In this section, we discuss the MLC methods used in the experimental evaluation. We first describe the problem transformation methods. Next, we describe the algorithm adaptation methods and their ensemble variants. Finally, we provide a discussion on how each of the methods is addressing two properties of the MLC task: label dependencies and high dimensional label spaces (including computational complexity method analysis of the methods).

3.1. Problem transformation methods

There are two main ideas in the problem transformation methods — decomposition of the problem of a set of binary problems or a (set of) multi-class problem(s). Fig. 3 depicts the used problem transformation methods and their organization into a taxonomy of methods.

The first idea observes a multi-label dataset as a composition of multiple single-target datasets sharing the same feature space. Such an approach has the benefit of providing a straightforward application of single-target binary methods. At prediction time all trained single target models are invoked to produce the result for the new test sample. This approach, however, loses information about the dependency between the labels. Regarding the process of creating the multiple single binary target datasets, this group of methods is further grouped into `One-Vs-One`-like and `One-Vs-All`-like methods. In the former approach (also known as binary relevance or pairwise approach (Gibaja & Ventura, 2015)), each pair of labels is considered producing a quadratic number of single-target binary datasets. In the latter approach, the problem is transformed directly to $|\mathcal{L}|$ single target multi-class problems by using the unique label sets as a separate class. This approach is also known as label powerset.

Both approaches allow for the use of simpler classifiers: the binary relevance uses binary classifiers, while the label powerset uses multi-class classifiers. The advantages of the latter over the former are that a single model is learned (compared to a quadratic number of models for the first) and the label dependencies are preserved (compared to the complete obliteration of this information in the binary relevance approach). However, a strong limitation of the label powerset methods is the inability to generalize beyond the label-sets present in the training dataset. The shortcomings of these two approaches are addressed to some extent by using them in the context of ensemble learning. We next discuss these methods in more detail.

3.1.1. Binary relevance methods

The Binary Relevance method **(BR)** (Tsoumakas & Katakis, 2007) transforms the MLC problem into $|\mathcal{L}|$ binary classification problems that share the same feature (descriptive) space as the original descriptive space of the multi-label problem. Each of the binary problems has assigned one of the labels as a target. It trains one base binary classifier for each of the transformed problems. It has only one hyperparameter — the base classifier. This method generalizes beyond the label-sets present in the training samples. It is not suitable for a large number of labels and ignores the label correlations. Due to the necessity of building models for each label, the training of the method can be time-consuming, especially if the computational complexity of the base learning method is large.

Calibrated Label Ranking **(CLR)** is a pairwise technique for multi-label ranking. It provides a built-in mechanism to extract bipartitions and thus can be used as a MLC method. The core of the pairwise methods is creating $\frac{|\mathcal{L}|(|\mathcal{L}|-1)}{2}$ single target binary datasets from the multi-label dataset with label-set of size $|\mathcal{L}|$, maintaining the original descriptive space. The binary target is generated in such a way that if one of the labels in a given pair, chosen as positive, is different from the other, the example in the newly created dataset obtains a value of 1 and 0 otherwise. If the labels are the same, the example is excluded. In such way $\frac{|\mathcal{L}|(|\mathcal{L}|-1)}{2}$ binary datasets are created. A base classifier is built on these datasets. CLR introduces one artificial label (Brinker, 2006; J. et al., 2008). This artificial label acts as a complementary label for each of the original labels, thus introducing $|\mathcal{L}|$ more models to be built. When ranking for each of the labels is obtained, the artificial variable acts as a split point between the relevant and irrelevant labels, producing bipartition. It has one hyperparameter to be chosen — the base learner. A strong advantage of this method is that it generates both ranking and bipartition. The main drawback is that it is not so suitable for datasets with a large number of labels, due to the large exploration space and time complexity.

Classifier Chains **(CC)** (Read et al., 2011) learning procedure involves two steps. It consists of training $|\mathcal{L}|$ single target binary classifiers as in BR connected in a chain. Each classifier deals with a single target problem of augmented feature space consisting of all the descriptive features and the predictions obtained from the previous classifier in the chain (the first classifier on the chain is learned only using the descriptive features). The only hyperparameter to set is the base classifier. A strength of this method is the introduction of label correlation to some extent (the order of the labels in the chain is important) and can generalize beyond seen label-sets. The two limitations of the method include its prohibitive usage for datasets with many labels (it is applying BR), and the dependence on the ordering of the labels along the chain.

3.1.2. Label powerset methods

Label Powerset **(LP or LC)** (Tsoumakas & Katakis, 2007) transforms the MLC method into a multi-class classification problem in such a way that it treats each unique label-set as a separate class. Any classifier suitable for solving a multi-class classifier can be applied to solve the newly created single target multi-class problem. It has only one hyperparameter, the base multi-class classifier. An advantage of the method is that preserves the label relationships. The limitations of this method are that it cannot predict novel label combinations and is prone to underfitting when the number of unique label sets is large.

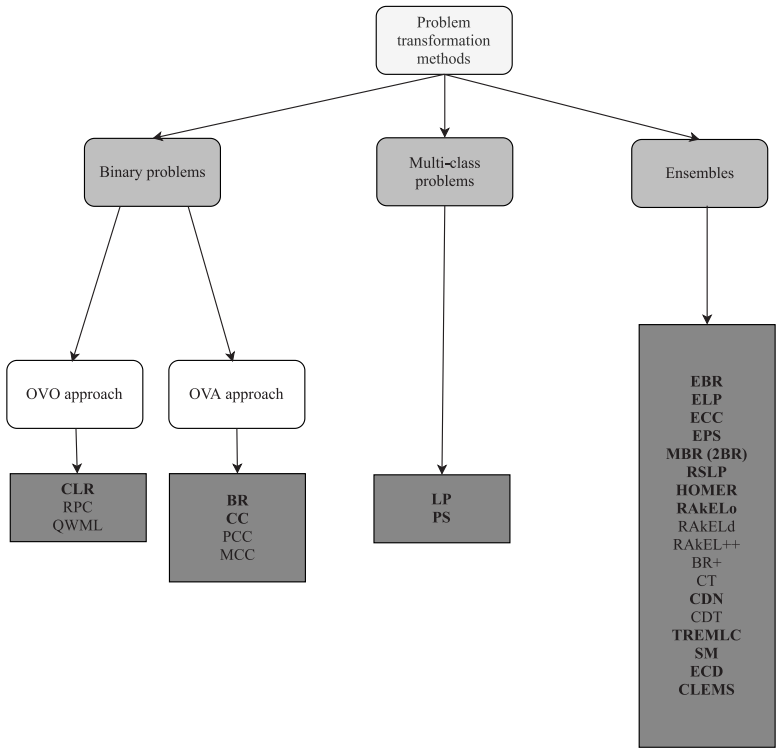


Fig. 3. Problem transformation methods. The methods used in this study are shown in bold.

The method of Pruned Sets **(PSt)** (Read et al., 2008) aims at reducing the number of unique classes (label-sets) appearing when a multi-label problem is approached with LP. To achieve this goal, the method has two phases. The first step is the so-called pruning step. The pruning step removes the infrequently occurring label sets from the training data. The decision on what means infrequent label-set is a hyperparameter of the method. The second phase consists of introducing the removed examples into the training set. It is done by subsampling the label sets of the infrequent samples for label subsets that satisfy the pruning criterion. The method introduces this as a tuning hyperparameter, and it defines the maximal number of frequent label sets to subsample from the infrequent ones. On such a newly created dataset, LP is trained. Additional improvement of the method is the introduction of a threshold function that enables new label combinations to be created at prediction time (Read, 2010). In total, there are three hyperparameters for the method: the base multi-class classifier, the pruning value (if the count of the label-sets in the datasets exceeds this number, the example is preserved), and the maximal number of frequent label-sets to be reintroduced. An advantage of this method is its efficiency — it is much faster than Label Powerset. A limitation of the method is that the assumptions can break, and the method (without threshold parameter) is unable to introduce novel label sets.

3.1.3. Ensembles of problem transformation methods

This category of problem transformation methods groups all MLC methods that utilize ensemble-like techniques such as stacking, bagging, random sub-spacing or employ different transformations of the datasets, such as embeddings.

Conditional Dependency Network **(CDN)** (Guo & Gu, 2011) aims at encapsulating the dependencies between the labels using dependency networks. Dependency networks are cyclic directed graphical models, where the parents of each variable are its Markov blanket (Heckerman et al., 2001). Markov blanket in the graphical model literature represents a set of nodes around a specific node that shield it. The Markov blanket of a node is the only knowledge needed to predict the behaviour of that node and its children (Pearl, 1988). The label dependency information is encoded into the graphical model parameters — the conditional probabilistic distribution associated with each label. The probabilistic distributions are modelled via simple binary classifier models, that as input take the whole feature space augmented by all other labels, with the exclusion of the label being modeled. In the inference phase, it uses the standard model for inference in graphical models — the Gibbs sampling method. This method assumes that one of the labels can change, assuming that all others are fixed. First, random ordering of the labels is chosen and each label is initialized to some value. In each sampling iteration, all the nodes modeling the labels are visited and the new value of the label being modelled is re-sampled according to the probability model that represents the current label being predicted. It has three tunable hyperparameters, i.e., the model trained at each node of the network, the number of iterations to perform until achieving stationarity of the chain the network and the *burnin* number of operations. This model preserves the label dependencies, however, if there are many labels the inference performed by the Gibbs sampling needs a longer convergence time, and stationarity may not be achieved.

Meta Binary Relevance **(MBR)** (Tsoumakas et al., 2009), also known as the 2BR method, consists of two consecutive stages of applying BR. First, $|\mathcal{L}|$ binary base models are built. At the second (meta) stage, the feature space is augmented with the predictions from the first stage ($|\mathcal{L}|$ features are added). New $|\mathcal{L}|$ binary models are trained as in BR. There exist a few approaches to generate the predictions in the first stage. The predictions can be generated using the full training set, via k fold cross-validation or by ignoring the irrelevant variables into the meta-level. The cross-validation approach is slow since it requires training of each model at the first level k times, but it passes non-biased information to the meta-level. The irrelevant information can be filtered using the Φ correlation coefficient to determine if two labels are correlated or not. If they are not correlated, the label is not introduced into the meta-level. Moyano et al. (2018) show that the version of this method where the full training set is used at the first stage is better regarding the other two. To further reduce the bias towards the label being predicted, Alvares-Cherman et al. (2012) suggest reducing the number of meta-labels to $|L|-1$ (the label being predicted is excluded).

This method is known as **BR+**. MBR has one hyperparameter to tune, the single target base method. A limitation is that MBR and its variants inherit the drawbacks of BR, i.e., it is not suitable if there are many labels.

Ensemble of Classifier Chains **(ECC)** (Read et al., 2011) creates an ensemble of CC built on sampled instances from the original dataset. The sampling is done with replacement. In Read et al. (2011) is argued that sampling with replacement provides better results compared with sampling without replacement. Choosing the percentage of the data for building the models (bag size) is allowed. So the hyperparameters of the method are the number of CC models in the ensemble and the bag size. In this method also a random ordering of the chain is considered to provide compensation for introducing non-existence dependency between labels. Using the different random subspaces of the training set and utilizing different ordering in the chain introduce diversity in the ensemble. This method takes into account the label correlation but has a drawback of the large time for training.

