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
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

Should abstract be included?

Abstract (German)

Und falls ja, dann zweisprachig?

Keywords: Automated Machine Learning, Meta Learning, Regression, Label Ranking, Algorithm Ranking, Algorithm Selection

Acknowledgement

Including acknowledgement expected or frowned upon for undergrad thesis?

Contents

1 Introduction

This introduction emphasizes why there is a need for automation in machine learning, and how this thesis relates to that problem.

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 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

[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).
[3] Hyperparameters as above.

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.

Since there is a need for automated machine learning and selecting a fitting classification algorithm for a new data set is an important but difficult task in the machine learning pipeline, facilitating this process by automatically suggesting well-performing classifiers for a new data set is a significant problem. 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 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.

Thus, the aim of this thesis is to

algorithm selection problem at heart but want ranking to allow the user iterative testing

could work because regression models have been used successfully to predict the performance of an algorithm dependent on the hyperparameter configuration [Egg+15]

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.

Chapter 4 - Implementation

Following the details of the approach, the implementation thereof is presented. Used software libraries are pointed out.

Chapter 5 - 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 6 - 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 7 - 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, firstly the concept of machine learning and the

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. 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 the 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 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 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 $(x_1, y_1), \ldots, (x_n, y_n)$ from f such that $f(x_i) = y_i$. Together, they form a data set D

Feature	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.: Example values for the predictive accuracy of classifiers (data set not relevant in this context). The predictive accuracy denotes the percentage of instances for which the classifier correctly perdicted the class membership.

which is one of all possible data sets $\mathbb{D}$ for the problem. He then chooses 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 $A : \mathbb{D} \to H$.

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. 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, 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 consist of a finite set of labels $\{y_1, \ldots, y_n\}$, so that each instance 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. 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 to 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.

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.

Ranking = Permutation of the actual labels! Ordering =

A ranking for a set of items defines a strict total order

The ordering sorts the labels according to their given score. It thus may also implicitly be given by ordering the items themselves, as the labels are aliases for them. Therefore, an ordering for a set of labels $\{1 \ldots k\}$ is a permutation π of the labels where $\pi(i)$ is the

It must be noted that the definitions given here are not transferable to the general case of a ranking or ordering problems, since a strict total order is not required in all contexts for either ranking or ordering.

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.: Example values for the predictive accuracy of classifiers (data set not relevant in this context). The predictive accuracy denotes the percentage of instances for which the classifier correctly perdicted the class membership.

The values in table **??** result in the ranking $[2, 3, 1]$, whereas the ordering is $[3, 1, 2]$.

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 $X \rightarrow S_Y$ where the output space S_Y contains all permutations over the set of labels Y [FH10]. An instance $x \in X$ therefore is assigned a permutation $y_{\pi_x^i}$

Although called label ranking actually label ordering!

2.1.4 Meta Learning

2.2 Evaluation

2.2.1 Kendall Rank Correlation Coefficient

To evaluate the quality of the prediction that the ranker will make, a statistic is needed to compare a predicted ranking with other rankings like a baseline or an optimal ranking. serves as indication of causal connection

The Kendall Rank Correlation Coefficient is a measure of association between two rankings of the same items also known as Kendall's tau. For two independent variables X and Y and and observations of values $(x_1, ..., x_m)$ for X and $(y_1, ..., y_n)$ 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

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

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

2.2.2 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 b expressed in measure called loss [**DBLP:conf/mldm/LeiteBV1**], 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 among the considered set of classifiers.

2.2.3 Mann-Whitney U Test

too compare the ranking alternatives and to the basline to see if statistically significantly better.

3 Approach

Approach of ranking classifier implementations here realized by two concepts, regression-based and preference-based ranking, which will be explained in detail. Also predicting predictive accuracy here. Thus target function of the form $metafeatures^n \rightarrow classifiers^m$. performance measure chosen predictive accuracy

Since the aim is to generate a ranking of classification algorithms, first, performance values of a number of classifiers are recorded on a few data sets. Then, meta features are computed for each data set. This is for training the ranking model. This generates a table that contains all necessary informatino for training both types of rankers. A ranking of the classifiers for a new instance is then achieved by computing the meta features for the data set and subsequently uing it to query the underlying model.

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.: This is a caption text.

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 into separate data sets that each contain all meta features for all data sets, but only the performance values of one classifier. 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 $metafeatures \rightarrow classifierperformancevalue$ 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.

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.: This is a caption text.

