METALEARNING USING STRUCTURE-RICH PIPELINE REPRESENTATIONS FOR BETTER AUTOML

Anonymous authors
Paper under double-blind review

ABSTRACT

Automatic machine learning (AutoML) systems have been shown to perform better when they learn from past experience. Examples include Auto-sklearn, which warm-starts the ML pipeline search using existing programs known to perform well on "similar" tasks, and AlphaD3M, which uses online reinforcement learning to search the ML pipeline space. These metalearning approaches, as well as many others, depend on simplifying assumptions about the pipeline search space and/or the pipeline representation. We show that we can estimate ML pipeline performance without simplifying the representation of the pipeline structure. We evaluate this approach on tabular classification tasks and find that it produces the best estimates of pipeline performance and yields pipeline rankings with the highest normalized discounted cumulative gain (nDCG) and the lowest regret.[1]

1 INTRODUCTION

Automatic machine learning (AutoML) systems can use metalearning to aid in machine learning (ML) program search. Such metalearning can help users with any level of ML expertise to find a good program for a dataset of interest by leveraging past results (Brazdil et al., 2008). A modern ML program usually contains multiple ML algorithms (e.g., missing value imputation, feature preprocessing, classification, etc.) that are composed into an ML *pipeline*. In order to represent the potentially complex computational flow of data through a pipeline (e.g., an ensemble like Figure 1), the pipeline must be represented as a directed acyclic graph (DAG). The many ways to compose available ML algorithms into pipelines creates an unscalable number of possible pipelines to consider. Thus, learning to pre-rank candidate pipelines allows AutoML systems to prioritize pipeline execution and allocate limited computational resources to the pipelines with the most potential.

Metalearning for AutoML can be categorized into offline and online approaches. AutoML systems that use offline metalearning leverage the results of running pipelines on a collection of previously seen datasets to inform pipeline search on a new dataset, for example, Auto-sklearn (Feurer et al., 2015), SmartML (Maher & Sakr, 2019), and DeepLine (Heffetz et al., 2020). Conversely, online metalearning approaches utilize only the results of running pipelines on a dataset of interest. Examples include systems like AlphaD3M (Drori et al., 2018), TPOT (Olson & Moore, 2016), and Layered-TPOT (Gijsbers et al., 2018) or approaches like Bayesian optimization (Bergstra et al., 2015; Snoek et al., 2012; Hutter et al., 2011). Both of these types of metalearning are orthogonal to each other and can be implemented simultaneously in a system, as in some of the above examples. For a broader related work on AutoML and metalearning in general, see Hutter et al. (2019).

In these previous works on metalearning for AutoML, their approaches (whether offline or online) make simplifying assumptions about the pipeline search space and/or the pipeline representations to enable metalearning. Auto-sklearn and SmartML treat pipelines as black boxes by using only pipeline performance and dataset metafeature similarity in their metamodels. AlphaD3M considers only sequential pipelines, so it can use an LSTM (Hochreiter & Schmidhuber, 1997) to approximate the value function in its reinforcement learning approach. DeepLine and RankML (Laadan et al., 2019) also use an LSTM metamodel, but consider a broader pipeline search space (e.g., ensembles and parallel sub-pipelines) albeit they linearize the pipeline directed acyclic graph (DAG) structure.

[1] Code and data will be made available after the anonymity period.

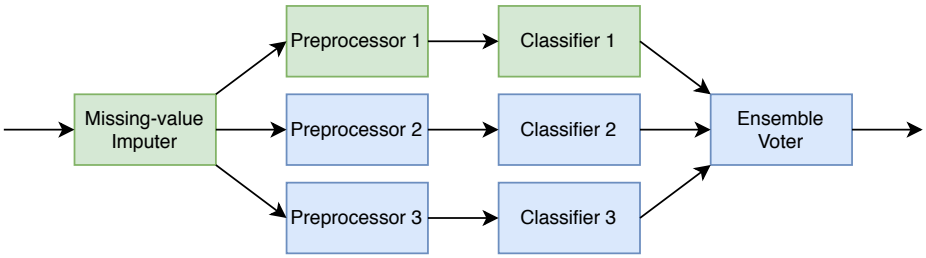

Figure 1: A pipeline can be represented as a directed acyclic graph (DAG). This example pipeline is a template for the pipelines used in our metadataset. A dataset is input on the left and flows from left to right, following the arrows. Each node transforms the data with some ML algorithm and the final predictions are output on the right.