Ensemble of Binary Relevance **(EBR)** (Read et al., 2011) build ensemble of BR as base learner. The sampling is done with replacement. The hyperparameter of the method is the number of BR models in the ensemble. Although it can provide novel label-sets at prediction, it still has the assumption of labels independence. It can be treated as a binary relevance approach with bagging as a meta learner.

Chi-dep (Tenenboim et al., 2009) is a multi-label method that is based on the identification of label dependencies using statistical tests between the labels. χ^2 statistical test for independence for each possible combination of two labels is used. It first tries to identify groups of dependent and independent labels. After their identification, the BR approach for the independent groups, and LP for the dependent labels are trained. At prediction time, the sample is processed by each of the models and the prediction is generated accordingly. This method provides a trade-off between the assumption of independence of the labels of the BR method, and the problem of a large number of unique label sets the LP method is facing. The hyperparameters of the method are the base learners for the BR and LP method and the selection of a confidence level for the test. The positive aspect of the method is that it provides a trade-off between the high bias and variance of the BR method, and the low bias and high variance of the LP method.

Ensemble of Chi-dep **(ECD)** (Tenenboim et al., 2010) builds several Chi-dep models. First, it generates a large number of possible label-set partitions at random. Each of the partitions is represented by the normalized χ^2 score of all the label pairs inside the partition, based on the inside pairwise χ^2 scores. Then, the top m distinct sets with the highest scores are included in the ensemble. The hyperparameters of the method are the number of ensemble members and the number of partitions to evaluate. The positive aspects of the method are that it can further reduce the variance of a single Chi-dep method, however, it suffers from large time complexity. Thus a fast base learning method is recommended.

Ensemble of Label Powersets **(ELP)** (Moyano et al., 2018) create an ensemble of LP method on sampled prototypes from the original set. The sampling is done with replacement. The hyperparameters of the method include the number of LP models built in the ensemble and the type of base models. It provides an opportunity to enable LP to predict unseen label combinations (through the ensemble voting), however, it inherits its large computational complexity, and it is practically inefficient for datasets with a large number of unique label sets.

Ensemble of Pruned Sets **(EPS)** (Read et al., 2008) creates an ensemble from the PSt method on sampled prototypes from the original set. The sampling is done without replacement with a specific percentage of the dataset being sampled. Additional parameters of the method are the number of members of the ensemble as well as the number of examples to be sampled from the training size (bag size). This method can predict novel label sets, thus diminishing one of the disadvantages of a standalone PSt method without thresholding. Its disadvantage is that it is not able to perform well when there are many diverse label-sets without frequent reoccurring of some of the label-sets. This is due to reducing the training set to a handful of training examples due to the pruning strategy.

Random k Labelsets **(RAkEL)** (Tsoumakas, Katakis, & Vlahavas, 2011) is an MLC ensemble. It uses multiple LP models trained on random partitions of the label space. Usually, the size of the label set is small. Each of the LP methods should learn 2^k classes instead of $2^{|\mathcal{L}|}$, where $k \ll |\mathcal{L}|$. Moreover, the resulting multi-class problems have a much better-balanced distribution of the classes. In Tsoumakas, Katakis, and Vlahavas (2011), two versions of the method are introduced. The first version does not allow for overlap between the groups when creating the label sets and is called RAkEL disjoint. The second version allows for overlapping between the labels in the created label sets. This gives the advantage for the same label to be included by the different LP models. The predictions are obtained by voting. Further improvements of the method are proposed in Rokach et al. (2014). They propose using the classification confidence intervals instead of voting. However, Moyano et al. (2018) show that voting versions of RAkEL achieve better results. There are three hyperparameters to be tuned: the size of label-sets k and the number of models m, as well as the base method. The underlying base method can be either LP or PSt. The positive aspect of RAkEL is that uses a smaller number of classifiers than BR and can provide better generalization and is not underfitting as LP. However, it does not scale well in time, as the number of labels and number of instances increases.

Hierarchy of Multi-label Classifiers **(HOMER)** (Tsoumakas et al., 2008) is an ensemble based on the transformation of the problem into a tree-shaped hierarchy of simpler, better-balanced, MLC problems, utilizing the divide and conquer strategy. The tree is constructed in such a manner that, at the leaves, there are the singleton labels, while the internal nodes represent joint label sets. A node will contain a training sample if and only if the sample is annotated with at least one of the labels of the label-set contained in a node. The method consists of two phases: first, the tree is built such that labels from the parent node are distributed to the children nodes using a balanced clustering algorithm. Second, the multi-label model is trained on a reduced label-subset, and the process is repeated until all nodes are with one label. Such an approach provides the opportunity to cluster dependent labels into a single node. The hyperparameters of the method are the number of children for a parent node (number of clusters) and the base learner. It is predominantly useful in tasks with a large number of labels where it is shown to have the best predictive performance (Madjarov et al., 2012). However, the constructed hierarchy is not utilized in problems with a smaller number of labels, hence this method does not show its full potential on datasets with such property (Moyano et al., 2018).

Random Subspace **(RS)** multilabel method is an extension of the Random Subspace methodology for single target prediction (Ho, 1998) into the area of MLC. It works with a random sampling of the features. Additionally, one can subsample the instances from the training set. For each subsample generated alongside the two dimensions of features and instances, either problem transformation or algorithm adaptation method can be used. There are four hyperparameters to tune: the percentage of the attribute space to be used, the percentage of sample space to be used, the number of models in the ensemble to be built and the multi-label classifier at the base level. This ensemble method is usually faster than bagging and other ensemble methods likewise conditioned on the base multilabel learners.

The AdaBoost **(AdaBoost, AdaBoost.MH)** (Schapire & Singer, 2000) method is introducing a set of weights maintained both on the examples (as in classical AdaBoost method Freund & Schapire, 1997) and the labels. The formula for calculating the weights incorporates the example-label pairs that are miss-classified by the base classifier. At each iteration, the method builds a simple classifier (e.g., decision stump — a decision tree of depth 1). The classifier uses weights to focus more on the examples that are hard to predict. The base classifier

should provide confidences, that are used to obtain a prediction. The final prediction is obtained by combining the confidences of each of the base models, weighted by the corresponding model weights. The parameter of the method is the number of boosted decision trees. This method is the same as applying AdaBoost to $|\mathcal{L}|$ binary datasets as in BR (Moyano et al., 2018; Schapire & Singer, 2000).

Cost-Sensitive Multi-label Embedding **(CLEMS)** (Huang & Lin, 2017) belongs to a special type of family of multi-label methods, known as Label Embedding methods. In general, these methods try to embed the label-space into a particular number of dimensions using some embedding technique. It is assumed that the embedded space represents a latent structure of the labels. For learning, either problem transformation or algorithm adaptation method is applied to the augmented feature space. At prediction time, embedding methods employ regression techniques to predict the value of the embedded features. One type of label embedding method is known as Cost-Sensitive Embedding. It considers the performance criteria being optimized, as a parameter. In particular, the method considered here employs weighted multidimensional scaling as an embedding technique (Kruskal, 1964). It embeds the cost matrix of unique label combinations. The cost matrix contains the cost of mistaking a given label combination for another. The hyperparameters of CLEMS are the performance/cost function, underlying MLC method, the regression method used to predict the values of the embedding features and the number of embedding dimensions. The most effective value for the number of embedding dimensions is the number of labels. The positive aspect of the method is that it can provide good results for a specific cost function being optimized. On the negative side, this method is dependent on the underlying MLC method and requires building a specific model for each cost function for optimal performance per measure.

Triple Random Ensemble **(TREMLC)** (Nasierding et al., 2010) is an ensemble for MLC that combines three ensemble strategies: a sampling of the instance space, sampling of the feature space and sampling of the target space. It is in essence combination of a Random Forest with RAkEL as a base classification method. The parameters of the method are bag size, the number of features to subsample, the size of label sets and the number of models to be built. Its drawback is that it inherits the large computational time of the RAkEL method, however, it scales better regarding the number of instances and features since the Random Forest method reduces the instance and label space.

Subset Mapper **(SM)** (Freund & Schapire, 1997) uses Hamming distance to make mapping between the output of a multi-label classifier and a known label combination seen in the training set. From the predicted probability distribution, SM will produce a labeled subset and will calculate the hamming distance to the labels of the training instances. The new test sample as prediction will take the labelset that resulted in the smallest distance. The parameter of this method is the base learning MLC method. This method does not generalize beyond the labelsets seen in the training set.

3.2. Algorithm adaptation methods

The adjustments of the underlying algorithm of the single target methods are the core idea on which the algorithm adaptation methods are built. For example, the adjustments for the trees to handle the multi-label problem are two-fold. First, the nodes in the trees are allowed to predict multiple labels at once. Second, the splits are generated by adjusting the impurity measure to take into account the membership and non-membership of a label in the set of relevant and irrelevant labels for the samples. These two adjustments allow for each of the labels to contribute when creating the splits and obtaining the predictions. Neural networks are inherently designed to tackle multiple targets simultaneously. This is usually done by allowing each of the output neurons to generate score estimates from 0 to 1 in the output neurons. Instance-based such as kNN (k Nearest Neighbours) based methods can also be used for MLC by design: the search of nearest neighbours is in the descriptive features space, the only difference is the calculation of the prediction. For example, Ml-KNN uses Bayesian posterior probability for the estimation of the scores. Support vector machine-based methods use SVM principles when building the model, often with modified cost function to optimize. Probabilistic models, in general, try to use the Bayes formula or Gaussian mixture models in a multi-label scenario. Fig. 4 depicts the algorithm adaptation methods used in this study.

3.2.1. Singleton algorithm adaptation methods

Predictive Clustering Trees **(PCTs)** (Blockeel et al., 1998) are decision trees viewing the data as a hierarchy of clusters. This method uses a standard TDITD algorithm for induction of the tree (Quinlan, 1986). At the top node, all data samples belong to the same cluster. This cluster is recursively partitioned into smaller clusters, such that the variance (impurity measure) is reduced. The variance function and the prototype function are selected for the task at hand. In the case of MLC, the variance function is computed as the sum of the Gini indices of the labels. The prototype function returns a vector of probabilities that a sample is labelled with a particular label. As stopping criteria for growing the tree the F-test is used. That is the only hyperparameter of the model that needs to be tuned. The positive aspects of this method include the fast time for training and prediction, and it is one of the rear MLC methods that can provide interpretable results. As negative aspects are that a single tree may be poor in performance, however, an ensemble of PCT can be a powerful learning model.

Back-propagation Neural Networks **(BPNN)** (Read & Perez-Cruz, 2014; Zhang & Zhou, 2006) is a neural network approach to the problem of MLC. It is the standard multi-layer perception method. It uses the back-propagation algorithm to calculate the parameters of the network. The hyperparameters of the method are the learning rate, the number of epochs and the number of hidden units. The positive aspects of this method are that it can provide good performance if a large number of training samples are available, and inherently target multi-target problems. It faces drawbacks on the time needed for hyperparameter optimization. A popular approach in this family of methods is the stacking of multiple layers of hidden units, thus increasing the neural network architecture in depth. This is part of a much broader range of methods referred to as deep learning. Given its popularity, a separate paragraph is dedicated to one deep-learning approach used to model higher-level features, Restricted Boltzmann Machines **(RBMs)**.

