Ranking of Classification Algorithms in AutoML

Helena Graf

January 5, 2018
Version: My First Draft

PADERBORN UNIVERSITY
The University for the Information Society

Department of Electrical Engineering,
Computer Science and Mathematics
Warburger Straße 100
33098 Paderborn

Intelligent Systems Group (ISG)

Bachelor's Thesis

Ranking of Classification Algorithms in AutoML

Helena Graf

1. Reviewer Prof. Dr. Eyke Hüllermeier
Department of Computer Science
Paderborn University

2. Reviewer Prof. Dr. Axel-Cyrille Ngonga-Ngomo
Department of Computer Science
Paderborn University

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

January 5, 2018

Helena Graf
Ranking of Classification Algorithms in AutoML
Bachelor's Thesis, January 5, 2018
Reviewers: Prof. Dr. Eyke Hüllermeier and Prof. Dr. Axel-Cyrille Ngonga-Ngomo
Supervisors: Prof. Dr. Eyke Hüllermeier and Prof. Dr. Axel-Cyrille Ngonga-Ngomo

Paderborn University
Intelligent Systems Group (ISG)
Department of Computer Science
Pohlweg 51
33098 and Paderborn

Abstract

Although the amount of data available worldwide is increasing, the portion of analyzed data remains low. This problem is likely caused by a lack of experts in the field that are able to carry out an analysis effectively, as it is a difficult task for a layperson. Especially regarding the challenge of classification there exist many algorithms with varying performances from data set to data set - there is no one best algorithm that can always be recommended. One response to this problem is automated machine learning (AutoML): tools that recommend a learning algorithm for a new data set that they estimate to perform best on this particular data set. Current AutoML approaches, however, often search for a fitting algorithm without explicitly taking into account how properties of the data set in question might influence the performances of classifiers. For the most part, they also suggest a single algorithm only. In comparison to the recommendation of a ranking of classifiers this has the disadvantage that no alternatives to the suggested algorithm are presented, which, due to a certain inaccuracy of the prediction, may have an equal or even better performance than the first suggestion regarding the examined data set. The problem considered in this thesis is thus the construction of a ranking of classification algorithms in relation to properties of a data set. Two different approaches that aim to solve this problem are presented and compared. The first alternative works on the basis of regression models: one regression model for each classifier predicts a performance value for that classifier. On the basis of the predicted performance values, a ranking then is constructed. In contrast, for the second approach, a preference-based model directly learns and returns rankings. In the evaluation of the two alternatives, it became apparent that the regression-based alternative is superior to the preference-based approach. However, the tested preference models also achieved a significant advantage compared to the static baseline used in the evaluation, at least in certain scenarios.

Abstract (German)

Die weltweit verfügbare Menge an Daten steigt immer weiter an, und doch bleibt der Anteil der tatsächlich analysierten Daten gering. Dieses Problem entsteht, da nicht genug Experten verfügbar sind, um diese Analyse auszuführen, und es für Laien schwierig ist, das richtige Verfahren auszuwählen. Besonders im Bereich der Klassifikation gibt es viele Algorithmen, deren Performanz jedoch von Datensatz zu Datensatz variieren kann - es gibt also keinen einen besten Algorithmus, der immer eine gute Wahl darstellt. Eine Antwort auf diese Problem ist automatisiertes Machine Learning. Daher werden in dieser Arbeit zwei verschiedene Ansätze, welche für einen neuen Datensatz ein Ranking von Klassifizierungsalgorithmen aufgrund von Eigenschaften des Datensatzes vorhersagt, vorgestellt und miteinander verglichen. Der erste der beiden Ansätze arbeitet auf Grundlage von Regressionsmodellen; ein Ranking wird hier konstruiert indem ein einzelnes Regressionsmodell für jeden Klassifizierungsalgorithmus einen Performanzwert vorhersagt, auf deren basierend dann das Ranking erstellt wird. Der zweite Ansatz beschäftigt sich mit Präferenzmodellen, welche direkt Rankings lernen und zurückgeben. In der Evaluation stellte sich heraus, dass der Regressionsansatz dem Präferenzmodell überlegen ist; das beste der getesteten Regressionsmodelle, basierend auf dem Random Forest Algorithmus, erreicht einen Wert von Kendalls Tau von bis zu 0,495 und Performanzverlustwerte von durchschnittlich nur 1,308 Prozentpunkten. Jedoch sind auch manche der getesten Präferenzmodelle in einigen Punkten wenigstens der zum Vergleich benutzten Baseline überlegen. Daher kann die Schlussfolgerung gezogen werden, dass wenigstens einige der benutzen Eigenschaften der Datensätze mit den Performanzwerten der Klassifizierungsalgorithmen zusammenhängen, und dieser Zusammenhang durch Regressionsmodelle sowohl als auch Präferenzmodelle ausgenutzt werden kann, wobei jedoch der Regressionsansatz letzterem im Rahmen der durchgeführten Evaluation überlegen ist.

Acknowledgement

Contents

1 Introduction

This introduction emphasizes why there is a need for automation in machine learning, and how this thesis relates to that problem in that it proposes two different approaches for ranking classification algorithms. Furthermore, an overview of the thesis structure is given.

1.1 Motivation and Problem Statement

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 [GR12]. 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 [GR17]. 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 the process is not accessible enough to non-experts. It 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 [Har99], the predictive accuracy for the Multilayer Perceptron[1] is 0.7887 [Cac17c]. The predictive accuracy of the Random Forest[2] algorithm on the same data set is 0.9236 [Cac17b], a much higher value. On a different data set, with the topic of vehicle silhouettes [Sie87], we get a predictive accuracy of 0.7979 for the Multilayer Perceptron [Cac17d], and

[1] With standard 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).

0.7518 for Random Forests [Cac17a], 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, and this example just illustrates the choice between two algorithms - in reality, the of available algorithms is much larger. 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 and the importance but difficulty of selecting a fitting algorithm, 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.

Thus, the aim of this thesis is the implementation and evaluation of two different approaches to ranking classification algorithms based on past-performances of the algorithms and according to properties of the data set. The two approaches include one regression-based approach that breaks down the problem of predicting a ranking into predicting a single performance value for each algorithm and then ranking them accordingly, and a preference based approach that is concerned with learning the rankings directly. Especially in the case of regression based ranking there is reason to believe that such a prediction is possible, as regression models have been used successfully to predict the performance of an algorithm dependent on the hyperparameter configuration [Egg+15].

A tool that efficiently predicts which algorithms are likely to perform well compared to their alternatives regarding a specific data set simplifies the process of selecting an algorithm for inexperienced users and experts alike. A user would send a request to the tool for their specific data set at hand and then evaluate highly ranked algorithms manually, choosing the one with the best performance. The advantage is that the user does not need to know characteristics of the classification algorithms or data set in order to achieve relatively good results in a short amount of time. A further speed-up and simplification could then possibly be achieved by embedding this tool in a solution that also takes care of the sampling of selected classifiers on the data set, for example existing solutions in the context of automated machine learning (auto-ml) context.

[3] Hyperparameters as above.

1.2 Thesis Structure

The following paragraphs give an outline of the thesis structure by providing a brief summary of each chapter.

Chapter 2 - Fundamentals

First, some preliminaries are discussed. A brief overview of tasks from the field of machine learning which are relevant to this thesis is given, and methods used for evaluation are explained.

Chapter 3 - Approach

The next chapter continues by describing the approach of this thesis for ranking classification algorithms. The two different proposed methods are contrasted. Following the details of the approach, the implementation thereof is presented. Used software libraries are pointed out.

Chapter 4 - Evaluation

The evaluation of the different ranking implementations is described by first clarifying the experimental setup used for the evaluation, followed by a discussion of the results.

Chapter 5 - Related Work

After the approach of this thesis has been laid out in detail, the scope is extended to related work in the area of auto-ml in general and ranking of learning algorithms more specifically. Differences and similarities in the approaches are discussed briefly.

Chapter 6 - Conclusion

In the last chapter, the results which have been achieved are revisited with the goals in mind. Finally, an outlook for possible future work extending more eval is provided.

2 Fundamentals

The core topic of this thesis falls under the category of machine learning. Thus, first the concept of machine learning and the tasks related to the approach of this thesis are explained. Second, means used for evaluation of the approach are discussed.

2.1 Machine Learning

In this section, the aspects of the field of machine learning which are relevant for this thesis are going to be introduced briefly. The core of machine learning is learning algorithms, which are used to induce a general model from a set of data samples. The concept of machine learning will be explained here with the help of an example.

Suppose an aspiring gardener wants to learn how to distinguish between different species of iris plants, a genus of ornamental plants with colorful flowers. More specifically, the focus lies on the three species iris versicolor, virginica and setosa. In order to do so, the gardener has observed four different *features*, namely the length and width of their petals and sepals, for a number of different individuals for which he knows the species, which is illustrated in Table 2.1. The goal of the gardener is then to learn how to distinguish the species based on these features, that is to derive a *model* from the data that will predict which species (out of the considered three) an unknown iris plant is. The gardener decides to build a decision tree from the data with forks on the basis of feature values, so that they can determine the species of the plant without much calculation. They observe that all iris setosa plants from his sample have a petal width of ≤ 0.6 cm, and that of the plants with a petal width of > 0.6 cm, most plants with a petal width ≤ 1.7 cm are of the iris versicolor species. The remaining plants with a petal width of > 1.7 cm are mostly iris virginica plants. While this model (illustrated in Figure 2.1) will not correctly predict the species of all iris plants, the gardener is settles with the approximation they have found.

