Automatic Ranking of Classification Algorithms

A Comparison of Regression-Based Ranking and Preference Models

Bachelor's Thesis Proposal & Work Plan

Helena Graf

matriculation number: 7011643

hgraf@mail.upb.de

December 1, 2017

Supervisors:
Prof. Dr. Eyke Hüllermeier
Prof. Dr. Axel-Cyrille Ngonga Ngomo

1 Motivation

The potential of big data is evident, and an increasing amount of information is collected and available for analysis - but this potential is not utilized. In a white paper, the International Data Corporation claims that in 2012, out of the 2.8 zettabytes (ZB) of available data only 3% were tagged as to enable further processing, and only 0.5% were analyzed [6]. A follow-up paper in 2017 projects that in 2025, 15% of the estimated 163ZB of global data will be tagged, and approximately 3% analyzed [7]. While this is more optimistic, it still shows that there is a huge gap between the amount of data that could potentially be used and the amount of data actually available. This indicates that the demand of data to be analyzed cannot be covered by data scientists alone, and thus calls for automation of the process in a way that not much expertise in the field of machine learning is needed to gain insights about the collected data.

One of the most prominent machine learning tasks is classification: A class is assigned to an instance, for example clients of a bank may be either deemed creditworthy or not, based on factors like other existing credits or the job of the client. But selecting a fitting classifier for a new data set is difficult, since algorithm performances can vary substantially among data sets, and it is not feasible to simply apply a large number of them to empirically find a good match. For example, on a data set about the electricity prices in the Australian state New South Wales [9], the predictive accuracy for the Multilayer Perceptron[1] is 0.7887 [3]. The predictive accuracy of the Random Forest[2] algorithm on the same data set is 0.9236 [2], a much higher value. On a different data set, with the topic of vehicle silhouettes [13], we get a predictive accuracy of 0.7979 for the Multilayer Perceptron [4], and 0.7518 for Random Forests [1], showing an advantage of the former on this data set[3]. So in each case, one would have picked a different algorithm in order to achieve the best results. In general, this means that for a different data set, a different algorithm might yield the best performance.

Since there is no one best classifier for all data sets, it is likely that how well a classifier performs on a given data set is dependent on properties of the data set, at least to some degree. Combined with the need for automated machine learning, this calls for an approach that considers past performances of classifiers for data sets in relation to properties of these data sets to automatically suggest well-performing classifiers for a new problem.

[1] With standart hyperparameters (L:0.3,M:0.2,N:500,V:0,S:0,E:20,H:a).
[2] With standart hyperparameters (P:100,I:100,num-slots:1,K:0,M:1.0,V:0.001,S:1).
[3] Hyperparameters as above.

2 Related Work

The demand for aid in the process of selecting an algorithm has already led to the development of tools that automate machine-learning (AutoML). In the following paragraphs, three of those tools work are outlined briefly.

Auto-WEKA is an AutoML tool that both selects a machine learning algorithm and optimizes its hyperparameters by using Bayesian optimization [14]. It was first released in 2013 as an extension to the popular data mining software WEKA [8] to assist the large number of novice users of the software in selecting parameterized algorithms for their problems. The tool has since grown in popularity and is in version 2.0 as of March 2016 [12]. In Auto-WEKA, the problem of selecting an algorithm and its hyperparameters is combined by treating the algorithm itself as a hyperparameter and searching the joint space of algorithms and hyperparameters for the best solution. An input data set is first preprocessed by means of feature selection. Then, Sequential Model-Based Optimization for General Algorithm Configuration (SMAC) is used to 'iterate[...] between fitting models and using them to make choices about which configurations to investigate' [10]. In the case of Auto-WEKA, this means that during the optimization process, a model is built, a configuration of hyperparameters that is promising regarding the current model and training data is tried out, and the result is fed back to the model. This cycle is then repeated until the allocated time has run out. Auto-WEKA exploits meta-knowledge, that is considering past performances of algorithms, to make decisions by always trying algorithms like Random Forests, which perform well on a large number of data sets, first.

AUTO-SKLEARN has been described as a sister-package to Auto-WEKA and is an AutoML tool which is based on scikit-learn, a machine learning library for Python [5]. It works very similar to AutoML but extends it by adding a meta-learning pre-processing step to warmstart the Bayesian Optimization and automatically constructing ensembles during optimization. During the pre-processing phase, performance values for the classifiers available in AUTO-SKLEARN are recorded on a set of data sets. For each data set, the algorithm which shows the best empirical performance is noted. Then, certain meta-features are calculated for each data set. The first step of the tool when given a new problem is to calculate meta-features of the data set. Then, the Manhattan distance to the other data sets is determined according to the meta-features, and the algorithms that are associated with the k-nearest data sets are used as a starting point for further optimization. The authors observe that the additional meta-learning and ensemble construction result in a more efficient and robust application. Their results show that Meta-Learning can be used to improve the overall AutoML process.