RBMs is a type of deep-learning method that aims to discover the underlying regularities of the observed data (Hinton & Salakhutdinov, 2006). A Boltzmann machine can be represented as a fully connected network. The restricted Boltzmann machine additionally has the restriction of connections between neurons in the same layer. Usually, the parameters of the network are learned by minimizing contrastive divergence (Hinton, 2002). The stacking of multiple RBMs creates so-called Deep Belief Networks **(DBNs)**. The standard back-propagation algorithm can be used to fine-tune the parameters of the network in a supervised fashion. Using DBNs one can generate new features as different representations of the data. Those features can be used as input to any multi-label classifier. The hyperparameters of this method include the same as for the BPNN method and an additional two: the number of hidden layers and the output multi-label classifier. The novel representation of the input data provided by DBNs can lead to improved performance, on the cost of increased time and space complexity for training the method (Read & Perez-Cruz, 2014).

Multi-label ARAM **(MLARAM)** network (Sapozhnikova, 2009) is an extension of Adaptive Resonance Associative Map neural-fuzzy networks. ARAM networks for supervised learning consists of two self-organizing maps sharing the same output neurons. The first self-organizing map tries to encode the input space into prototypes, while simultaneously trying to characterize the prototypes with a mapping encoding the labels. A parameter called vigilance is used to control the specificity of the prototypes. Larger values indicate more specific

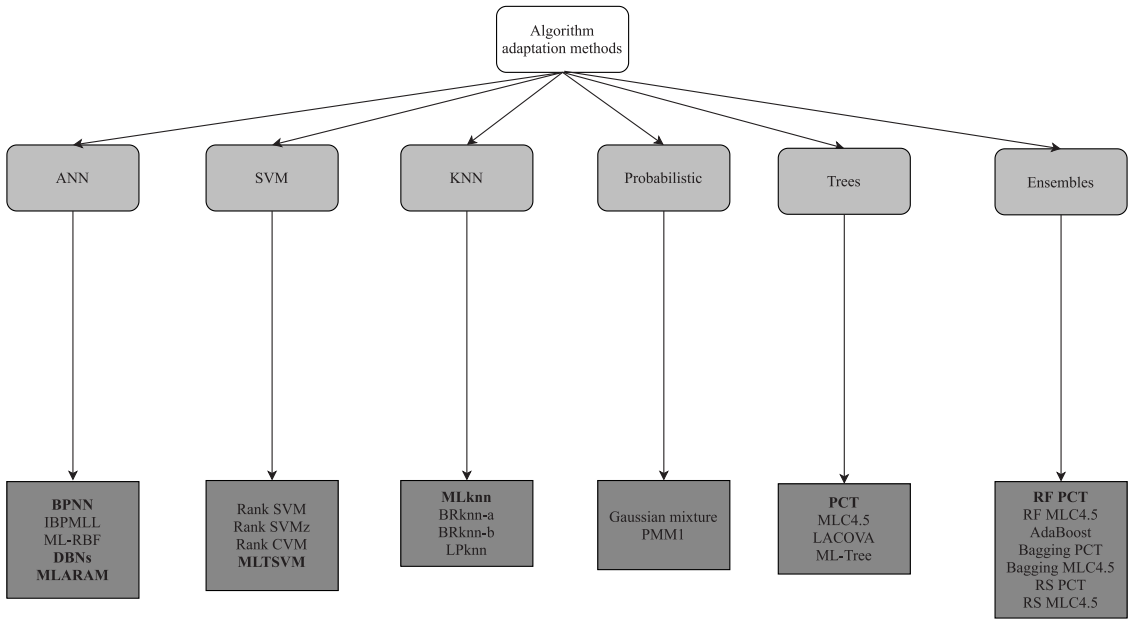


Fig. 4. Algorithm adaptation methods. The methods used in this study are bold.

prototypes (Tan, 1995). MLARAM is an extension of ARAM in such a way that it allows flexibility in determining when a particular node is activated, taking into consideration label dependencies. The output predictions may vary due to the order in which training examples are presented. The flexibility of inclusion depends on a threshold parameter. The parameters to be tuned are vigilance and threshold. The positive aspect is that is fast to train and is useful in text classification of a large volume of data, however since it is based on Adaptive Resonance Theory neural-fuzzy networks it has generalization limitations if too many prototypes are built.

Twin Multi-Label Support Vector Machine **(MLTSVM)** (Chen et al., 2016) tries to fit multiple nonparallel hyperplanes to the data to capture the multi-label information embedded in the data. It follows the Twin SVM concept (Jayadeva & Chandra, 2007), where (in the binary classification case) one tries to find two nonparallel hyperplanes such that each one is closer to its class, but it is further than the others. At the training phase, this method constructs multiple nonparallel hyperplanes to exploit the multi-label information via solving several quadratic programming problems using fast procedures. The prediction is obtained by calculating the distance of the test sample to the different hyperplanes. The hyperparameters of the method are the threshold above which a label is assigned, empirical risk penalty (determines the trade-off between the loss terms in the loss function) and a regularization parameter. Chen et al. (2016) show that MLTSVM outperforms other SVM-based methods for MLC (based on Hamming loss and ranking evaluation measures). Its advantage is that it is fast to train because of the fast underlying procedures for solving the quadratic problem formulated by SVM.

Multi-label k Nearest Neighbour **(MLkNN)** (Zhang & Zhou, 2005) method is an adaptation of the Nearest Neighbour (Ruiz, 1986) paradigm for multi-label problems. It finds the k nearest neighbours of a given sample as in a single target kNN algorithm. It constructs prior and conditional probabilities from the training data and thus can use the Bayes formula to obtain the posterior probability for a given label on a given test sample. The parameter of the method is the number of neighbours. It is fast to build, but as a lazy method, obtaining the prediction is a more expensive operation. Thus it may not be suitable in a situation where fast predictions are required.

3.2.2. Ensemble of algorithm adaptation methods

Random Forest of Predictive Clustering Trees **(RFPCT)** (Kocev, 2011; Madjarov et al., 2012) uses the random forest method (Breiman, 2001) with a PCT as a base learning model. It samples both the instance space (sampling with replacement) and the feature space (at random at each tree node). The parameters of the method are the number of features to be used when building the trees and the number of ensemble members. The positive aspects of the method are that it is fast and can tackle the correlation between the labels inherently. On the negative side, RFPCT is not suitable for datasets with large sparse feature vectors. Due to the process of a random selection of attributes, it often can happen that these sparse features will be chosen for building the trees. In such a scenario, the trees will have low predictive performance and that will hurt the overall predictive performance of the ensemble.

3.3. Addressing specific properties of MLC

Since there exist multiple targets to predict, the task of MLC is more challenging than binary classification (Herrera et al., 2016). While in binary classification the complexity of the models depends on the number of relevant features and number of samples, the MLC task has an additional complexity along the target dimension. These issues present specific challenges (i.e., label dependencies and high dimensional multi-label space), and influence the application and the development of MLC methods.

3.3.1. Label dependencies

Label dependence has a central position in the definition of the MLC task. It presents how the labels are related among themselves — for example, consider labelling an image of a seaside; Given the presence of the label ‘sea’, it is more probable that it will also be labelled with ‘beach’ than ’city street’. Exploiting these label dependencies can strongly influence the performance of a given MLC method. In the extreme case of the non-existence of such dependencies then the best way to approach MLC is by looking at the task as separate L tasks, i.e., binary relevance. However, in real-world applications, typically there is a strong influence of the label dependencies on the performance of the MLC methods.

The most straightforward problem-transformation method, BR (and its corresponding ensemble - EBR), does not exploit the dependence information, and it is its most common referenced drawback (Read

et al., 2011). To bridge that gap, there are various ways to introduce the dependency information, hence the appearance of different methods such as CC, CDN, MBR, SM, CLEMS and ECC. Binary relevance's counterpart, the LP method, takes into consideration the label dependencies. However, it also considers the nonexistent dependency. Pruned sets method utilizes LP and thus has the same positive aspect and drawback.

The ensemble of MLC models such as HOMER, where the groups of similar labels are being joined together, exploit the label dependency explicitly. HOMER is regrouping the labels into smaller groups, such that dependent labels belong to closer nodes in the tree. Chi-dep's has a statistical driven built-in mechanism for resolving the dependencies between labels. The ensembles of LP and PSt exploit the label dependencies given their base MLC classification method. RAkEL and TREMLC provide an opportunity to exploit the dependencies between the labels. Since they are randomly subsampling the label space, independent labels may be grouped, thus non-existing dependencies are modeled. Nevertheless, if a large number of base models are built it is expected that these non-existing dependencies will be averaged out. The random sampling of the labels increases the bias of the methods. However, the variance of the methods is decreased with the averaging.

On the other side of the spectrum, most of the algorithm adaptation methods (and their corresponding ensembles) in the names of PCT, RFPCT, BPNN, DBNs, MLARAM and MLTSVMs, have the built-in mechanism to deal with this challenge. The presence of label dependencies in methods that have as hyper-parameter a MLC method is tackled depending on the choice of the particular method. Since AdaBoost can be viewed as applying AdaBoost as base-learner to a BR (Schapire & Singer, 2000), this method has no mechanism of dealing with dependencies between the labels. MLkNN cannot exploit label dependencies since it is similar to applying BR with kNN as a base learner.

All in all, different methods have different approaches to how they tackle the dependencies between the labels. Some try to augment the descriptive space, others try to model the dependencies modifying the dataset, others are modifying the learning method. In general, it is expected that adding additional information to the method can help to improve performance. This means that it is somewhat expected for methods considering the label-dependencies to perform better.

3.3.2. High-dimensional label space

The challenge of high-dimensional label space mimics the well-known problem of *curse of dimensionality* (Bellman, 1954; Guyon & Elisseeff, 2003) appearing in the feature space. The curse of dimensionality references the issues in datasets with many features (Kira & Rendell, 1992). Following the same analogy, the large number of labels imposes a problem for the multi-label methods. It influences the time needed to obtain a prediction and the performance of the methods (Herrera et al., 2016). The *curse of dimensionality* in the input space exist in MLC tasks also. We discuss these issues in more detail in the remainder of this section.

The curse of dimensionality in the label space is best illustrated through the computational complexity analysis for both training and testing time provided in Table C.3 in the Appendix, available online. We give the complexity analysis as provided by the authors proposing the specific method. If this is not available, then we performed the analysis and the summary of it is given in the table (as reference these methods are marked as [ours]). The training time complexity is the time needed to learn the predictive models, while the testing time complexity is the time needed to predict for a given example.

BR and CC scale linearly with the number of labels. CLR scales quadratically with the number of labels since it needs to build pairwise base learner models. The time complexity of LP scales exponentially with the number of labels in the worst case. However, in practice, the number of classifiers is limited to the $min(2^{|\mathcal{L}|}, n_{tr})$, where n_{tr} is the number of training instances. Additionally, this method should take into consideration the applied strategy for solving the multi-class problem by the base learner. If the base learner is SVM and if the applied strategy is one vs one *(OVO)* the complexity scales quadratically with the number of unique label sets. If the applied strategy is one-vs-all *(OVA)*, the complexity scales linearly with the number of unique label sets. Since PSt utilize the LP method, at worst if no label-set is removed from the training set its computational complexity is equivalent to the LP method. However, in practice, it is much faster since depending on the pruning parameter infrequent label-sets are removed.

