Edit this form

28 responses

View all responses

Summary

Team name

djajetic
AAD Freiburg
abhishek4
sjahandideh
Test
a
Team Bennett
Research Group on Learning, Optimization, and Automated Algorithm Design (aad_freiburg)
tadej
Ideal Intel Analytics
jrl44
asml.intel.com
backstreet.bayes
aad_freiburg_gpu
agrigorev_GPU
aad_freiburg-GPU
djajetic_GPU
abhhishek-GPU
postech.mlg_exbrain

Team website URL

http://aad.informatik.uni-freiburg.de/
https://github.com/djajetic
aad.informatik.uni-freiburg.de
http://chalearn.org
b
http://tdj.si
http://ideal.intel.com/
https://github.com/postech-mlg-exbrain/AutoML-Challenge

Team Leader Name

Frank Hutter
Damir Jajetić
James Lloyd
Abhishek Thakur
Isabelle
c
Tadej Štajner
abhishek thakur
Samad Jahandideh
Alexey Grigorev
Dr. Frank Hutter
Damir Jajetic
Jungtaek Kim

Contact email

automl2015@informatik.uni-freiburg.de
abhishek4@gmail.com
james.robert.lloyd@gmail.com
toto@gmail.com
d
bennek@rpi.edu
tadej@tdj.si
eugene.tuv@intel.com;igor.chikalov@intel.com
samad_jahandideh@yahoo.com
The @data declaration is a single line denoting the start of the data segment in the file.
alexey.s.grigoriev@gmail.com
aad_freiburg@fhutter.de
jtkim@postech.ac.kr

Team Leader Address and phone number

Sveta Nedelja, Croatia
Institut für Informatik Albert-Ludwigs-Universität Freiburg Sekretariat Nebel/GKI Georges-Köhler-Allee 052 79110 Freiburg, Germany +49 761 203-67740
898 werer aoube
ef
Institut für Informatik Albert-Ludwigs-Universität Freiburg Sekretariat Nebel/GKI Georges-Köhler-Allee 052 79110 Freiburg, Germany +49 761 203-67740
Tadej Štajner Pražakova ulica 14 1000 Ljubljana Slovenia +38640504984
Trinity College Cambridge CB2 1TQ UK 00447890215148

Office: Q2.431, Warburger Str. 100, 33098, Paderborn, Germany +4915254783954
Sanford-Burnham Medical Research Institute, La Jolla, California 8586463100 Ext. 4047
111 York Street, Cambridge, CB1 2PZ UK 00447890215148
Georges-Köhler-Allee 52 Sekretariat Nebel/GKI 79110 Freiburg Germany
Frank Hutter Arbeitsgruppe für Lernen, Optimierung, und Automatisches Algorithmendesign c.o. Sekretariat Nebel/GKI Institut für Informatik Albert-Ludwigs-Universität Freiburg Georges-Köhler-Allee 52 79110 Freiburg im Breisgau Germany +49 761 203-67740
Arbeitsgruppe Machine Learning for Automated Algorithm Design Institut für Informatik Albert-Ludwigs-Universität Freiburg Georges-Köhler-Allee 52 79110 Freiburg im Breisgau Germany
Croatia
Damir Jajetić
Langhansstr 70 13086 Berin Germany 48 177 490 5706
Habersaathstr. 26, 10115, Berlin, Germany
Institut für Informatik Albert-Ludwigs-Universität Freiburg Sekretariat Nebel/GKI Georges-Köhler-Allee 052 79110 Freiburg, Germany Phone +49 761 203-67740
Machine Learning Group, Department of Computer Science and Engineering, POSTECH, 77 Cheongam-ro, Nam-gu, Pohang-si 37673, Gyungsangbuk-do, Republic of Korea

Other team members

aosoasdasfaf
f
Manuel Blum Katharina Eggensperger Stefan Falkner Matthias Feurer Aaron Klein Jost Tobias Springenberg Farooq Zuberi
Single Member
Emma Smith Rowan McAllister Natasha Latysheva Alex Davies
Matthias Feurer Aaron Klein Katharina Eggensperger Jost Tobias Springenberg Manuel Blum
Matthias Feurer Katharina Eggensperger Aaron Klein Stefan Falkner Marius Lindauer Manuel Blum Jost Tobias Springenberg
Matthias Feurer Jost Tobias Springenberg Katharina Eggensperger Aaron Klein Marius Lindauer Manuel Blum Stefan Falkner
Hector Mendoza Aaron Klein Matthias Feurer
Hector Mendoza Matthias Feurer Aaron Klein
Matthias Feurer Katharina Eggensperger Aaron Klein Hector D. Mendoza Jost Tobias Springenberg Manuel Blum Marius Lindauer Frank Hutter
Jongheon Jeong