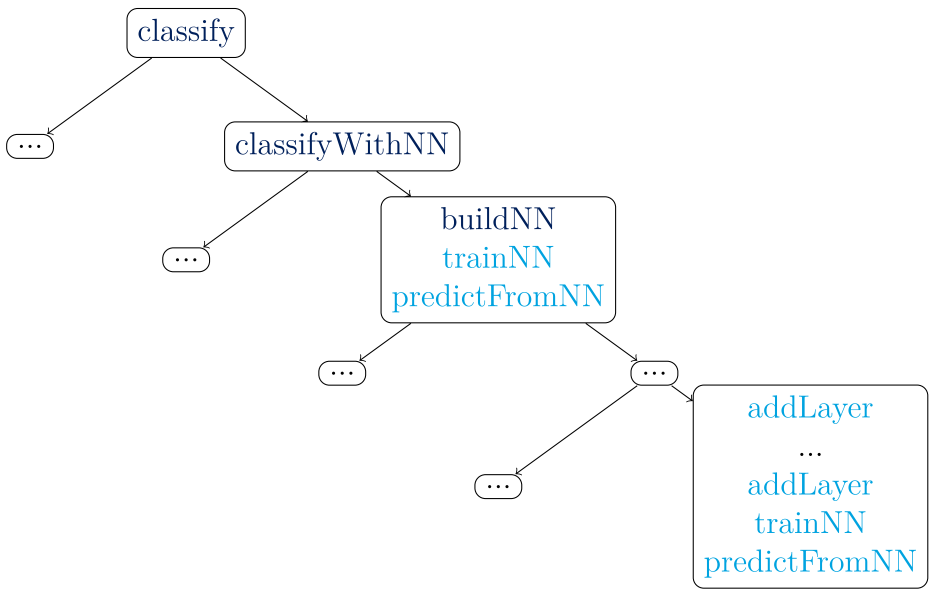

Figure 1: An example for how the complex task 'classify' might be broken down by ML-Plan. The figure is loosely adapted from [15]. Nodes containing '...' represent an undefined amount of subtrees. Complex tasks and primitive tasks are distinguished by their color.

ML-Plan is an AutoML tool that instead of concentrating on hyperparameter optimization, aims to optimize the whole machine-learning pipeline [15]. This is achieved by viewing machine-learning as a task, building a hierarchical task network out of those tasks, and then searching for a solution in the induced tree structure. In the tree, the root node contains the complex task of building a machine learning pipeline, inner nodes represent incomplete pipelines consisting of complex and possibly also primitive tasks, and leaf nodes are complete pipelines that include only primitive tasks. An example of this might be 'classify' as the root node, with an intermediate node on some level that contains the tasks 'build NN', 'train NN' and 'predict from NN'. The complex task 'build NN' would then further be decomposed, and could lead to a leaf node with n tasks 'Add layer', 'build NN' and 'predict from NN', which are all primitive tasks that do not need to be further decomposed. A best-first search algorithm in a modified variant is then used to find good solutions in this task network. While this variant does not use meta-learning in the process of optimizing the pipeline, the authors find that their results exceed those achieved by Auto-WEKA.

3 Goals

The aim of this thesis is to test the assumption that given a new data set, one can derive an accuracy-based ranking of classification algorithms from their past performances by means of regression-based compared with preference-based ranking, and to compare the two methods. It is derived from the need for AutoML tools and the fact that related solutions so far have not exploited Meta-Learning in this way. If the hypothesis is true, this tool could be used as a pre-processing step in another Auto-ML framework similarly to how Meta-Learning is used in AUTO-SKLEARN, as a pre-processing step to speed up the whole process. The aim presented consists of multiple goals, which can be divided into required and optional goals.

3.1 Required Goals

Based on the general objective of the thesis, several goals should be achieved:

1. Implementing a tool with the following functionality:
    - 1.1 A pre-processing phase in which the performances of the available set of classifiers is calculated for a set of data sets, a regression model is fit to each of the classifiers, and a preference-ranker is fit to the ranks of the algorithms which can be deduced from their performance values.
    - 1.2 A method where the tool is given a data set and returns a ranking of classifiers for this data set.
    - 1.3 The possibility to choose between regression-based and preference-based ranking.
2. Evaluate the predictions of both the regression and preference-ranking variant of the implementation by comparing them against each other, an oracle that represents the best ranking according to actual algorithms performances, and a best-algorithm baseline.

3.2 Optional Goals

If the required goals have been met in less time than originally allocated, and the initial results are promising, addressing additional goals may be considered. The following list suggests possible optional goals. However, they may be replaced if while working on the thesis it becomes clear that more appropriate expansions exist.

