S. Krajči (ed.): ITAT 2018 Proceedings, pp. 44–51
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Jan Motl and Pavel Kordík

Do We Need to Observe Features to Perform Feature Selection?

Jan Motl, Pavel Kordík

Czech Technical University in Prague,
Thákurova 9, 160 00 Praha 6, Czech Republic,
jan.motl@fit.cvut.cz, pavel.kordik@fit.cvut.cz

Abstract: Many feature selection methods were developed in the past, but in the core, they all work the same way — you pass a set of features to the algorithm and get a reduced set of the features. But can we perform a non-trivial feature selection without first observing the features? This is an important question because if we were actually able to predict feature importance before observing the features, we would reduce computation requirements of all stages of machine learning process beginning with feature engineering. In this article, we argue that it is possible to predict feature importance before feature vector observation. The trick is that we use meta-features about the features to perform the feature selection. We evaluate the concept on 15 relational databases. On average, it was enough to generate the top decile of all features to get the same model accuracy as if we generated all features and passed them to the model.

Keywords: meta-learning, feature engineering, feature selection, relational database, propositionalization

1 Introduction

Data in relational databases are in the form of many tables, but common classification algorithms require input data in the form of a single table. Propositionalization solves this discrepancy by converting data from the form of many tables into a single table.

But there are two significant problems with the propositionalization [3]. It produces a lot of features. And many of them are redundant. These two issues result in high computational requirements during both, propositionalization and classification.

Contrary to the common approach (e.g., [10], [7], [8]), we deal with these two issues by performing feature selection *before* the propositionalization and not *after* the propositionalization. The key idea is that we collect metadata about the attributes in the database (e.g., attribute data type), meta-data about the feature generative functions (e.g., id of the feature function), calculate landmarking features on a small subset of all features and pass their performance to a meta-learner, which predicts the optimal order, in which the remaining features should be calculated.

2 Related Work

The presented work is at the border between feature engineering and feature selection. Hence, we review related work from both these disciplines.

2.1 Meta-learning for Feature Engineering

Meta-learning was originally concerned with algorithm selection[21]. Nevertheless, Nargesian [16] trained a neural network to predict, which feature transformations are going to improve the accuracy of a classifier based on the feature histograms.

We extend the idea of using the data-based meta-features (in Nargesian's case a histogram) for feature engineering with landmarking.

2.2 Meta-learning for Feature Selection

Reif [20] applies meta-learning to accelerate forward selection. The key concept is that the performance of all candidate feature subsets in each forward step is first estimated with a meta-learner. And only the top x percent of the candidates get evaluated on the data to get the true subset performance. Based on the reported results, it is sufficient to evaluate only the top 10% of all candidate subsets on the data to get results comparable to classical forward selection.

The difference between our approach and Reif's approach is that Reif calculates meta-features from the features, while we calculate meta-features directly from the attributes that are used to calculate the features (in Figure 1 we use only the left table, while Reif uses the right table). Consequently, in Reif's case, we have to calculate the features first, to perform feature selection. While in our case, we can perform the feature selection before feature calculation.

3 Method

A high-level schema of our approach is in Figure 2. The whole process is divided into two phases. During the offline phase, meta-features and feature performance are collected on many databases and passed to a meta-learner as training data. During the online phase, the trained meta-learner is used to rank candidate features in the descending order of their estimated utility. Following paragraphs define the *feature utility*.

There are many properties that a feature should posses [12], but we focus on predicting properties measurable directly from the data: relevance to the task, redundancy to other features and runtime of the feature calculation.

feature function

id	class	att1	att2	...	attn
1	+	10	apple	...	12:03
1	+	12	cinnamon	...	7:53
2	-	4	banana	...	19:21
3	-	3	cherry	...	12:20
3	-	6	banana	...	8:21

attribute space

$\longrightarrow$

id	class	feature
1	+	10
2	-	4
3	-	3

feature space

Figure 1: An example of a feature generative function *min* applied on attribute *att1*, which converts the multi-instance problem into a single-instance problem solvable with a common attribute value classifier. In this trivial example, the feature space contains only a single feature vector but it may generally contain thousands of feature vectors.