In formal terms, the gardener is searching for an unknown *target function* $f : X \to Y$ from the input space X to the output space Y [ALM12] that represents an ideal way of identifying iris species. X denotes the possible inputs, in this case all combinations of the four features that have been defined, whereas Y are the outputs, here the species of iris plant. The examples that the gardener recorded are samples

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Iris setosa
5.0	3.5	1.6	0.6	Iris setosa
5.0	3.4	1.6	0.4	Iris setosa
5.6	3.0	4.5	1.5	Iris versicolor
6.7	3.1	4.4	1.4	Iris versicolor
5.9	3.2	4.8	1.8	Iris versicolor
7.2	3.0	5.8	1.6	Iris virginica
5.9	3.0	5.1	1.8	Iris virginica
6.9	3.1	5.1	2.3	Iris virginica

Tab. 2.1.: Features of different iris plants. This is an excerpt from the well known iris data set [36]

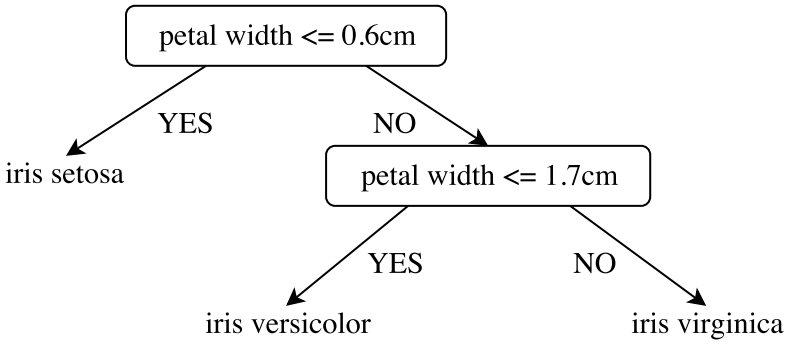

Fig. 2.1.: The decision tree constructed as a simple model to classify iris plants in the gardening example.

$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ from f such that $f(x_i) = y_i$. Together, they form a data set D which is one of all possible data sets $\mathbb{D}$ for the problem. They then choose a specific approximation $g : X \to Y, g \approx f$, the decision tree they built, from the hypothesis space H, which in this case consists of possible decision trees. The learning algorithm itself thus maps a data set to a hypothesis from the hypothesis space and can be defined as $A : \mathbb{D} \to H$. An overview of this learning problem is provided in Figure 2.2.

2.1.1 Supervised Learning

The problem described above has some additional properties besides being a machine learning problem in general. Firstly, it falls in the category of supervised machine learning. Supervised learning is one of the three main learning paradigms of machine learning, together with reinforcement learning and unsupervised learning [ALM12]. In the context of this thesis, the latter will be neglected as only supervised learning is used. It is characterized by the fact that the learning algorithm is provided with a set of inputs *together* with the outputs, which could be viewed as a kind of teacher,

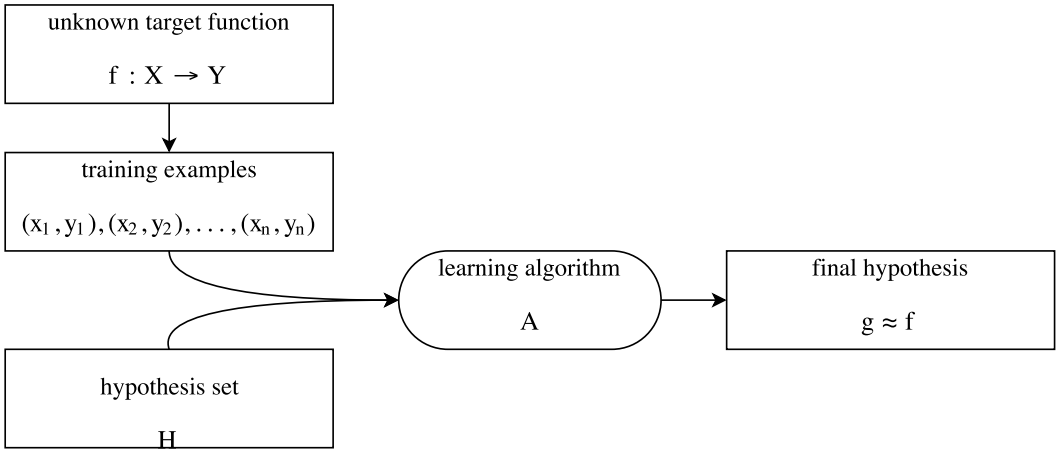

Fig. 2.2.: The learning problem, adapted from [ALM12].

or supervisor, explaining expected results to the learner. In the other cases, no such detailed feedback is provided to the learner.

2.1.2 Classification and Regression

In addition to falling into the category of supervised learning, the problem is a classification problem. The nature of the prediction is to assign one of the three classes, or labels, 'iris setosa', iris virginica' and 'iris versicolor' to a new plant. Formally, this means that the output space Y of the target function consists of a finite set of labels $\{y_1, \ldots, y_n\}$, so that each instance of the data set is associated with a label $y_i \in Y$, in this case $Y = \{'iris - setosa', 'irisvirginica', 'irisversicolor'\}$.

Apart from classification, another important category of machine learning problems is regression problems. In the case of regression, the output space of the target function is no longer finite, but instead consist of real numbers. To reconsider the gardening example, the gardener could want to predict the height of a plant on the basis of the same features as used above. Likewise, they would have to observe a number of plants to gather feature values together with the expected target value, which instead of the species would then be the height of the plant in centimeters, that is $Y = \mathbb{R}$.

2.1.3 Preference Learning

Preference Learning is a relatively new subfield of machine learning [FH10]. It is dedicated to the problem of learning how to rank, the precise definition of what this encompasses being defined by the specific task one considers. Out of the three main tasks of preference learning, namely label ranking, instance ranking and object

ranking, label ranking is the relevant one in the context of this thesis. But before going into the details of label ranking, it first has to be explained what is meant with a ranking in this context and how it is distinguished from the concept of an ordering, as it is easily confused.

Ranking and Ordering

To clarify the meaning of ranking and ordering, we return to the gardening example. After spending some time studying the different flowers, the gardener has realized he prefers certain iris species to others. He expresses his preference for a plant in a score, and labels all the plants in no particular order (see Fig. 2.2). An *ordering* then is the result of simply ordering the labels in decreasing order of their respective score. It thus can also be given implicitly by just ordering the original classes, the flower names, as such being a permuation of the labels. A ranking, on the other hand, assigns a fixed position to each label; for instance the rank of 'iris virginica' will always be in the first position in the ranking, that is the number in that position denotes the *rank* of the label, which is in turn defined as the label's position in the ordering. Therefore the values in the example in Table 2.2 result in the ranking $[2, 3, 1]$, whereas the ordering is $[3, 1, 2]$. It is notable that these can be the same, and are easily convertible. However, in the remainder of this thesis mostly the term ranking is used to identify both ranking and ordering interchangeably, especially as usually only the term 'ranking' is used.

Species		iris virginica	iris versicolor	iris setosa	
Label	[	1,	2,	3	]
Score	[	0.17,	0.12,	0.89	]
Ranking	[	2,	3,	1	]
Ordering	[	3,	1,	2	]

Tab. 2.2.: Ranking and ordering in direct comparison.

Label Ranking

Similar to classification, in label ranking there is a finite set of labels $Y = \{y_1, \ldots, y_n\}$, but it is not the output space. Instead, Fürnkranz and Hüllermeier define the target function for label ranking as $f : X \to S_Y$ where the output space S_Y contains all permutations over the set of labels Y [FH10]. To come back to the topic of rankings and orderings, it becomes clear now that even though it is called label ranking, the labels are actually ordered. An example of label ranking could be, to stay in the domain of gardening, to learn the preferences of different gardeners towards iris plants. A label ranking data set used to learn this connection could, for example,

map the favourite colour and height of a gardener to a ranking of the three iris plants.

2.1.4 Meta Learning

The previous discussion of machine learning problems has conceptually been about the fitting of different machine learning algorithms to problems in the form of data sets. In the context of meta learning regarding machine learning, it is researched whether it is 'possible to learn about the learning process itself, and in particular that a system could learn to profit from previous experience to generate additional knowledge that can simplify the automatic selection of efficient models summarizing the data' [Bra+08]. That is, the original algorithm selection problem [Ric76] is viewed as a learning problem itself that is to be tackled.

The approach of this thesis clearly falls under that category, as it is trying to learn whether a connection exists beween properties of a data set, that is its meta features, and the performance values of certain classification algorithms, and how well it can be exploited by regression and preference models respetively, if it exists. The benefit from meta learning in the conext of machine learning is that learning how properties of data sets influence the learning process of the classifiers could save much work; Often, much work is put into the fitting of an algorithm to a specific data set, but it is not transferred to other poblems that may be of the same nature.

2.2 Evaluation

After having discussed the preliminaries that the theoretical approach of this thesis is built upon, how a data set can be split into training and test data to estimate the performance of learning algorithms and the measures that will be used in the evaluation of the different ranking methods are briefly reviewed. These include means of comparing the predicted rankings with actual rankings as well as a statistic that reveals if an advantage regarding one measure can be viewed as significant.

2.2.1 Estimation Procedures

To find out how well a classifier performs, it first needs to be decided which measure consider. There are a few possibilities for this, however in this thesis the term performance measure refers to the predictive accuracy of a classifier, that is the percentage of classes that a classifier correctly identified in a given data set. But since using the whole data set to train a classifier, that is build a model, would skew

the analysis - the classifier has already 'seen' these examples - a method for dividing data sets into train and test sets is needed.

