Explainable Model-specific Algorithm Selection for Multi-Label Classification

Ana Kostovska, Carola Doerr, Saso Dzeroski, Dragi Kocev, Pance Panov, Tome Eftimov

Explainable Model-specific Algorithm Selection for Multi-Label Classification

1st Ana Kostovska
Knowledge Technologies Department,
Jožef Stefan International Postgraduate School
Jožef Stefan Institute
Ljubljana, Slovenia
0000-0002-5983-7169

2nd Carola Doerr
CNRS, LIP6
Sorbonne Université
Paris, France
0000-0002-4981-3227

3rd Sašo Džeroski
Knowledge Technologies Department
Jožef Stefan Institute
Ljubljana, Slovenia
0000-0003-2363-712X

4th Dragi Kocev
Knowledge Technologies Department
Jožef Stefan Institute
Ljubljana, Slovenia
0000-0003-0687-0878

5th Panče Panov
Knowledge Technologies Department
Jožef Stefan Institute
Ljubljana, Slovenia
0000-0002-7685-9140

6th Tome Eftimov
Computer Systems Department
Jožef Stefan Institute
Ljubljana, Slovenia
0000-0001-7330-1902

***Abstract*—Multi-label classification (MLC) is an ML task of predictive modeling in which a data instance can simultaneously belong to multiple classes. MLC is increasingly gaining interest in different application domains such as text mining, computer vision, and bioinformatics. Several MLC algorithms have been proposed in the literature, resulting in a meta-optimization problem that the user needs to address: which MLC approach to select for a given dataset? To address this algorithm selection problem, we investigate in this work the quality of an automated approach that uses characteristics of the datasets – so-called features – and a trained algorithm selector to choose which algorithm to apply for a given task. For our empirical evaluation, we use a portfolio of 38 datasets. We consider eight MLC algorithms, whose quality we evaluate using six different performance metrics. We show that our automated algorithm selector outperforms any of the single MLC algorithms, and this is for all evaluated performance measures. Our selection approach is explainable, a characteristic that we exploit to investigate which meta-features have the largest influence on the decisions made by the algorithm selector. Finally, we also quantify the importance of the most significant meta-features for various domains.**

***Index Terms*—automated algorithm selection, multi-label classification, XAI**

I. Introduction

Multi-label classification (MLC) is a predictive modeling task that involves predicting the presence of multiple class labels which are not mutually exclusive class labels. It is present in many research areas (e.g., text categorization, where documents might be assigned multiple topics simultaneously [1], [2], computer vision [3], [4], and bioinformatics [5], [6]). For solving the MLC task, many different algorithms have been proposed in the literature. Previous experiments have demonstrated that there is no single algorithm that performs best on all possible datasets on a given machine learning (ML) learning task. The diversity of different MLC algorithms, therefore, poses a new meta-optimization problem: which algorithm to select when a new dataset becomes available to maximize the performance metric under consideration? This meta-optimization problem is known as the *algorithm selection* (AS) problem.

AS is a task of meta-learning [7], where an ML model is learned to predict the best performing algorithm for new datasets (or even data instances). This requires a set of benchmark datasets that will be used for training the AS model, often referred to as a dataset portfolio; dataset characteristics presented in the same vector space for all datasets defined meta-features (this is known as meta-representation or representation of the problem landscape); and data on the performance achieved by a set of algorithms out of which the best algorithm will be selected for each dataset, often referred to as algorithm portfolio.

The AS task has already been explored for different learning tasks, based on the availability of the above-mentioned resources for the learning task: the dataset portfolio, a meta-representation for the datasets, and the performance data for an algorithm portfolio. There are several frameworks, such as OpenML [8], ASlib [9] and OPTION [10] that support the development of AS for different ML and optimization tasks.

OpenML [8] is an open platform for sharing datasets, algorithms, and experimental results that can further be used for meta-learning about ML tasks. ASlib [9] is a repository that contains a large number of ML datasets, together with data about the performance of the algorithms achieved on datasets especially designed and stored for performing AS. OPTION [10] is an ontology developed to store and make experimental data from single-objective optimization interoperable. Currently, it contains meta-features for describing

The authors acknowledge the support of the Slovenian Research Agency through research core grants No. P2-0103 and P2-0098, project grants No. J2-9230 and N2-0239, the young researcher grant No. PR-09773 to AK, as well as the EC through grant No. 952215 (TAILOR). Our work is also supported by the CNRS INS2I project RandSearch and by the ANR T-ERC project VARIATION (ANR-22-ERCS-0003).

single-objective optimization problem instances coming from the COCO benchmark suite [11], together with performance data obtained from COCO, IOHprofiler [12], and the Meta framework Nevergrad [13]. It is also worth mentioning that the DACBench library [14], where instead of AS, data for performing algorithm configuration (AC) (i.e., selecting the best hyperparameters for a given algorithm) is stored for evolutionary computation, AI planning, and deep learning tasks. The same data can be also used in an AS learning scenario, where different hyper-parameters for the same algorithm will be treated as different algorithms.