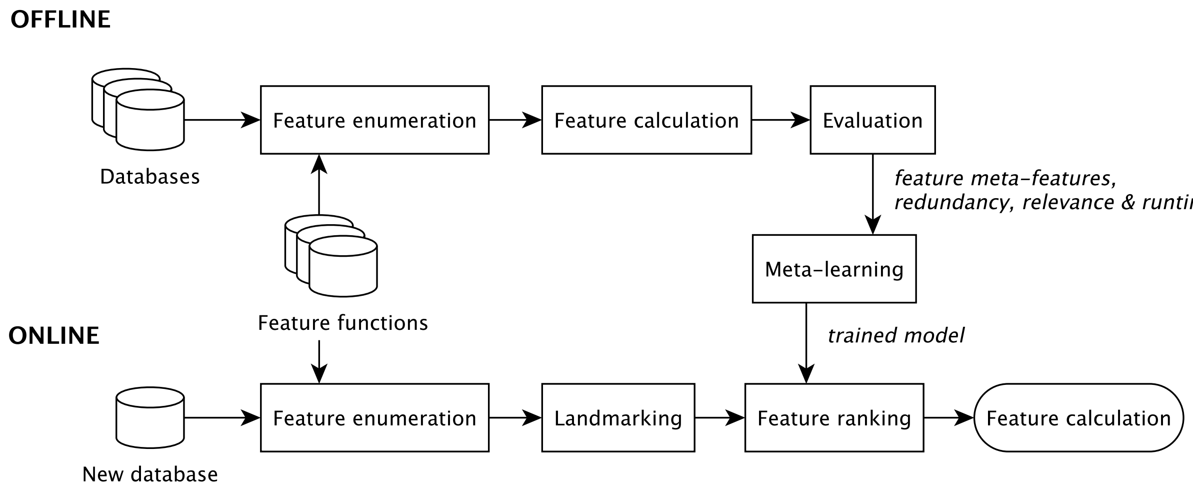

Figure 2: Flowchart of meta-learning on features.

Relevance Without loss of generality, we assume that we want to utilize the calculated features for classification. We use *Chi*^{2}*statistics* [4, Section A.6.1] between the feature and the label as the measure of the relevance (if the feature is continuous, we first discretize the feature with equal-width binning). But in theory, any other measure can be used.

Runtime The runtime is defined as the time needed to calculate a particular feature vector. If two feature vectors are otherwise identical, we prefer the one that has a smaller runtime.

Redundancy In the analyzed databases (discussed further in Section 4.1), 38% of all calculated features are redundant. We define that nominal feature f_1 is *redundant* to nominal feature f_2 iff a bijection exists between values in f_1 and f_2. A numerical feature f_1 is redundant to numerical feature f_2 if a linear transformation from f_1 to f_2 and back exists.

We use this (weaker) definition of redundancy instead of the *identity* of the features because it corresponds better with the notion of redundancy in many models (e.g., in logistic regression with one shot encoding of categorical features). To speed up the identification of redundant features, we use *Chi*^{2} as a hash function to identify potential redundant features [19, Section 2.1].

3.1 Feature Utility

We calculate features[1] in descending order of the estimated *relevance*/*runtime* ratio [1] since we prefer to calculate highly relevant and fast features first. Furthermore, we penalize the feature i proportionally to the estimated probability that the feature is redundant $\hat{p}_i$. Because each dataset has a different proportion of redundant features (see Table 1) and the tested meta-learning models had difficulties to model these differences, we employ median thresholding instead of a fixed threshold:

$$utility_i = (\hat{p}_i > median(\hat{p})\,?\,1-\hat{p}_i\,:\,1)\frac{relevance_i}{runtime_i}, \quad (1)$$

where *redundancy* is a vector of estimated redundancy probabilities for a database.

[1]In the production, we would calculate only the top n features that we would use to build a production classifier. But to demonstrate the meaningfulness of such approach, we calculate all features.

4 Experiment

4.1 Data

We used 15 databases listed in Table 1 from relational repository [15].

4.2 Features

For propositionalization, we used Relaggs [9], which was modified to work with 31 different feature (generative) functions, listed in Figure 2. The detail description of the employed feature functions is at `http://predictorfactory.com`.

4.3 Meta-features

We employ three sources of meta-features: landmarking features, database meta-data and feature function meta-data.