The complexity of the ensembles for MLC built from BR, CC, PSt and LP preserve the same complexity concerning the labels as their base MLC models. Their complexity differs in the number of built multi-label models. However, this is not true for HOMER. HOMER requires splitting the label space into smaller clusters when building the hierarchy, this means that its speed is dependent on the clustering algorithm. In Tsoumakas et al. (2008), it is shown that the balanced k-means method scales linearly with the number of labels. MBR, CDN, ECD, MLTSVM and AdaBoost scale linearly with the number of labels. For RFPCT and PCT, the computational complexity depends on a logarithmic function of the number of labels.

An important aspect of the computational complexity analysis for a specific method is the base learner it uses (and especially the susceptibility of the base learner to the 'curse of dimensionality'). For example, using SVM as a base learner requires calculation of the kernel, which amounts to computational complexity of $O(n_{tr}f^2)$ for $f < n_{tr}$. If the dataset has a large number of samples then the time needed for training the method will be larger: $O(n_{tr}^3)$ if $f \sim n_{tr}$.

The high-dimensional label space influences both problem transformation and algorithm adaptation approaches to MLC, not just in terms of computational complexity but also in terms of making the problem more imbalanced. Namely, the high-dimensional label spaces encountered in real-life datasets are usually also sparse: low label cardinality (average number of labels per example) and low label density (frequency of labels). The sparsity then poses a challenge for both problem transformation and algorithm adaptation approaches as follows. In the former case, in binary relevance and label power set, the simpler classification tasks are imbalanced. Hence, once could resort to specific approaches addressing this issue, thus, even more, increasing the computational cost. In the latter case, the sparse output spaces could make the learning of predictive models more difficult (for example, see PCTs Kocev et al., 2013, extreme MLC Jain et al., 2016). A way to approach this is to embed the sparse space in a more compact space through matrix factorization or deep embedding methods (for example, see Stepišnik & Kocev, 2020).

4. Experimental design

In this section, we discuss the experimental design. First, we present the experimental methodology adopted for conducting the comprehensive study. Second, we present the details on the specific experimental setup used throughout the experiments. Third, we give the specific parameter instantiations used to execute the experiments. Next, we present the evaluation measures used to access the performance of the methods. Finally, we discuss the statistical evaluation used to analyse the results from the study.

4.1. Experimental methodology

At the basis of any large scale, comprehensive study lies a suitable experimental methodology for hyperparameter optimization of the MLC methods and their base classifiers as well as measures and procedures for accessing the (predictive) performance of the methods on the datasets. In this study, we adopted and adapted the experimental methodology presented in Caruana and Niculescu-Mizil (2006). Fig. 5 depicts the four-stages experimental methodology used in this work.

In the first stage, the multi-label datasets considered in this study come in predefined train-test splits. We first sample 1000 examples

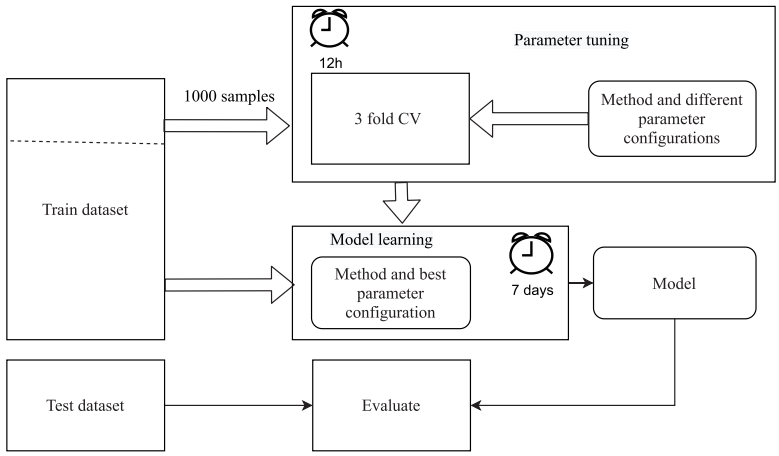


Fig. 5. The design of the experimental setup and protocol.

from the training set using iterative stratification (Sechidis et al., 2011) (for the datasets with less than 1000 examples, we take all of them).

In the second stage, on the selected portion of the data we perform 3-fold cross-validation to select the optimal hyper-parameters under a time-budget constraint of 12 h (similarly as in AutoML Hutter et al., 2019). It means that we allow for evaluation as much as possible (uniformly randomly selected) parameter combinations within the time budget and select the best one out of these. For each of the methods, we evaluate a multitude of hyperparameter combinations defined with ranges taken from the literature (Madjarov et al., 2012; Moyano et al., 2018; Read et al., 2011; de Sá et al., 2018; Tsoumakas, Katakis, & Vlahavas, 2011) (detailed range values for the parameters are given in the Supplementary Material Sec. Appendix B, available online). After the expiration of the time budget or evaluating all of the combinations, whichever comes first, the hyperparameter combination that leads to the smallest Hamming loss is selected as best. If the time budget did not allow for the evaluation of at least one combination then the literature recommended values are used.

In the third stage, we learn a predictive model using the complete training set and the selected optimal hyperparameter combination. Also in this stage, we set a time budget for learning a predictive model to 7 days. In the cases where this occurs, the performance of that specific method is marked as `(DNF)`.

In the fourth and final stage, we evaluate the predictive performance of the method using the test set. The test set is used only to assess the predictive performance of the learned models, and it has not been used at any other stage of the experimental evaluation. For the methods that did not yield a model from the previous stage, their performance was set to the worst possible value for each evaluation measure.

4.2. Implementation of the experimental methodology

For undertaking such an extensive experimental study, we needed to implement a multi-platform experimental methodology. The implementation of the experimental methodology is done using the Python programming language. It follows the design principles and guidelines of the `skmultilearn` (Szymański & Kajdanowicz, 2019) and `sckit-learn` (Buitinck et al., 2013) ecosystem. We designed a unified experimental methodology. To include methods and their well-tested implementations from `CLUS`, `MULAN` (Tsoumakas, Spyromitros-Xioufis et al., 2011) and `MEKA` (Read et al., 2016), a unified experimental setup was designed. An `Ubuntu-Xenial` image was built using `Singularity` (Kurtzer et al., 2017) to provide the same experimental conditions for the experiments.

The methods are abstracted into a generic form to provide a unique way of accessing. Using these libraries require specific pre-processing and formatting of the data: For example, `MULAN` requires `XML` files storing the names of the labels. After learning the model on a given dataset with a specific method, the predictions (as raw scores) are stored. Next, the raw prediction scores are used as input to the evaluation measures. The `sckit-learn` implementation of the measures is used. The one error measure is not implemented in `sckit-learn`, therefore, we implemented it. Furthermore, we used a wrapper to access the `MEKA` library the `skmultilearn`. The wrapper provides a uniform way to obtain the scores, predictions and additional information describing the models. Methods accessed through this library are MBR, BR, LP (LC), PSt, CC, RAkEL, EBR, ELP, ECC, CDN, EPS, BPNN, RSLP, DBPNN, SM, TREMLC. Additionally, the methods CLR and HOMER were accessed via `MEKA`'s wrapper for `MULAN`. Methods from `MEKA` that do not provide score estimates are LP, CC and SM. To use `MULAN`, a suitable wrapper around it was used to access the CDE method. Next, `CLUS` was used to access RFPCT and PCT. RFPCT and PCT do provide prediction scores that can be seen as the probability of a given example being labeled with a given label. Finally, the `skmultilearn` library was used to access MLRAM, MLkNN, CLEMS, AdaBoost, RFDTBR and MLTSVM. MLTSVM does not provide score estimates. It has an internal mechanism for providing predictions. AdaBoost and RFDTBR required the implementation of supporting code to retrieve probability estimates.

4.3. Parameter tuning

The goal of the experimental study is to provide the same conditions for all methods and to provide an opportunity for each method to give its best results. Hence, we need to select the optimal parameters for each of the methods. This is especially relevant for MLC methods that use as base classifiers methods that require tuning (e.g., SVMs). Based on our experimental methodology outlined in Fig. 5, we select the optimal combination of parameters for each method using parameter ranges as defined in the literature. Detailed description of the specific parameter ranges and values evaluated in this study are provided in Appendix B, available online.

4.4. Evaluation measures

We use 18 predictive performance and two efficiency criteria to evaluate the performance of the methods across the datasets. Fig. 6 depicts a taxonomy of the evaluation measures (or *criteria*, *scores*) (Madjarov et al., 2012).

For the evaluation of the predictive performance of the methods, `sckit-learn` implementation of the measures for MLC is used. The evaluation measures requiring score estimates as input are provided with both the calculated scores and the ground truth labels. Since LP, CC, SM and MLTSVM do not generate scores, score-based evaluation

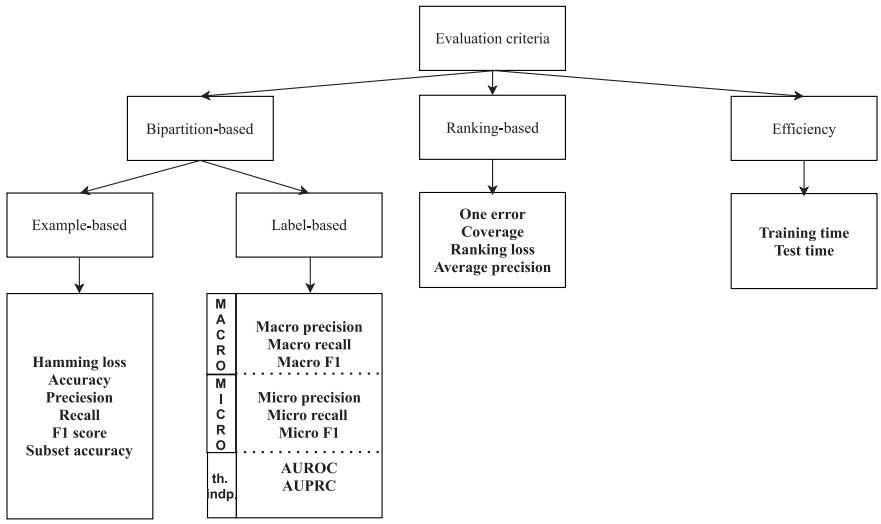


Fig. 6. Taxonomy of the evaluation criteria. There are two types of measures, one used to evaluate the performance of the methods and the other one used to evaluate the efficiency of the methods. The performance evaluation can be done either via evaluating bipartitions or via evaluating the relevance of a label for the sample.

criteria are not calculated for them and the predictions as generated by the implementations of the methods are used. Most of the methods used in this study return raw scores as predictions. These scores then need to be thresholded to obtain the label predictions. Hence, we use the global PCut thresholding method (Read et al., 2011): It selects a threshold using an iterative procedure such that the label cardinality of the training set is equal to the cardinality of the test set. A detailed description of the evaluation measures is given in the Supplementary material as well as in the referenced literature (especially Gibaja & Ventura, 2015; Herrera et al., 2016; Madjarov et al., 2012; Moyano et al., 2018; Rivolli et al., 2020; Zhang et al., 2018; Zhang & Zhou, 2014).

The assessment of the performance of an MLC method relies on the type of predictions it produces: relevance scores per label or a bipartition. If a method produces relevance scores, special postprocessing techniques can be employed to produce bipartitions (Reem et al., 2014). In the case of bipartitions, the evaluation measures can be separated into example-based and label-based measures. The latter can then be micro- or macro-averaged, based on the fact whether the joint statistics for all labels are used to calculate the measure or the per label measures are averaged into a single value. In the case of providing relevance scores per label, threshold independent or ranking-based measures can be calculated. The different methods might be more biased towards optimizing a given evaluation measure than other methods. For example, the methods predicting label sets have favourable evaluation using example-based measures, while the methods predicting each label with a different model and combining the predictions have favourable evaluation using macro-averaged measures. Hence, for an unbiased view of the performance of the MLC methods, one needs to consider multiple evaluation measures.