The existence of the above-mentioned and similar libraries has led to several studies in AS [15]. Tornede et al. [16] presented a general framework for performing AS, especially focusing on meta-learning and ensemble learning methods. Shawkat and Smith [17] investigate AS in a classification learning scenario involving 8 different classifiers and 100 benchmark datasets. Cohen-Shapira and Rokach [18] presented an approach for AS in clustering by using supervised graph embeddings, 210 clustering datasets, and 17 clustering algorithms. Kotthoff [19] provided an overview of different methods that can be used in AS for combinatorial optimization problems. Jankovic et al. [20] investigated per-problem and per-instance AS for single-objective continuous optimization problems by using a meta-representation calculated from trajectory data of the algorithms and their global state variables that are changing during the optimization process. Kostovska et al. [21] investigated a per-run AS for single-objective continuous optimization using trajectory data of the optimization algorithms and explored the transferability of the AS results across different dataset portfolios or benchmark suites.

However, AS for MLC is still largely unexplored. In recent work, Bogatinovski et al. [22] presented a study of automated algorithm performance prediction for MLC algorithms, the performance of 26 MLC algorithms was considered on a set of 40 MLC datasets described with 50 meta-features. From this data, they trained a multi-target regressor (with predictive clustering trees) to predict the performance of the algorithms with regard to several performance metrics. The main findings describe the importance of the meta-features for algorithm performance prediction with predictive clustering trees. This study motivated us to go one step beyond and instead of automated algorithm performance prediction to perform AS for MLC, i.e., select the best performing algorithm for each dataset separately. In addition, we also utilize explanation techniques to explain the AS decisions.

The contributions of our paper can be summarized as follows:

- Using ML, we trained MLC algorithm selector for six different evaluation measures on a portfolio comprised of eight different MLC algorithms. The results show that the algorithm selector provides better results across the different evaluation metrics when compared to the best single algorithms, confirming our hypothesis that state-of-the-art algorithm selection techniques provide a promising alternative to a manual choice.
- To provide explanations for the choices made by the algorithm selector, we investigate the impact of the MLC meta-features on the automatic algorithm performance prediction for each algorithm separately and combine the contribution of the MLC meta-features based on the selected best algorithms.
- We present domain-specific explanations for the algorithm selector to explore the differences in the landscape of MLC meta-features when training an algorithm selector for each domain separately.

Outline of the Paper: In Section II, we introduce the problem of algorithm selection and provide a background of the MLC meta-features used to characterize the landscape of MLC datasets. Section III provides details of the experimental setup and the quality estimation of the MLC selector. In Section IV, we discuss the results of our empirical evaluation. The domain-specific explanations of the algorithm selector are given in Section V. Finally, Section VI concludes our study by highlighting the main contributions and discussing possible directions for future work.

Availability of data and code for reproducibility: Source code, performance and landscape data, results, and figures produced for our study are available in the Zenodo repository accompanying this paper [23]. Please note that the repository also contains additional information, not described here for reasons of space limitation.

II. Background

A. Algorithm Selection

Algorithm selection is a meta-algorithmic design technique that addresses the problem of choosing a well-performing algorithm from a finite, performance-complementary algorithm portfolio, on a per-instance basis. There are different strategies for building an algorithm selector (e.g., parallel algorithm portfolios, algorithm schedules, or ML-based automated algorithm selection). In ML, algorithm selection can be treated as a classification or regression task. When treated as multi-class classification, the input is the landscape features (or meta-features) of the dataset instance, and the output (or target) is the best-performing algorithm out of an algorithm portfolio. Alternatively, pairwise classification can be employed, where the relative performance of pairs of algorithms is compared and the algorithm with the most "wins" is selected [24]. When treated as a regression task, separate regression models for performance prediction are trained for each algorithm in the portfolio. The algorithm with the best-predicted performance is selected.

To estimate the quality (or the performance gain) of an algorithm selector, two baselines are commonly used in the literature [15]: (i) the performance of the algorithm that maximizes mean performance on the dataset portfolio (called single best solver (SBS)); (ii) the oracle performance or the virtual best solver (VBS) – a hypothetical, perfect selector that chooses the best performing algorithm for each dataset instance.

B. MLC Meta-features

Meta-learning is a sub-field of ML concerned with learning from past experiences i.e., data on past machine learning experiments, commonly referred to as meta-data [25]. The main goal of meta-learning is to enable the automation of parts of the machine learning pipeline, i.e., the selection of machine learning algorithms that are most suitable for a given dataset. The meta-data usually includes dataset characteristics (or meta-features) that are relevant to the learning task. These meta-features allow for grouping the datasets according to their similar characteristics, which can be used for transferring knowledge from one dataset to other datasets in the same group.

Defining a proper set of meta-features for specific learning tasks has been a question of interest for data scientists. Moyano et al. [26] defined a list of meta-features specific for multi-label classification datasets, categorized into five meta-feature groups: (1) dimensionality, e.g., number of features, number of labels, number of instances; (2) label distribution, e.g., frequency, cardinality and density of labels; (3) label imbalance, e.g., mean of inter-class imbalance ratio; (4) labels relationship, e.g., proportion of distinct labelsets; and (5) attribute metrics, e.g., number of binary attributes. In this study, we use these meta-features to represent the landscape of MLC datasets.