Landmarking features Just like the accuracy of a few classifiers can be used as meta-features for the recommendation of the best classifier on the data (e.g., [18]), we define a subset of feature functions as landmarking feature functions for the recommendation of the best features.

Without loss of generality, we used following set of landmarking features: Direct field (a simple copy of the value), Aggregate (e.g., *min*, *max*,...), WOE (Weight of Evidence), Count (of tuples), Aggregate WOE, Time aggregate since. These feature functions were selected for their low runtime (see Table 11 in the appendix) and good coverage of different data types (numerical/character/temporal) and relationships between the label and the data (1:1/1:n). Note that we do not use multivariate feature functions for landmarking due to the potential combinatorial explosion.

Database meta-data Basic descriptive and statistical meta-features are frequently employed in meta-learning (e.g., [11]) and we do not differ in this respect. A noteworthy difference is that we do not calculate statistics of the attributes but rather reuse statistics maintained by the relational database for query plan optimization [14]. This slight deviation allows us to collect estimates of the statistics in time independent on the count of tuples (records) in the database.

Feature function meta-data Feature function meta-data consists of feature function name (e.g., Aggregate) and feature function parameters (e.g., *min*).

4.4 Measures

Anytime algorithm We formulate feature engineering as anytime algorithm [25], which aims to deliver the best subset of calculated features in any time. The quality of anytime algorithm can be expressed with a performance profile, where we measure quality of the solution at the given time (see example in Figure 3). To assign a single number to the performance profile, we calculate the area between the *archived* curve $a(t)$ and the expected *random* curve $r(t)$ (which we obtain from averaging the curve from many random permutations), divided by the area between the *perfect* curve $p(t)$ and the expected random curve $r(t)$:

$$POP = \frac{\int a(t)dt - \int r(t)dt}{\int p(t)dt - \int r(t)dt}, \tag{2}$$

where t is time. The obtained ratio then represents the "percentage of perfect" solution [2]. In our case, $a(t)$, $r(t)$ and $p(t)$ are the Chi^2 of the feature calculated at time t. The only difference between these functions is then the order, in which the features are calculated. The perfect feature ordering is based on a complete knowledge of relevance, redundancy and runtime of all the features. While archived ordering is based only on the estimates of these feature properties (the only exception are landmarking features, which are calculated in a pseudorandom order).

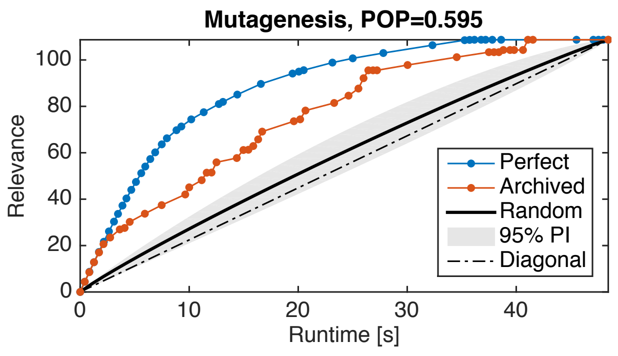

Figure 3: Performance profile. The shaded area represents the 95% prediction interval for a random curve.

Individual models To assess the ability of relevance and runtime prediction models to rank, we use Spearman correlation (ρ). The quality of redundancy estimation (a classification task) is evaluated with area under receiver operating characteristic curve (AUROC).

4.5 Methodology

Algorithms were evaluated with leave-one-out validation, where all but the tested database was used for the training of the models. Obtained accuracies are reported for three different algorithms: *generalized linear model* (GLM), *gradient boosting machine* (GBM) and *deep learning* (DL), all from H2O.

Permutation testing To assess, whether the obtained performance profiles are significantly better than random, we generate 1,000 random orderings of the features to estimate 95% prediction intervals.

Table 1: Used databases. The range of relevant features is estimated with forward & backward selection with a decision tree (the percentage of features when meta-learning feature selection reaches accuracy corresponding to accuracy obtained on all the features).