We propose to estimate ML pipeline performance without simplifying the representation of the pipeline structure. The fine-grained pipeline metadata, including both the ML algorithms used in the pipeline and the pipeline structure (i.e., how the algorithms are composed), may contain pertinent information for the metamodel. We validate this hypothesis in the offline metalearning context by creating a metadataset consisting of about 400,000 instances made up of more than 4,000 pipelines (e.g., Figure 1) run on almost 200 tabular classification datasets. We evaluate nine metamodels that utilize varying levels of pipeline metadata granularity to show how the pipeline representation affects metalearning performances.

2 METADATASET

To test our hypothesis, we construct a large metalearning dataset, or metadataset. We scope this experiment to tabular classification datasets and select 194 datasets collected as part of the Data Driven Discovery of Models (D3M)[2] program that were originally sourced from OpenML (Vanschoren et al., 2014), UCI Machine Learning Repository (Dheeru & Karra Taniskidou, 2017; Lichman, 2013), and Zenodo (Guzey et al., 2014).

Following traditional metalearning research, we extract metafeatures (Vanschoren et al., 2014; Feurer et al., 2015; Brazdil et al., 2008) from each of these datasets. We compute 72 metafeatures using a Python port of Reif (2012), with additional metafeatures from Vanschoren et al. (2014). These include simple, statistical, information-theoretic, landmarking, and model-based metafeatures that were tractable and relevant to all of our selected datasets. A complete list of metafeatures is found in Appendix F.

Next, we generate pipelines representative of typical, human-made machine learning pipelines using the D3M machine learning framework (Milutinovic et al., 2020) and Scikit-learn (Pedregosa et al., 2011). These pipelines include straight pipelines and ensembles, which require a DAG representation. The straight pipelines consist of a missing value imputation step, one of 10 feature preprocessing algorithms (e.g., PCA (Wold et al., 1987) and no-op), and one of 14 final classifiers (e.g., Decision Tree (Swain & Hauska, 1977)). A complete list of these ML algorithms can be found in Appendix C. The ensembles combine three straight pipelines with a majority vote algorithm. To limit the scope of this work, all ML algorithms used default hyperparameters. Figure 1 shows the pipeline template.

Altogether this process created 4,592 different pipelines, which we ran on the 194 datasets to get 413,567 distinct instances in our metadataset. Each instance thus consists of the dataset metfeatures, a DAG representation of the pipeline, and the test F1 score (though any metric could have been used). Note that not all pipelines ran successfully on all datasets, for example, some preprocessing methods created an intractable number of features that caused memory errors. We randomly partitioned the metadataset, grouped by dataset, into meta-train (125 datasets with 225,668 metadata instances), meta-validation (25 datasets with 90,131 instances), and meta-test (44 datasets with the remaining 97,768 instances) sets. For more experimental details, see Appendix B.

[2]D3M datasets: https://datasets.datadrivendiscovery.org/d3m/datasets

3 METAMODELS

We consider a broad range of metamodels, including naive baselines, classic machine learning models, and modern deep learning models. We categorize each metamodel into different types based on the pipeline representation they use. The categories include naive baseline, pipeline-agnostic, structure-agnostic, linear structure-aware, and DAG-structure aware. While many metamodels could have been chosen, we choose at least one for each of these types based on prevalence in related work.

These metamodels are implemented using Scikit-learn (Pedregosa et al., 2011), PyTorch (Paszke et al., 2019), and code from Auto-sklearn (Feurer et al., 2015). Each metamodel is trained to estimate the F1 macro score given the dataset metafeatures and some representation of the pipeline. We use these estimates of performance to rank pipelines, which is useful when not all candidate pipelines are known apriori (Appendix B).

Naive Baselines. The *Mean* metamodel is dataset metafeature and pipeline agnostic. Since the Mean metamodel is constant, we use the *Random* metamodel to randomly rank pipelines as a surrogate for evaluating the Mean metamodel's ranking performance.

Pipeline-Agnostic. Auto-sklearn uses dataset metafeature similarity to determine the k-nearest datasets, or *k-ND*, from their static metadataset to the given input dataset. From each of these nearest datasets, the best known pipeline is selected to warm-start the system. We use their implementation of k-ND, but use our metadataset. Note that this metamodel ranks pipelines and recommends k of them, but does not estimate performance.

Structure-Agnostic. These metamodels use the dataset metafeatures and an encoding of which ML algorithms are used in the pipeline, ignoring all pipeline structure metadata. We consider *Linear Regression* and *Random Forest*. We also bootstrap metamodel selection by passing our metadataset to Auto-sklearn for 24 hours and label the resultant metamodel *Meta Auto-sklearn*.

