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 (different language)

Und falls ja, dann zweisprachig?

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

to achieve the best results. In general, this means that for a different data set, a different algorithm might yield the best performance.

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

1.2 Thesis Structure

Chapter 2

Firstly, relevant preliminaries are discussed. A brief overview of tasks from the field of machine learning which are relevant to this thesis is given.

Chapter 3

This chapter continues by describing the approach of this thesis in tackling the problem of suggesting classifies

Chapter 4

Following the details of the approach the implementation thereof is presented.

Chapter 5

Chapter 6

Chapter 7

The final chapter concludes the thesis and provides an outlook for possible future work.

2 Fundamentals

This chapter lays out the fundamentals for Firstly,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 are 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 Classifier Evaluation

2.2.2 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.

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 has been 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

2.2.3 Loss

2.2.4 Root Mean Square Error

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 $metafeatures^n \rightarrow classifiers^m$.

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. 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.

EXAMPLE TABLE ALL DATA

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. A regression model therefore tries to learn the target function $metafeatures \rightarrow classifierperformancevalue$ for its respective classifier. Hence for a query instance, after computing the meta features, these are fed into each regression model. In a second step, the classifiers are ordered according to the predictions made by the models.

EXAMPLE TABLE DERIVED DATA SETS and METAFEATURES -> PREDICTIONS -> COMBINED -> ORDERING

The second ranking possibility considered here is using preference learning to predict a ranking. Each instance of the data set generated beforehand contains meta feature information for the considered data sets and performance values of classifiers This implies that the preference learning task at hand is label ranking.

EXAMPLE TABLE CONVERTED INFORMATION

Fig. 3.1.: Figure example: *(a)* example part one, *(c)* example part two; *(c)* example part three

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

5 Evaluation

5.1 Experimental Setup

Both variants compared against Best Algorithm baseline, which was implemented as a Ranker of type PreferenceRanker itself and evaluated the same way as the other PreferenceRankers. This baseline is computed iteratively by training an oracle on the same data set and then

5.2 Results

6 Related Work

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

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] 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 Forests, which perform well on a large number of data sets, first.

AUTO-SKLEARN has been described as a sister-package to Auto-WEKA and is an AutoML tool which is based on scikit-learn, a machine learning library for Python [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.

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. 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.

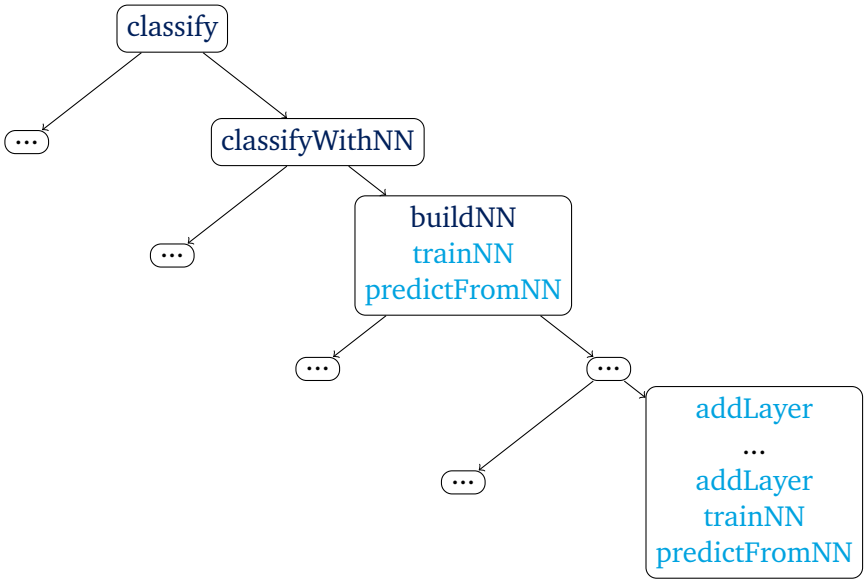

Fig. 6.1.: An example for how the complex task 'classify' might be broken down by ML-Plan. The figure is loosely adapted from [WMH]. Nodes containing '...' represent an undefined amount of subtrees. Complex tasks and primitive tasks are distinguished by their color.

7 Conclusion

7.1 Future Work

- deeper analysis of meta features, e.g. add some, remove some, consider trade-off time accuracy - 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 - more careful training by hand-selecting datasets for training (but then aains is this really the real world anymore? But then again duplicates may be contained) - hyperparameters neglected here, may include standard combinations in future work (or random ones)-> possible only if id by string - use this for regression - use this to try to predict other measures (e.g. time needed)

Example 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

[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 p. 15).

[FH10] Johannes Fürnkranz and Eyke Hüllermeier, eds. *Preference Learning*. Springer, 2010 (cit. on pp. 5, 6).

[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. 15).

[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. 15).

[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. 15).

[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. 15).

[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. 16, 17).

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