A popular approach is using a stratified split, for example splitting the data into 70% test and 30% training data, with the important aspect that both sub sets have equal (or at least very similar) ratios of classes [Reference Missing]. The learning algorithm is then trained with one part of the data (usually the larger), and its performance tested on the other. To get more stable results, often the process is repeated: The data is divided in k folds, k times, the learning algorithm is trained on k-1 folds, and tested on the left out subset. Results are averaged. This process is then called k-fold (stratified) cross validation [Reference Missing]. A variant of this is Monte Carlo crossvalidation, where after each time the algorithm has been trained and evaluated, a new split is computed [Reference Missing]. This especially causes the fact that the number of folds does not dictate the ratio of the split; one could for example have 5-folds Mone Carlo crossvalidation with a 70%/30% train/test split.

An alternative to using a stratified split is using a method called leave-one-out. This method essentially is, for a data set with n instances, n-fold crossvalidation (not stratified, naturally). This means that the algorithm is trained on n-1 instances and evaluated on the instance that has been left out.

2.2.2 Kendall Rank Correlation Coefficient

To evaluate the quality of the prediction that a ranker will make, a statistic is needed to compare a predicted ranking with the correct ranking. For this, the Kendall Rank Correlation Coefficient will be used here, as it expresses the similarity of two rankings.

The Kendall Rank Correlation Coefficient, also known as Kendall's tau, is a measure of association between two rankings of the same items [Ken38]. For two independent variables X and Y and and observations of values $(x_1, x_2, ..., x_m)$ for X and $(y_1, y_2, ..., y_n)$ for Y with unique x_i and y_i respectively, the coefficient is based on comparing pairs of observations. If for two pairs (x_i, y_i) and (x_j, y_j) both $x_i < x_j$ and $y_i < y_j$ (or the opposite) holds, they are viewed as concordant. If $x_i < x_j$ but $y_i > y_j$ (or the opposite), they are viewed as discordant. Other cases are not considered. The simplest version of the statistic, called T_A is then defined as

$$T_A = \frac{n_c - n_d}{n_0},$$

with
n_c, the number of concordant pairs,
n_d, the number of discordant pairs, and
$n_0 = n * (n + 1)/2$.

As we have seen in example above, can also have same rank for items. In order to account for this, a second statistic, T_B is defined as

$$T_B = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}},$$

with
n_c, n_d, n_0 as above,
$n_1 = \sum_i t_i(t_i - 1)/2$,
$n_1 = \sum_j t_j(t_j - 1)/2$,
t_i, the number of ties observed for X in the ith position of the ranking
u_j, the number of ties observed for Y in the ith position of the ranking

Thus, the value of the tau (for both versions) lies in the interval $[-1, 1]$, with 1 indicating a perfect agreement of rankings, -1 perfect disagreement, and 0 corresponding to no correlation. A high value is therefore desirable in this context, as it indicates that the prediction of the ranker agrees with the actual ranking of algorithms.

2.2.3 Loss

While the Kendall rank correlation coefficient is an indication as to what degree a predicted ranking agrees with the correct ranking of classifiers for a certain data set, it does not convey the difference in performance experienced by the user when choosing the first of the recommended algorithms. This difference can be expressed in measure called loss [LBV12], which is defined as the difference of the performance of the actual best algorithm and the actual performance of the first recommended algorithm.

Additionally, the ranker recommends a ranking instead of a single algorithm in order to allow a user (or automated tool) to test a few of the top-ranked algorithms, which therefore also needs to be taken into account when thinking about loss. This is done here by computing a measure called best three loss, which denotes the loss resulting in the selection of the best of the first three recommended classifiers in comparison to the actual best-performing classifier in the considered set of classifiers.

2.2.4 Mann-Whitney U Test

The measures previously presented can indicate if one of the ranking alternatives is superior to another regarding that measure, but one cannot conclude from those results alone whether this advantage is significant. The question here is whether the null-hypothesis that two distributions belong to the same population can be rejected. This can be determined with the help of the Mann-Whitney U, a non-parametric hypthesis test [MW47]. The Mann-Whitney U is computed by, for two distributions one wants to analyze, counting how many times each value frome one distribution is lower than another value from the other distribution. If the U is very high (or low, depending on the hypothesis), the null hypothesis can be rejected. Whether this significance level is reached is expressed by the p-value: for a p-value below 0.05, The null hypothesis can be rejected. Therefore, if two distributions, for example regarding the loss values of two ranking approaches, suggest that one is superior to the other, and the p-value for the Mann-Whitney U is $<= 0.05$, this suggests that the difference is statistically significant.

3 Approach

In this chapter the two alternatives considered in this thesis to tackle the problem of ranking classification algorithms regarding properties of the data set. First, the theoretical approach is presented. Then, implementation details are discussed.

3.1 Theory

In general, the aim is to predict a ranking of algorithms based on properties, or meta features of a data set. This means, that as training data for a ranking approach (ranker), a number of performance values of classifiers need to be recorded on data sets in a first pre-processing phase, for which also meta features are to be computed. The table of this collected data is illustrated in Table **??**, for n meta features, k classifiers and n data sets. This approach to ranking is realized here, as pointed out before, in a regression based and preference based approach, which are explained in detail in the following sections.

MF 1	MF 2	...	MF n	PV 1	PV 2	...	PV k
v_{11}	v_{12}	...	v_{1n}	p_{11}	p_{12}	...	p_{1k}
v_{21}	v_{22}	...	v_{2n}	p_{21}	p_{22}	...	p_{2k}
...	...	...	...	...	...	...	...
v_{m1}	v_{m2}	...	v_{mn}	p_{2m}	p_{m2}	...	p_{mk}

Tab. 3.1.: Training data for the rankers.

3.1.1 Regression Based Ranking

One possibility to derive a ranking of classifiers from this information when given a new data set is to use regression models. The idea is to use separate regression models to predict a performance value for each classifier, and to then derive an ordering from these predictions. This is done by splitting the data set compiled beforehand (see Tab. **??**) into separate data sets that each contain all meta features for all data sets, but only the performance values of one classifier, as illustrated in Table 3.2. The target feature on these individual data sets, the performance value of the classifier, is a numeric value, and thus a regression model can be trained on each of them. All regression models therefore try to learn the target function

metafeatures→performance value for their respective classifier. Hence for a query instance, after computing the meta features, these are fed into each regression model and the predicted performance value for each classifier is saved. In a second step, the classifiers are ordered according to the predictions. This process the ranker passes through for making a prediction is presented in Figure 3.1.

Meta Feature 1	...	Meta Feature n	Performance Value 1
v_{11}	...	v_{1n}	p_{11}
v_{21}	...	v_{2n}	p_{21}
...	...	...	...
v_{m1}	...	v_{mn}	p_{m1}
Meta Feature 1	...	Meta Feature n	Performance Value 2
v_{11}	...	v_{1n}	p_{12}
v_{21}	...	v_{2n}	p_{22}
...	...	...	...
v_{m1}	...	v_{mn}	p_{m2}
...			
Meta Feature 1	...	Meta Feature n	Performance Value k
v_{11}	...	v_{1n}	p_{1k}
v_{21}	...	v_{2n}	p_{2k}
...	...	...	...
v_{m1}	...	v_{mn}	p_{mk}

Tab. 3.2.: How the regression ranker processes the training data.

3.1.2 Preference Based Ranking

The second possibility considered in this context is using preference learning to predict a ranking. Each instance of the data set generated beforehand (see Fig **??**) contains meta feature information for the considered data set and performance values of all classifiers. Similar to the regression alternative, the performance values can be converted to preference information by sorting the corresponding classifiers by their performance values for each data set. This leads to an ordering of classifiers associated with each instance, and implies that the preference learning task at hand is label ranking. In this case, there exists only one label ranking model which is fed the computed meta data for a new data set and returns an ordering of classifiers, which is depicted in Figure 3.2. Thereby it attempts to learn the mapping from meta features to an ordering of classifiers directly.

MF 1	MF 2	...	MF n	Ranking
v_{11}	v_{12}	...	v_{1n}	r_1
v_{21}	v_{22}	...	v_{2n}	r_2
...	...	...	...	...
v_{m1}	v_{m2}	...	v_{mn}	p_m

Tab. 3.3.: How the preference ranker processes the training data.