Linear Structure-Aware. Recent metalearning approaches for estimating pipeline performance (Heffetz et al., 2020; Laadan et al., 2019; Drori et al., 2018) linearize the pipeline DAG structure to be able to use the *LSTM* (Hochreiter & Schmidhuber, 1997), which we also include. This metamodel thus assumes that data flows though the pipeline sequentially, without parallel branches, which is not always true. We also include the *Transformer* (Vaswani et al., 2017) sequence model, which could learn to attend to the relevant parts of the pipeline to estimate performance.

DAG Structure-Aware. The final two metamodels do not simplify the pipeline metadata representation, but rather utilize the fine-grained pipeline metadata, including both the ML algorithms used in the pipeline and the pipeline structure. The *DAG LSTM* (Song et al., 2018; Zhu et al., 2016) is an extension of an LSTM for DAGs. We also use a Neural Module Network (Andreas et al., 2016), or *NMN*, adapted from the visual question answering task. We simplify the NMN to use only the dynamic network component, removing the LSTM component, to evaluate its metalearning potential in isolation. This metamodel has a neural network module for each ML algorithm being modeled and dynamically changes its network structure to match the structure of the input pipeline.

We tune each metamodel's hyper-parameters using random search over 70+ different hyper-parameter configurations on the validation split of the metadataset. We report final metamodel scores on the test split of the metadataset using the best hyper-parameter configuration found on the validation split. We combine the train and validation splits of metadata to train the metamodels for the final run on the test split. We run each metamodel five times with different random seeds and report the average and standard deviation over those runs. For more experimental settings about the training procedure, see Appendix B.

4 RESULTS

Table 1 shows the performance of each metamodel. The RMSE column shows how well each metamodel can estimate the F1 score of a given ML pipeline on a given dataset. The nDCG and Regret columns in Table 1 measure the quality of the top 25 ranked pipelines averaged over all datasets in the meta-test split. Additional results for the nDCG and Regret metrics for other values of k can be found in Appendix B.

Metamodel Type	Metamodel	RMSE ↓	nDCG@25 ↑	Regret@25 ↓
Naive Baseline	Mean / Random	0.302 ± 0.000	0.759 ± 0.005	**0.014** ± 0.002
Pipeline Agnostic	k-ND	-	0.805 ± 0.000	**0.011** ± 0.000
Structure Agnostic	Linear Regression	0.265 ± 0.000	**0.886** ± 0.000	0.024 ± 0.000
	Random Forest	0.211 ± 0.001	**0.906** ± 0.007	0.026 ± 0.002
	Meta Auto-sklearn	0.213 ± 0.001	**0.900** ± 0.001	0.029 ± 0.001
Linear Structure Aware	LSTM	0.376 ± 0.130	**0.834** ± 0.059	**0.045** ± 0.040
	Transformer	0.193 ± 0.005	**0.899** ± 0.007	0.044 ± 0.005
DAG Structure Aware	DAG LSTM	**0.182** ± 0.006	**0.890** ± 0.017	**0.042** ± 0.018
	NMN	0.198 ± 0.005	0.872 ± 0.013	**0.046** ± 0.012

Table 1: Metamodel performance on the test set, including mean and SD over five random seeds. Arrows indicate the direction of better scores. Scores in **bold** are not significantly different than the best score in the column (two-sample t-test, $p > 0.05$). We use $k = 25$ to match Feurer et al. (2015). Note that RMSE is not reported for k-ND because it does not estimate pipeline performance.

In the RMSE column, we observe that the metamodels that utilize more of the pipeline metadata are associated with better estimates of performance. The DAG LSTM gives the best estimates of performance with much more reliability than the LSTM (which is used in related work, e.g. Heffetz et al. (2020); Laadan et al. (2019); Drori et al. (2018)).

The normalized discounted cumulative gain at 25 (nDCG@25) measures the aggregate performance of the top 25 ranked pipelines by accounting for both the true pipeline score and the estimated pipeline rank. We observe that utilizing some pipeline metadata allows the metamodels to select 25 pipelines that are among the best performing pipelines (compared to those that use no pipeline metadata, like k-ND (Feurer et al., 2015)). In other words, using dataset metafeature similarity alone is not an effective method for finding a pool of high ranking pipelines.

