An Extensive Checklist for Building AutoML Systems

Thiloshon Nagarajah[1] and Guhanathan Poravi[2]

[1] University of Westminster, New Cavendish Street, London, UK
[2] Informatics Institute of Technology, Ramakrishna Road, Colombo 6, Sri Lanka
thiloshon@gmail.com

Abstract. Automated Machine Learning is a research area which has gained a lot of focus in the recent past. But the required components to build an autoML system is neither properly documented nor very clear due to the differences and the recentness of researches. If the required steps are analyzed and brought under a common survey, it will assist in continuing researches. This paper presents an analysis of the components and technologies in the domains of autoML, hyperparameter tuning and meta learning and, presents a checklist of steps to follow while building an AutoML system. This paper is a part of an ongoing research and the findings presented will assist in developing a novel architecture for an autoML system.

Keywords: AutoML, Hyperparameter, Meta-learning, Algorithm-Selection.

1 AutoML

The umbrella term AutoML coined from 'Automated Machine Learning' [1] refers to the large scale automation of a wide spectrum of the machine learning process beyond the traditional model-creation such as data pre-processing, meta-learning [2–5], feature learning, model searching, hyperparameter optimization [6], training [7–9], workflows generation [9–12], data acquisition and reporting. These black-box learning machines gained popularity after ChaLearn initiated AutoML competitions [1] in 2015. Started as a 'benchmark for automated machine learning systems that can be operated without any human intervention' the challenge focused on automating hyperparameter tuning and model selection for classification learnings.

Even though there were many promising systems emerged from these competitions and recently we have been introduced to some commercial level AutoML systems by Google [13] and H2O.ai [14], the majority of the concepts and researches are in very early stages [15]. Researchers have used varieties of statistical theories like regularization, Bayesian priors [16], Minimum Description Length (MDL), Structural Risk Minimization (SRM) and genetic programming while further researches are required to find the best suiting techniques that are generic and works consistently. In this paper, the different components that are required to build an autoML system and the technologies available to develop those are explored.

The 1st Interdisciplinary Workshop on Algorithm Selection and Meta-Learning in Information Retrieval (AMIR), 14 April 2019, Cologne, Germany. Editors: Joeran Beel and Lars Kotthoff. Co-located with the 41st European Conference on Information Retrieval (ECIR). http://amir-workshop.org/

In section 2, we present the methodology we used to conduct this survey and collect data. Section 3 covers the architecture designs proposed by researchers. Section 4 covers the preprocessing techniques that can be automated. Section 5 deals with algorithm selection and meta learning methods to find best candidate algorithms. Section 6 covers hyperparameter optimization techniques used in this domain and their reviews. Section 7 covers how to automate the evaluation of models to choose the best one. Section 8 deals with how to benchmark the developed autoML system and in the last chapter conclusion is provided.

2 Methodology

We started gaining domain knowledge with a literature survey in the domain of autoML systems. We came across 48 such primary studies and identified the different approaches used for autoML in the available work. We short listed techniques used by the available work in order to achieve hyperparameter optimization, meta learning and algorithm selection. These techniques were selected to be discussed in this paper, according to the majority of use.

3 AutoML Architecture

Throughout the researches on the autoML domain, the final goal has always been to automate the entire pipeline of the machine learning. However, it has proved to be a difficult task as a whole. Thus several work has been conducted in automating at least some part of machine learning, with the intention of putting all these together at the end. Liu [17], realized this limitation and came up with two categories to differentiate this.

1. Narrow AutoML deals with partial automation or concentration of autoML systems, that is mainly fueled by commercial needs.
2. Generalized AutoML aims to automate the entire process, which would lead way to Artificial General Intelligence and is predominantly seen in academic researches.

According to him, even though most of the work available are narrow autoML, it is eminent to achieve pivotal progresses in generalized autoML. The same concept has been covered by Guyon et al [1] as well, who termed these two categories as *semi-automated* and *fully-automated* autoML. With this understanding, more focus is given to generalized autoML in this paper.

An autoML system will need to automate all the parts of machine learning in its architecture. Das and Cakmak [18] came up with a requirement list that reflects this. It had moving and interconnected components of machine learning that needs to be automated including feature preprocessing, feature selection, model selection and hyperparameter optimization. They defined all the required components to achieve, full automation as follows,

Table 1. Components in modal AutoML Architecture [18]

Phases	Components	Methods
Data Sources	Historical data	Databases, Flat files
	Real-time data	Streaming data, Data providers (API)
Data Processing	Clean, format, quality check	Find, replace, modify, delete Ensure accuracy and consistency
	Feature transformations	Encoding, indexing, scaling, assembling, expanding, normalizing, binarization
	Feature selection	Backward/forward elimination, vector slicing, chi-squared selection
Model Training	Algorithm selection	Supervised, unsupervised, semi-supervised
	Evaluation	Cross validation, performance metrics
	Hyperparameter Optimization	Optimal hyperparameter settings
	Versioning	Model and pipeline versioning
Deployment	Workload type	Online, batch, streaming deployments
	Monitoring	Handling model performance decay
	Continuous learning	Re-training model as new data comes