1. Adding automatic optimization of regression and ranking models with the help of ML-Plan and jPL[4] respectively [11].
2. Extend the evaluation by comparing the tool with other AutoML solutions.
3. Extend the evaluation by evaluating against a more intelligent baseline.
4. Adding another layer to the search by including hyperparameters, possibly by means of iterative requests to the tool.
5. Adding runtime and/ or complexity prediciton to allow for combined predictions.

4 Approach

As described in Chapter 3, the goal is to implement an AutoML tool that returns a ranking of classification algorithms for a given data set based on past performances of the classifiers by using regression and ranking models and evaluating the results. The implementation of this tool will be done in Java 8. Since the aim is to ensure a basic functionality, preliminary algorithms are going to be used both for regression and ranking models.

The first step therefore is to implement the pre-processing phase, which consist of computing the performances of classifiers on data sets. For the implementation of the classifiers, WEKA 3.8.1 is going to be used, which dictates the set of classifiers that will be included[5]. The data sets will be fetched from OpenML.org, an open-source project where users can contribute data sets, implementations for machine learning algorithms, and run algorithm runs. Since the results of the runs are available publicly on the website, some of the required data may be extracted instead of computed. OpenML.org provides the data sets in .ARFF format, and supplies

[4] A framework for the evaluation of preference learners.

a WEKA plug-in, which should ensure that these libraries work together smoothly.

The next step is to add the functionality to fit preliminary regression models and a preference ranker to the performance data, based on properties of the data sets. This step also includes providing a method that can be given a data set and returns a ranking, based on the choice of ranking method either based on regression or a preference model. This method first needs to compute properties of the new data set. Then, for regression, it is going to use the preliminary regression model of each algorithm to get a predicted performance of each classifier for the new data set. These predictions are turned into a ranking. For the ranking model, the computed properties of the new data set can be fed to the model directly.

For the evaluation, it is necessary to generate a baseline and an oracle. Both can be obtained from the data of the pre-processing phase. Due to all performances of classifiers being recorded on all data sets, one can obtain an optimal ranking of classifiers for each data set from that same table. The baseline which is going to be used is counting the number of data sets for which a classifier is the best option, and then ranking them from overall best performance (best choice for most data sets) to worst. This can be obtained from the table of the pre-processing phase as well. The rankings of the oracle, the baseline, and the results of the tool are then going to be compared using the Kendall Rank Correlation Coefficient.

[5] Meta-Classifiers and Ensemble-Classifiers will not be supported.

5 Preliminary Document Structure

The following structure will serve as an orientation during the writing process. While the chapters should remain the same, subsections may be added, removed or moved.

1. Introduction
2. Fundamentals
   - 2.1 Meta-Learning
   - 2.2 Kendall Rank Correlation Coefficient
   - 2.3 JPL
   - 2.4 WEKA
3. Approach
4. Implementation
   - 4.1 Regression-based Ranking
     - 4.1.1 Preliminary Regression Algorithm
     - 4.1.1 Automatic Algorithm Selection
   - 4.2 Preference Models
     - 4.2.2 Preliminary Ranking Algorithm
     - 4.2.2 Automatic Algorithm Selection
5. Evaluation
   - 5.1 Computing the Accuracy of the Results
   - 5.2 Assessment of the Accuracy
6. Related Work
7. Conclusion
8. Literature
9. Appendix

6 Schedule

The working time is going to be structured according to the schedule shown in figure 2. Variations due to unforeseeable circumstances may occur.