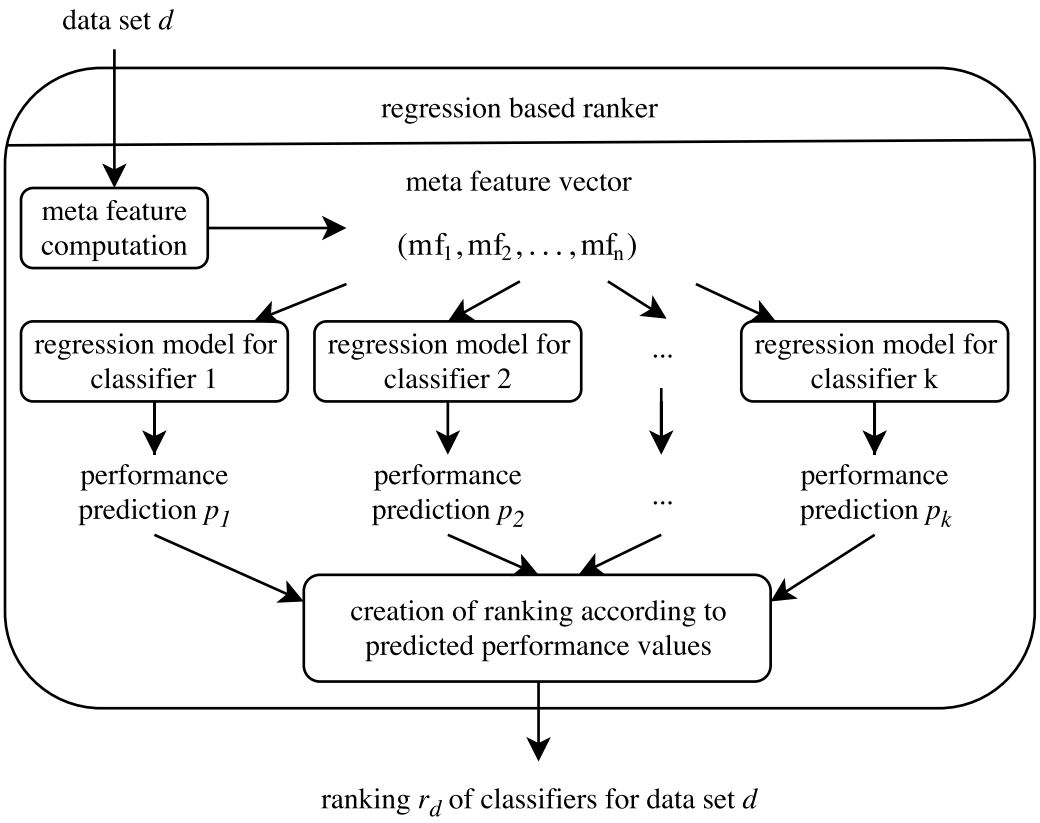

Fig. 3.1.: An overview of how a regression based ranker constructs a ranking.

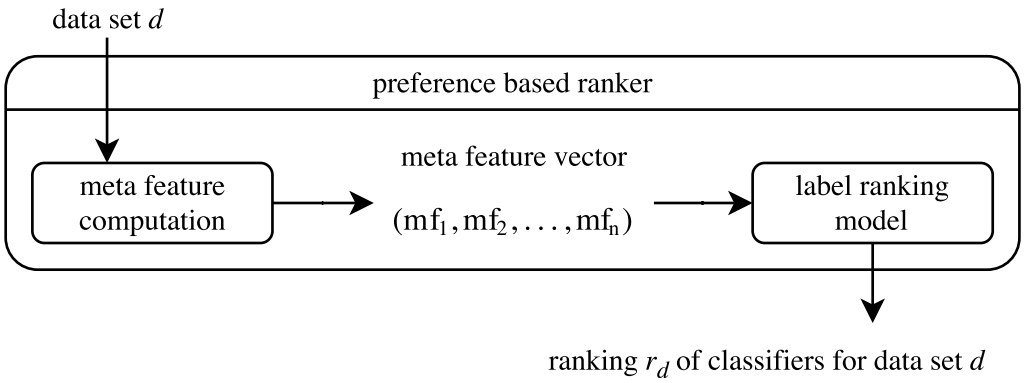

Fig. 3.2.: An overview of how a preference based ranker constructs a ranking.

3.2 Implementation

In general, all implementations for the approach are carried out in Java, particularly Java 8. However, some additional packages where used, which are pointed out briefly. For the implementation of the regression models, WEKA was used, a popular libary of machine learning algorithms that includes algorithms for classification, regression, and provides help in the evaluation of these learning algorithms [Hal+09]. Regarding the preference algorithms, the jPL framework was used [Int17]. The jPL framework is a java framework for the evaluation of preference learning algorithms, it implements several tasks from the context of preference learning, including label ranking, and also provides a representation for data sets that contain label ranking information. The list of meta features used is taken from OpenML [Van+13], a website that not only provides a large number of openly accessible data sets, but also records performance values of learning algorithms on these data sets, and features properties of data sets. Thus the meta features where calculated with the help of an implementation provided by OpenML in its Evaluation Engine [Int18]; The full list of meta features is included in Table A.1 in the Appendix.

4 Evaluation

This chapter is dedicated to the evaluation of the proposed ranking approaches. First, the oracle and baseline against which the rankers are compared are explained. Second, the experimental setup, that is conditions under which the evaluation was conducted, is set forth. Last, results are presented and discussed.

4.1 Baseline and Oracle

In order to evaluate the implemented rankers, their outputs are compared to a perfect output generated by an oracle. The oracle is implemented as a ranker that, after having been trained on a meta data - performance data set returns correct rankings when queried for any instance of that data set. The correct ranking is implied by the performance values (highest first, as the measure used here is predictive accuracy), with ties being handled in a sequential manner, i.e. for the same performance values, the algorithm that was encountered first when constructing the ranking will be ranked before all others with the same value.

As rankings returned by this oracle represent the ideal solution, the predicted rankings are compared with them through the measures presented in chapter 2. These include the Kendall rank correlation coefficient, loss and best three loss. For all three measures, it is computed if any advantage or disadvantage one solutions has is significant, by means of the Mann-Whitney U and its P-value. The regression based rankers pose a special case: internally, they predict a performance value for each classifier, which therefore can be compared to the actual performance values to show how well they are predicted. This is done by computing the root mean square error between the predicted and actual performance values. In summary, it is desirable for the rankers to come as close as possible to the correct ranking, which is reflected by a high rank correlation and a low loss and root mean square error.

Furthermore, we are interested in the question as to what degree knowing about the properties of a data set influences the quality of rankings. Therefore, a ranker that is agnostic of the meta features of a query data set, that is that will always return the same ranking for any data set, is used as a baseline. More specifically, a best algorithm strategy that iteratively determines a ranking by counting the number of

data sets where an algorithm is the best choice, is implemented. That means the first algorithm in the best algorithm ranking is the classifier that performs best on most data sets, the second is the best on most data sets when only rankings excluding the first algorithm are considered, and so forth. This baseline is evaluated in the same way as the preference rankers; as it does not predict performance values, the root mean square error of predicted performance values cannot bet computed. Ideally, rankings returned by the preference and regression based rankers should then be statistically significant better than the baseline.

4.2 Experimental Setup

In the following paragraphs, it is briefly summarized under which conditions the experiments which led to the results discussed in the next section where observed. This includes which classifiers, data sets and meta features where chosen in the evaluation, what the specifications of the machines the evaluation was executed on where, how the evaluation was carried out, and which specific regression based and preference based implementations where used.

The classifiers considered in the rankings were the learning algorithms implemented in WEKA that are fit for classification, excluding meta and ensemble methods. The full list of the 22 classifiers can be gathered from Table 4.6. They where all used in their default configurations as specified by WEKA.

Data sets for the analysis where gathered from OpenML. From all data sets considered as 'active' on OpenML, which yields 2050 data sets that have been approved with no severe issues found for them so far [nd], all data sets that comply with the constraints of the learning problem at hand where selected. Only data sets with a defined, nominal target feature that are in the .ARFF format which where not specifically designed for streaming analysis[1] where considered, which resulted in a reduced selection of 812 instances. Since this amount of data sets hindered the evaluation considerably, large data sets with more than 1000 instances or 100 features where removed as well, leading to a final selection of 448 data sets. Even though evaluation was only carried out on the shorter list, both the list of 812 and 448 data sets are included in the supplementary material for completeness.

The performance data of the classifiers for the selected data sets was generated on a linux cluster, each node consisting of two Intel(R) Xeon(R) E5-2670 processors running at 2.6GHz, 16GB RAM with usage of up to 150 nodes. There was a timeout

[1] Data sets that contained the substring 'BNG' in their name for Bayesian Network Generator, which contain data artificially generated by a Bayesian Network [Rij+14] for the sake of data stream analysis where not included in the evaluation.

of eight hours set for the computation of a single performance value of a classifier on a data set. For each of the classifiers, the predictive accuracy (the percentage of correctly classified instances in the data set) was recorded on each data set by means of five times stratified Monte-Carlo cross validation with a 70/30 split in training and test data. The stratified split was generated with the help of JAICore, 'A Java library for infrastructural classes and basic algorithms in the area reasoning, search, and machine learning' [Reference Missing]. When a timeout occurred or the evaluation failed otherwise, the predictive accuracy was set to zero.

All other experiments where carried out on a Windows Machine running Windows 10 Education (version 1709) with a Intel(R) Core(TM) i5-4200U CPU running at 1.60GHz (2.30GHz max) with 12 GB RAM. It is notable that especially the time-related results are to be viewed in regard to this setup. No timeouts where used in this evaluation, and the rankers where evaluated on the training data by means of leave-one-out estimation, that is for the n data sets for which performance values where recorded, they where trained with the meta data - performance examples for n - 1 data sets, and queried for the remaining data set. As n = 448 in this case, the detailed values for each data set are not included in the discussion of the results. They can, however, be found in the supplementary material, offering clues to how each ranker performed in the prediction of a ranking for each of the used data sets, identified by their ID on OpenML. Furthermore, the evaluation was carried out twice for all measures: once with the full meta data used, and one time using only the meta features not generated by landmarkers, that is the ones that require probing to be carried out for a new data set.

For the computation of the Mann-Whitney U and Kendall rank correlation coefficient, Apache Commons implementations where used. Regarding the Mann-Whitney U, the strategy for ties as set to averaging, and the NaNs where set to be removed before the analysis was started.