Database	Domain	Attributes	Features	Redundant [%]	Relevant [%]
Accidents	Government	43	305	39	2–12 (7)
AustralianFootball	Sport	77	794	45	1–6 (1)
BasketballMen	Sport	195	865	41	1–49 (1)
Biodegradability	Medicine	17	71	25	6–66 (4)
Chess	Sport	45	127	16	65–72 (91)
Financial	Financial	55	493	32	1–59 (7)
Hepatitis	Medicine	26	152	42	4–42 (5)
Mondial	Geography	167	1524	45	1–9 (1)
Mutagenesis	Medicine	14	65	40	6–46 (6)
Nations	Geography	118	191	76	2–21 (3)
PremierLeague	Sport	217	667	23	2–27 (7)
PTE	Medicine	76	691	58	1–33 (1)
StudentLoan	Education	15	41	7	15–66 (21)
VisualGenome	Education	20	42	64	10–10 (14)
Walmart	Retail	27	545	19	1–22 (4)
average		74	438	38	8–36 (11)

5 Results

First, we report the accuracy of the individual models. Second, we comment on the meta-feature importance as reported by L1 & L2 regularized GLM. Third, we report the obtained POPs.

5.1 Accuracy

The obtained accuracies are depicted in Table 3. Since the difference between the models is not significant, we use GLM for all following experiments.

5.2 Feature Importance

Relevance The most important meta-features for feature relevance prediction is the average relevance of the landmarking features on individual attributes and the type of the employed feature function (see Table 4).

Redundancy There are two main sources of redundant features [10]: redundancy in the input data and redundancy introduced by the feature functions. The redundancy in the input data is covered by landmarking *landmark_is_redundant* and *data_type*. While the introduced redundancy is explained with *feature_function* and *feature_parameters* (see Table 5).

Runtime The runtime of a feature function calculation is a function of two factors: the type of the feature function and data property. Nevertheless, these two factors are dominated by the landmarking *landmark_runtime* (see Table 6).

5.3 Percentage of Perfect

The quality of anytime learning for all 15 datasets is reported in Table 7 in the penultimate column.

6 Discussion

6.1 What is the contribution of the individual models to POP?

To evaluate the contribution of the individual models to POP, we performed an experiment with a 2-level full factorial design for presence/absence of runtime, relevance and redundancy models (8 combinations in total) on all databases. To deal with the variability across databases (some are easier than others), we treat the database name as a random factor.

Conclusion: The result of the factor analysis is in Table 8. As expected, the intercept is not significantly different from zero, since POP measure should on average be 0 when we randomly rank the features. The biggest contributions to the accuracy are from redundancy and relevance prediction. The interaction between redundancy and relevance has a negative estimate because we do not reward calculation of redundant features even if they are highly relevant. Hence, prediction of the relevance helps only on the subset of unique features from the set of all candidate features.

6.2 What is the effect of meta-learning on model accuracy?

To evaluate the effectivity of the meta-learning, we iteratively train a classification model on increasing percentage

Table 2: Taxonomy of feature functions (data type they work on: c-character, n-numeric, t-temporal). The horizontal axis differentiates between feature functions working on a single attribute and multiple attributes. The vertical axis differentiate between feature functions working on a single tuple and multiple tuples.

	Univariate		**Multivariate**
1:1	Direct field (any)		Time diff (t+t)
	Text length (c)		
	Time day part (t)		
	Time is weekend (t)		
	Time part (t)		
	Time since (t)		
	WOE (c)		
1:n	Aggregate (n)	Existential count (any)	Aggregate frame (n+t)
	Aggregate distinct (n)	Log product (n)	Correlation (n+t)
	Aggregate range (n)	Null ratio (any)	Intercept (n+t)
	Aggregate text length (c)	Time aggregate (t)	Slope (n+t)
	Aggregate WOE (c)	Time aggregate since (t)	Time aggregate diff (t+t)
	Coefficient of variation (n)	Time aggregate since event (t)	
	Count (any)	Time frequency (t)	
	Distinct count (any)	Time range (t)	
	Duplicate ratio (any)	Time WOE (t)	

Table 3: Leave-one-out accuracy of individual models.

Algorithm	Relevance [ρ]	Runtime [ρ]	Redundancy [AUROC]
DL	**0.558** $\pm$ 0.255	0.302 $\pm$ 0.186	0.798 $\pm$ 0.097
GBT	0.556 $\pm$ 0.252	0.206 $\pm$ 0.259	0.787 $\pm$ 0.106
GLM	0.551 $\pm$ 0.276	**0.369** $\pm$ 0.267	**0.810** $\pm$ 0.102

Table 4: Meta-features for relevance prediction.