The Regret@25 metric is the difference in pipeline F1 score between the best of the model's top 25 ranked pipelines and the true best pipeline on a given dataset. This is a measure of how much worse using any of the metamodels is compared to simply running all pipelines and choosing the best one. Interestingly, we observe that k-ND (Auto-sklearn's approach) on average finds a pipeline from the 25 nearest datasets within about 1 point of the true best pipeline. This corroborates the intuition that pipelines behave similarly on similar datsets. Similar to k-ND, Random produces a pipeline that performs close to the best when 25 are sampled (see Appendix D for a theoretical justification).

Overall Results. In practice, offline metalearning systems for AutoML use a metamodel to recommend many high-scoring pipelines to warm start their optimization process (Feurer et al., 2015; Maher & Sakr, 2019). This would translate to a metalearning approach that scores highly in all three metric categories: returning a highly ranked pool of piplines (nDCG), having pipelines that score close to the true best pipeline (Regret), and estimating pipeline performance accurately (RMSE). Examining Table 1 altogether, we find that models that contain more structural information perform better across categories: those that are structure or pipeline agnostic perform well in only one metric, while those that use the linear structure (e.g. LSTM) can perform well in two measures (nDCG/Regret), and those that use all metadata (e.g. DAG-LSTM) can score well in all three metrics. Although this result may not be surprising, we are the first to look at using the full structure in the metalearning context, and our results suggest it could significantly improve current systems.

5 Conclusion

We have shown that metalearning benefits from utilizing all available pipeline metadata, without simplifying the representation of the pipeline structure. We find that these structure-aware metamodels have the highest performance across several performance estimation and ranking measures. By implementing this strategy, AutoML systems could potentially reduce the computation time needed to find top-performing pipelines, spending the extra saved time exploring other fruitful segments of the pipeline search space. Future work in this area would include considering the effects of specific hyperparameter configurations, broadening the scope of the study to include more diverse task types, or creating new metamodels that take advantage of DAG structures.

REFERENCES

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016.

James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015.

Pavel Brazdil, Christophe Giraud Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Petros Drineas and Michael W Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(Dec):2153–2175, 2005.

Iddo Drori, Yamuna Krishnamurthy, Remi Rampin, Raoni Lourenço, Jorge One, Kyunghyun Cho, Claudio Silva, and Juliana Freire. Alphad3m: Machine learning pipeline synthesis. In *AutoML Workshop at ICML*, 2018.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pp. 2962–2970, 2015.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

Pieter Gijsbers, Joaquin Vanschoren, and Randal S Olson. Layered tpot: Speeding up tree-based pipeline optimization. *arXiv preprint arXiv:1801.06007*, 2018.

Onur Guzey, Gihad Sohsah, and Muhammed Unal. Classification of word levels with usage frequency, expert opinions and machine learning, October 2014. URL `https://doi.org/10.5281/zenodo.12501`.

Yuval Heffetz, Roman Vainshtein, Gilad Katz, and Lior Rokach. Deepline: Automl tool for pipelines generation using deep reinforcement learning and hierarchical actions filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2103–2113, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pp. 507–523. Springer, 2011.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

Aapo Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks*, 10(3):626–634, 1999.

David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

Doron Laadan, Roman Vainshtein, Yarden Curiel, Gilad Katz, and Lior Rokach. Rankml: a meta learning-based approach for pre-ranking machine learning pipelines. *arXiv preprint arXiv:1911.00108*, 2019.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Mohamed Maher and Sherif Sakr. Smartml: A meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms. In *EDBT: 22nd International Conference on Extending Database Technology*, Mar 2019.

Mitar Milutinovic. *Towards Automatic Machine Learning Pipeline Design*. PhD thesis, UC Berkeley, 2019.

Mitar Milutinovic, Brandon Schoenfeld, Diego Martinez-Garcia, Saswati Ray, Sujen Shah, and David Yan. On evaluation of automl systems. In *Proceedings of the ICML Workshop on Automatic Machine Learning*, 2020.

Randal S Olson and Jason H Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, pp. 66–74, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, and et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pp. 61–74. MIT Press, 1999.

Matthias Reif. A comprehensive dataset for evaluating approaches of various meta-learning tasks. In *ICPRAM (1)*, pp. 273–276, 2012.

Robert Harry Riffenburgh. *Linear discriminant analysis*. PhD thesis, Virginia Polytechnic Institute, 1957.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pp. 583–588. Springer, 1997.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. N-ary relation extraction using graph-state lstm. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2226–2235, 2018.