The second possibility considered in this context is using preference learning to predict a ranking. Each instance of the data set generated beforehand contains meta feature information for the considered data set and performance values of all classifiers. Similarly to the regression alternative, the performance values can be converted to preference information by sorting the corresponding classifiers by their performance values. 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 mode which is fed the computed meta data for a new data set and returns an ordering of classifiers. Thereby it attempts to learn the mapping from meta features to an ordering of classifiers directly.

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. 3.3.: This is a ranking

META FEATURES USED

4 Implementation

4.0.1 Frameworks used

4.0.2 jPL framework

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. Furthermore, the framework introduces the Generic Preference Representation Format (GPRF) to store preference information.

4.0.3 WEKA

4.1 Generating Performance Values

Performance values generated by

4.2 Stuff

Rankers trained with data set of the following form, so each ranking algorithm can derive its on performance

5 Evaluation

5.1 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 a ranking returned by this oracle represents the ideal solution,

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. The baseline is evaluated in the same way as the preference rankers; as it does not predict performance values, the RMSE of predict performance values cannot bet computed. Ideally, rankings returned by the preference and regression based rankers should then be statistically significant better than the baseline, and come as close as possible to the correct ranking.

5.2 Experimental Setup

Performance Values acquired with 5 times MCCV

5.3 Results

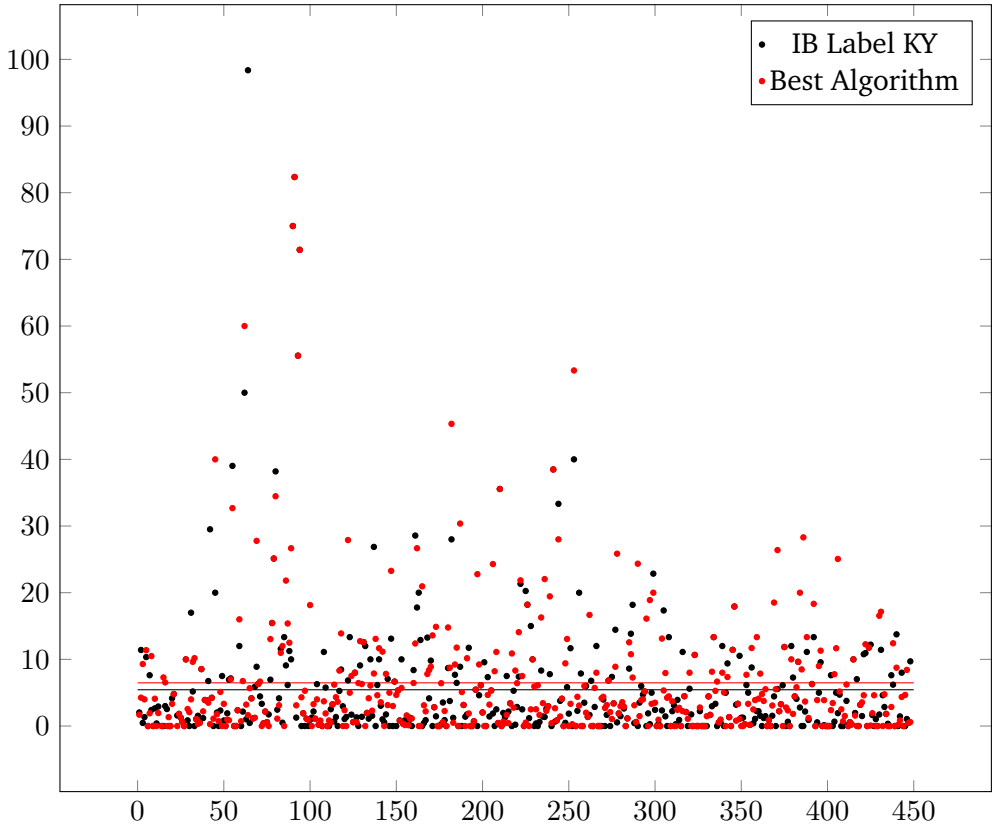
5.3.1 Accuracy

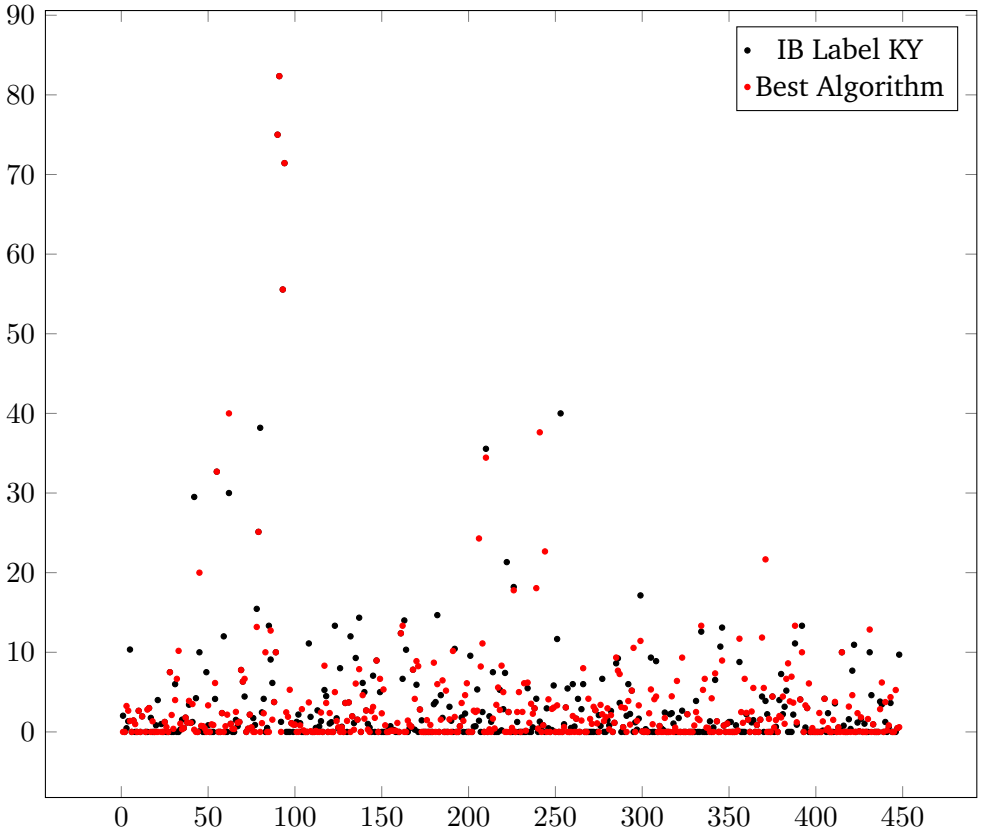
Ranker	Kendall's Rank Correlation				Loss				BestThreeLoss			
	Min	Max	Mean	Stdv	Min	Max	Mean	Stdv	Min	Max	Mean	Stdv
LinearRegression	-0.255	0.896	0.473	0.221	0	86.667	3.469	7.244	0	31.220	1.267	3.011