Meta-feature	Comment	Weight
landmark_relevance	average on the attribute	9.81
feature_function	e.g., null_ratio is inferior to direct field	5.58
feature_parameters	e.g., aggregate=min is inferior to aggregate=avg	1.90
data_type	e.g., enums are superior to datetimes	0.34
avg_length	extremely long attributes like text are subpar	0.25
is_primary_key	surrogate primary keys make inferior features	0.10

Table 5: Meta-features for redundancy prediction.

Meta-feature	Comment	Weight
landmark_redundancy	average on the attribute	16.70
feature_parameters	e.g., aggregate=min is inferior to aggregate=avg	13.65
feature_function	e.g., null_ratio is inferior to count	2.61
data_type	e.g., integers are inferior to doubles	0.02

Table 6: Meta-features for runtime prediction.

Meta-feature	Comment	Weight
landmark_runtime	average on the attribute	5.00
feature_function	complicated features take more time	3.34
table_rows	more data means higher runtime	0.10

of the top features, as estimated with meta-learning. As the classification model, we use a decision tree because it can model interactions between the features, it is undemanding on data preprocessing and it is reasonably fast. As the evaluation measure, we use misclassification error as all databases have reasonably balanced classes in the label.

An example of the obtained curve is depicted in Figure 4, where we can observe that the decision tree slightly overfits when we use all the features. Nevertheless, forward selection still outperforms meta-learning feature selection, as it can observe all the features (our approach does not) and it is a wrapper (our approach is a filter [5]).

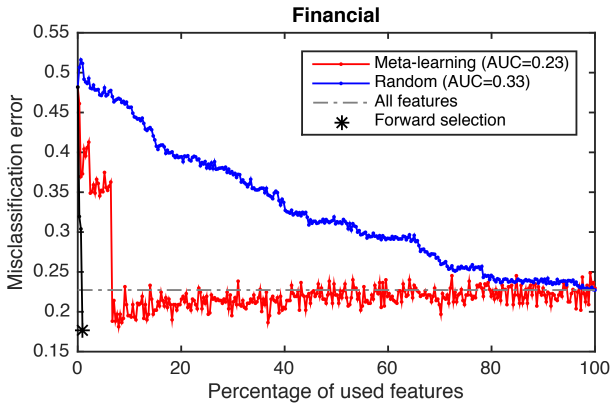

Figure 4: Area under misclassification error. Smaller is better.

Conclusion: The result of the factor analysis is in Table 9. Prediction of relevance significantly reduces misclassification error. Prediction of runtime insignificantly increases the misclassification error, because this evaluation does not reward fast features. Redundancy prediction does not significantly decrease the classification error. Based on our inspection of the results, this is because this evaluation rewards early discovery of a few highly relevant features much higher (since the best possible decision tree may use just a few features) than it penalizes redundancy (a redun-

Table 7: POPs for all databases based on the used individual models. PI column contains the upper 95% prediction interval of POPs for random ordering of the features. The best values are in bold.

redundance	0	1	0	0	1	1	0	1	
relevance	0	0	1	0	1	0	1	1	
runtime	0	0	0	1	0	1	1	1	PI
Accidents	0.11	0.45	0.61	−0.18	**0.68**	0.44	0.60	0.67	0.33
AustralianFootball	0.03	0.33	0.35	−0.09	0.40	0.33	0.35	**0.40**	0.36
BasketballMen	0.08	0.53	−0.52	0.28	0.62	0.56	−0.37	**0.62**	0.11
Biodegradability	−0.18	0.31	−0.44	0.24	**0.34**	−0.21	−0.32	0.32	0.31
Chess	0.05	0.46	0.72	0.31	0.71	0.80	**0.90**	0.87	0.29
Financial	−0.01	0.24	−0.02	−0.01	0.30	0.29	0.02	**0.35**	0.32
Hepatitis	0.07	0.37	−0.01	0.03	0.34	**0.48**	0.04	0.37	0.28
Mondial	−0.00	0.19	0.30	−0.09	0.32	0.27	0.26	**0.32**	0.11
Mutagenesis	0.05	0.14	0.09	0.20	0.24	**0.63**	0.62	0.59	0.22
Nations	0.16	0.59	0.87	0.18	0.75	0.79	**0.88**	0.76	0.34
PremierLeague	−0.07	0.12	0.27	0.08	0.17	0.35	**0.35**	0.34	0.17
PTE	0.01	0.39	0.32	−0.43	0.54	0.31	0.24	**0.53**	0.20
StudentLoan	0.25	0.17	0.62	0.14	0.61	0.53	**0.65**	0.61	0.39
VisualGenome	−0.11	0.44	0.95	−0.03	0.95	0.74	**0.96**	0.94	0.82
Walmart	−0.18	0.20	0.77	−0.06	0.49	0.17	**0.81**	0.52	0.42
average	0.02	0.33	0.32	0.04	0.50	0.43	0.40	**0.55**	0.31
win count	0	0	0	0	2	2	**6**	5	0