Method summary

Title of your contribution

AutoCompete Again : Better selection of Algorithms
sdfasdf
a
Testing challenge
AutoSklearn
AutoKit: pipeline selection and hyper-parameter optimization
boosted trees with soft dynamic feature selection
A quick implementation of Freeze-Thaw Bayesian Optimization
Phase0
RF-CLASSIFIER
Method developer
asml.intel.com
Rational allocation of computational resources for ensemble construction via stacking
auto-sklearn
3rd place in Phase Final 2
djajetic
djajetic AutoML3
1st place final 3; 1st place auto4
djajetic Final3
djajetic AutoML4
Autonet
agrigorev_GPU
Automated Configuration of Neural Networks
GPU Final4
AutoML5
1st place Final 4 & AutoML5
Automated Machine Learning Framework Using Random Space Partitioning Optimizer

General description

asdfasdfasdfasfdafas sdasd
b
Bayesian Optimization with Random Forests in SMAC [Hutter et al. 2011] applied to a flexible configuration space describing scikit-learn [Pedregosa et al. 2011] as done in Auto-WEKA [Thornton et al. 2013]. We initialized SMAC with Meta-Learning[Feurer et al. 2015] and constructed ensembles [unpublished] with CMA-ES.Bayesian Optimization with Random Forests applied to a flexible configuration space describing scikit-learn.
The method is based on the hyperopt and hyperopt-sklearn [1] packages to pose the automatic machine learning problem as a hyperparameter optimization problem. The approach that is used in this submission extends this model to include additional learning model selection that is able to

determine admissible learning models given the problem description. It also includes different pre-processing approaches in the hyperparameter search space. This allows us to not only tune individual approaches, but also enrich the data with different representation, such as clustering to have a lower dimensional representation, or kernel approximation to cover non-linearities in the model. [1] Komer, Brent, James Bergstra, and Chris Eliasmith. "Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn." ICML workshop on AutoML. 2014.
Gradient boosting of trees built on a random subspaces dynamically adjusted to reflect learned features relevance. Huber loss function is used, no pre-processing was done.
I quickly implemented the work described in Swersky, K., Snoek, J. & Adams, R. P. Freeze-Thaw Bayesian Optimization. arXiv preprint 1406.3896 (2014). at <http://arxiv.org/abs/1406.3896> to select between 4 variants of random forest and 4 variants of gradient boosting machines.
Since the dataset was known in phase0, I built separate models for each of them. Mainly used were GBM and SVM. For the digits dataset cross validation was performed to select appropriate number of PCA components
Briefly, I have used a random-forest-based method for classification. Also, I have used gini index for feature selection.
I used classical GBT technique with prior feature selection step.
I modified the freeze thaw Bayesian optimisation algorithm (http://arxiv.org/abs/1406.3896) to be applicable to ensemble construction. Several algorithms are run, an ensemble is formed by stacking, and then various probabilistic models are used to predict which computational action will most improve the performance of the ensemble.
We used a predecessor of auto-sklearn. auto-sklearn combines the machine learning library scikit-learn with the state-of-the-art SMBO method SMAC to find suitable machine learning pipelines for a dataset at hand. This is basically a reimplementation of Auto-WEKA. To speed up the optimization process we employ a meta-learning technique which starts SMAC from promising configurations of scikit-learn. Furthermore, we use the outputs of all models and combine these into ensemble using ensemble selection.
We use auto-sklearn together with ensemble selection (but no meta-learning) as described in Section 6 of the paper reference below. Instead of a simple holdout split we used 5-fold cross validation.
We use auto-sklearn (https://github.com/automl/auto-sklearn) with a new python version of SMAC (http://www.cs.ubc.ca/labs/beta/Projects/SMAC/) for the auto phase. We used auto-sklearn with the Java version of SMAC for the tweakathon to tune auto-sklearn and deep neural networks implemented in Lasagne/Theano. For the tweakathon we used the following setting: * alexis: ? * dionis: 25 jobs; one day each; 8GB RAM; 5fold CV * grigoris: 25 jobs; one day each; 8GB RAM; 5 fold CV * jannis: 25 jobs; one day each; 4 GB RAM; 5 fold CV * wallis: 25 jobs; one day each; 4 GB RAM; 5 fold CV
Autosklearn with Neural Networks instead of scikit-learn
neural networks using keras
We use Bayesian Optimization to find a good configuration (instance) for hyper-parameters (learning rate, regularization factor, etc.) used by a Neural Network.
Main method: * auto-sklearn [1] Novelties: * AutoML: use EIPS [2] and fit the model of SMAC [3]

on a lot of meta-data used on previous datasets [unpublished] Tweakathon: * Use Deep Neural Networks and blend them with solutions found by running auto-sklearn

Automated machine learning framework solves the given task without any intervention. It selects all algorithm configurations such as a kind of algorithms, its hyperparameters, and its model parameters. General machine learning encounters the problem to find the best model parameters. To resolve the automated machine learning problem with the same manner, we parameterize an algorithm configuration and optimize it to use sequential model-based Bayesian optimization. As a result, we propose the novel Mondrian forests optimizer based on random space partitioning method, and improve one of the state-of-the-art automated machine learning framework, auto-sklearn. Random space partitioning optimizer divides the algorithm configuration space without a response, and predicts a response after building the model. It lets that the optimizer works in parallel and waits to acquire the response until the observation from the unknown function is sampled. In particular, Mondrian forests optimizer introduced in this paper is extended from Mondrian forests regression to handle categorical variables and run with the unknown function asynchronously. Moreover, it holds the forests and grows them when new algorithm configuration is inserted. These properties become a strong point for sequential model-based Bayesian optimization. The experiment results show that Mondrian forests optimizer has the better performance than other existing optimizers for the conventional global optimization benchmarks and simplified real problems.