Though this covers all major aspects of machine learning, few important steps like train/test splitting, models ensembling and reporting are missed in this list. Most of the architectures proposed in autoML domain (AutoWeka, Hyperopt-Sklearn, Auto-Sklearn, TPOT, H2O.ai) contains at least some subset of these components. Olson et al proposed an architecture [48] where these systems are arranged into different engines. Addition to visualization and graph engines that offers insights which are not listed in the above list, they also added a 'Human Engine', which takes human inputs for maintenance of the autoML system. With this understanding of architecture, let's analyze how each component can be developed.

4 Automated preprocessing

Raw data used in machine learning is often unclean, skewed and noisy. Cleaning data and feature transformations have proven to improve accuracy of machine learning systems substantially. Thus the first step of an AutoML system is data preprocessing. The following section covers some important preprocessing steps that can be automated at least to certain extent.

Data transformation. For *numerical data*, some of the common processes to automate are,

- scaling - standardization and normalization
- missing values imputation - using global constant, using mean / median, using indicator variable, predicting the most probable value or simply removing record
- outlier detection: univariate - interquartile range and filtering, 'Winsorizing' or trimming: multivariate - one class SVM, Local Outlier Factor (LOF) and isolation forest

- binning – equal width binning, equal frequency binning**;** log and power transformations
- identifier detection

For *categorical variables*, some of the common processes are,

- encoding - label encoding, one-hot encoding, frequency-based encoding, target mean encoding, binary encoding, hash encoding
- replacing missing values with the mode

Additionally, for other data types such as text or video, several other preprocessing techniques like tokenization, normalization or substitution are available. An interesting point to note is that, tree based supervised models such are random forests are able to handle feature or data abnormalities by themselves, whereas non-tree based supervised learning algorithms, are much sensitive to abnormalities.

Feature selection. Some of the feature selection that can be done before creating a model are as follows,

- Identifying highly correlated variables and treating them
- Excluding features with low variance or univariate feature selection
- Recursive feature elimination - Measuring information gain for the available set of features and choosing the top N features accordingly
- Dimensionality reduction with PCA - transforming the data in the high-dimensional space to a space of fewer dimensions.

And, if we are to do feature selection after creating baseline model,

- Using linear regression and selecting variables based on p values
- Using stepwise selection for linear regression and selecting the important variables
- Feature selection using random forest - Using random forest and selecting the top N important variables
- Feature generation - It is also possible to generate new features from the intrinsic data with techniques such as numerical feature generation, pairwise feature creation, categorical feature creation, temporal feature creation, etc.

These preprocessing steps can improve efficiency and accuracy of the subsequent machine learning workflows. Several of these preprocessing can be decided to be used based on simple statistical metrics. For example, multi-collinearity between features can be found with Pearson's Correlation Coefficients. These variables can be treated with stepwise regression or principle component analyses. Such automatic decision-execution pairs can help build a powerful autoML system.

5 Automated Algorithm Selection and Model Initiation

Next step is to automatically find candidate algorithms suitable for the dataset. The categories of machine learning algorithms that needs to be considered are as follows,

Table 2. Types of Machine Learning

Machine Learning	Supervised Learning	Regression Algorithm
		Classification Algorithms
	Unsupervised Learning	
	Semi supervised Learning	
	Reinforcement Learning	
	Transfer Learning	

There are other aspects data scientists worry about in algorithm selection, such as computational complexity, differences in training and scoring time, linearity versus non-linearity, etc. and it's useful if these are to be considered while automating. The main quantitative techniques in this paradigm can be categorized as rules-based and meta-learning. The following section discusses these techniques.

5.1 Rules Based

In rules based systems, we try to mimic how a data scientist manually does algorithm selection, which is a mixture of initial exploration of the dataset and his experience. Certain characteristics of the dataset and the domain of the dataset can suggest the possible candidates of algorithms to build machine learning experiments with. A rules system can be implemented to reflect these characteristics with the help of many cheat sheets of algorithms publicly available in the internet. For example, Scikit Learn python package has a map [19] to help its users find the best classifier available in its package. Rules Based Machine Learning is a "method that identifies, learns, or evolves 'rules' to store, manipulate or apply" [20] for decision making mechanisms. Set of rules in the format {IF 'condition' THEN 'result'} makes up this rules system or knowledge base. In autoML space, characteristics can be modelled as conditions and algorithms as results. For example, in python language, *Skope-rules* is used to perform RBML. Under RBML, **Learning Classifier Systems** [21] uses genetic algorithm for discovery component and usual machine learning classifiers as learning component.