M5P	-0.29	0.870	0.470	0.219	0	82.353	3.78	6.535	0	82.353	1.508	4.757
RandomForest	-0.281	0.922	0.495	0.228	0	60	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	9.44	0	82.353	3.62	8.540
InstanceBased[1]	-0.429	0.887	0.340	0.252	0	98.367	5.437	10.323	0	82.353	3.294	8.367
InstanceBased[2]	-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	-0.437	0.489	0.057	0.159	0	82.353	6.480	10.168	0	82.353	3.420	8.195

Tab. 5.1.: Evaluation results with full meta data

[1] Rankaggregation: Kemeny-Young

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


IB Label KY
Best Algorithm
100
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
350
400
450


IB Label KY
Best Algorithm
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
350
400
450

5.3.2 Time

Ranker	Ranker Build Time (ms)				Ranker Prediction Time (ms)			
	Min	Max	Mean	Stdv	Min	Max	Mean	Stdv
LinearRegression	1454	2060	1580	36	0	86.667	3.469	7.244
M5P	3145	4916	3226	89	0	82.353	3.78	6.535
RandomForest	6048	9720	6236	259	0	60	3.097	5.745
REPTree	599	1264	629	38	0	82.353	4.829	7.948
InstanceBased	66	550	90	25	0	82.353	5.401	9.44
InstanceBased[3]	66	138	88	12	0	98.367	5.437	10.323
InstanceBased[4]	66	163	87	12	0	82.353	5.382	9.402
PairwiseComparison	11784	15096	12623	348	0	97.822	9.762	13.135
BestAlgorithm	193	321	221	23	0	82.353	6.480	10.168

Tab. 5.2.: Build and Prediction Times of the Rankers in ms

5.3.3 Further Insights

Ranker	Ranker Build Time (ms)			
	Min	Max	Mean	Stdv
LinearRegression	1454	2060	1580	36
M5P	3145	4916	3226	89
RandomForest	6048	9720	6236	259
REPTree	599	1264	629	38

Tab. 5.3.: Root Mean Square Error of Predicted Performance Values for the Regression-Based Rankers

[3] Rankaggregation: Kemeny-Young

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

Classifier	Number of Data Sets Placed First
Logistic	28
Naive Bayes	29
IBk	23
KStar	24
LMT	17
VotedPerceptron	16
ZeroR	14
J48	13
NaiveBayesMultinomial	6
RandomTree	2
SimpleLogistic	1
DecisionStump	1
MultilayerPerceptron	1
RandomForest	1
DecisionTable	1
PART	1
SGD	1
BayesNet	1
REPTree	1
JRip	1
SMO	313
OneR	448

Tab. 5.4.: The ranking of Classifiers returned by the Best Algorithm Baseline

6 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.