Table 8: Contribution of models to POP. Adjusted R^2: 0.564.

	Estimate	Std. Error	$\Pr(>\|t\|)$	
(Intercept)	0.020	0.0586	0.6877	
relevance	0.292	0.0149	4.0527×10^{-5}	***
redundance	0.318	0.0149	2.9095×10^{-5}	***
runtime	0.053	0.0122	0.0124	*
rel:red	−0.1723	0.0244	0.0021	**

dant feature only pushes all subsequent features one step later).

Table 9: Contribution of models to reduction of the area under misclassification error curve. Adjusted R^2: 0.486.

	Estimate	Std. Error	$\Pr(>\|t\|)$	
(Intercept)	0.294	0.020	1.3357×10^{-5}	***
relevance	−0.053	0.016	0.0016	**
redundance	−0.018	0.014	0.2402	
runtime	0.010	0.014	0.5167	

6.3 Which meta-features are important?

To analyze the importance of the three categories of the meta-features (*landmarking*, *database*, *feature-function*), we design an experiment, in which we vary the set of the used meta-features.

Table 10: Contribution of meta-feature categories to the reduction of the count of engineered features needed to reach or surpass model accuracy obtained on a complete set of features. Adjusted R^2: 0.308.

	Estimate	Std. Error	$\Pr(>\|t\|)$	
(Intercept)	43.328	11.569	0.0013	**
database	−0.741	11.007	0.9472	
featureFunction	−3.755	11.007	0.7377	
landmarking	−30.586	11.007	0.0140	*

Conclusion: Based on the results reported in Table 10, only landmarking meta-features help to significantly[2] reduce the count of features that have to be engineered to reach model accuracy obtained on all features. Table 10 also tells us that if all meta-features are used, it is in average sufficient to engineer only the top 8.25% of the features to match or surpass the classification accuracy of the model trained on all features.

6.4 Do we need so many feature functions?

We may wonder whether it is not enough to just engineer the 6 *landmarking features* and do not continue with the engineering of the remaining 25 (e.g., multivariate) features. We compared accuracies of the models trained only on the landmarking features with accuracies obtained on all

[2]The reported p-values do not incorporate correction for repeated evaluation of serially correlated observations

features. Based on Wilcoxon signed-rank test, we have to reject the null hypothesis that the additional features do not improve accuracy (p-value = 0.00048). The median improvement is 1.2 percent point in classification accuracy (average improvement is 2.7 percent point).

Conclusion: The additional features improve the accuracy of the model over the accuracy of the model build only on the landmarking features by a small but significant amount.

6.5 Feature Selection vs. Feature Meta-learning

The described feature meta-learning bears similarity with filter-type feature selection methods like *Correlated Feature Selection* (CFS)[6] and *Minimum Redundancy Maximum Relevance* (mRMR)[17]. Both these methods attempt to quickly select relevant non-redundant features. And so does our method. But in comparison to these methods, we perform feature selection before the feature engineering.

Difficulty It can be argued that feature meta-learning is at least as difficult problem as feature selection since we can always convert feature selection problem to feature meta-learning by throwing away the computed features (and recalculating them on request).

6.6 Limitations

We performed experiments only on relational data and features from propositionalization. Propositionalization is known to produce a lot of duplicate features (38% on average on the tested databases) and many of the features are irrelevant to the task (64% on average on the tested databases based on backward selection). These properties make it possible to obtain substantial gains from feature selection. However, the performed experiments do not tell us how the described approach is going to generalize on non-relational data.