To evaluate the efficiency of the MLC methods, the training and test time is measured. Training time measures the time needed to learn the predictive model, and the testing time measures the time needed to make predictions for the available test set. While we are aware that the execution times are dependent on the specific implementation of a given method, measuring these times have practical relevance. They provide a glimpse into the time a method needs to produce a model or a prediction and can serve as a guideline for a practitioner when a decision needs to be made on the use of a specific method. These efficiency estimates should be examined together with the computational complexity of the methods (as provided in Table C.3).

4.5. Statistical evaluation

Accessing the overall differences in performance across the datasets to determine if the differences in performance of the methods are statistically significant, we used the corrected Friedman test (Iman & Davenport, 1980) and the post-hoc Nemenyi test (Nemenyi, 1963). Friedman test is a non-parametric multiple hypothesis test (Friedman, 1940). It ranks the methods according to their performance for each data separately. Then it calculates the average ranks of the methods and calculates the Friedman statistics. Due to the conservatives of the test the corrected Friedman statistics is preferred.

If statistical significance between the methods exists, the post-hoc Nemenyi test is used to identify the methods with statistically significant differences in performance. The performance of the two methods is statistically significant if their ranks differ more than the critical distance (calculated for a given number of methods, datasets and a significance level). The significant level α is set to 0.05.

Limitations of the study. While having great practical relevance for detailed depicting the landscape of a learning task, these forms of studies have inherent limitations. One of the drawbacks of making large experimental studies relates to that they are computationally expensive. For example, the time of computation for hyperparameter tuning as well as building the models on all of the 42 datasets, while optimizing single performance criteria took approximately 126720 CPU hours. This time is linearly dependent on the number of performance criteria one is optimizing. Thus optimization over all of the performance criteria is practically infeasible with a lack of appropriate infrastructure. Moreover, the task of making sense out of an abundance of results that will emerge is challenging.

To overcome this challenge, we adopt design choices following recognized literature standards (Caruana & Niculescu-Mizil, 2006). The iterative stratified cross-validation strategy preserves the frequency of the labels. In the study, there are 19 datasets with more than 1000 samples and 7 datasets with more than 5000 labels (2 with more than 9000). The iterative stratified sampling strategy sub-samples the datasets, trying to preserve the frequency of the labelsets (Sechidis et al., 2011). Thus, the potential effect of overfitting to the data is not expected to have a noticeable influence over the choice of the best method in the given experimental scenario for all of the datasets. Even if it has, the influence will be small, and will not hurt the conclusions.

Following Madjarov et al. (2012), we selected Hamming loss as optimization criteria as it is analogous to error rate in single target classification. It provides penalization for the miss-classification of

individual labels. Optimizing for other measures, e.g., F1, precision and recall (micro, macro and example-based), have inherited bias towards specific paradigms that are correlated with the assumptions done by the families of methods. Considering threshold independent measures discard methods that cannot produce rankings. Accuracy example-based evaluates just the correctly predicted labels, while the subset accuracy is blind to correct prediction of right and incorrect prediction of wrongly predicted labels. Thus, Hamming loss seems like the fairest choice for optimization.

Another challenge when performing large scale studies emerges from the included datasets. The conclusions from such a study find their validity to hold in the meta-space constrained by the values of the meta-features of the included datasets. For MLC, to the best of our knowledge, this is the greatest amount of datasets and methods being evaluated. Thus we believe that it depicts the current state of the field pointing out guidelines for both practitioners and experts to design and choose the most suitable methods for their MLC problem, and further expand the field of MLC as an important task in machine learning.

While deep learning methods have significantly gained in popularity in recent years and managed to push the predictive performance boundaries of machine learning models, they still do not perform as well on tabular data as they do on image and text data (Gorishniy et al., 2021). Furthermore, their greatest power resides in the ability to extract useful feature representations by leveraging large quantities of raw data typically given as images or textual documents. However, most of the MLC problems from the domains of bioinformatics or medicine typically do not have many samples needed by deep learning methods to achieve state-of-the-art performance (Liu et al., 2020). Considering these obstacles, we evaluated the performance of two established neural network architectures for MLC on tabular data — Deep Boltzmann machines (DBM) and Multi-layer Perception for MLC (BPNN) (Read & Perez-Cruz, 2014; Zhang & Zhou, 2006).

5. Results and discussion

This section paints the landscape of MLC methods by providing a discussion on the results from the comprehensive empirical study. The discussion is organized into four parts: (1) comparison of problem transformation methods, (2) comparison of algorithm adaptation methods, (3) analysis of selected best-performing methods and (4) computational efficiency analysis of the methods. We focus the discussion on the predictive performance using four evaluation measures (Hamming Loss, F1 example-based, Micro precision and AUPRC), thus ensuring the inclusion of all the different groups of measures. The complete results and their detailed analysis are provided in the Appendix (available online), and at http://mlc.ijs.si.

5.1. Problem transformation method comparison

Fig. 7 depicts the average rank diagrams comparing the problem transformation methods. At a first glance, we can observe that the problem transformation methods that are utilizing BR outperform all other methods across all evaluation measures. More specifically, the best performing method is RFDTBR — it is the best-ranked method on 12 evaluation measures (and second-best on three more) and is the most efficient in terms of both training and testing time. EBRJ48, AdaBoost.MH, ECC J48, TREMLC and PSt are often among the top-ranked methods, while CDN, SM and HOMER are the worst-performing methods (CDN is worst-ranked on 13 evaluation measures). In the remainder of this section, we analyse the performance of all methods in more detail, along with the different types of evaluation measures and different subgroups of methods, finally selecting the most promising problem transformation methods.

We first summarize the performance of the methods according to the different groups of evaluation measures. Considering the example-based evaluation measures, the best performing methods are RFDTBR, ECC J48 and RSLP, while the worst performing methods are MBR, HOMER and CDN. Next, focusing on the label-based evaluation measures, we can make the following observations: i) on the threshold-independent measures (AUCROC and AUPRC), the best performing methods are RFDTBR and EBR J48, while RAkEL, HOMER, and CDN are on the losing end; ii) on the micro averaged measures, the best performing methods are RFDTBR, TREMLC, ECC J48, AdaBoost.MH; and iii) on the macro averaged measures, BR, ECC J48, CLR and AdaBoost.MH perform best. It is interesting to observe the drop of RFDTBR in the ranking based on macro-averaged measures. This suggests that RFDTBR as a base learner fails to provide good individual predictions per label, as opposed to BR with SVMs. Furthermore, analysing the performance on ranking-based measures, RFDTBR, EBRJ48, AdaBoost.MH, and PSt has the best performance, while RAkEL, HOMER, and CDN are the worst. Finally, considering efficiency, the best training times are obtained with RFDTBR, HOMER and PSt, and the worst with ECC J48, EBR J48 and AdaBoost.MH, The best testing times are obtained with RFDTBR, CDE, SSM, and the worst with LP, RAkEL, and CLR.

When it comes to using ensemble methods to approach MLC with problem transformation (using the BR, CC or LP approach), it is very important to select the proper base predictive models of those ensembles (Madjarov et al., 2012; Moyano et al., 2018; Rivolli et al., 2020). There are two widely used options concerning the base predictive models: J48 and SVMs.

We performed an additional experimental study concerning this choice. Namely, we evaluated EBR, ECC and ELP built with J48 trees and SVMs as base predictive models. The results showed that using J48 as a base predictive model is generally beneficial in terms of predictive performance: on a large majority of predictive performance measures, the ensembles using J48 are better ranked than their SVM counterparts. Moreover, EBR with J48 is the best-ranked method on 14 out of 18 predictive performance measures. The ensembles with SVMs perform better on the macro aggregated evaluation measures and subset accuracy. In terms of efficiency, ensembles with J48 are undoubtedly the preferred choice: they are faster to learn and make predictions than their SVM counterparts. In addition, J48 does not need parameter tuning, while two SVMs parameters need to be tuned. A detailed discussion of the evaluation is available in the Supplementary material accessible online.

Several observations can be made by comparing the performance of the different ensembles of problem transformation methods (EBR, ECC, ELP). First, it can be observed that EBR tends to perform best according to ranking-based measures, micro-averaged label-based and threshold-independent measures. The predictions of EBR are characterized with good precision scores hence they provide more exact predictions (i.e., the labels predicted as relevant are truly relevant). Conversely, according to the example-based and macro-averaged label-based measures, the ECC method ranks best. ECC has high values for recall on the example-based measures, meaning that its predictions are more complete (i.e., the truly relevant labels are indeed predicted as relevant). EBR performs well on precision, and ECC performs well on recall. EBR and ECC models with J48 tend to provide good results (EBR on micro and ranking-based measures and ECC on micro and example-based measures), do not require tuning of parameters and are fast to build. ELP shows the worst performance in general.

LP-based architectures of problem transformation methods have good performance as measured by the recall, and consequently accordingly to the F1 measure. More specifically, according to example-based measures, these methods produce complete predictions, where truly relevant labels are indeed predicted as relevant. In contrast, they fall short on precision-based measures, meaning that not all the predicted labels are truly relevant for the examples. Similar observations can be made for the micro-averaged label-based measures, but not for the macro-averaged label-based measures, where LP-based methods suffer reduced performance. Since LP-based methods predict partitions, they can preserve the label sets. This reflects the good rankings achieved

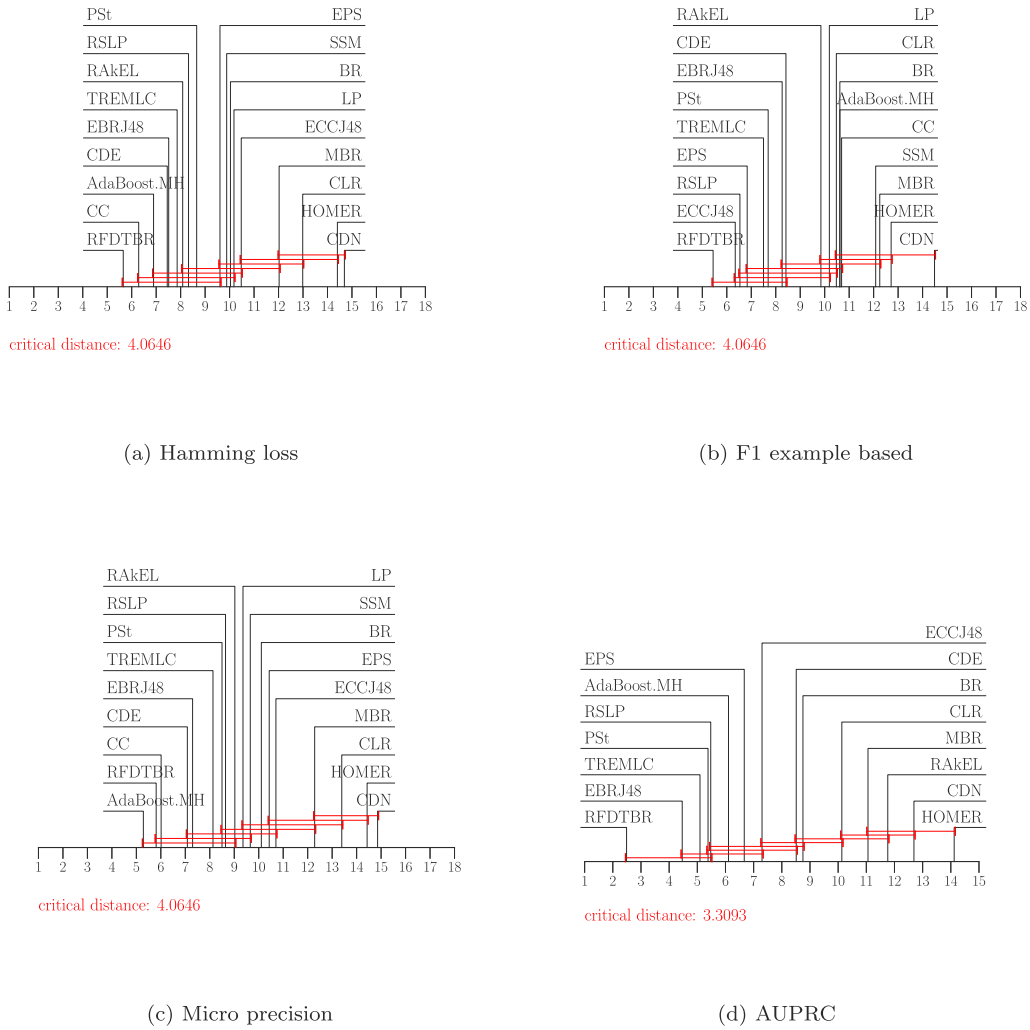


Fig. 7. Average rank diagrams comparing the predictive performance of problem transformation methods. The performance of the methods connected with a line is not statistically significantly different.