References

sdas vavads

c

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In Proc. of LION-5, 507–523 Feurer, M.; Springenberg, T and Hutter, F. 2015. Initializing Bayesian Hyperparameter Optimization via Meta-Learning. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. JMLR 12:2825–2830 Chris Thornton, Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classifiaction Algorithms. In Proc. of KDD 2013,. Hansen, N. and A. Ostermeier (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, pp. 312-317;

Besides the hyperopt-sklearn reference, no papers describe the particular method in this submission. URL to codebase: https://github.com/tadejs/autokit

Borisov A., Eruhimov V. and Tuv, E. Tree-Based Ensembles with Dynamic Soft Feature Selection, In Feature Extraction Foundations and Applications Series: Studies in Fuzziness and Soft Computing , Vol. 207, Guyon, I.; Gunn, S.; Nikravesh, M.; Zadeh, L.A. (Eds.), Springer, 2006

They should say I implemented a slightly simplified version of Swersky, K., Snoek, J. & Adams, R. P. Freeze-Thaw Bayesian Optimization. arXiv preprint 1406.3896 (2014). at <http://arxiv.org/abs/1406.3896>

Improving the chances of successful protein structure determination with a random forest classifier S Jahandideh, L Jaroszewski, A Godzik Acta Crystallographica Section D: Biological Crystallography 70 (3), 627-635
Work not yet published. Cite github for the moment (https://github.com/jamesrobertlloyd/automl-phase-2).
@INPROCEEDINGS{feurer-automl15a, author = {M. Feurer and A. Klein and K. Eggensperger and J. Springenberg and M. Blum and F. Hutter}, title = {Methods for Improving Bayesian Optimization for AutoML}, booktitle = {ICML 2015 AutoML Workshop}, year = {2015}, month = jul, } Other relevant references are given in the paper.
@inproceedings{feurer-nips2015, booktitle = {Proceedings of the Neural Information Processing Systems Conference (NIPS)}, month = {December}, title = {Efficient and Robust Automated Machine Leraning}, author = {M. Feurer and A. Klein and K. Eggensperger and J. Springenberg and M. Blum and F. Hutter}, year = {2015}, pages = {}, }
Feurer, M. and Klein, A. and Eggensperger, K. and Springenberg, J. and Blum, M. and Hutter, F. Efficient and Robust Automated Machine Leraning In: Advances in Neural Information Processing Systems 28
Not published yet
Efficient and Robust Automated Machine Learning - Feurer, M. and Klein, A. and Eggensperger, K. and Springenberg, J. and Blum, M. and Hutter, F
[1] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, “Efficient and Robust Automated Machine Learning,” in Advances in Neural Information Processing Systems, 2015, pp. 2944–2952. [2]J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian Optimization of Machine Learning Algorithms,” in Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012, 2012, pp. 2951–2959. [3]F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential Model-Based Optimization for General Algorithm Configuration,” in Proceedings of the conference on Learning and Intelligent OptimizatioN, 2011, vol. 6683, pp. 507–523.
F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for gen- eral algorithm configuration (extended version). Technical Report 10-TR-SMAC, UBC, 2010. B. Lakshminarayanan, D. M. Roy, and Y. W. Teh. Mondrian forests for large-scale regression when uncertainty matters. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), 2016. M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In Advances in Neural Information Processing Systems (NIPS), volume 28, 2015. C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pages 847–855, 2013.

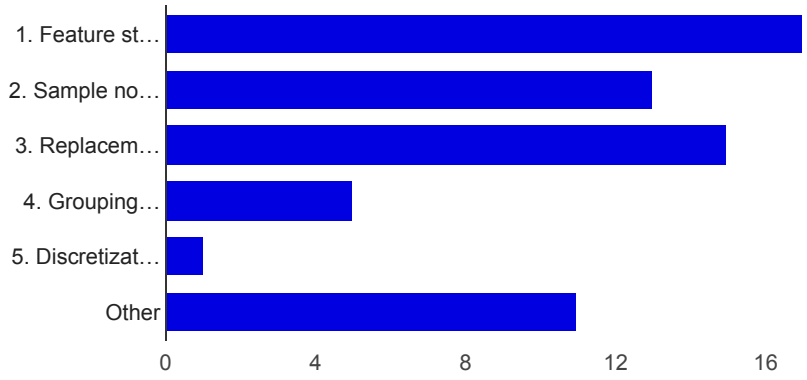
Supplementary on-line material

https://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf
sdfadsf
d

http://www.cs.ubc.ca/labs/beta/Projects/SMAC/
https://github.com/tadejs/autokit/blob/master/DESCRIPTION.md
https://github.com/jamesrobertlloyd/automl-phase-2
http://aad.informatik.uni-freiburg.de/papers/15-AUTOML-AutoML.pdf
http://aad.informatik.uni-freiburg.de/papers/15-NIPS-auto-sklearn-preprint.pdf
https://github.com/djajetic/AutoML2
https://github.com/djajetic/AutoML3
https://github.com/automl/auto-sklearn