III. Experimental Setup

This section describes the experimental setup, which includes the description of the dataset portfolio, the landscape data associated with the datasets, the MLC algorithm portfolio, and the performance data. Following that, we present details on how the regression models that are the basis for building the algorithm selector are trained. Finally, we describe how we build the algorithm selector.

A. Dataset Portfolio and Landscape Data

The dataset portfolio consists of 38 MLC datasets that have previously been used in various studies for benchmarking MLC methods. The datasets come from five different application domains (i.e., text, multimedia, bioinformatics, medical, and chemistry). Further, the portfolio covers datasets with a diverse number of labels (4-274), data instances (139-17190), and descriptive features (33-49060). Our ML pipeline for building the algorithm selector relies on having meta-descriptors (or meta-features) of the MLC datasets in order to train the regression models. For that purpose, we reuse a set of 63 MLC meta-features that have already been proposed in the literature [27] and are shown to provide promising results when predicting algorithm performance [22]. We reduce the initial set of 63 meta-features to 17 by calculating Pearson correlation pairwise and removing one of the features in the pair with a correlation larger than 0.75. All datasets and the related MLC meta-features have been downloaded from the publicly available MLC data catalogue [28].

TABLE I: RF hyperparameter names and their corresponding values considered in the grid search.

Hyperparameter	Search space
n_estimators	$[50, 100]$
max_features	[AUTO, SQRT, LOG2]
max_depth	$[4, 8, 15,$ NONE$]$
min_samples_split	$[2, 5, 10]$

B. MLC Algorithm Portfolio and Performance Data

The performance data we use here comes from a comprehensive comparative study of MLC algorithms [29]. The study evaluates 26 MLC algorithms over 42 datasets (including the 38 datasets mentioned above) using 18 predictive performance evaluation measures and two efficiency performance measures.

In this study, we are only concerned with selecting the algorithm that performs best and we ignore the efficiency component, i.e., we ignore the two efficiency performance measures. Since the evaluation measures are correlated, we remove the evaluation measures with a Pearson correlation larger than 0.90. This leaves us with six evaluation measures: average precision, macro F1, one error, AUROC, Hamming loss, and micro precision.

Next, to create a portfolio of MLC algorithms with complementarity in their performance, for each combination of dataset and evaluation measure, we count the number of times a given algorithm is performing the best. In the portfolio, we include the algorithms that performed the best on at least eight out of the 38 datasets for any of the six evaluation metrics. Following this criteria, the final MLC method portfolio consists of eight algorithms (i.e., AdaBoost [30], Calibrated Label Ranking (CLR) [31], [32], Deep Belief Networks (DEEP4 version) [33], Hierarchy of Multi-label Classifiers (HOMER) [34], Multi-label Adaptive Resonance Associative Map (MLARM) [35], The method of Pruned Sets (PSt) [36], Binary relevance with random forest (RFDTBR) [37], and Random Forest of Predictive Clustering Trees (RFPCT) [**?**], [38]).

C. Regression Models

For training the regression models we considered two scenarios:

- Single target regression (STR) – we train a separate regression model for each evaluation metric per MLC algorithm. That leaves us with 48 different regression models (8 MLC methods $\times$ 6 evaluation metrics).
- Multi-target regression (MTR) – we train one regression model per MLC method where we simultaneously try to predict the performance of the method according to the six evaluation metrics. In this scenario, we train eight different regression models since we have an MLC method portfolio of size eight.

The regression models are built with the Random Forest (RF) algorithm as implemented in the Python package `scikit-learn` [39]. The RF hyperparameters are tuned using the grid search methodology. We tune four different RF

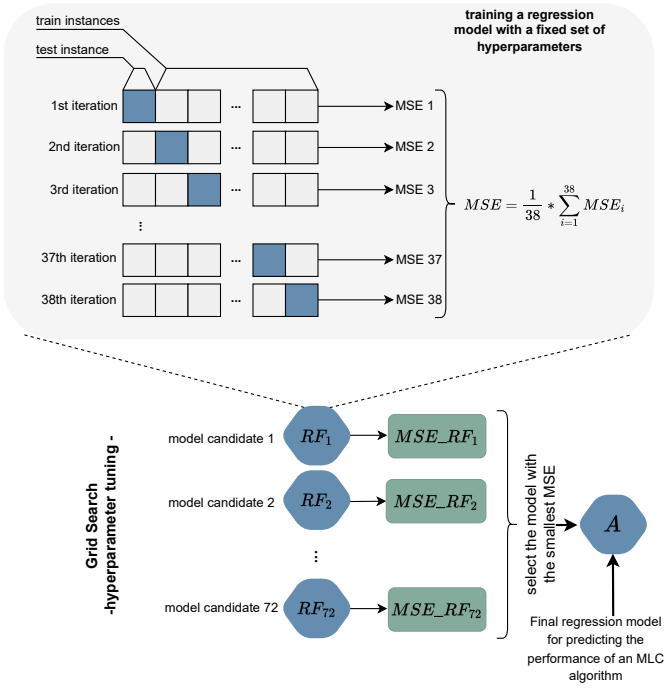


Fig. 1: An illustration of the process of training a regression model for predicting the performance for an MLC algorithm.