in example-based measures and micro-averaged label-based measures that are calculated by taking the labels jointly, before averaging them. However, when macro-averaged label-based measures are considered, the preservation of the label-sets seems not to be beneficial and the methods are unable to produce sufficient diversity per label. These conclusions are further confirmed by analysing the performance of the methods on the ranking and threshold-independent measures (see Fig. 7(d)). Again, LP-based methods are ranked lower as compared to the BR-based methods.

Comparing the results of LP-based and BR-based singletons utilizing SVMs as base learners, it can be observed that PSt is the best ranked, except for the macro measures. PSt prunes the infrequent label sets and trains an LP method on the modified dataset. The better ranking of PSt shows that the infrequent label-sets hurt the performance of the LP-based approaches. PSt is superior to its counterpart BR according to the example-based and micro-averaged label-based measures. Comparison of the BR-based and LP-based singletons versus the corresponding architectures shows better performance for the architectures, which most often is significantly large, for the best-performing methods.

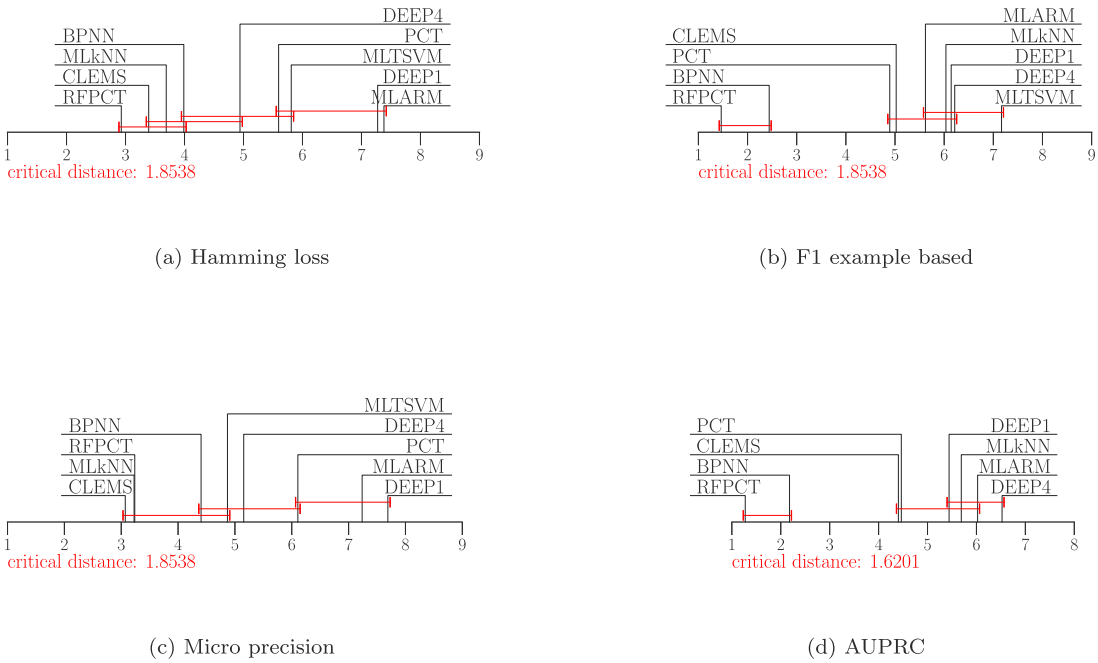
Based on the above discussion and the empirical evidence, we select RFDTBR, AdaBoost.MH, ECCJ48, TREMLC, PSt and EBRJ48 as the best performing group of the problem transformation methods.

5.2. Algorithm adaptation method comparison

Fig. 8 depicts the average rank diagrams for the algorithm adaptation methods. At a first glance, we can make the following observations: (i) The best performing method is RFPCT — it is best ranked according to 17 out of 18 performance evaluation measures; (ii) It is closely followed by BPNN — ranked second according to 16 out of 18 performance evaluation measures, with performance differences to RFPCT that are not statistically significant; and (iii) the worst-ranked methods are MLTSVM, DEEP 1 and DEEP 4 (DEEP 1 and 4 are the two architectures using DBNs to create a lower-dimensional representation of the input, and then using BPNNs or ECC as a second stage classifier).

By inspecting in detail the results across all types of evaluation measures, we find that the above observations hold: RFPCT and BPNN are typically the best-ranked methods. Here, we mention the two evaluation measures where this is not the case: Hamming Loss and micro-averaged precision. For both evaluation measures, CLEMS and MLkNN achieve good predictive performance (according to micro-averaged precision, CLEMS and MLkNN are the top-ranked methods, see Fig. 8(c)). The good performance on precision indicates that these methods are more conservative in assigning relevant labels. Usually, this means a weaker performance on recall-based measures.

Next, we have performed an extensive evaluation of four different architectures of deep belief networks (DBNs), as representatives of deep learning methods for MLC. The 4 DBN-based models were trained on



(a) Hamming loss

(b) F1 example based

(c) Micro precision

(d) AUPRC

Fig. 8. Average rank diagrams comparing the performance of algorithm adaptation methods. The performance of the methods connected with a line is not statistically significantly different.

the whole training set to increase their chance of preventing overfitting due to the small number of instances. These four architectures are obtained as a Cartesian product of two sets of parameters of the optimizer (learning rate and momentum) and the MLC classifier with the fixed-parameter at the second stage (one of BPNN or ECC).

Detailed analyses of the DBN results, given in the Supplementary material reveal the following. Using ECC as a MLC classifier in the second stage, is beneficial according to the example-based and label-based measures while using BPNN for that purpose is beneficial according to the threshold-independent label-based measures and the ranking-based measures. The better-ranked architectures for the two different MLC classifiers (DEEP1 and DEEP4) were selected for comparison with the other algorithm adaptation methods. Still, these two architectures have much worse performance as compared to other methods. This might be because the benchmarking datasets are of different sizes and there is a good portion of them with a small number of examples, which makes the DBNs overfit (Read & Perez-Cruz, 2014): This prompts for better exploration of the parameter space of DBNs in this context.

Based on the discussion and all of the empirical evidence presented, we select RFPCT and BPNN as the best performing group of algorithm adaptation methods.

5.3. Selected MLC methods performance comparison

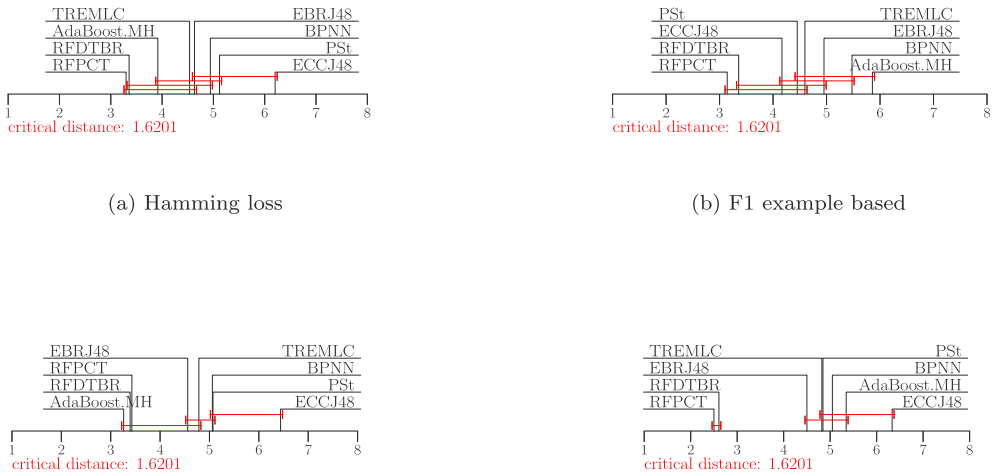
We further compare the results of the selected best-performing methods from both groups, i.e., the problem transformation methods and the algorithm adaptation methods. Fig. 9 depicts the results of this comparison. As explained above, we selected six problem transformation methods and two algorithm adaptation methods. At a glance, the results shown here, as well as the detailed results from the Appendix, clearly identify that tree-based model as the state-of-the-art, especially tree-based ensembles based on random forests. Below, we first discuss the performance of all considered methods along the lines of the different types of evaluation measures and then drill down to the performance of each selected method.

We start with discussing the results for the example-based evaluation measures (Figure D.11). Here, RFPCT is best ranked according to 4 out of 6 measures and second-best on the other two. RFDTBR is best ranked on one measure and second-best on 4. These two methods are the best performers, except on recall (where ECC J48 is top-ranked). It means that the predictions made by RFPCT and RFDTBR assign relevant labels more conservatively. Also, on the threshold-independent measures (Figure D.13), which provide the most holistic view on MLC method performance, RFPCT and RFDTBR are dominant (according to AUPRC, they statistically significantly outperform the competition).

In terms of the label-based evaluation measures (Figure D.14), the situation is not as clear. ECCJ48 is the best performing method according to 3 evaluation measures and worst-performing according to two evaluation measures. Namely, ECCJ48 is strong according to the recall measures (for the macro-averaged recall it even statistically significantly outperforms all competitors). This comes at the price of it being the worst-ranked method on precision. On precision-based measures, AdaBoost.MH is the best performing (with RFPCT and RFDTBR following closely) and among the worst-performing methods on recall-based measures. Interesting to note here is the difference in the performance of RFPCT due to the averaging of the recall: with macro averaging, it is worst-ranked, while with micro averaging, it is ranked second best. It indicates that RFPCT focuses on predicting the more frequent labels correctly at the cost of misclassifying the less frequent ones. This is in line with the understanding that the BR-type of methods are more appropriate for macro-averaging of the performance: they try to predict each of the labels separately as well as possible. Conversely, the methods that predict the complete or partial label set (such as LP-based methods and algorithm adaptation methods, incl. RFPCT) are well suited for micro-averaged measures.