Philip H Swain and Hans Hauska. The decision tree classifier: Design and potential. *IEEE Transactions on Geoscience Electronics*, 15(3):142–147, 1977.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Dag-structured long short-term memory for semantic compositionality. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pp. 917–926, 2016.

A Additional Results at Various Levels of K

Our main results in Table 1 use $k = 25$ for the nDCG and Regret scores because Feurer et al. (2015) use that value when evaluating their metalearning approach. We evaluated our results using many values of k, but found that the results are not substantially different for $k > 1$. Table 2 and Table 3 show nDCG and Regret, respectively, for $k \in \{1, 25, 100, 4592\}$. One thing to note is that for Regret@1, almost any algorithm except Random is effective - however, Random becomes more effective in Regret as k increases. For a theoretical explanation of why Random is effective at larger k values, see Appendix D. We further find that the DAG-LSTM continues to provide equal or better performance than all other models across the different metrics and k values.

Model	nDCG@1	nDCG@25	nDCG@100	nDCG @ 4592
Random	0.760 ± 0.008	0.759 ± 0.005	0.775 ± 0.004	0.948 ± 0.001
k-ND	**0.848** ± 0.000	0.805 ± 0.000	0.793 ± 0.000	0.949 ± 0.000
Linear Regression	**0.880** ± 0.000	**0.886** ± 0.000	**0.898** ± 0.000	**0.981** ± 0.000
Random Forest	**0.901** ± 0.011	**0.906** ± 0.007	**0.916** ± 0.007	**0.983** ± 0.003
Meta Auto-sklearn	**0.890** ± 0.003	**0.900** ± 0.001	**0.914** ± 0.000	**0.984** ± 0.000
LSTM	**0.823** ± 0.064	**0.834** ± 0.059	0.843 ± 0.056	**0.973** ± 0.014
Attention	**0.900** ± 0.011	**0.899** ± 0.007	**0.912** ± 0.006	**0.986** ± 0.003
DAG LSTM	**0.888** ± 0.018	**0.890** ± 0.017	**0.902** ± 0.018	**0.981** ± 0.008
NMN	0.856 ± 0.018	0.872 ± 0.013	0.886 ± 0.009	0.972 ± 0.006

Table 2: Mean and std of metamodel results for the nDCG metric at various levels of k over five runs with different seeds. Scores in **bold** are not significantly different than the best score in the column from a two-sample t-test ($p > 0.05$). Higher is better.

Model	Regret@1	Regret@25	Regret@100
Random	0.095 ± 0.012	**0.014** ± 0.002	**0.005** ± 0.001
k-ND	**0.049** ± 0.000	**0.011** ± 0.000	**0.002** ± 0.000
Linear Regression	**0.050** ± 0.000	0.024 ± 0.000	0.018 ± 0.000
Random Forest	**0.047** ± 0.007	0.026 ± 0.002	0.018 ± 0.002
Meta Auto-sklearn	**0.053** ± 0.002	0.029 ± 0.001	0.020 ± 0.001
LSTM	**0.091** ± 0.055	**0.045** ± 0.040	**0.021** ± 0.017
Attention	**0.056** ± 0.005	0.044 ± 0.005	0.028 ± 0.004
DAG LSTM	**0.055** ± 0.016	**0.042** ± 0.018	**0.027** ± 0.013
NMN	0.075 ± 0.018	**0.046** ± 0.012	**0.022** ± 0.005

Table 3: Mean and std of metamodel results for the Regret metric at various levels of k over five runs with different seeds. Note that Regret@4592 is 0 by definition. Scores in **bold** are not significantly different than the best score in the column from a two-sample t-test ($p > 0.05$). Lower is better.

B EXPERIMENTAL SETTINGS

We use Scikit-learn (Pedregosa et al., 2011) implementations for feature preprocessing and classification algorithms and execute them in the ML framework developed as part of the Data-Driven Discovery of Models (D3M) program (Milutinovic, 2019; Milutinovic et al., 2020) that handles train/test splits and pipeline execution.

Experiments to gather the metadataset ran for two weeks on 20 computers with 4 Intel Core i7-4770K CPUs and 32 GB DIMM1 RAM. We kept pipeline results that successfully ran, indicating that the algorithm worked on the data (e.g. an algorithm that could not process non-numeric text would fail on a dataset that included it) and did not cause an out of memory error. Although this reduced our overall number of pipeline runs, we were still able to gather the cross product of all executable combinations of pipelines on the 194 datasets.