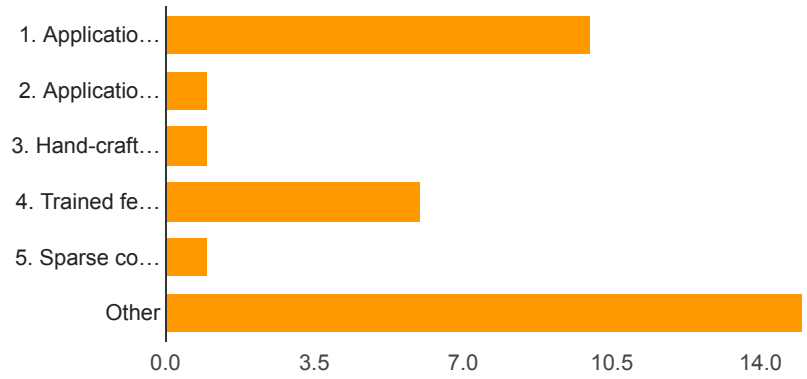
Preprocessing and feature construction

Normalization

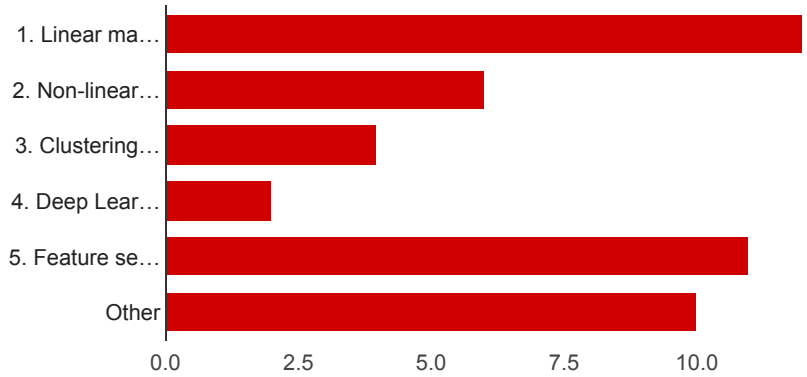


1. Feature standardization (for numerical variables)	**17**	60.7%
2. Sample normalization (for numerical variables)	**13**	46.4%
3. Replacement of the missing values	**15**	53.6%
4. Grouping modalities (for categorical variables)	**5**	17.9%
5. Discretization (for numerical variables)	**1**	3.6%
Other	**11**	39.3%

Feature extraction



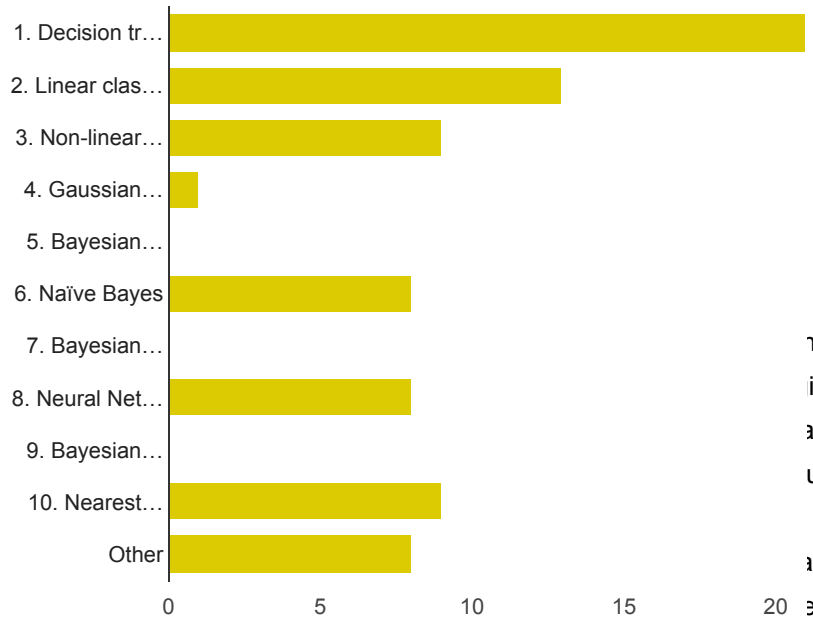
Dimensionality reduction



1. Linear manifold transformations (e.g. factor analysis, PCA, ICA)	**12**	42.9%
2. Non-linear dimensionality reduction (e.g. KPCA, MDS, LLE, Laplacian Eigenmaps, Kohonen maps)	**6**	21.4%
3. Clustering (e.g. K-means, hierarchical clustering)	**4**	14.3%
4. Deep Learning (e.g. stacks of auto-encoders, stacks of RBMs)	**2**	7.1%
5. Feature selection	**11**	39.3%
Other	**10**	35.7%