Furthermore, in terms of ranking-based measures (Figure D.12), the best performing method is RFDTBR — it is top-ranked on all four evaluation measures, while RFPCT is second-best on three evaluation measures. These results indicate that by further improving the thresholding method for assessing whether a label is relevant or not, one can expect a further improvement of the performance of these methods on the other evaluation measures. Here, the worst-performing method is ECC J48 — it has the worst ranks according to three performance evaluation measures.

We next dig deeper into the performance of each of the selected methods. We start with the random forest approach to learning ensembles for MLC. When used either in local/problem transformation (RFDTBR) or global/algorithm adaptation (RFPCT) context, random



(a) Hamming loss (b) F1 example based

(c) Micro precision (d) AUPRC

Fig. 9. Average rank diagrams for the best-performing methods from both groups (problem transformation and algorithm adaptation) were selected after the per-group analysis of the results. The performance of the methods connected with a line is not statistically significantly different. The average ranking diagrams for all evaluation measures are available in Appendix A, available online.

forests show the best performance across the example-based, micro-averaged label-based and ranking-based measures, as well as threshold-independent measures. When considering the macro-averaged label-based measures, both of the methods, despite having good rankings on macro precision measures, are not able to predict as relevant all instances where the labels are truly relevant (per label) and thus have worse rankings on macro recall. This leads us to the observation that these methods are rather conservative when deciding whether a label is relevant or not.

ECCJ48 is best ranked on the recall-based measures but underperforms on the precision-based measures, where it is often being ranked worst. It indicates that ECCJ48 is rather liberal when assigning labels as relevant, i.e., it indeed truthfully predicts most of the relevant labels, but at the cost of also predicting irrelevant labels as relevant.

In contrast to ECCJ48, AdaBoost.MH shows good results on precision, as compared to the results on recall. The weights over the samples help AdaBoost to be conservative in its predictions. Additionally, it is ranked favourably according to the ranking-based measures (indicating that there is room for further improvement of the scores for the other measures by adjusting the thresholding method). For coverage and ranking loss, it is among the best-ranked methods.

EBRJ48 shows competitive performance on the ranking measures. It closely follows the best-ranked methods, often not statistically significantly different from them. On the threshold-independent measures, it is also ranked as the 3-rd best method. However, it suffers on the other measures, even though it is not statistically significantly worse than the best method in 4 out of 12 remaining measures (after excluding the ranking-based and the threshold-independent based ones).

The PSt method has good ranks for the example-based measures — it is not statistically significantly different from the best-ranked method in four out of six measures. However, on the remaining measures, it has hood ranks for two measures — macro F1 and coverage. The rationale behind this behaviour of PSt is that it predicts label-sets hence it can provide good performance on the per example-based measures. TREMLC, as an LP-based architecture, has better ranks for macro and micro-based measures, as compared to the singleton LP-based method, PSt. However, it has statistically significantly worse ranks than the best-ranked methods, similar to PSt. The average rank diagrams show that LP-based methods, in general, are not competitive on ranking-based performance measures.

BPNN has the weakest ranking across the measures — it is often statistically significantly different from the best-ranked method and it is only ranked as not statistically significantly different to the best method in two out of 18 predictive performance measures. These results point out that BPNN is very sensitive to its architectural design and parameter settings. Finding the best configuration for a BPNN, for a given problem is a computationally expensive challenge on its (Elsken et al., 2019). Moreover, this observation follows the general observation for single target tasks, for tabular data, where ensemble methods have the competitive edge (Lundberg et al., 2020).

5.4. MLC methods efficiency comparison

We focus the discussion now on comparing the efficiency of the selected best-performing methods in terms of training and testing times. Fig. 10 shows that RFPCT is the most efficient method — it learns a predictive model fastest and makes predictions the fastest. It is then followed by RFDTBR. The differences in the efficiency of these two methods as compared to the rest of the methods are statistically significant for both the training time (except for the PSt method) and the testing time.

We next consider the speed up of RFPCT (as the top efficient method) relative to the remaining methods across all datasets[1]. In a nutshell, RFDTBR is slower than RFPCT $\sim$ 2.5 times, AdaBoost.MH $\sim$ 28.1 times, PSt $\sim$ 29.6 times, TREMLC $\sim$ 48.1, EBRJ48 $\sim$ 61 times, BPNN $\sim$ 63 times and ECCJ48 $\sim$ 76.7 times. We believe that the difference in efficiency between the two top-ranked methods (RFPCT and RFDTBR) is because RFPCT typically learns shallower/smaller trees (Kocev et al., 2013). Notwithstanding this difference, the comparison of efficiency identifies RFPCT and RFDTBR as the most efficient MLC methods.

6. Conclusion

In this paper, we present the most comprehensive comparative study of MLC methods to date. It gives an in-depth theoretical and empirical analysis of a variety of MLC methods. Considering the ever-increasing interest in MLC by the research community and its increased practical relevance, this study maps the landscape of MLC methods, and provides guidelines for practitioners on the usage of the MLC methods, on

[1] The relative speedup is calculated as the average over the datasets of the ratio of the times needed to learn a model by the other methods and by using RFPCTs.

(a) Train time (b) Test time

Fig. 10. Average rank diagrams comparing the efficiency (running times) of the best performing methods. The performance of the methods connected with a line is not statistically significantly different.

selecting the best baselines when proposing a new method, or selecting the first methods to try out on new MLC datasets.

The theoretical analysis of the MLC methods focuses on aspects covering different viewpoints of the methods, such as (1) detailing the inner working procedures of the methods; (2) stressing their strengths and weaknesses; (3) discussing their potential to address specific properties of the MLC task, i.e., exploit the potential label dependencies and handle the high-dimensionality of the label space; and (4) analysis of the computational cost for training a predictive model and making a prediction using it. We divide the methods into two groups: problem transformation and algorithm adaptation. While the former group of methods decomposes the MLC problem into a simpler problem(s) that are addressed with standard machine learning methods, the latter group of methods holistically addresses the MLC problem — it learns a model predicting all labels simultaneously.

Our empirical study of the methods is by far the *largest empirical study for MLC methods to date*: It considers 26 MLC methods learning predictive models for 42 datasets, and evaluating them by 18 predictive performance measures and two efficiency criteria. The datasets stem from various domains, including text (news, reports), medicine, multimedia (images and audio), bioinformatics, biology and chemistry. The 18 predictive evaluation criteria provide a whole range of viewpoints on the performance of the MLC methods, including their capability of predicting bi-partitions (per example and label), label ranking, and the independence of the predictions regarding the threshold.

Regarding the experimental design, we adhere to the literature recognized standards for conducting large experimental studies in the machine learning community. It includes time-constrained hyperparameter optimization of the methods' parameters on a sub-sampled portion of the datasets. The parameterization is performed using literature recognized values for the parameters. For analyses of the results, we use the Friedman and Nemenyi statistical tests and present their outcomes using average ranking diagrams.

We analyse and discuss the results of the experiments in detail, first separately for problem transformation and algorithm adaptation methods, and then on a selection of the best-performing methods from both groups. Based on the analysis of the performance within each group, we selected 8 best performing methods (RFDTBR, AdaBoost.MH, ECCJ48, TREMLC, PSt and EBRJ48 among problem transformation, and RFPCT and BPNN among algorithm adaptation methods). We then compare these to identify an even more compact set of methods as best performing.

The evaluation outlines RFPCT, RFDTBR, EBRJ48, AdaBoost.MH and ECCJ48 as best performing, considering the 18 evaluation measures. These methods have their strengths and weaknesses and should be selected based on the context of use. For example, ECCJ48 is very strong on recall-based measures and weak on precision-based measures — this is reversed for AdaBoost.MH. Notwithstanding, RFPCT and RFDTBR are the top-performing methods (having the top-ranked positions across the majority of the evaluation measures) as well as the most efficient MLC methods (within the selected best performing group of methods).

Being very comprehensive in terms of MLC methods, datasets and evaluation measures, this study opens several avenues for further research and exploration. To begin with, while it is important to provide different viewpoints by using different evaluation measures, this makes it difficult to select an optimal method. To alleviate this problem, we aim to use our empirical results to study the evaluation measures and identify relevant relationships between them. Furthermore, the results of the large set of experiments will allow us to relate the data set properties and the performance of the methods in a meta learning study and investigate the influence of dataset properties on predictive performance. For this purpose, we will first describe the MLC datasets with features that describe their specific MLC task properties and then use these features to learn meta models. Next, we will investigate further the potential of deep learning methods for MLC, with a special focus on methods for transfer learning as well as different data augmentation strategies. Finally, considering that we store the actual prediction scores of the models, we can further experiment with thresholding functions that separate the relevant from irrelevant labels. We will analyse different MLC thresholding techniques, to improve existing or design novel thresholding methods.

CRediT authorship contribution statement

Jasmin Bogatinovski: Conceived and designed the experimental study, Performed initial analysis and visualization of the results, Prepared the initial draft of the manuscript, Collected the datasets, Implementations of the methods, Implemented the experimental framework, Performed the experiments, All the authors contributed in verifying the results, All authors participated in the manuscript revision. **Ljupčo Todorovski:** Design of the study, Analysis of the results and reviewed and edited the manuscript, All the authors contributed in verifying the results, All authors participated in the manuscript revision. **Sašo Džeroski:** Design of the study, Analysis of the results and reviewed and edited the manuscript, All the authors contributed in verifying the results, All authors participated in the manuscript revision. **Dragi Kocev:** Conceived and designed the experimental study, Performed initial analysis and visualization of the results, Prepared the initial draft of the manuscript, Supervised the work, All the authors contributed in verifying the results, All authors participated in the manuscript revision.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.eswa.2022.117215.

Acknowledgments

We acknowledge the financial support of the Slovenian Research Agency via the grants P2-0103, J2-9230, N2-0128, P5-0093, and V5-1930 and the European Commission through the project TAILOR - Foundations of Trustworthy AI - Integrating Reasoning, Learning and Optimization (grant No. 952215). The computational experiments presented here were executed on a computing infrastructure from the Slovenian Grid (SLING) initiative, and we thank the administrators Barbara Krašovec and Janez Srakar for their assistance. All authors read and approved the final manuscript.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.eswa.2022.117215.

References

Alvares-Cherman, E., Metz, J., & Monard, M. C. (2012). Incorporating label dependency into the binary relevance framework for multi-label classification. *Expert Systems with Applications*, *39*, 1647–1655. http://dx.doi.org/10.1016/j.eswa.2011.06.056.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, *60*, 503–515. http://dx.doi.org/10.1073/pnas.38.8.716.

Blockeel, H., Džeroski, S., & Grbović, J. (1999). Simultaneous prediction of multiple chemical parameters of river water quality with TILDE. In *Principles of data mining and knowledge discovery* (pp. 32–40). Berlin, Heidelberg: Springer.

Blockeel, H., Raedt, L. D., & Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of the 15th international conference on machine learning* (pp. 55–63). San Francisco, CA, USA: Morgan Kaufmann Publishers.

Boutell, M., Luo, J., Shen, X., & Brown, C. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*, 1757–1771. http://dx.doi.org/10.1016/j.patcog.2004.03.009.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32. http://dx.doi.org/10.1023/A:1010933404324.