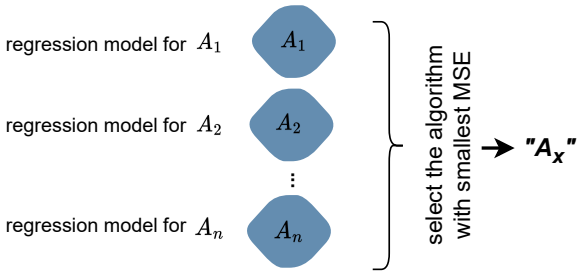


Fig. 2: An illustration of the process of selecting the best performing algorithm for an algorithm portfolio of size N.

hyperparameters: (1) *n_estimators* – the number of trees in the random forest; (2) *max_depth* – the maximum depth of the trees ; (3) *max_features* – the number of features used for making the best split ; and (4) *min_samples_split* – the minimum number of samples required for splitting an internal node. Thus, for each trained model, we consider 72 different model candidates. The full list of tuned hyperparameters and the corresponding search spaces are given in Table I.

The regression models are evaluated with the *leave-one-instance-out* strategy, where an instance is one MLC dataset. Since we have 38 MLC datasets, we perform training 38 times where we hold one instance for testing and 37 for training. Finally, we compute the mean squared error and average it across all test instances (see Figure 1.

D. Construction and Quality Estimation of the MLC Algorithm Selector

After training the regression models for performance prediction of each of the MLC algorithms separately, we select the best performing algorithm (the one with the best-predicted performance) on every dataset, w.r.t. each evaluation metric (see Figure 2). Note that the selection process is the same for the single and multi-target settings and the two settings only differ in the prediction step.

To estimate the quality (or the performance gain) of the algorithm selector, we take two approaches: (i) we treat the algorithm selection as a multi-class classification problem and report the macro f1 score, and (ii) we compute the absolute difference between the target performance value reached by the selected best algorithm and the true best algorithm.

IV. Evaluation Results

We first present the results of the algorithm selector where we estimate its performance on the corresponding multi-class classification problem. Fig. 3 provides the heatmap of the macro F1 scores with each of the MLC algorithms in the portfolio as chosen/predefined as the single best solver and when we take predictions of the two algorithm selectors we build (single and multi-target) for each evaluation metric.

First of all, we should note that the results are highly dependent on the evaluation metric. For example, RFPCT can be considered the single best solver according to the AUROC metric (it has a 0.142 macro F1 score – the highest when compared to the other algorithms in the portfolio). However, for 3 out of the 6 evaluation metrics, it has a macro F1 score of 0, which means that in the available performance data, this algorithm did not perform the best for any of the datasets (for the 3 metrics considered).

Further, the results indicate that there is a performance gain when using our methodology for algorithm selection. For instance, the macro F1 score increases from 0.142 for the single best solver to 0.272/0.288 for the algorithm selector build in the single/multi-target setting, when considering AUROC as an evaluation metric. The single/multi-target algorithm selector achieves the best performance, 0.401/0.471, for the ONE ERROR evaluation metric.

Since the evaluation measures (the multiple targets in our experimental setup) are correlated, we wanted to investigate whether the performance of the selector improves when we train multi-target regression models and make predictions simultaneously for the multiple targets (or evaluation measures). We can observe that for 4 out of the 6 evaluation measures considered in the study, the performance indeed improves. However, the performance slightly drops when considering Hamming loss as the evaluation metric (from 0.395 to 0.368) and there is a significant drop in performance in the case of macro F1 (from 0.396 to 0.108). This could be explained by the fact that Hamming loss has a lower correlation with the other evaluation metrics. However, even though macro F1 has strong a correlation with the other metrics, performance suffers in the multi-target setting. We intend to look into

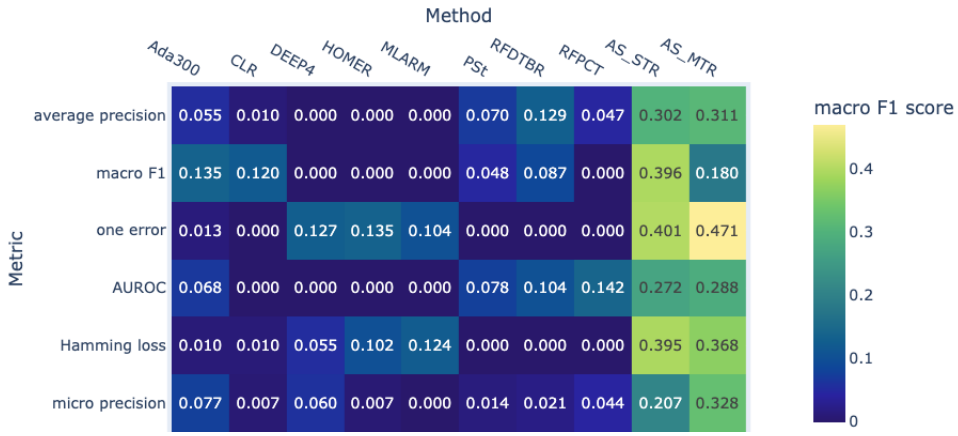


Fig. 3: A heatmap showing the f1-macro of the multi-class classification for each of the methods chosen as single best solvers and for the two algorithm selectors (single and multi-target) across the six different MLC evaluation metrics.

this phenomenon in more detail in subsequent research. All correlation plots can be found at [23].