Furthermore, four alternatives where selected for a regression based and preference based alternative each, to get a general idea of how well the respective approach might be suited for the problem. For the regression based approach, the algorithms chosen where random forest, REPTree, M5P, and linear regression. For all of them, the WEKA implemenation [Hal+09] was used, and the algorithms where used with the standard hyperparameters set by WEKA. As the preference based ranking falls under the category of label ranking, and in the jPL-framework, which was used for the implementation of the label ranking algorithms and data set representations, only two alternatives for label ranking are implemented [Int17], the other two alternatives where generated by modifying the hyperparameters. The implemented alternatives are label ranking by pairwise comparison and instance based label ranking. As in early tests it became apparent that the instance based approach might

hold more potential, for label ranking by pairwise comparison, only the default configuration dictated by jPL was used. For instance based label ranking, the default configuration, a configuration where the rank aggregation algorithm used by the label ranker was set to Kemeny-Young, and a configuration with Kemeny-Young rank aggregation and the number of neighbors of the base learner of the label ranker, namely k nearest neighbor (kNN), set to the square root of the number of instances in the training data set. In the following section, these eight variants are compared with the baseline and oracle mentioned in the previous section, and among each others.

4.3 Results

In the following subsections, results of the evaluation are discussed in detail. First, the quality of the predictions of the rankers is examined. Then, the times required for building the rankers and for predictions made as well as for the calculation of the meta features is reviewed. Last, additional insights gathered during the evaluation are pointed out.

4.3.1 Accuracy

When looking at the results depicted in Table 4.1, it must first be noted that positive findings regarding the Kendall rank correlation can be reported. For the full set of meta data, six out of the eight implemented alternatives outperform the baseline, all of them significantly, as can be learned from Table 4.2 which shows the results of the significance tests, with random forest having the highest correlation of 0.495. In contrast, the two approaches that did not overtake the baseline, instance based label ranking in the default configuration and label ranking by pairwise comparison, show a significant disadvantage compared to the baseline, and thus also compared to the other ranking approaches. Regarding loss, only linear regression and random forest show a significant advantage, with all label ranking approaches except for instance based label ranking having a significant disadvantage in contrast. Moving on to best three loss, the picture largely remains the same, except that surprisingly, REPTree now also has a significant advantage over the baseline. The lowest value here is achieved by random forest with a best three loss of 1.308, in comparison to 2.440 for the best algorithm baseline. These values are illustrated in Figure 4.1, with a scatter plot for individual best three loss values on the data sets.

As for the rank correlation most of the rankers had a significant advantage over the baseline, it is interesting to compare them among each other. From Table 4.1 it is clear that the regression rankers are superior to the preference rankers, and

this advantage is significant, as the weakest of the regression rankers, REPTree, is still significantly better than the strongest label ranking approach, instance based label ranking with Kemeny-Young rank aggregation ($p \approx 4.305E-6$). Out of the four regression rankers, three approaches are furthermore significantly better than REPTree.

Compared to the evaluation on the full set of meta data, the results on the reduced set without probing are less distinctive. For the Kendall Rank correlation, linear regression, M5P and random forest are significantly better than the baseline, and for label ranking, both of the instance based approaches with non-default parameters as well, while the other label ranking approaches are significantly worse than the baseline. In total, random forest is still the best alternative with a correlation of 0.439, slightly worse than the correlation achieved on the full meta data. For both loss measures however, while the disadvantages of the label ranking approaches remain, none of the eight alternatives achieves a significantly better loss or best three loss than the baseline anymore.

With reduced meta data, out of the alternatives that pose a significant advantage compared to the baseline, only random forest is significantly better than all other approaches, the others are not significantly different. With loss and best three loss, no such statement can be made.

Summarizing the observed results, in general, a regression based approach shows an indication to be stronger than a preference based approach. Specifically, using random forest as a regression model is the best solution observed in the evaluation, and label ranking by pairwise comparison the weakest, as it falls behind the other ranking approaches and the baseline often A surprising result is that the label ranking approaches achieve very similar results with or without meta data.

Ranker	Kendall's Rank Correlation				Loss				BestThreeLoss			
	Min	Max	Mean	Stdv	Min	Max	Mean	Stdv	Min	Max	Mean	Stdv
LinearRegression	-0.255	0.896	0.473	0.221	0	86.667	3.469	7.244	0	31.220	1.267	3.011
M5P	-0.29	0.870	0.470	0.219	0	82.353	3.78	6.535	0	82.353	1.508	4.757
RandomForest	-0.281	0.922	0.495	0.228	0	60.000	3.097	5.745	0	34.634	1.308	3.150
REPTree	-0.229	0.896	0.412	0.213	0	82.353	4.829	7.948	0	34.634	1.759	3.515
InstanceBased	-0.429	0.870	0.221	0.249	0	82.353	5.401	9.440	0	82.353	3.620	8.540
InstanceBased[2]	-0.429	0.887	0.340	0.252	0	98.367	5.437	10.323	0	82.353	3.294	8.367
InstanceBased[3]	-0.429	0.870	0.335	0.249	0	82.353	5.382	9.402	0	82.353	3.511	8.493
PairwiseComparison	-0.870	0.576	0.014	0.234	0	97.822	9.762	13.135	0	82.353	4.001	8.600
BestAlgorithm	-0.351	0.758	0.296	0.213	0	82.353	4.480	9.205	0	82.353	2.440	7.625
LinearRegression	-0.532	0.835	0.350	0.235	0	82.353	4.921	9.109	0	82.353	2.211	6.744
M5P	-0.290	0.887	0.344	0.219	0	87.455	5.367	9.662	0	82.353	2.060	5.466
RandomForest	-0.359	0.913	0.439	0.233	0	82.353	3.538	6.803	0	40.000	1.587	3.784
REPTree	-0.290	0.844	0.301	0.215	0	91.154	6.771	10.710	0	82.353	3.013	7.162
InstanceBased	-0.429	0.870	0.221	0.249	0	82.353	5.401	9.429	0	82.353	3.615	8.531
InstanceBased[2]	-0.429	0.887	0.340	0.252	0	98.367	5.437	10.311	0	82.353	3.294	8.357
InstanceBased[3]	-0.429	0.870	0.335	0.249	0	82.353	5.382	9.391	0	82.353	3.511	8.483
PairwiseComparison	-0.870	0.524	0.013	0.232	0	91.667	9.486	13.158	0	82.353	4.096	8.669
BestAlgorithm	-0.351	0.758	0.296	0.213	0	82.353	4.479	9.205	0	82.353	2.440	7.625

Tab. 4.1.: Evaluation results with full meta data (top) and no probing (bottom).

[2] Rankaggregation: Kemeny-Young
[3] Rankaggregation: Kemeny-Young, Baselearner KNN with n=$\sqrt{\text{number of instances}}$

Ranker	Kendall's Rank Correlation		Loss		BestThreeLoss	
	Mann-Whitney U	P-Value	Mann-Whitney U	P-Value	Mann-Whitney U	P-Value
LinearRegression	144397.5	5.798-30	108781.0	0.030	112695.5	0.001
M5P	144007.0	1.831E-29	100897.5	0.888	110031.5	0.012
RandomForest	148620.0	1.208E-35	110970.0	0.006	113020.0	0.001
REPTree	129582.0	4.471E-14	107225.0	0.076	102345.0	0.607
InstanceBased	117813.5	6.540E-6	110030.5	0.012	112178.0	0.002
InstanceBased[4]	110063.0	0.012	106943.5	0.089	107195.5	0.077
InstanceBased[5]	109909.0	0.014	108406.5	0.038	110058.5	0.012
PairwiseComparison	163081.0	5.457E-59	138148.5	1.702E-22	123784.5	1.451E-9
LinearRegression	114370.5	2.955E-4	106405.5	0.118	102563.0	0.568
M5P	110774.0	0.007	107366.5	0.070	100699.0	0.929
RandomForest	134467.0	1.278E-18	106410.0	0.118	107453.0	0.067
REPTree	101292.0	0.808	120574.0	1.781E-7	111526.0	0.004
InstanceBased	117813.5	6.540E-6	110030.5	0.012	112178.0	0.002
InstanceBased[4]	110063.0	0.012	106943.5	0.089	107195.5	0.077
InstanceBased[5]	109909.0	0.014	108406.5	0.038	110058.5	0.012
PairwiseComparison	163420.0	1.310E-59	138082.0	2.016E-22	124838.0	2.587E-10

Tab. 4.2.: Significance of evaluation results (compared to the baseline) with full meta data (top) and no probing (bottom).