Briggs, F., Lakshminarayanan, B., Neal, L., Fern, X. Z., Raich, R., Hadley, S., Hadley, A., & Betts, M. (2012). Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach. *The Journal of the Acoustical Society of America*, *131*, 4640–4650. http://dx.doi.org/10.1121/1.4707424.

Brinker, K. (2006). On active learning in multi-label classification. In *From data and information analysis to knowledge engineering* (pp. 206–213). Berlin, Heidelberg: Springer.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., Vanderplas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. arxiv. http://arxiv.org/abs/arXiv:1309.0238 [arXiv:arXiv:1309.0238].

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on machine learning* (pp. 161–168). New York, USA: ACM.

Chen, W. J., Shao, Y. H., Li, C. N., & Deng, N. Y. (2016). MLTSVM: a novel twin support vector machine to multi-label learning. *Pattern Recognition*, *52*, 61–74. http://dx.doi.org/10.1016/j.patcog.2015.10.008.

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, *20*, 1–21, http://jmlr.org/papers/v20/18-598.html.

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139. http://dx.doi.org/10.1006/jcss.1997.1504.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of *m* rankings. *The Annals of Mathematical Statistics*, *11*, 86–92. http://dx.doi.org/10.1214/aoms/1177731944.

Gibaja, E., & Ventura, S. (2015). A tutorial on multilabel learning. *ACM Computing Surveys*, *47*, 52:1–52:38. http://dx.doi.org/10.1145/2716262.

Gorishniy, Y., Rubachev, I., Khrulkov, V., & Babenko, A. (2021). Revisiting deep learning models for tabular data. In *Proceedings of the 35-th conference on advances in neural information processing systems*. Curran Associates, Inc.

Grady, L., & Funka-Lea, G. (2004). Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials. In *Computer vision and mathematical methods in medical and biomedical image analysis* (pp. 230–245). Berlin, Heidelberg: Springer Berlin Heidelberg.

Guo, Y., & Gu, S. (2011). Multi-label classification using conditional dependency networks. In *Proceedings of the 22nd international joint conference on artificial intelligence* (pp. 1300–1305). Barcelona, Spain: AAAI Press.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., & Kadie, C. (2001). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, *1*, 49–75.

Herrera, F., Rivera, A. J., del Jesus, M. J., & Charte, F. (2016). *Multilabel classification: problem analysis, metrics and techniques*. Springer Cham, Switzerland: Springer.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computing*, *14*, 1771–1800. http://dx.doi.org/10.1162/089976602760128018.

Hinton, G., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*, 504–507. http://dx.doi.org/10.1126/science.1127647.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 832–844. http://dx.doi.org/10.1109/34.709601.

Huang, K. H., & Lin, H. T. (2017). Cost-sensitive label embedding for multi-label classification. *Machine Learning*, *106*, 1725–1746. http://dx.doi.org/10.1007/s10994-017-5659-z.

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automatic machine learning: methods, systems, challenges*. Berlin, Heilderberg: Springer, (Chapter 2).

Iman, R., & Davenport, J. (1980). Approximations of the critical region of the friedman statistic. *Communications in Statistics-theory and Methods*, *9*, 571–595. http://dx.doi.org/10.1080/03610928008827904.

J., Fürnkranz, Hüllermeier, E., Loza Mencía, E., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, *73*, 133–153. http://dx.doi.org/10.1007/s10994-008-5064-8.

Jain, H., Prabhu, Y., & Varma, M. (2016). Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 935–944). Association for Computing Machinery.

Jayadeva, Khemchandani R., & Chandra, S. (2007). Twin support vector machines for pattern classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*, 905–910.

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 discovery challenge*.

Kira, K., & Rendell, L. (1992). The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the 10th national conference on artificial intelligence* (pp. 129–134). San Jose, California: AAAI Press.

Kocev, D. (2011). *Ensembles for predicting structured outputs*. (Ph.D. thesis), Ljubljana, Slovenia: Jožef Stefan International Postgraduate School.

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, *46*, 817–833. http://dx.doi.org/10.1016/j.patcog.2012.09.023.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, *29*, 1–27. http://dx.doi.org/10.1007/BF02289565.

Kurtzer, G. M., Vanessa, S., & Bauer, W. M. (2017). Singularity, scientific containers for mobility of compute. *PLoS One*, *12*(5), Article e0177459. http://dx.doi.org/10.1371/journal.pone.0177459.

Liu, W., Shen, X., Wang, H., & Tsang, I. W. (2020). The emerging trends of multi-label learning. arXiv:2011.11197 (preprint).

Lundberg, M., S, Erion, G., & Chen, H. e. a. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, *2*, 56–67. http://dx.doi.org/10.1038/s42256-019-0138-9.

Madjarov, G., Kocev, D., Gjorgjevikj, D., & Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, *45*, 3084–3104. http://dx.doi.org/10.1016/j.patcog.2012.03.004.

Moyano, J. M., Galindo, E. L. G., Cios, K. J., & Ventura, S. (2018). Review of ensembles of multi-label classifiers: Models. *Experimental Study and Prospects Information Fusion*, *44*, 33–45. http://dx.doi.org/10.1016/j.inffus.2017.12.001G.

Nasierding, G., Kouzani, A., & Tsoumakas, G. (2010). A triple-random ensemble classification method for mining multi-label data. In *IEEE International conference on data mining workshops* (pp. 49–56). Washington, DC, USA: IEEE Computer Society.

Nemenyi, P. (1963). *Distribution-free Multiple Comparisons*. (Ph.D. thesis), Princeton, USA: Princeton University.

Pearl, J. (1988). Markov and bayesian networks: two graphical representations of probabilistic knowledge. In J. Pearl (Ed.), *Probabilistic reasoning in intelligent systems* (pp. 77–141). San Francisco (CA): Morgan Kaufman Publishers.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106. http://dx.doi.org/10.1007/BF00116251.

Ratnarajah, N., & Qiu, A. (2014). Multi-label segmentation of white matter structures: Application to neonatal brains. *NeuroImage*, *102*, 913–922. http://dx.doi.org/10.1016/j.neuroimage.2014.08.001.

Read, J. (2010). *Scalable Multi-Label Classification*. (Ph.D. thesis), Hamilton, New Zeland: University of Waikato.

Read, J., & Perez-Cruz, F. (2014). Deep learning for multi-label classification. http://arxiv.org/abs/arXiv:1502.05988 arXiv:1502.05988 (pre-print).

Read, J., Pfahringer, B., & Holmes, G. (2008). Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th IEEE international conference on data mining* (pp. 995–1000). Washington, DC, USA: IEEE Computer Society.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(333), http://dx.doi.org/10.1007/s10994-011-5256-5.

Read, J., Reutemann, P., Pfahringer, B., & Holmes, G. (2016). MEKA: a multi-label/multi-target extension to WEKA. *Journal of Machine Learning Research*, *17*, 1–5.

Reem, A.-O., Flach, P., & Meelis, K. (2014). Multi-label classification: A comparative study on threshold selection method. In *1st International workshop on learning over multiple contexts*.

Rivolli, A., Read, J., Soares, C., Pfahringer, B., & de Carvalho, A. C. P. L. F. (2020). An empirical analysis of binary transformation strategies and base algorithms for multi-label learning. *Machine Learning*, *109*.

Rokach, L., Schclar, A., & Itach, E. (2014). Ensemble methods for multi-label classification. *Expert Systems with Applications*, *41*, 7507–7523. http://dx.doi.org/10.1016/j.eswa.2014.06.015.

Ruiz, E. V. (1986). An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Letters*, *4*, 145–157. http://dx.doi.org/10.1016/0167-8655(86)90013-9.

de Sá, A. G. C., Pappa, G. L., & Freitas, A. (2018). Multi-label classification search space in the MEKA software. arXiv:1811.11353.

Sapozhnikova, E. (2009). ART-based neural networks for multi-label classification. In *Advances in intelligent data analysis VIII* (pp. 167–177). Berlin, Heidelberg: Springer.

Schapire, R., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*, 297–336. http://dx.doi.org/10.1023/A:1007614523901.

Schapire, R., & Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, *39*, 135–168. http://dx.doi.org/10.1023/A:1007649029923.

Schulz, A., Loza Mencía, E., & Schmidt, B. (2016). A rapid-prototyping framework for extracting small-scale incident-related information in microblogs: Application of multi-label classification on tweets. *Information Systems*, *57*, 88–110. http://dx.doi.org/10.1016/j.is.2015.10.010.

Sechidis, K., Tsoumakas, G., & Vlahavas, I. (2011). On the stratification of multi-label data. In *Machine learning and knowledge discovery in databases* (pp. 145–158). Berlin, Heidelberg: Springer.

Stepišnik, T., & Kocev, D. (2020). Hyperbolic embeddings for hierarchical multi-label classification. In *International symposium on methodologies for intelligent systems* (pp. 66–76). Springer.

Szymański, P., & Kajdanowicz, T. (2019). A scikit-based python environment for performing multi-label classification. *Journal of Machine Learning Research*, *20*, 209–230.

Tan, A. H. (1995). Adaptive resonance associative map. *Neural Networks*, *8*, 437–446. http://dx.doi.org/10.1016/0893-6080(94)00092-Z.

Tenenboim, L., Rokach, L., & Shapira, B. (2009). Multi-label classification by analyzing labels dependencies. In *Proceedings of the 1st international workshop on learning from multi-label data* (pp. 117–131).

Tenenboim, L., Rokach, L., & Shapira, B. (2010). Identification of label dependencies for multi-label classification. In *2nd International workshop on learning from multi-label data* (pp. 53–60).

Tsoumakas, G., Anastasios, D., Eleftherios, S., Vasileios, M., Ioannis, K., & Vlahavas, I. P. (2009). Correlation-based pruning of stacked binary relevance models for multi-label learning. In *1st International workshop on learning from multi-label data* (pp. 101–116).

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, *2007*, 1–13.

Tsoumakas, G., Katakis, I., & Vlahavas, I. P. (2008). Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of the workshop on mining multidimensional data at ECML/PKDD 2008* (pp. 53–59).

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2011). Random K-labelsets for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, *23*, 1079–1089. http://dx.doi.org/10.1109/TKDE.2010.164.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., & Vlahavas, I. (2011). Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, *12*, 2411–2414.

Wang, H., Li, Z., Huang, J., Hui, P., Liu, W., Hu, T., & Chen, G. (2020). Collaboration based multi-label propagation for fraud detection. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence* (pp. 2477–2483). International Joint Conferences on Artificial Intelligence Organization.

Xu, J., Liu, J., Yin, J., & Sun, C. (2016). A multi-label feature extraction algorithm via maximizing feature variance and feature-label dependence simultaneously. *Knowledge-Based Systems*, *98*, 172–184. http://dx.doi.org/10.1016/j.knosys.2016.01.032.

Zhang, M.-L., Yu-Kun, L., Xu-Ying, L., & Geng, X. (2018). Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, *12*, 191–202. http://dx.doi.org/10.1007/s11704-017-7031-7.

Zhang, M.-L., & Zhou, Z.-H. (2005). A k-nearest neighbor based algorithm for multi-label classification. In *IEEE international conference on granular computing* (pp. 718–721). Washington, DC, USA: IEEE.

Zhang, M.-L., & Zhou, Z.-H. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, *18*, 1338–1351. http://dx.doi.org/10.1109/TKDE.2006.162.

Zhang, M. L., & Zhou, Z. H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*, 1819–1837. http://dx.doi.org/10.1109/TKDE.2013.39.