Treating the algorithm selection as a multi-class classification problem might not give us an accurate estimate of the performance gain. It can happen that the selected algorithm is not the same as the virtual best solver, but has very similar performance. The opposite is also true, the performance difference between the selected and the virtual best solver can be very large. To take into consideration this discrepancy in the target value, we also provide boxplots that depict the absolute difference between the target value reached by the selected best and the virtual best solver (see Fig. 4). Here, we can also observe that the algorithm selector improves performance when comparing it to the single best solvers. However, the performance gain (the gap closed between the VBS and the SBS) largely depends on the evaluation metric. More specifically, for the one error, micro precision, macro F1, and Hamming loss we have a large performance gain, whereas for the AUROC and average precision the performance of the algorithm selector is very similar to the one of RFDTBR and RFPCT taken as single best solvers.

V. Explainable Algorithm Selection

To provide better explanations of the algorithm selection models, we proceed with the calculation of the Shapley values, i.e. the MLC meta-feature importance scores for the AS models. The Shapely values quantify the marginal contribution of the input features on the predictions made by the predictive AS model [40]. However, the calculation of the Shapley values is computationally expensive as it considers each possible combination of features (a power-set of features) to determine their contribution to the prediction process. Therefore, we apply the SHAP (SHapley Additive exPlanations) algorithm [41].

A. General explanations

In order to obtain the Shapley values for the algorithm selector, for each MLC dataset, we first check which is the selected algorithm and then get the Shapley values by using the regression model trained for the selected algorithm. Next, we provide a summary plot that illustrates the positive and negative relationships of the meta-features with the quality of the prediction. Each dot in the plots represents a dataset and the MLC meta-features are listed in descending order of importance. The colors used indicate the magnitude of the MLC meta-feature value (red representing higher values and blue representing lower values). Finally, the position on the horizontal axis presents the impact of the MLC meta-feature value on the prediction of the target. Fig. 5 shows the summary plot for the single target algorithm selector with respect to the *one error* evaluation metric. We can see that *Density*, *Ratio of unconditionally dependent label pairs by chi-square test* and *Mean of mean of numeric attributes*, appear as the top 3 most important meta-features. Because of the limited number of pages available, the summary plots for the other evaluation metrics and for the multi-target algorithm selector are not included here but are available at our Zenodo repository [23].

B. Domain-specific Explanations

Next, we investigate the MLC meta-features' importance in the predictive performance for the algorithm selector at the level of domains from which datasets originate. The 38 MLC datasets taken into account in this study come from 5 distinct application domains, including text (15), multimedia (5), bioinformatics (15), medical (2), and chemistry (1)). In the following analysis, for simplicity, we focus only on the text, multimedia, and bioinformatics domains as they appear as the most frequent domains in our dataset portfolio. To obtain domain-specific explanations, we take the Shapley values of the MLC meta-features, group them by domain and compare the feature importance ranks to see if they vary across the different domains.

Figure 6 depicts Venn diagrams with the top 5 most important MLC meta-features for the text, multimedia and bioinformatics domain w.r.t. the *one error* and *AUROC* evaluation measures. An interesting observation is that the most important MLC meta-features do not overlap for the bioinformatics and