4.3.2 Computation Times

Because the ranking itself is implemented as a means of speeding up the process of algorithm selection, the time it takes to rank is not negligible. The full results of how long it takes to build the respective ranking models and how much time they need for predictions can be taken from Table 4.3. Results are rounded to the full millisecond, as due to the times being measured with a stop watch in the code, more accurate measurements are not possible. The times for building the regression rankers are not surprising; the regression models take far more time to be built than most of the label ranking methods, most likely due to the fact the regression based rankers have to train 22 regression models in order to be trained themselves. However, it is notable that while the ranking by Pairwise Comparison was not among the best solutions concerning the accuracy of the results, it is the ranker with the highest mean build time for full meta data, and second highest for a reduced meta data set, whereas in other cases the high quality of results is correlated with a high time invested in the building of the ranker. The Random Forest ranker, for example, is the ranker with the highest build time, and also with the rankings of the generally highest quality. But the most important fact to be taken away from Table 4.3 is that

[4] Rankaggregation: Kemeny-Young
[5] Rankaggregation: Kemeny-Young, Baselearner KNN with n=$\sqrt{\text{number of instances}}$

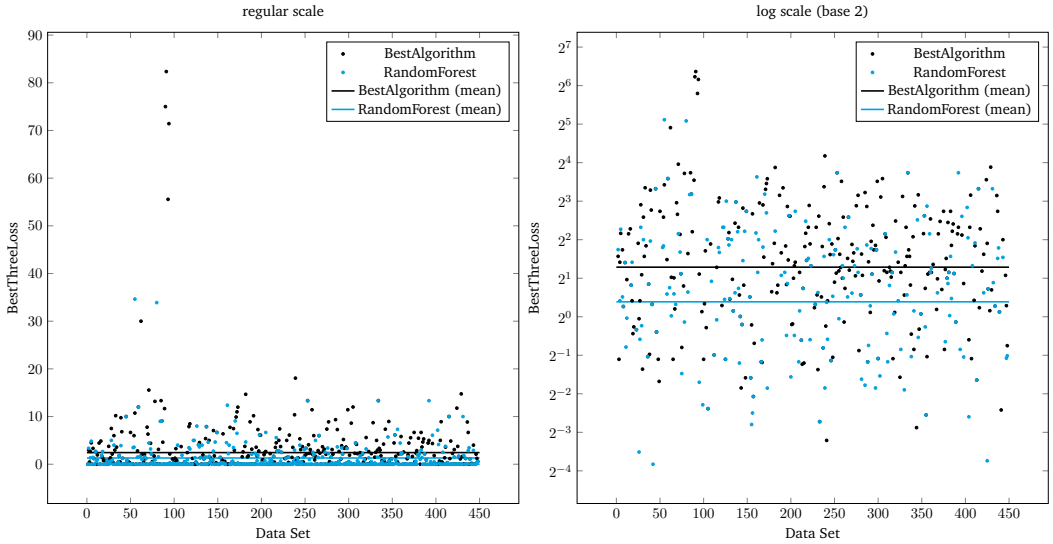

Fig. 4.1.: The best three loss for the baseline compared to the random forest ranker (full meta data).

the prediction times for all rankers are short enough to be negligible, with the mean prediction time never being higher than five milliseconds and the maximum never exceeding 50 milliseconds. Furthermore, while the build times are higher, these are not as important due to the fact that each model only needs to be trained once. Although in a real-world scenario, calculation times for meta features also have to be taken into account when ranking, which therefore will be discussed in the following paragraph.

Table 4.4 shows the times required for the computation of groups of meta features as discussed in chapter 3. As can be taken from the table, the computation of the full meta features takes approximately 135 milliseconds on average, while the simple meta features without probing add approximately 3 milliseconds to the prediction times on average. Regarding these computation times it is also notable that among the probing meta feature groups, some are considerably more expensive than others, so that in the future it might be sensible to consider only light probing with probing meta feature groups that are not as expensive.

4.3.3 Further Insights

Further insights to be noted are the accuracy of predictions of performance values for classifiers by the regression models, and the static ranking constructed by the best algorithm baseline. The accuracy of the predictions achieved by the regression based rankers are depicted in Table 4.5, for full and reduced meta data respectively. The

[6] Rankaggregation: Kemeny-Young
[7] Rankaggregation: Kemeny-Young, Baselearner KNN with n=$\sqrt{\text{number of instances}}$

Ranker	Ranker Build Time (ms)				Ranker Prediction Time (ms)			
	Min	Max	Mean	Stdv	Min	Max	Mean	Stdv
LinearRegression	1454	2060	1580	36	0	1	0	0
M5P	3145	4916	3226	89	0	16	0	1
RandomForest	6048	9720	6236	259	0	16	3	2
REPTree	599	1264	629	38	0	16	1	16
InstanceBased	66	550	90	25	0	47	5	8
InstanceBased[6]	66	138	88	12	0	19	1	3
InstanceBased[7]	66	163	87	12	0	16	1	2
PairwiseComparison	8456	15063	9096	449	0	16	0	2
BestAlgorithm	204	1250	227	52	0	0	0	0
LinearRegression	421	500	446	10	0	0	0	0
M5P	2219	4225	2282	148	0	16	0	1
RandomForest	6175	10535	5419	479	0	16	3	6
REPTree	484	1078	509	32	0	15	0	1
InstanceBased	62	141	77	7	0	16	4	7
InstanceBased[6]	62	125	79	8	0	16	1	3
InstanceBased[7]	62	110	77	7	0	16	1	§
PairwiseComparison	6705	7893	7253	169	0	16	0	2
BestAlgorithm	206	405	232	23	0	0	0	0

Tab. 4.3.: Build and prediction times of the rankers rounded to full milliseconds with full meta features (top) and no probing (bottom).

first thing that attracts attention in this table is the absurdly high values for Linear Regression and M5P, however, these are all caused by the very high maximum value on one data set with id 685. If these extreme maxima remain out of consideration for Linear Regression and M5P (the other algorithms do not have this problem) more reasonable values are obtained. For M5P, on the full meta data the new maximum is 311.526, mean 8.127 and standard deviation 18.407; on the reduced meta data it is a maximum of 474.692, mean 15.444 and standard deviation 27.433. For Linear Regression, on the full meta data the new maximum is 142.996, mean 9.281 and standard deviation 8.438; on the reduced meta data it is a maximum of 131.456, mean 14.351 and standard deviation 10.299. However, these values are still worse than the ones obtained for the Random Forest and REPTree variants. It comes as little surprise that the Random Forest Ranker also delivers the most accurate results for performance estimates, but as these best values are more than 6% off on average for the full meta data and even more than 10% off on average for the reduced meta data, it is questionable whether these prediction can be useful. The usefulness is dependent on the task these predictions would be used for an whether if requires very accurate predictions of performance values.

The second interesting thing to consider is the ranking which is created by the best algorithm baseline. This ranking constitutes meta-knowledge about the learning process itself in that it shows which algorithms are generally a good choice for many data sets. This ranking is depicted in Table 4.6. The table is to be read top to bottom; of the 22 algorithms, the algorithm in the ith row is placed first on most data sets

Meta Feature Group	Computation Time (ms)			
	Min	Max	Mean	Stdv
NominalAttDistinctValues	0	16	0.127	1.128
SimpleMetaFeatures	0	16	0.221	1.614
Cardinality	0	19	0.654	2.723
Statistical	0	69	1.520	5.442
DecisionStump, 2 folds	0	54	1.536	4.710
RandomTreeDepth1, 2 folds	0	29	1.955	4.785
RandomTreeDepth2, 2 folds	0	18	2.114	4.793
RandomTreeDepth3, 2 folds	0	30	1.636	4.454
REPTreeDepth1, 2 folds	0	56	2.980	6.805
REPTreeDepth2, 2 folds	0	67	2.955	6.999
REPTreeDepth3, 2 folds	0	60	3.259	7.206
J48.001., 2 folds	0	201	5.129	13.935
J48.0001., 2 folds	0	101	4.944	10.657
J48.00001., 2 folds	0	116	4.464	10.744
NaiveBayes, 2 folds	0	200	6.853	17.679
kNN1N, 2 folds	0	1118	22.250	69.212
CfsSubsetEval kNN1N, 2 folds	2	1096	25.806	52.236
CfsSubsetEval NaiveBayes, 2 folds	2	132	23.304	12.918
CfsSubsetEval DecisionStump, 2 folds	2	150	23.087	12.296

Tab. 4.4.: Computation times for groups of meta features

when compared to the next 22-i algorithms. The associated number represents the number of data sets on which the algorithm was placed first. It is, however, not a particularly surprising result, as for example the placement of Random Forest as the first algorithm was to be expected. Random Forest is a learning algorithm that is known to often deliver good performances, which was proven again in the results discussed previously - out of the reviewed ranking approaches, Random Forest was the best choice.

Ranker	Root Mean Square Error			
	Min	Max	Mean	Stdv
LinearRegression	2.269	454695.445	1028.751	21505.865
M5P	1.487	50757.174	121.914	2400.412
RandomForest	0.938	23.788	6.625	3.873
REPTree	2.276	23.676	7.616	4.223
LinearRegression	3.503	372808.391	850.213	17632.512
M5P	1.934	561871.676	1275.212	26574.835
RandomForest	0.868	67.892	10.692	7.429
REPTree	1.573	78.048	13.492	8.815

Tab. 4.5.: Root mean square error of predicted performance values for the regression based rankers with full meta features (top) and no probing (bottom).

Classifier	Number of Data Sets Placed First
RandomForest	80
MultilayerPerceptron	66
BayesNet	51
Naive Bayes	448
Logistic	50
SGD	65
SMO	313
SimpleLogistic	131
LMT	120
IBk	66
KStar	95
DecisionTable	88
JRip	117
PART	104
J48	140
REPTree	110
RandomTree	133
OneR	177
DecisionStump	253
VotedPerceptron	255
ZeroR	378
NaiveBayesMultinomial	6

Tab. 4.6.: The ranking of classifiers generated by the baseline.

5 Related Work

The demand for aid in the process of selecting an algorithm has already led to the development of numerous solutions that automate machine-learning (AutoML). In the following sections, the workings of a few such tools that are related to this work are outlined briefly, loosely organized by their scope of operation. Each tool's usage of meta knowledge is discussed shortly.

5.1 Predicting Rankings

In contrast to ranking solely based on classifier performances, Abdulrahman et al. investigated an approach to extend existing ranking methods by incorporating the time needed for the evaluation of the classifiers [Abd+18]. They call this combined measure of accuracy and time A3R, and integrate it into two different ranking approaches. In both cases, the authors find that this leads to a better mean interval loss, which is a measure that expresses the mean loss experienced in respect to the time that has been allowed for the construction of the ranking, that is time taken to evaluate top-ranked classifiers.