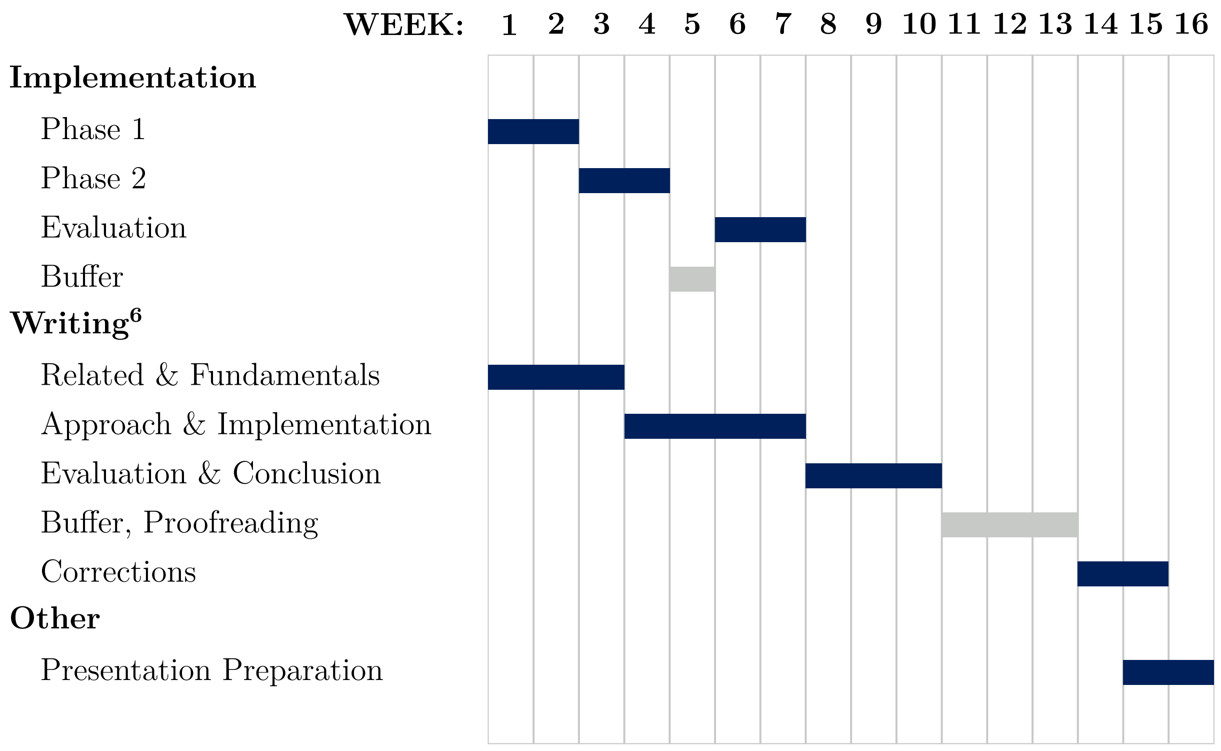

Figure 2: Sketch of the Work Schedule for the thesis.

[6] The chapter 'Introduction' is not included since it will be written in parallel to other chapters, in small bits according to the progress of the thesis.

References

[1] Miguel Cachada. *Run 2210139*. May 17, 2017. URL: `https://www.openml.org/r/2210139` (visited on 11/30/2017).

[2] Miguel Cachada. *Run 2221382*. May 18, 2017. URL: `https://www.openml.org/r/2221382` (visited on 11/30/2017).

[3] Miguel Cachada. *Run 2290623*. May 20, 2017. URL: `https://www.openml.org/r/2290623` (visited on 11/30/2017).

[4] Miguel Cachada. *Run 2290766*. May 20, 2017. URL: `https://www.openml.org/r/2290766` (visited on 11/30/2017).

[5] Matthias Feurer et al. "Efficient and robust automated machine learning." In: *Advances in Neural Information Processing Systems*. 2015, pp. 2962–2970.

[6] John Gantz and David Reinsel. *THE DITIGAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East*. Tech. rep. IDC 1414_v3. International Data Corporation, Nov. 2012. URL: `https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf` (visited on 11/30/2017).

[7] John Gantz and John Rydning. *Data Age 2025: The Evolution of Data to Life-Critical. Don't Focus on Big Data; Focus on the Data That's Big*. Tech. rep. International Data Corporation, Apr. 2017. URL: `https://www.seagate.com/files/www-content/%our-story/trends/files/Seagate-WP-DataAge2025-%Mar%h-2017.pdf` (visited on 11/30/2017).

[8] Mark Hall et al. "The WEKA data mining software: an update." In: *ACM SIGKDD explorations newsletter* 11.1 (2009), pp. 10–18.

[9] Mark Harris. *Splice-2 comparative evaluation: Electricity pricing*. Tech. rep. The University of South Wales, 1999. URL: `http://www.inescporto.pt/%7Ejgama/ales/ales_5.html` (visited on 11/30/2017).

[10] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. "Sequential Model-Based Optimization for General Algorithm Configuration." In: *LION* 5 (2011), pp. 507–523.

[11] IntelligentSystemsGroup. *jPL framework*. June 30, 2017. URL: `https://github.com/Intelligent-Systems-Group/jpl-framework` (visited on 11/30/2017).

[12] Lars Kotthoff et al. "Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA." In: *Journal of Machine Learning Research* 17 (2016), pp. 1–5.

[13] J Paul Siebert. *Vehicle recognition using rule based methods.* Tech. rep. TIRM87018. Turing Institute, 1987.

[14] Chris Thornton et al. “Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms.” In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM. 2013, pp. 847–855.

[15] Marcel Wever, Felix Mohr, and Eyke Hüllermeier. “Automatic Machine Learning: Hierarchical Planning Versus Evolutionary Optimization.” Unpublished: Proc. 27. Workshop Computational Intelligence, Dortmund, 23. -24.11.2017.

Hereby supervisor and student confirm that this proposal is the basis for the topic assignment of the described work. The timetable and topic description are accepted by both sides as laid out in this proposal.

Supervisor	Student
(Prof. Dr. Eyke Hüllermeier)	(Helena Graf)