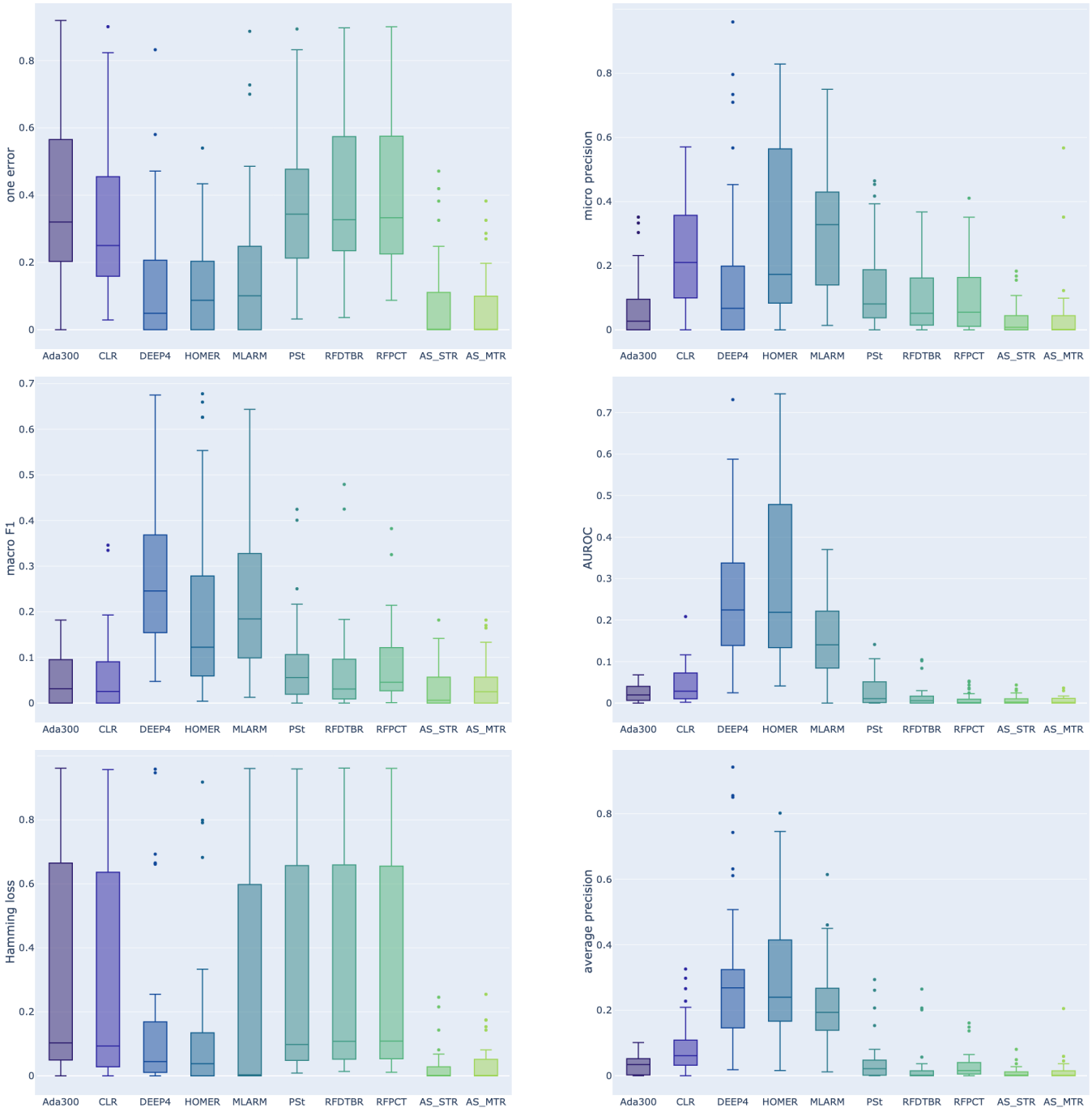


Fig. 4: Boxplots depicting the absolute difference in the achieved target value by the single best solver (SBS) and the target value of the virtual best solver (VBS) for each of the evaluation metrics.

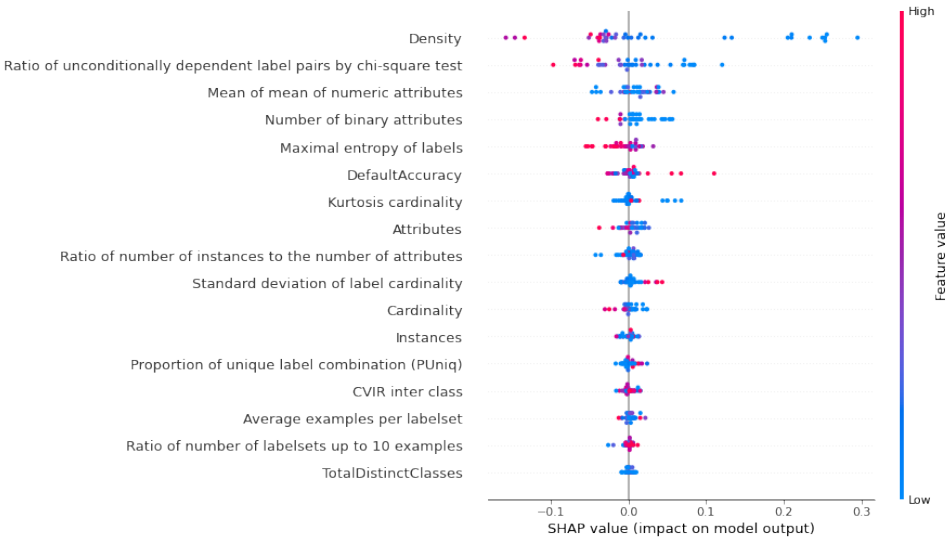
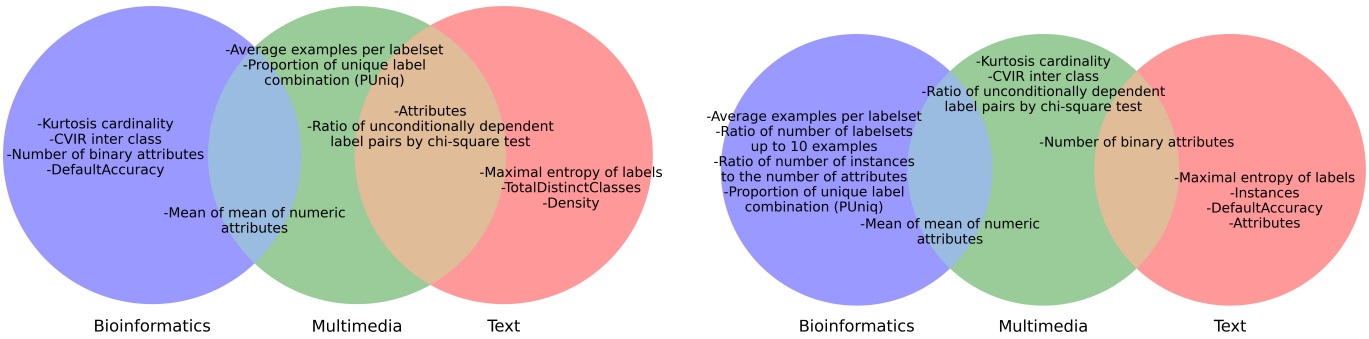