We trained the neural metamodels using mean squared error loss. Ranking the pipelines was done by using the predicted pipeline performance to estimate rank. Ranking pipelines by estimating performance has several advantage over directly learning to rank a set of pipelines. Ranking pipelines directly requires the metamodel to operate simultaneously on all candidate pipelines. Adding a single pipeline to that ranking requires re-ranking all pipelines. With an estimate of performance, adding a new pipeline to the ranking requires that the metamodel process only the new pipeline instead of all previously considered pipelines. This advantage applies to AutoML systems like Auto-sklearn [18], TPOT [35], and AlphaD3M [15, 16], where pipelines are generated dynamically at runtime.

We choose to use a meta-holdout set instead of using meta-cross-validation due to computational tractability in tuning the metamodels' hyper-parameters on the meta-validation set.

C ML ALGORITHMS

Missing value imputation The imputation ML algorithm sampled values from each column independently and uniformly over all instances with known values.

Feature Preprocessing FastICA (Hyvarinen, 1999), GenericUnivariateSelect, KernelPCA (Schölkopf et al., 1997), MinMaxScaler, Nystroem (Drineas & Mahoney, 2005), PCA (Wold et al., 1987), SelectFwe, SelectPercentile, StandardScaler, and the No-Op.

Classification BaggingClassifier (Breiman, 1996), BernoulliNB (Duda et al., 2012), DecisionTreeClassifier (Swain & Hauska, 1977), ExtraTreesClassifier (Geurts et al., 2006), GaussianNB (Duda et al., 2012), GradientBoostingClassifier (Friedman, 2001), KNeighborsClassifier (Cover & Hart, 1967), LinearDiscriminantAnalysis (Riffenburgh, 1957), LinearSVC (Platt, 1999), LogisticRegression (Kleinbaum et al., 2002), PassiveAggresiveClassifier (Crammer et al., 2006), RandomForestClassifier (Breiman, 2001), SGDClassifier (Platt, 1999), SVC (Platt, 1999)

D WHY IS RANDOM SO GOOD?

As shown in Table 1, the Random metamodel offers a competitive baseline in terms of choosing at least one really good pipeline with its low Regret@25 score. It seems counter-intuitive that the most naive model outperforms the models that get to look at the training data. Suppose for some dataset that the pipeline F1 macro scores were distributed uniformly across that interval $[0, 1]$ Then, choosing k pipelines at random, or in other words ranking the pipelines randomly and choosing the top k from that ranking, would likely yield k pipelines that were more or less uniformly distributed on $[0, 1]$. The best of the k would, of course, be close to 1.

More formally, the score of the best pipeline under these assumptions is more generally known as the *largest order statistic*, whose expected value when sampling from a uniform distribution is $\frac{k}{k+1}$. When k is 25, we would thus expect the Random metamodel to pick a pipeline with a score of $25/26 \approx 0.962$. The Regret@k, assuming that a pipeline with a perfect score exists, would be 0.038. We observe in Table 1 that the Random metamodel has slightly lower, but still comparable result to this approximation. Although an approximation, this analysis suggests one possible reason why sampling pipelines at random will generate at least one that is close to the best possible pipeline for a given dataset.

E DATASETS

We used the following datasets to build our metadataset, grouped by their sources:

Dataset from Zenodo (Guzey et al., 2014):

- Word Level Classification Dataset

Dataset from UCI (Dheeru & Karra Taniskidou, 2017):

- Hepatitis Dataset

Datasets from UCI (Lichman, 2013):

- Breast Cancer Wisconsin Diagnostic Dataset
- DrivFace Dataset
- Breast Cancer Wisconsin Original Dataset
- SPECT Heart Dataset

Datasets from OpenML (Vanschoren et al., 2014):