Another limitation of the reported work is that it ignores interactions between the feature vectors in the downstream model. This can reduce the accuracy of the downstream model because a univariate oraculum meta-learner would not recommend calculation of features that are useful only in the combination with other features (a trivial example where this may happen is XOR problem [13]). Possible solutions to this problem are briefly mentioned in the future work Section 7.

6.7 Applications

Feature meta-learning is desirable in domains, where a single universal approach to feature extraction does not exist or is not known ahead. An exemplary domain are relational data, which may contain highly diverse content ranging from structured to unstructured data.

Additionally, feature selection before feature engineering is applicable to complex or large data, where it is not feasible or convenient to calculate and evaluate all possible features due to limited resources.

7 Conclusion

In this article, we evaluated an idea of performing feature selection *before* feature engineering. To guide the search, we exploited *meta-learning*. Nargesian [16] used meta-features calculated from the original data. But we found out that landmarking meta-features work better. When we evaluated the implementation on 15 databases, we concluded that it is on average enough to engineer only *the top decile* of features to get accuracy comparable to accuracy obtained on all features. This finding is similar to Reif's [20] finding, who applied meta-learning to feature selection. However, Reif performs feature selection *after* feature engineering while we perform feature selection *before* feature engineering.

7.1 Future Work

In this exploratory work, we optimize an ersatz measure called POP, which is easy to reason about. One possible extension of this work is to improve individual components of the meta-learning model. For example, we can detect redundancy based on a fuzzy comparison of equal-height histograms estimated with the database engine or quantile sketch. With this change, we would detect duplicates that are identical up to a monotonic transformation, leading to better alignment with models that are invariant to monotonic transformations of the features (e.g., decision trees in theory). Or we could replace the redundancy detection with a precomputed correlation matrix describing similarities between the feature functions [24, p. 148]. Alternatively, we could estimate a transition matrix describing the optimal order in which to apply feature functions (or give up on the given attribute). To improve non-redundancy and relevance together, we could train a fast model (e.g., naive Bayes) on streaming features (e.g., [23] or [22, p. 19]). The possibilities are vast.

Another possible direction is to directly optimize the measure we are interested in (e.g., improvement to model's AUROC over time). This can be done by training a single model (e.g., [16]). And this article provides an extended set of meta-features, on which such model could be trained.

8 Acknowledgement

We would like to thank Adéla Blažková for her help. We furthermore thank the anonymous reviewers, their comments helped to improve this paper. The reported research has been supported by the Grant Agency of the Czech Technical University in Prague (SGS17/210/OHK3/3T/18) and the Czech Science Foundation (GAČR 18-18080S).

References

[1] S. M. Abdulrahman and P. Brazdil. Measures for combining accuracy and time for meta-learning. *CEUR Workshop Proc.*, 1201:49–50, 2014.

[2] T. Brandenburger and A. Furth. Cumulative Gains Model Quality Metric. *J. Appl. Math. Decis. Sci.*, 2009:1–14, 2009.

[3] L. De Raedt. *Inductive Logic Programming*, volume 1446 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 1998.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, 2 edition, 2000.

[5] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. An Introduction to Variable and Feature Selection Isabelle. *Mach. Learn.*, 46(1/3):389–422, 2002.

[6] M. A. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, The University of Waikato, 1999.

[7] J. M. Kanter. Deep Feature Synthesis: Towards Automating Data Science Endeavors. *IEEE DSAA*, 2015.

[8] C. S. Kheau, R. Alfred, and L. H. Keng. Dimensionality Reduction in Data Summarization Approach to Learning Relational Data. *ACIIDS*, 7802:166–175, 2013.

[9] M.-A. Krogel and S. Wrobel. Transformation-Based Learning Using Multirelational Aggregation. In *ILP*, pages 142–155. Springer, London, 2001.

[10] M.-A. Krogel and S. Wrobel. Propositionalization and Redundancy Treatment. In *Databases, Doc. Inf. Fusion*, Hannover, 2002. CEUR.

[11] C. Lemke, M. Budka, and B. Gabrys. Metalearning: a survey of trends and technologies. *Artif. Intell.*, 44(1):117–130, 2015.