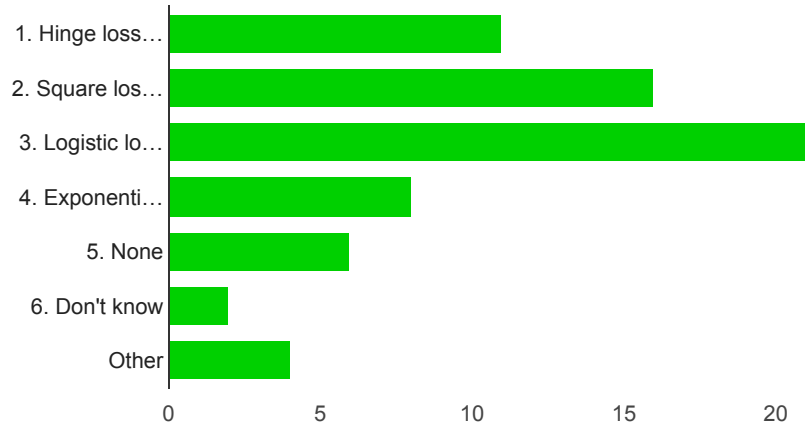
Prediction

Base predictor



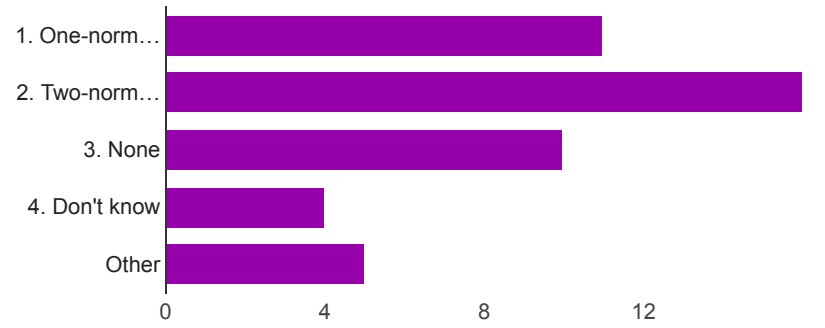
Random Forest	**21**	75%
ıear regression)	**13**	46.4%
istic regression)	**9**	32.1%
ıussian Process	**1**	3.6%
ussian Process)	**0**	0%
6. Naïve Bayes	**8**	28.6%
ın Naïve Bayes)	**0**	0%
earning Method)	**8**	28.6%
Neural Network	**0**	0%
10. Nearest neighbors	**9**	32.1%
Other	**8**	28.6%

Loss function



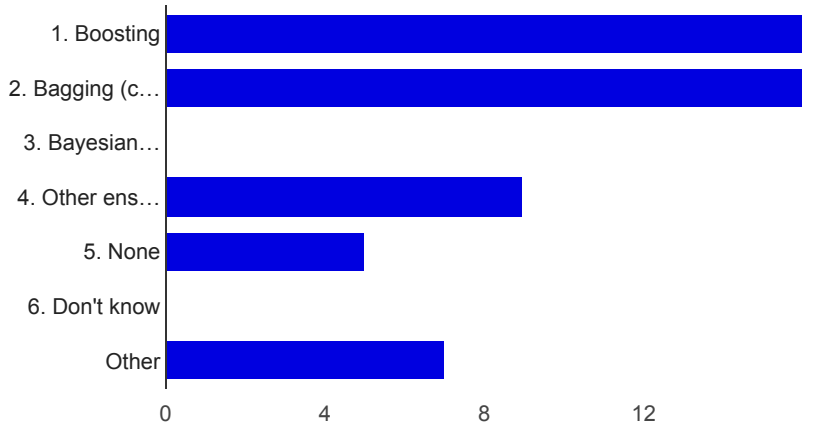
1. Hinge loss (like in SVM)	**11**	39.3%
2. Square loss (like in ridge regression)	**16**	57.1%
3. Logistic loss or cross-entropy (like in logistic regression)	**21**	75%
4. Exponential loss (like in boosting)	**8**	28.6%
5. None	**6**	21.4%
6. Don't know	**2**	7.1%
Other	**4**	14.3%