- Abalone Dataset (3 classes)
- ADA Agnostic Dataset
- Prior Dataset
- Adult Dataset
- Allrep Dataset
- Analcatdata Authorship Dataset
- Analcatdata Birthday Dataset
- Analcatdata BroadwayMult Dataset
- Analcatdata DMFT Dataset
- Analcatdata GermanGSS Dataset
- Analcatdata GSSSexSurvey Dataset
- Analcatdata HallOfFame Dataset
- Analcatdata Lawsuit Dataset
- Analcatdata Reviewer Dataset
- Anneal Dataset
- Arsenic Female Bladder Dataset
- Arsenic Male Bladder Dataset
- Artificial Characters Dataset
- Audiology Dataset
- Australian Dataset
- AutoHorse Dataset
- AutoUniv-Au1-1000 Dataset
- AutoUniv-Au4-2500 Dataset
- AutoUniv-Au6-1000 Dataset
- AutoUniv-Au6-400 Dataset
- AutoUniv-Au6-750 Dataset
- AutoUniv-Au7-1100 Dataset
- AutoUniv-Au7-500 Dataset
- AutoUniv-Au7-700 Dataset
- Backache Dataset
- Badges2 Dataset
- Balance Scale Dataset
- Banana Dataset
- Bank Marketing Dataset
- Banknote Authentication Dataset
- Baseball Dataset
- Biomed Dataset
- Blood Transfusion Service Center Dataset
- Breast Cancer Dataset
- BreastTumor Dataset
- Breast W Dataset
- CalendarDOW Dataset
- Car Dataset
- Cardiotocography Dataset
- Cardiotocography Dataset (3 classes)
- Cars Dataset
- CastMetal1 Dataset
- Chscase Funds Dataset
- Chscase Vine2 Dataset
- Chscase Whale Dataset
- Click Prediction Small Dataset
- Climate Model Simulation Crashes Dataset
- CMC Dataset
- Colic Dataset
- Collins Dataset

- Corral Dataset
- CostaMadre1 Dataset
- Credit Approval Dataset
- Credit G Dataset
- CRX Dataset
- Cylinder Bands Dataset
- Dermatology Dataset
- Diabetes Dataset
- Diggle Table A2 Dataset
- Dresses Sales Dataset
- E. Coli Dataset
- Eucalyptus Dataset
- Eye Movements Dataset
- First Order Theorem Proving Dataset
- FishCatch Dataset
- Flags Dataset
- Flare Dataset
- GAMETES Epistasis 2 Way 20atts 0.1H EDM 1 1 Dataset
- GAMETES Epistasis 2 Way 20atts 0.4H EDM 1 1 Dataset
- GAMETES Epistasis 3 Way 20atts 0.2H EDM 1 1 Dataset
- GAMETES Heterogeneity 20atts 1600 Het 0.4 0.2 50 EDM 2 001 Dataset
- GAMETES Heterogeneity 20atts 1600 Het 0.4 0.2 75 EDM 2 001 Dataset
- Glass Dataset
- Grub Damage Dataset
- Haberman Dataset
- Hayes Roth Dataset
- Heart C Dataset
- Hepatitis Dataset
- Hill Valley Dataset
- Housing Dataset
- Indian Liver Patient Dataset
- Ionosphere Dataset
- Irish Dataset
- Japanese Vowels Dataset
- JM1 Dataset
- KC1 Dataset
- KC2 Dataset
- KC3 Dataset
- King+Rook versus King+Pawn Dataset
- Led24 Dataset
- LED Display Dataset
- Lowbwt Dataset
- Machine CPU Dataset
- MC2 Dataset
- MeanWhile1 Dataset
- MegaWatt1 Dataset
- MFeat Fourier Dataset
- MFeat Karhunen Dataset
- MFeat Morphological Dataset
- MFeat Zernike Dataset
- MiceProtein Dataset
- M of N 3 7 10 Dataset
- Monks Problems 1 Dataset
- Monks Problems 3 Dataset
- MW1 Dataset
- Nursery Dataset
- One Hundred Plants Margin Dataset
- One Hundred Plants Texture Dataset
- OptDigits Dataset
- Ozone Level 8hr Dataset
- Page Blocks Dataset
- Parity 5 Plus 5 Dataset
- Parkinsons Dataset
- PC1 Dataset
- PC3 Dataset
- PC4 Dataset
- PenDigits Dataset
- Phoneme Dataset
- PieChart1 Dataset
- Pima Dataset
- PizzaCutter1 Dataset
- Planning Relax Dataset
- PopularKids Dataset
- PRNN Crabs Dataset
- PRNN Synth Dataset
- pwlinear Dataset
- QSAR Biodegradation Dataset
- Ringnorm Dataset
- RMfTSA Cyprus Tourist Arrivals Dataset
- RMfTSA Sleep Dataset
- Robot Failures LP5 Dataset
- South Africa Heart Disease Dataset
- Seeds Dataset
- Segment Dataset
- Seismic Bumps Dataset