[12] A. McNab and D. A. Ladd. Information quality: The importance of context and trade-offs. In *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pages 3525–3532, 2014.

[13] M. Minsky and S. A. Papert. *Perceptrons. An Introduction to Computational Geometry*. MIT, jan 1969.

[14] J. Motl and P. Kordík. Foreign Key Constraint Identification in Relational Databases. In *ITAT*, pages 106–111. CEUR, 2017.

[15] J. Motl and O. Schulte. The CTU Prague Relational Learning Repository. *arXiv*, page 7, nov 2015.

[16] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. Turaga. Learning Feature Engineering for Classification. *IJCAI*, (August):2529–2535, 2017.

[17] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE TPAMI*, 27(8):1226–1238, aug 2005.

[18] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier. Meta-Learning by Landmarking Various Learning Algorithms. In *ICML*, volume 951, pages 743–750, 2000.

[19] A. Popescul and L. H. Ungar. Structural Logistic Regression for Link Analysis. *MRDM*, (August):92–106, 2003.

[20] M. Reif and F. Shafait. Efficient feature size reduction via predictive forward selection. *Pattern Recognit.*, 47(4):1664–1673, 2014.

[21] J. Rice. The Algorithm Selection Problem. *Adv. Comput.*, 15(C):65–118, 1976.

[22] J. Tang, S. Alelyani, and H. Liu. Feature Selection for Classification: A Review. *Data Classif. Algorithms Appl.*, pages 37–64, 2014.

[23] K. Yu, W. Ding, D. Simovici, H. Wang, J. Pei, and X. Wu. Classification with Streaming Features: An Emerging-Pattern Mining Approach. *ACM TKDD*, 9(4):1–31, 2015.

[24] Z. Zhou and H. Liu. *Spectral Feature Selection for Data Mining*. CRC, New York, NY, 2011.

[25] S. Zilberstein. Using Anytime Algorithms in Intelligent Systems. *AI Mag.*, 17(3):73–83, 1996.

Appendix

Reproducibility

The used code is published at:
`https://github.com/janmotl/metalearning`.
The used databases are published at:
`https://relational.fit.cvut.cz`.

Table 11: Expected standardized relevance (bigger is better), runtime (smaller is better) and redundancy (smaller is better) of feature functions (sorted by the feature utility).

Feature function	Relevance	Runtime	Redundancy	Utility
Aggregate frame	−2.11	−0.17	0.49	−2.19
Time aggregate diff	−1.53	0.05	0.54	−1.95
Time diff	−0.92	−0.01	0.48	−1.34
Time day part	−1.33	0.02	0.02	−1.31
Time since	−0.89	−0.04	0.45	−1.22
Null ratio	−0.99	−0.02	0.18	−1.06
Time frequency	−0.17	0.35	−0.10	−0.71
Existential count	−0.67	−0.06	0.06	−0.47
Slope	−0.49	0.03	0.03	−0.47
Time WOE	0.51	0.38	0.07	−0.42
Time is weekend	−1.03	−0.12	−0.21	−0.40
Time part	−0.45	0.00	0.05	−0.40
Text length	−0.76	−0.07	−0.09	−0.37
Intercept	1.42	0.36	0.37	−0.21
Correlation	1.32	0.26	0.37	−0.05
Time aggregate	0.57	0.05	0.21	0.19
Aggregate text length	−0.24	−0.05	−0.15	0.29
Aggregate range	0.34	−0.02	0.15	0.34
Duplicate ratio	0.32	0.17	−0.26	0.39
Aggregate distinct	0.56	0.03	0.11	0.44
Time range	0.06	0.06	−0.26	0.45
Coefficient of variation	0.36	0.01	−0.07	0.62
Time aggregate since event	0.19	0.01	−0.24	0.75
Direct field	0.26	−0.05	−0.09	0.79
Distinct count	0.21	−0.04	−0.16	0.82
Aggregate	0.49	0.04	−0.16	0.82
Log product	0.62	−0.02	0.02	0.87
Time aggregate since	1.25	0.27	−0.24	0.95
Count	0.30	−0.08	−0.18	1.13
WOE	1.27	−0.03	−0.01	1.69
Aggregate WOE	1.04	0.01	−0.39	2.05