Regularizer



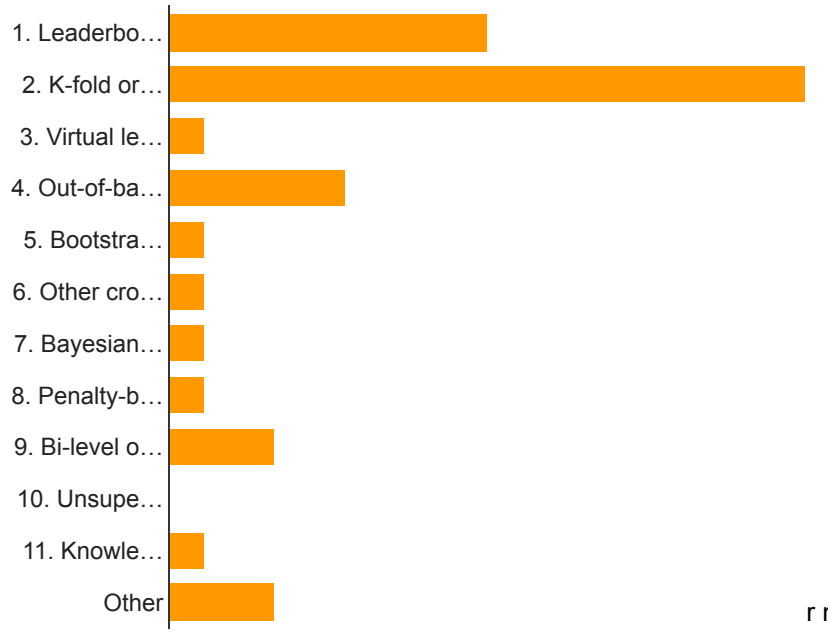
1. One-norm (sum of weight magnitudes, like in Lasso)	**11**	39.3%
2. Two-norm (||w||^2, like in ridge regression and regular SVM)	**16**	57.1%
3. None	**10**	35.7%
4. Don't know	**4**	14.3%
Other	**5**	17.9%

Ensemble method



1. Boosting	**16**	57.1%
2. Bagging (check this if you use Random Forest)	**16**	57.1%
3. Bayesian ensemble	**0**	0%
4. Other ensemble method	**9**	32.1%
5. None	**5**	17.9%
6. Don't know	**0**	0%
Other	**7**	25%

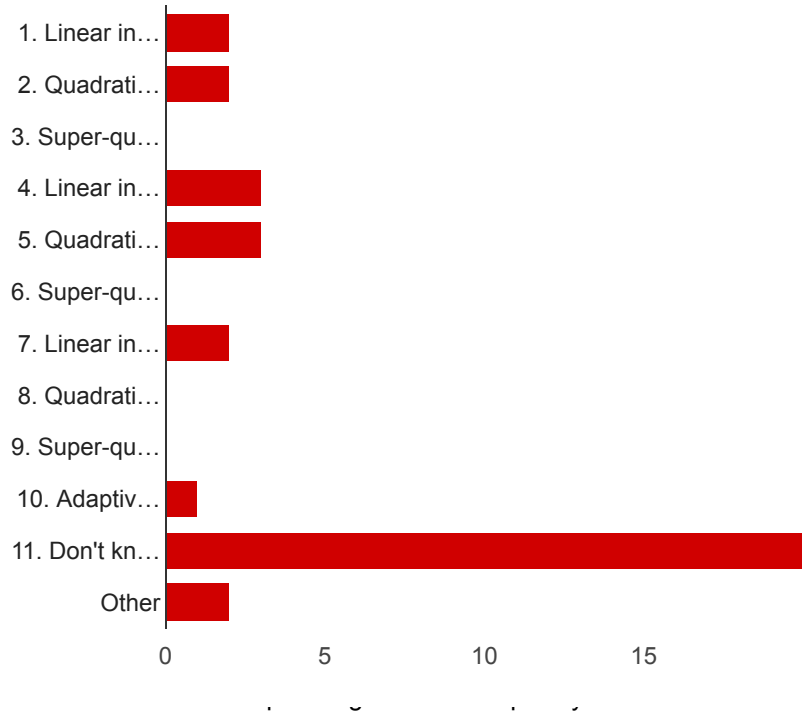
Model selection and transfer learning



r model selection	**9**	32.1%
ing training data)	**18**	64.3%
classifier training)	**1**	3.6%
4. Out-of-bag estimation (for bagging methods such as Random Forest)	**5**	17.9%
5. Bootstrap estimation (other than out-of-bag)	**1**	3.6%
6. Other cross-validation method	**1**	3.6%
7. Bayesian model selection	**1**	3.6%
8. Penalty-based method (non-Bayesian)	**1**	3.6%
9. Bi-level optimization	**3**	10.7%
10. Unsupervised learning with unlabeled validation and/or test data or transduction	**0**	0%
11. Knowledge transfer from the development data of past phases	**1**	3.6%
Other	**3**	10.7%