One of those two approaches is called average-ranking. It utilizes meta knowledge to suggest a ranking of classifiers for a new data sets, which in this case consist of recorded past performances of classifiers on data sets. That means this algorithms aggregates the performances of classifiers on all rankings once and then always recommends the same ranking. The new measures is integrated here by, instead of ordering only according to performance values, the classifiers are ordered according to the combined measure A3R.

The second approach does not initially use meta knowledge and is called active testing. Active testing is a strategy that 'intelligently select the most useful tests' [Abd+18] to perform on a given data sets. Tests in this context mean the evaluation of a classifier that is considered for the ranking on the given data set. The authors aim to improve this strategy by reducing the time needed to identify a good algorithm by, similarly to the rank aggregation, incorporating time in the estimations made by the strategy which previously only compared accuracy values.

5.2 Algorithm and Hyperparameter Selection

Taking it a step further than predicting rankings of classification algorithms with fixed hyperparameters are tools that in addition two selecting an algorithm also optimize its hyperparameters, which has been defined as 'the combined algorithm selection and hyperparameter optimization problem (short: CASH)' [Tho+13]. Two widely used approaches of this kind are AUTO-WEKA and AUTO-SKLEARN. It has to be noted that thus these approaches also go further than the solution proposed in this thesis, which currently only takes into account one fixed hyperparameter configuration for each classification algorithm.

Auto-WEKA is an AutoML tool that both selects a machine learning algorithm and optimizes its hyperparameters by using Bayesian optimization [Tho+13]. It was first released in 2013 as an extension to the popular data mining software WEKA [Hal+09], which also offers a user-friendly GUI in addition to a command-line interface and an API, 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 [Kot+16]. 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' [HHL11]. 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 Forest, 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 [Feu+15]. 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.

5.3 Constructing pipelines

Before a classifier is evaluated on a data set, often a number of pre-processing steps are executed first. These could include selecting promising features and discarding other and normalizing the data. The 'sequence of tasks that need to be performed to classify instances belonging to a given dataset into a set of predefined categories' [Sá+17] can therefore be defined as a machine learning pipeline. Two tools which construct such complete pipelines for data sets are MP-Plan and the RECIPE framework.

ML-Plan is an AutoML tool that instead of concentrating on hyperparameter optimization, aims to optimize the whole machine-learning pipeline [WMH]. 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. For the actual implementation of the learning algorithms, WEKA is used. 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.

Similar to ML-PLAN, the RECIPE framework constructs whole classification pipelines for new problems [Sá+17]. However, this is done by means of grammar-based genetic programming: the tasks that the pipeline is composed of are represented by

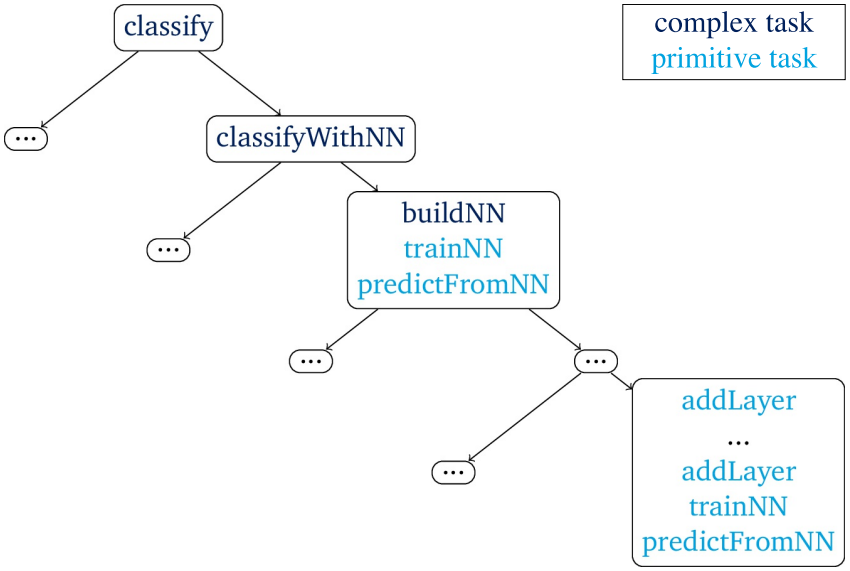

Fig. 5.1.: An example for how the complex task 'classify' might be broken down by ML-Plan, adapted from [WMH]. Nodes containing '...' represent an undefined number of subtrees.

a grammar, which 'is used to generate the initial population, as well as to constraint the crossover and mutation operations, which always need to be valid according to the grammar' [Sá+17]. This means that generated individuals representing complete pipelines can never be invalid. So for a new data set combined with a grammar that represents the valid pipelines, the tool first initializes the first generation according to the grammar. The fitness of individuals is then evaluated by mapping them to their respective implementations in the scikit-learn framework, and a new generation is created that takes this newly gained information into account. Evaluation by the authors indicates that the RECIPE framework is able to compete with AUTO-SKLEARN and a different evolutionary approach, although it does not incorporate meta-knowledge in the search.

6 Conclusion

In this thesis, the problem of predicting a ranking of classification algorithms for a new data set on the basis of meta features of the data set and past performances of the algorithms has been considered. Being able to predict a ranking of such algorithms is desirable since this potentially speeds up and simplifies the process of algorithm selection, which is important due to a rapidly increasing amount of available data, and more importantly, data that is available but has not yet been analyzed. This problem has been addressed by implementing two different approaches, regression-based and preference-based ranking. Both have been evaluated extensively against a baseline and an oracle.

The evaluation showed a significant advantage regarding the Kendall rank correlation for all tested methods in comparison to the best algorithm baseline, except instance based label ranking in the standard configuration and label ranking by pairwise comparison for the full meta data. However, regarding the practically possibly more important measure of best three loss, only the three regression based variants with linear regression, M5P and random forest as the regression learners can hold a significant advantage. When evaluating best three loss on reduced meta data that does not incorporate probing, although the evaluated data indicates a slight advantage of those same regression models over the baseline still, it is not statistically significant anymore. Furthermore, some label ranking methods even show a significant disadvantage in comparison to the baseline for the best three loss and loss for both evaluation on the full and reduced meta data. Thus it can be included that a regression based approach seems to be generally more promising than a preference based approach, but is not better than the baseline regarding the best three loss when evaluated on the selected meta data without probing. Out of the regression based variants, random forest seems to be the most promising one.

The main limitation of the findings in this thesis is that the data sets used in the evaluation process where not very large, being limited to a maximum of 100 features and 1000 instances. Often, data sets can get much larger and the question whether this ranking approach would then still pose an efficient regarding calculation of meta features and prediction times and, maybe even more importantly, an accurate enough solution for predicting classifiers remains.

On the basis of these results, it can be concluded that a there is an indication of a connection between certain meta features of a data set and the predictive accuracy of classification algorithms for this data set, which can be exploited to a degree by regression models and label ranking models to predict a ranking of classification algorithms. A practical application of these findings, apart from the direct use as a ranker, may be to incorporate the implementation or parts of it in another Auto-ML tool as a search heuristic, similar to how some Auto-ML solutions like AUTO-SKLEARN already benefits from meta learning [Feu+15]. However, some additional work may have to be done in extension to this thesis in order for a sensible integration.

6.1 Future Work

As the results have shown that the approach taken in this thesis has worked out reasonably well, for example when considering the regression based approach using random forests, many possibilities for additional work arise. An intuitive thought would be to extend the evaluation of the tool, for instance in terms of meta features. Additional sets of meta features could be evaluated, like adding implementations for additional meta features, or considering using only light probing by incorporating the performance values for usually cheap algorithms in the meta features. Since the significant advantage of any approach was lost regarding best three loss when moving from the full meta data to the reduced meta data without probing, this might be especially interesting. One could consider adding the meta feature groups for DecisionStump with two folds, RandomTree with a depth of two with two folds, REPTree with a depth of two and two folds, and J48.0001 with two folds, which would add approximately 12 milliseconds to the already existing 3 milliseconds of calculation times for the meta features in the mean, which seems to be reasonable. An effect on the time and quality of predictions for larger data sets would have to be examined still.

Furthermore, more insights could be gained about whether this approach is competitive by comparing it with other ranking approaches, for example like the one proposed by Abdulrahman et al. [Abd+18]. In this regard it would also be interesting to implement the loss - and time - loss curves proposed by the authors, which show how loss decreases as one advances in the predicted ranking and tries out the recommended algorithms, and at what cost in the sense of time the sampling of algorithms comes respectively.

Additionally, so far the evaluation of the tool has been confined to predicting algorithms for classification. Since this has been relatively successful, it could be

tested whether similar predictions can also be made for other machine learning tasks like regression or clustering with the same approach by extending the tool to handling these functions. Also, one could try to include a few parameterized versions of algorithms, as so far only one variant of each classifier with standard hyperparameters has been used.

In a similar manner to extending the algorithms for which a performance measure is predicted, one could use the tool to predict different performance measures. For this, the tool does not even need to be adapted, as it is agnostic to the semantics behind the measure it predicts, it only needs to be numeric. One possibility for this would be the prediction of build or prediction times of learning algorithms, or a measure that combines time and accuracy like A3R [Abd+18].