Fig. 5: Summary plot of the Shapley values obtained for the single-target algorithm selector and the *one error* evaluation metric across all datasets.



(a) one error (b) AUROC

Fig. 6: Venn diagrams with the top 5 most important MLC meta-features for the *Bioinformatics*, *Multimedia*, and *Text* domains w.r.t. the (a) *one error* and (b) *AUROC* evaluation measures.

the text domain. The same pattern can be observed for the other evaluation measures as well (check our Zenodo repository [23] that includes all resources and generated figures). This might be due to the nature of the datasets, but it is something we that should be investigated further.

VI. Conclusions

In this paper, we have investigated the potential of automated algorithm selection for the multi-label classification (MLC) learning task. We have trained random forest models in both single- and multi-target scenarios, to predict the performance of the algorithms with regard to six performance metrics. We used 38 datasets, represented with 17 selected meta-features and an algorithm portfolio of eight MLC algorithms. The results of the performance prediction were used to select the best algorithm for each dataset separately. The evaluation results showed that algorithm selection yields performance gains over the scenario, which uses a single MLC algorithm (the one which is, on average, the best for all datasets) regardless of the evaluation measure that is predicted. By combining the explanations obtained for each performance prediction model separately, we have additionally provided explanations about which meta-features most influence the decisions made by the algorithm selector (AS).

We come to the conclusion that there is an overlap between the meta-features from the multimedia datasets and the datasets from the bioinformatics and text domains separately after looking at the explanations provided for various datasets and examining them based on the dataset domain. The most significant meta-features, however, that are used to help the algorithm choose between datasets from bioinformatics and text, do not overlap.

For our future work, we intend to build on this work by incorporating additional datasets for the MLC learning task. We will also look at how the datasets, described by their meta-feature representation are distributed in the landscape and choose datasets that are evenly distributed in the landscape space that will be used to train the AS. By doing this, we hope

to reduce the bias of the AS toward certain types of dataset distribution landscapes. Finally, we consider presenting a more thorough analysis of the justifications offered by the AS and their intersection with various dataset domains and evaluation metrics.

Acknowledgment

The authors acknowledge the support of the Slovenian Research Agency through research core grants No. P2-0103 and P2-0098, project grant No. N2-0239, and the young researcher grant No. PR-09773 to AK, as well as the EC through grant No. 952215 (TAILOR). Our work is also supported by Paris Ile-de-France region, via the DIM RFSI AlgoSelect project and via a SPECIES scholarship for Ana Kostovska.

References

[1] Z. Chen and J. Ren, "Multi-label text classification with latent word-wise label information," *Applied Intelligence*, vol. 51, no. 2, pp. 966–979, 2021.

[2] W.-C. Chang, H.-F. Yu, K. Zhong, Y. Yang, and I. Dhillon, "X-bert: extreme multi-label text classification with using bidirectional encoder representations from transformers," *arXiv preprint arXiv:1905.02331*, 2019.

[3] W. Ge, S. Yang, and Y. Yu, "Multi-evidence filtering and fusion for multi-label classification, object detection and semantic segmentation based on weakly supervised learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1277–1286.

[4] H. Guo, K. Zheng, X. Fan, H. Yu, and S. Wang, "Visual attention consistency under image transforms for multi-label image classification," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 729–739.

[5] Y. Guo, F.-L. Chung, G. Li, and L. Zhang, "Multi-label bioinformatics data classification with ensemble embedded feature selection," *IEEE Access*, vol. 7, pp. 103 863–103 875, 2019.

[6] J.-P. Zhou, L. Chen, and Z.-H. Guo, "iatc-nrakel: an efficient multi-label classifier for recognizing anatomical therapeutic chemical classes of drugs," *Bioinformatics*, vol. 36, no. 5, pp. 1391–1396, 2020.

[7] J. Vanschoren, "Meta-learning," in *Automated machine learning*. Springer, Cham, 2019, pp. 35–61.

[8] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo, "Openml: networked science in machine learning," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 2, pp. 49–60, 2014.