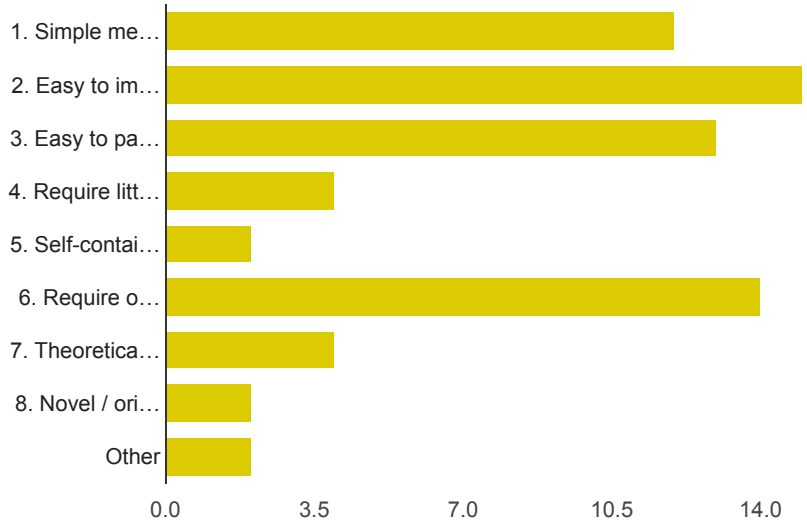
Method advantages

Algorithmic complexity



11. Don't know; too difficult to evaluate	**20**	71.4%
Other	**2**	7.1%

Qualitative advantages



1. Simple method	**12**	42.9%
2. Easy to implement	**15**	53.6%
3. Easy to parallelize	**13**	46.4%
4. Require little memory	**4**	14.3%
5. Self-contained (does not rely on third party libraries)	**2**	7.1%
6. Require only freeware libraries	**14**	50%

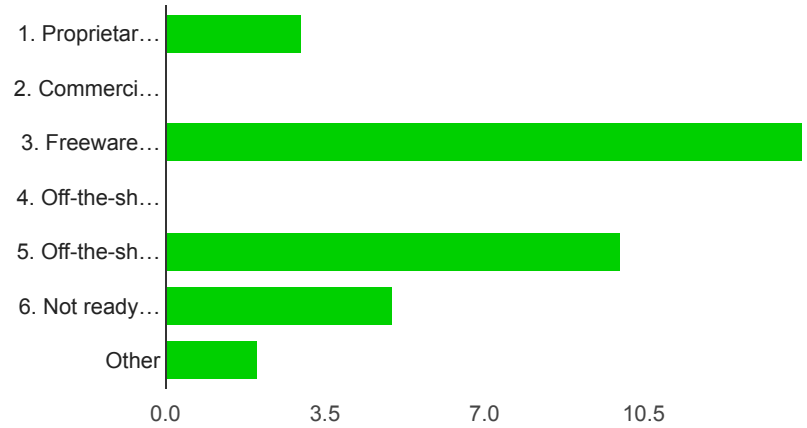
7. Theoretically motivated	**4**	14.3%
8. Novel / original	**2**	7.1%
Other	**2**	7.1%

Comparison with other methods

afdasfsafafafaf
asdf
We used an ensemble of many methods
Comparing to basic hyperopt-sklearn, the critical differences were searching across the space of preprocessing techniques.
Reasoning about benefits of different computational steps very helpful when constrained by time. Ensembles are always a good idea.
We did not try any other method
We chose a method which encompasses a lot of 'base' methods. In the reference which describes our method, one can find a comparison of our method to the 'base' methods.

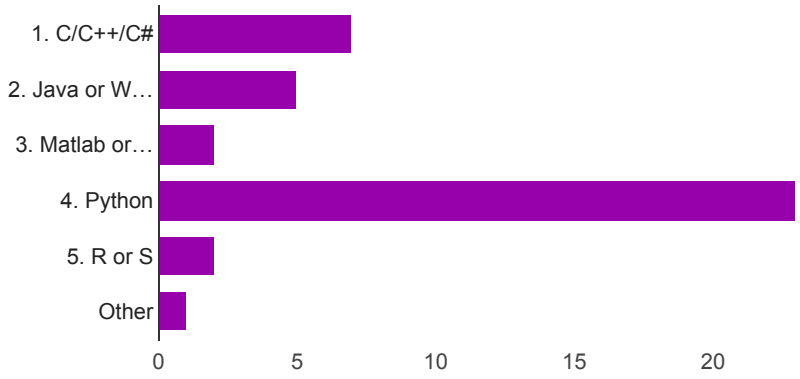
Software implementation

Availability



1. Proprietary in house software	**3**	10.7%
2. Commercially available in house software	**0**	0%
3. Freeware or shareware in house software	**14**	50%
4. Off-the-shelf third party commercial software	**0**	0%
5. Off-the-shelf third party freeware or shareware	**10**	35.7%
6. Not ready yet, but may share later	**5**	17.9%
Other	**2**	7.1%

Language



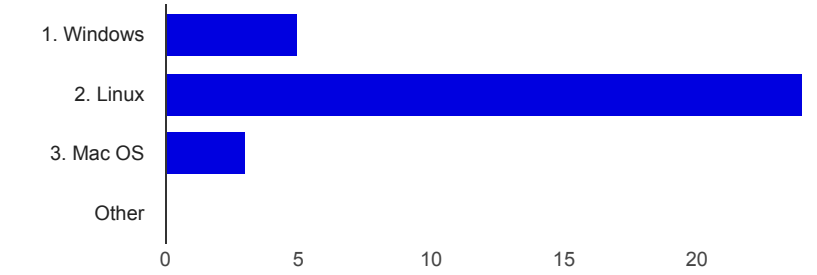
1. C/C++/C#	**7**	25%
2. Java or Weka	**5**	17.9%
3. Matlab or Octave	**2**	7.1%
4. Python	**23**	82.1%
5. R or S	**2**	7.1%
Other	**1**	3.6%