Depending on the fact whether one is willing to trade-off longer build and prediction times, it would also be possible to fit more sophisticated regression models for the regression based approach. This could include feature pre-processing, automating the choice of regression algorithm with one of the available tools [Tho+13] [Feu+15], or even using a whole pipeline prediction tool [WMH] [Sá+17]. Furthermore, the building and predictions for the regression models could be parallelized and evaluated on a cluster to simulate and test the usage of the ranking tool in a productive environment.

Last, it could be interesting to implement an additional baseline that averages ranks instead of counting how many times an algorithm is ranked first. This would help to determine if the ranking returned by the best algorithm baseline (see Fig. 4.6) still holds when using a different baseline. Naturally this new baseline then could also be used to re-evaluate the implemented ranking concepts.

Appendix

A

Meta Feature Group
NominalAttDistinctValues
MaxNominalAttDistinctValues, MinNominalAttDistinctValues, MeanNominalAttDistinctValues, StdvNominalAttDistinctValues
SimpleMetaFeatures
NumberOfInstances, NumberOfFeatures, NumberOfClasses, Dimensionality, NumberOfInstancesWithMissingValues, NumberOfMissingValues, PercentageOfInstancesWithMissingValues, PercentageOfMissingValues, NumberOfNumericFeatures, NumberOfSymbolicFeatures, NumberOfBinaryFeatures, PercentageOfNumericFeatures, PercentageOfSymbolicFeatures, PercentageOfBinaryFeatures, MajorityClassSize, MinorityClassSize, MajorityClassPercentage, MinorityClassPercentage, AutoCorrelation
Cardinality
MeanCardinalityOfNumericAttributes, StdevCardinalityOfNumericAttributes, MinCardinalityOfNumericAttributes, MaxCardinalityOfNumericAttributes, MeanCardinalityOfNominalAttributes, StdevCardinalityOfNominalAttributes, MinCardinalityOfNominalAttributes, MaxCardinalityOfNominalAttributes, CardinalityAtTwo, CardinalityAtThree, CardinalityAtFour
Statistical
MeanMeansOfNumericAtts, MeanStdDevOfNumericAtts, MeanKurtosisOfNumericAtts, MeanSkewnessOfNumericAtts, MinMeansOfNumericAtts, MinStdDevOfNumericAtts, MinKurtosisOfNumericAtts, MinSkewnessOfNumericAtts, MaxMeansOfNumericAtts, MaxStdDevOfNumericAtts, MaxKurtosisOfNumericAtts, MaxSkewnessOfNumericAtts, Quartile1MeansOfNumericAtts, Quartile1StdDevOfNumericAtts, Quartile1KurtosisOfNumericAtts, Quartile1SkewnessOfNumericAtts, Quartile2MeansOfNumericAtts, Quartile2StdDevOfNumericAtts, Quartile2KurtosisOfNumericAtts, Quartile2SkewnessOfNumericAtts, Quartile3MeansOfNumericAtts, Quartile3StdDevOfNumericAtts, Quartile3KurtosisOfNumericAtts, Quartile3SkewnessOfNumericAtts
Landmarkers[1]
DecisionStump, RandomTreeDepth1, RandomTreeDepth2, RandomTreeDepth3, REPTreeDepth1, REPTreeDepth2, REPTreeDepth3, J48.001., J48.0001., J48.00001., NaiveBayes, kNN1N, CfsSubsetEval kNN1N, CfsSubsetEval NaiveBayes, CfsSubsetEval DecisionStump

Tab. A.1.: The meta features computed by each meta feature group. More details can be found on www.openml.org.

[1] Each landmarker represents its own group of meta features consisting of AUC, ErrRate and Kappa. All are computed with 2-fold crossvalidation.

Bibliography

[Abd+18] Salisu Mamman Abdulrahman, Pavel Brazdil, Jan N. van Rijn, and Joaquin Vanschoren. „Speeding up algorithm selection using average ranking and active testing by introducing runtime“. In: *Machine Learning* 107.1 (2018), pp. 79–108 (cit. on pp. 29, 34, 35).

[ALM12] YS Abu-Mostafa, HT Lin, and M Magdon-Ismail. „Learning from Data: A Short Course: AMLbook“. In: *View Article PubMed/NCBI Google Scholar* (2012) (cit. on pp. 5–7).

[Bra+08] Pavel Brazdil, Christophe Giraud Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008 (cit. on p. 9).

[Egg+15] Katharina Eggensperger, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. „Efficient Benchmarking of Hyperparameter Optimizers via Surrogates“. In: *AAAI*. AAAI Press, 2015, pp. 1114–1120 (cit. on p. 2).

[Feu+15] Matthias Feurer, Aaron Klein, Katharina Eggensperger, et al. „Efficient and robust automated machine learning“. In: *Advances in Neural Information Processing Systems*. 2015, pp. 2962–2970 (cit. on pp. 30, 34, 35).

[FH10] Johannes Fürnkranz and Eyke Hüllermeier, eds. *Preference Learning*. Springer, 2010 (cit. on pp. 7, 8).

[GR12] 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 (cit. on p. 1).

[GR17] 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 (cit. on p. 1).

[Hal+09] Mark Hall, Eibe Frank, Geoffrey Holmes, et al. „The WEKA data mining software: an update“. In: *ACM SIGKDD explorations newsletter* 11.1 (2009), pp. 10–18 (cit. on pp. 16, 19, 30).

[Har99] Mark Harris. *Splice-2 comparative evaluation: Electricity pricing*. Tech. rep. The University of South Wales, 1999 (cit. on p. 1).

[HHL11] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. „Sequential Model-Based Optimization for General Algorithm Configuration.“ In: *LION* 5 (2011), pp. 507–523 (cit. on p. 30).

[Ken38] Maurice G Kendall. „A new measure of rank correlation“. In: *Biometrika* 30.1/2 (1938), pp. 81–93 (cit. on p. 10).

[Kot+16] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. „Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA“. In: *Journal of Machine Learning Research* 17 (2016), pp. 1–5 (cit. on p. 30).

[LBV12] Rui Leite, Pavel Brazdil, and Joaquin Vanschoren. „Selecting Classification Algorithms with Active Testing“. In: *MLDM*. Vol. 7376. Lecture Notes in Computer Science. Springer, 2012, pp. 117–131 (cit. on p. 11).

[MW47] Henry B Mann and Donald R Whitney. „On a test of whether one of two random variables is stochastically larger than the other“. In: *The annals of mathematical statistics* (1947), pp. 50–60 (cit. on p. 12).

[Ric76] John R Rice. „The algorithm selection problem“. In: *Advances in computers*. Vol. 15. Elsevier, 1976, pp. 65–118 (cit. on p. 9).

[Rij+14] Jan N van Rijn, Geoffrey Holmes, Bernhard Pfahringer, and Joaquin Vanschoren. „Algorithm selection on data streams“. In: *International Conference on Discovery Science*. Springer. 2014, pp. 325–336 (cit. on p. 18).

[Sá+17] Alex Guimarães Cardoso de Sá, Walter José G. S. Pinto, Luiz Otávio Vilas Boas Oliveira, and Gisele L. Pappa. „RECIPE: A Grammar-Based Framework for Automatically Evolving Classification Pipelines“. In: *EuroGP*. Vol. 10196. Lecture Notes in Computer Science. 2017, pp. 246–261 (cit. on pp. 31, 32, 35).

[Sie87] J Paul Siebert. *Vehicle recognition using rule based methods*. Tech. rep. TIRM87018. Turing Institute, 1987 (cit. on p. 1).

[Tho+13] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. „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 (cit. on pp. 30, 35).

[Van+13] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. „OpenML: Networked Science in Machine Learning“. In: *SIGKDD Explorations* 15.2 (2013), pp. 49–60 (cit. on p. 16).

[WMH] 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 (cit. on pp. 31, 32, 35).

Webpages

[36] *iris*. 1936. URL: `https://www.openml.org/d/61` (visited on Feb. 11, 2018) (cit. on p. 6).

[Cac17a] Miguel Cachada. *Run 2210139*. May 17, 2017. URL: `https://www.openml.org/r/2210139` (visited on Nov. 30, 2017) (cit. on p. 2).

[Cac17b] Miguel Cachada. *Run 2221382*. May 18, 2017. URL: https://www.openml.org/r/2221382 (visited on Nov. 30, 2017) (cit. on p. 1).

[Cac17c] Miguel Cachada. *Run 2290623*. May 20, 2017. URL: https://www.openml.org/r/2290623 (visited on Nov. 30, 2017) (cit. on p. 1).

[Cac17d] Miguel Cachada. *Run 2290766*. May 20, 2017. URL: https://www.openml.org/r/2290766 (visited on Nov. 30, 2017) (cit. on p. 1).

[Int17] IntelligentSystemsGroup. *jPL framework*. June 30, 2017. URL: https://github.com/Intelligent-Systems-Group/jpl-framework (visited on Nov. 30, 2017) (cit. on pp. 16, 19).

[Int18] IntelligentSystemsGroup. *Evaluation Engine*. Jan. 17, 2018. URL: https://github.com/openml/EvaluationEngine (visited on Feb. 11, 2018) (cit. on p. 16).

[nd] *OpenML Bootcamp*. n.d. URL: http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm (visited on Feb. 11, 2018) (cit. on p. 18).

List of Figures

List of Tables

Colophon

This thesis was typeset with LaTeX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

I hereby declare that I prepared this thesis independently and without illicit assistance and have not used any sources without declaration. Any statements that have been adopted literally or analogously have been identified as such. This thesis has not been submitted in the same or substantially similar version, not even in part, to another authrity for grading and has not been published elsewhere.

Declaration (German)

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

Paderborn, January 5, 2018

Helena Graf