[9] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Fréchette, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney *et al.*, "Aslib: A benchmark library for algorithm selection," *Artificial Intelligence*, vol. 237, pp. 41–58, 2016.

[10] A. Kostovska, D. Vermetten, C. Doerr, S. Džeroski, P. Panov, and T. Eftimov, "Option: optimization algorithm benchmarking ontology," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2021, pp. 239–240.

[11] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, "Coco: A platform for comparing continuous optimizers in a black-box setting," *Optimization Methods and Software*, vol. 36, no. 1, pp. 114–144, 2021.

[12] C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck, "IOHprofiler: A Benchmarking and Profiling Tool for Iterative Optimization Heuristics," *CoRR*, vol. abs/1810.05281, 2018, Up-to-date documentation of IOHprofiler available at https://iohprofiler.github.io/.

[13] J. Rapin and O. Teytaud, "Nevergrad - A gradient-free optimization platform," https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[14] T. Eimer, A. Biedenkapp, M. Reimer, S. Adriaensen, F. Hutter, and M. Lindauer, "Dacbench: A benchmark library for dynamic algorithm configuration," *arXiv preprint arXiv:2105.08541*, 2021.

[15] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, "Automated algorithm selection: Survey and perspectives," *Evolutionary computation*, vol. 27, no. 1, pp. 3–45, 2019.

[16] A. Tornede, L. Gehring, T. Tornede, M. Wever, and E. Hüllermeier, "Algorithm selection on a meta level," *Machine Learning*, pp. 1–34, 2022.

[17] S. Ali and K. A. Smith, "On learning algorithm selection for classification," *Applied Soft Computing*, vol. 6, no. 2, pp. 119–138, 2006.

[18] N. Cohen-Shapira and L. Rokach, "Automatic selection of clustering algorithms using supervised graph embedding," *Information Sciences*, vol. 577, pp. 824–851, 2021.

[19] L. Kotthoff, "Algorithm selection for combinatorial search problems: A survey," in *Data mining and constraint programming*. Springer, 2016, pp. 149–190.

[20] A. Jankovic, T. Eftimov, and C. Doerr, "Towards feature-based performance regression using trajectory data," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2021, pp. 601–617.

[21] A. Kostovska, A. Jankovic, D. Vermetten, J. de Nobel, H. Wang, T. Eftimov, and C. Doerr, "Per-run algorithm selection with warm-starting using trajectory-based features," *arXiv preprint arXiv:2204.09483*, 2022.

[22] J. Bogatinovski, L. Todorovski, S. Džeroski, and D. Kocev, "Explaining the performance of multilabel classification methods with data set properties," *International Journal of Intelligent Systems*, 2022.

[23] A. Kostovska, C. Doerr, S. Džeroski, D. Kocev, P. Panov, and T. Eftimov, "Explainable Model-specific Algorithm Selector for Multi-Label Classification," Jul. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6829671

[24] J. N. v. Rijn, S. M. Abdulrahman, P. Brazdil, and J. Vanschoren, "Fast algorithm selection using learning curves," in *International symposium on intelligent data analysis*. Springer, 2015, pp. 298–309.

[25] P. Brazdil, C. Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008.

[26] J. Moyano, E. Gibaja, and S. Ventura, "MLDA: A tool for analyzing multi-label datasets," *Knowledge-Based Systems*, vol. 121, pp. 1–3, 2017.

[27] J. M. Moyano, E. L. Gibaja, and S. Ventura, "Mlda: A tool for analyzing multi-label datasets," *Knowledge-Based Systems*, vol. 121, pp. 1–3, 2017.

[28] A. Kostovska, J. Bogatinovski, S. Džeroski, D. Kocev, and P. Panov, "A catalogue with semantic annotations makes multilabel datasets FAIR," *Scientific Reports*, vol. 12, no. 1, pp. 1–11, 2022.

[29] J. Bogatinovski, L. Todorovski, S. Džeroski, and D. Kocev, "Comprehensive comparative study of multi-label classification methods," *Expert Systems with Applications*, vol. 203, p. 117215, 2022.

[30] R. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, pp. 135–168, 2000.

[31] K. Brinker, "On active learning in multi-label classification," in *From Data and Information Analysis to Knowledge Engineering*. Berlin, Heidelberg: Springer, 2006, pp. 206–213.

[32] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker, "Multilabel classification via calibrated label ranking," *Machine Learning*, vol. 73, no. 2, pp. 133–153, 2008.

[33] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[34] G. Tsoumakas, I. Katakis, and I. P. Vlahavas, "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*, 2008, pp. 53–59.

[35] E. Sapozhnikova, "ART-based neural networks for multi-label classification," in *Advances in Intelligent Data Analysis VIII*. Berlin, Heidelberg: Springer, 2009, pp. 167–177.

[36] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *Proceedings of the 8th IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 995–1000.

[37] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.

[38] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognition*, vol. 45, pp. 3084 – 3104, 2012.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *JMLR*, vol. 12, pp. 2825–2830, 2011.

[40] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[41] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.