6.1 Predicting Rankings

One method of ranking algorithms called average-ranking utilizes meta-knowledge to suggest a ranking for a new data set. The meta-knowledge here consists of recording past performances of classifiers on data sets, and computing the average rank of each classifier over all data sets. Performance measures used are often related to the predictive accuracy of the classifier.

[Abd+18]

6.2 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.

6.3 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

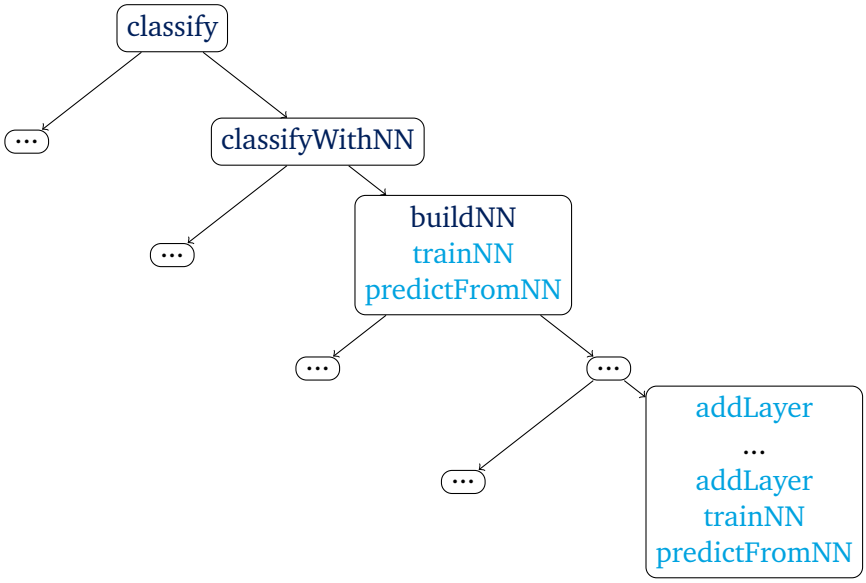


Fig. 6.1.: An example for how the complex task 'classify' might be broken down by ML-Plan. The figure is loosely adapted from [WMH]. Nodes containing '...' represent an undefined number of subtrees. Complex tasks and primitive tasks are distinguished by their color.

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 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.

7 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. The problem has been addressed by implementing to different approaches, regression-based and preference-based ranking, and the evaluations of both against each other, a baseline and an oracle.

- Results -

On the basis of the results, it can be concluded that a causal connection exists 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 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.

7.1 Future Work

- use this to try to predict other measures (e.g. time needed) - look at the loss curve + loss-time (log) curve - further investigate the better solution - integrate solution into other auto ml solution if sensible regarding time contraints for that solution (and accuracy) -e.g. use A3R - or runtime could be predicted separately by nn based on meta data [ELH17] - adding feature pre-processing

- better fitting of learning algorithms to meta data (more manual work, but of course consider danger of overfitting!

- compare against other tools that rank r.g. the Jan van Rijn one - deeper analysis of meta features, e.g. add some, remove some, consider trade-off time accuracy

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.

- hyperparameters neglected here, may include standard combinations in future work (or random ones)-> possible only if id by string - use this for regression

an average baseline like used in speeding up algo select might be better

Appendix

A

A.1 Appendix Section 1

Alpha	Beta	Gamma
0	1	2
3	4	5

Tab. A.1.: This is a caption text.

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 p. 21).

[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).

[ELH17] Katharina Eggensperger, Marius Lindauer, and Frank Hutter. „Predicting Runtime Distributions using Deep Neural Networks“. In: *CoRR* abs/1709.07615 (2017) (cit. on p. 25).

[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. 22, 25).

[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 p. 21).

[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. 22).

[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. 22).

[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. 23, 24).

[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 p. 21).

[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 p. 23).

Webpages

[Cac17a] Miguel Cachada. *Run 2210139*. May 17, 2017. URL: `https://www.openml.org/r/2210139` (visited on Nov. 30, 2017) (cit. on p. 1).

[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).

List of Figures

List of Tables

Colophon

This thesis was typeset with LaTeX 2$_\varepsilon$. 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