Details on software implementation

adsfafasdfad
sadf
Collection of scripts using afore-mentioned packages.
Based on: Python 2.7 NumPy SciPy scikits-learn hyperopt hyperopt-sklearn AutoML example code The main contribution of the autokit implementation is defining the space preprocessing and learning models and their hyperparameter spaces that are subsequently sampled and evaluated via hyperopt.
IDEAL is Intel's internal ML system
python + scikit-learn
Mostly python; uses externool tools SMAC (in Java) and runsolver (C++).
keras + theano
Uses standard theano and lasagne code for deep networks, with in house wrapper to use it with the automatic configuration machinery
Software freely available at github.com/automl/auto-sklearn

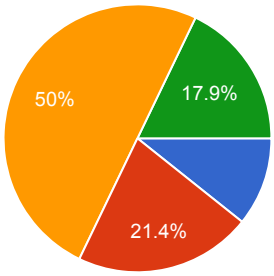
Hardware implementation

Platform



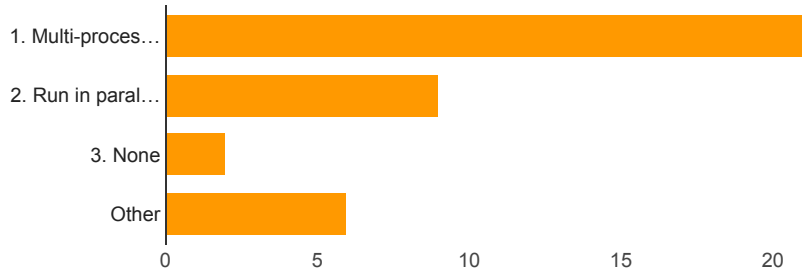
1. Windows	**5**	17.9%
2. Linux	**24**	85.7%
3. Mac OS	**3**	10.7%
Other	**0**	0%

Memory



1. <= 2 GB	**3**	10.7%
2. > 2GB but <= 8 GB	**6**	21.4%
3. > 8 GB but <= 32 GB	**14**	50%
4. > 32 GB	**5**	17.9%

Parallelism



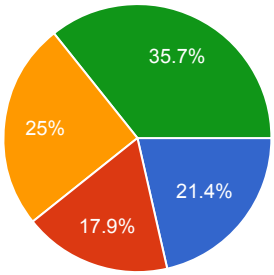
1. Multi-processor machine	**21**	75%
2. Run in parallel different algorithms on different machines	**9**	32.1%
3. None	**2**	7.1%
Other	**6**	21.4%

Code URL

https://github.com/abhishekkrthakur/AutoML
https://github.com/automl/auto-sklearn
sdfaffafadfsasdfa
asdfs
aad.informatik.uni-freiburg.de/downloads/automl_competition_2015_000.zip
https://github.com/tadejs/autokit
https://github.com/jamesrobertlloyd/automl-phase-1
https://github.com/vkocheganov/AutoML_Phase1
https://github.com/jamesrobertlloyd/automl-phase-2
http://aad.informatik.uni-freiburg.de/downloads/automl_competition_2015_001.zip
https://github.com/automl/ChaLearn_Automatic_Machine_Learning_Challenge_2015
https://github.com/djajetic/AutoML2
https://github.com/djajetic/AutoML3
https://github.com/djajetic/AutoML3Final
https://github.com/djajetic/AutoML4
https://github.com/djajetic/AutoML5
https://github.com/abhishekkrthakur/automl_gpu
https://github.com/postech-mlg-exbrain/AutoML-Challenge

Development effort

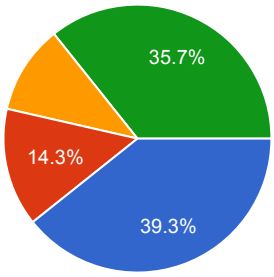
Total human effort



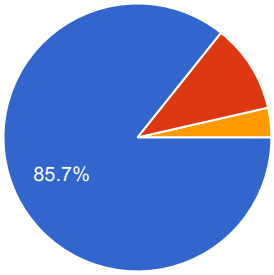
1. A few man hours	**6**	21.4%
2. A few man days	**5**	17.9%
3. 1-2 man weeks	**7**	25%
4. > 2 man weeks	**10**	35.7%

Total machine effort

1. A few hours	**11**	39.3%
2. A few days	**4**	14.3%



3. 1-2 weeks	**3**	10.7%
4. > 2 weeks	**10**	35.7%



1. Yes	**24**	85.7%
2. No, but I cannot spend more time	**3**	10.7%
3. No, please extend or run another round	**1**	3.6%

Final evaluation time (hours)

1.5
2
1
3
33
safsdf
1.5h on a eight-core machine
a few hours
It runs until the time limit by design
NA
<1
8
However long I was given. Similarly for memory usage - it uses as much as it is allowed to!
Tweakathon: ca 9600; Auto: ?
~10000
6
> 2500
4
1.67 hours
0

1.67
Bi-level optimization: > 250 CPU days; training models found by bi-level optimization: ~1 CPU day
20

Number of daily responses