Case Based Reasoning [22] solves new problems, based on the solutions of the past identical problems. It follows a four step method of retrieving, reusing, revising and retaining. Rule induction is a term to denote general area of ML where formal rules are extracted based on set of observations. In autoML landscape, all of these techniques can help build a rules system that can suggest algorithms based on the dataset properties and the domain the data is from.

5.2 Meta Learning

With metalearning (Machine Learning for meta-learning) [23, 24], we try to gain insights from the metadata of the machine learning experiments. Results of each model training is stored along with its dataset and performance details and used in the future runs. There has been substantial interest in the meta learning space in the recent past and many autoML systems (TPOT, Auto-Sklearn) have integrated these.

The first step in any meta learning solution is creating a meta-database. *OpenML* [25] is one such prominent database available now. These databases will contain information about datasets such as number of features, number of records, correlation of features, number of missing values, information about models such as algorithms, hyperparameter spaces and also performance information such as running time and accuracy. These metadata can be learned with machine learning algorithms and best algorithm settings can be predicted for new datasets [26, 27]. It can also be used to suggest the initial hyperparameter settings to start modelling. Feurer et al [27] in their research gathered 140 datasets from *OpenML* repository and created instantiation settings based on meta-features using Bayesian optimization (SMAC with cross-validation) that checks empirical performance for that dataset. Then, meta-features of new dataset are stripped and compared with L1 distances of offline datasets gathered before, to choose 25 nearest datasets and their parameter space. Optimization is then done starting from these initializations to get high model accuracies.

5.3 Graphical Methods

Algorithm selection is almost always backed by visualizing the dataset in graphical methods. These Exploratory Data Analysis (ETA) methods can be done in conjunction with other statistical methods. This helps one understand data beyond the statistical modelling or hypothesis testing procedures. The only issue with these methods are that these cannot be automated. This is done solely with the supervision of a human component. But viewing these in the AutoML system as part of the configuration step or report generation step will give additional insight to the user. The following table gives an overview of such exploratory methods.

Table 3. Graphical ETA methods

Method	Description	Used To
Ordination	Mainly used in data clustering, groups similar multivariate objects near each other and dissimilar objects farther. Most common ordination technique is Principle Component Analysis (PCA).	Very common and widely used.
Median polish	Uses the medians of the rows and columns to iteratively fit model for the data.	Not sensitive to outliers but very simple method.

Box plot	Used in numerical data to depict quartiles and variabilities outside quartiles.	To find spread, skewness and outliers in the data
Histogram	Used in numerical data to accurately depict distribution of the data. It represents the probability distribution of continuous variables, but is limited to a single variable per graph.	Find density of distributions and one of seven tools of quality control [28].
Run Chart	Used in time series data to display data in time sequence. Used as univariate graphical method.	Validate univariate data assumptions and find anomalies / outliers over time.
Scatter Plot	Drawn in Cartesian coordinates to compare two variables of the data. It is possible to add more dimensions in terms of color codes or point shapes.	Find correlation in data and is one of seven tools of quality control.
Parallel Coordinates	Used to visualize high dimension geometry and multivariate data. It is closely related to time series graphs but it doesn't have any time variables, thus do not have natural order.	Find relationship between dimensions.
Targeted projection pursuit	Used in very complex data to find features or patterns of interest.	Find 'interestingness' in data and as feature selection method

Now that we have analyzed several algorithm selection methods, the following table gives a summary of these techniques grouped under its three types.

Table 4. Comparison of algorithm selection methods

Technique	Advantage	Disadvantage	Used researches
Rules-based	Follows strict methodology. Easily interpretable. Mimics common human process.	Not intelligent nor adaptable.	-
Meta-learning	Automated and adaptable. Makes use of past learnings and data	High processing power required. Proper dataset and periodical update is required.	Hyperopt-Sklearn, AUTO-SKLEARN
Graphical Methods	Very intuitive and human explainable	Non automatable and requires subject expertise	*PennAI*, *H2O*, Cloud offerings like *GoogleML* and *AzureML*

5.4 Model Selection

After algorithm selection, the machine learning model and its features to learn will be customized. There are few quantitative methods data scientists use manually that can be automated as discussed below.

Akaike information criterion (AIC) / Watanabe–AIC (WAIC) calculates the relative quality of models for a dataset compared to other models, thus can be used for model selection [29]. It uses the amount of information lost by the model as the parameter of quality. It is very common and widely used.

Bayesian information criterion (BIC) [30] is very similar to AIC but is based on likelihood function. Model with smaller BIC value is considered the best. BIC cannot handle high dimension model selection tasks and at times, is less effective than AIC [29].