- Servo Dataset
- Smartphone Based Recognition Of Human Activities Dataset
- Solar Flare 1 Dataset
- Solar Flare 2 Dataset
- Sonar Dataset
- Soybean Dataset
- Spambase Dataset
- SPECTF Heart Dataset
- Splice Dataset
- Steel Plates Fault Dataset
- Synthetic Control Dataset
- Tamil Nadu Electricity Dataset
- Teaching Assistant Dataset
- Tecator Dataset
- Texture Dataset
- Thoracic Surgery Dataset
- Thyroid Allbp Dataset
- Thyroid Allhyper Dataset
- Thyroid Ann Dataset
- Thyroid Dis Dataset
- Thyroid New Dataset
- Tic Tac Toe Dataset
- Titanic Dataset
- Tokyo1 Dataset
- Triazines Dataset
- Twonorm Dataset
- User Knowledge Dataset
- Vehicle Dataset
- Vertebra Column Dataset
- Vertebra Column Dataset (2 classes)
- Volcanoes A1 Dataset
- Volcanoes A2 Dataset
- Volcanoes A3 Dataset
- Volcanoes A4 Dataset
- Volcanoes B1 Dataset
- Volcanoes B4 Dataset
- Volcanoes D1 Dataset
- Vote Dataset
- Vowel Dataset
- Wall Robot Navigation Dataset
- Wall Robot Navigation Dataset (3 attributes)
- Wall Robot Navigation Dataset (5 attributes)
- Waveform 5000 Dataset
- Breast Cancer Wisconsin Diagnostic Dataset
- Wholesale Customers Dataset
- Wilt Dataset
- Wine Quality White Dataset
- XD6 Dataset

F METAFEATURES

We compute 72 metafeatures using a Python port of Reif (2012), with additional metafeatures from Vanschoren et al. (2014). This follows previous work, e.g. Auto-sklearn uses only 30 metafeatures.

1. Number Of Instances
2. Number Of Features
3. Number Of Numeric Features
4. Number Of Categorical Features
5. Ratio Of Numeric Features
6. Ratio Of Categorical Features
7. Number Of Classes
8. Mean Class Probability
9. Skew Class Probability
10. Kurtosis Class Probability
11. Min Class Probability
12. Quartile 1 Class Probability
13. Quartile 2 Class Probability
14. Quartile 3 Class Probability
15. Max Class Probability
16. Minority Class Size
17. Majority Class Size
18. Dimensionality
19. Number Of Missing Values
20. Ratio Of Missing Values
21. Class Entropy
22. Naive Bayes Err Rate
23. Naive Bayes Kappa
24. kNN 1N Err Rate
25. kNN 1N Kappa
26. Decision Stump Err Rate

27. Decision Stump Kappa
28. Random Tree Depth 1 Err Rate
29. Random Tree Depth 1 Kappa
30. Random Tree Depth 2 Err Rate
31. Random Tree Depth 2 Kappa
32. Random Tree Depth 3 Err Rate
33. Random Tree Depth 3 Kappa
34. Mean Decision Tree Level Size
35. Total Metafeature Compute Time
36. Skew Decision Tree Level Size
37. Kurtosis Decision Tree Level Size
38. Min Decision Tree Level Size
39. Decision Tree Node Count
40. Max Decision Tree Attribute
41. Max Decision Tree Level Size
42. Linear Discriminant Analysis Kappa
43. Quartile 1 Decision Tree Level Size
44. Quartile 2 Decision Tree Level Size
45. Quartile 3 Decision Tree Level Size
46. Mean Decision Tree Branch Length
47. Skew Decision Tree Branch Length
48. Decision Tree Leaf Count
49. Decision Tree Height
50. Min Decision Tree Branch Length
51. Max Decision Tree Branch Length
52. Decision Tree Width
53. Mean Decision Tree Attribute
54. Kurtosis Decision Tree Branch Length
55. Number Of Features With Missing Values
56. Ratio Of Features With Missing Values
57. Number Of Instances With Missing Values
58. Ratio Of Instances With Missing Values
59. Standard Deviation Decision Tree Attribute
60. Skew Decision Tree Attribute
61. Kurtosis Decision Tree Attribute
62. Min Decision Tree Attribute
63. Quartile 1 Decision Tree Attribute
64. Quartile 2 Decision Tree Attribute
65. Quartile 3 Decision Tree Attribute
66. Standard Deviation Class Probability
67. Linear Discriminant Analysis Err Rate
68. Quartile 1 Decision Tree Branch Length
69. Quartile 2 Decision Tree Branch Length
70. Quartile 3 Decision Tree Branch Length
71. Standard Deviation Decision Tree Level Size
72. Standard Deviation Decision Tree Branch Length