Focused information criterion (FIC) is yet another method for selecting best model among possible competitors. Model with the best estimated precision is chosen. Unlike AIC or BIC, FIC [30] doesn't find overall fitness of models, rather, on the parameter of primary interest that gives different estimates in all candidates.

Mallows's Cp calculates fit of regression models, where a model with best subset of predictors among predictor variables, available for some outcome is chosen. Small value of Cp is considered to be more precise. Cp only works well in large sample sizes and can't handle complex collection of models.

Stepwise Regression (SR) chooses each feature in the dataset incrementally and finds accuracy of the models [33]. By following this for every feature, it chooses the set of features that increases accuracy of the models and removes others. This can be used as a feature selection mechanism. Typically, AIC, BIC, FDR or Mallows's Cp is used as the selection criterion. Stepwise regression if often criticized as data dredging and biased as it works on the data itself and favored over by ensembles.

6 Automated Hyperparameter optimization

In the entire machine learning landscape there are two types of parameters.

- Model parameters that are learned by the algorithm while learning, thus does not need to be automated
- Hyperparameters that needs to be set before beginning of the learning, thus needs to be automated

Optimizing the hyperparameter is a function with the objective of minimizing the loss / cost of the algorithm, which in turn helps keep balance between the model bias and variance. This is essential in getting a low cross-validation error at the end of the experiment. While automating machine learning process, it is also expected to automate tuning or optimizing hyperparameters that best fits the dataset. In the following section analysis of hyperparameter optimization techniques are provided.

6.1 Simple Search Approaches

The most trivial techniques of hyperparameter tuning is grid search and randomized search.

Grid Search. Grid search expects few set of values as parameter space and tries all combinations of these values to learn in brute force manner. Search will be guided by a metric, which is often cross validation error of the training data or evaluation on the test data. Grid search suffers from curse of dimensionality, because even when there are two hyperparameters and five distinct values of these parameters, it requires twenty-five times of modelling and evaluation. Besides, there is no feedback or adjustment mechanism, thus the algorithm is highly unintelligent.

Random Search. Random search [31] is very similar to grid search and does pretty much the same, but in a random combination of hyperparameters. It is proved to outperform Grid search, but performs poorly in real cases as there is not adjustment or feedback in the learning process based on the results of previous learning.

Because of the limitations of simple search approaches came a second technique called 'Sequential Model based global Optimization' (SMBO) [32, 33].

6.2 Heuristic Search

Sequential Model based global Optimization. In scenarios where the evaluation of fitness function is expensive and costly, these model-based systems evaluate fitness with a surrogate that is cheaper to calculate [34]. Among other options, Expected Improvement (EI) has turned out to be a good candidate for the surrogate fitness function. The concept is to use objective functions like Gaussian process to choose good hyperparameter values and then sequentially update values based on results. This makes use of the results of the previous iteration to find better hyperparameter values to try in the next iteration, thus, is considered smart.

Bayesian-based hyperparameter optimization is one of SMBO technique that is widely used in the autoML systems. Bayesian optimization is proved to work much better than other alternatives, as it is able to reason about the quality of runs before they even start. It has been proven Gaussian process [16] based BO [*Spearmint* [35]] to perform better on low-dimension data and tree based BO to perform better on high dimension data [27]. Within tree based systems, random forest based SMAC (Sequential Model-based Algorithm Configuration) works better than another high performing system, TPE (Tree-structured Parzen Estimator). SMAC is also faster as it uses cross validation fold wise and removing poor parameter settings early in the optimizations.

Building on top of Bayesian optimization, there has been other advancements as well. For example, the concept of meta-learning has been used to build the initial model for BO to optimize. By referring to the meta data of the hyperparameters and their performance on past similar datasets, new parameter spaces are developed that are more likely to fit well. Other than BO, there are Random Online Aggressive Racing (ROAR) as well.

6.3 Evolutionary Optimization

In evolutionary optimization [34], evolutionary algorithms follow a process inspired by biological concept of evolution. It first creates a random hyperparameter population as much as one hundred. It starts evaluating these and gets their fitness functions. Based on these relative fitness values, parameters are ranked. Worst performing tuples of parameters are replaced by new ones generated through crossover and mutation. This is repeated until the performance is not improved. Though mainly used in deep learning tasks, these have started to be used in typical machine learning as well.

In the recent times, there has been interests in developing techniques out of this standard scope, to achieve hyperparameter optimization. Genetic programming [36], transfer learning [37] and reinforcement learning [38] are some of the techniques used. Genetic programming is mainly used in neural networks and SVMs. Bandit-based approaches have been developed which uses small subsets of data to find settings space to try on complete data, making the process much more efficient.

7 Automated Evaluation

Typically, a model evaluation can contain statistical methods as well as the business rules specific for the problem. While the statistical methods are general to all learning problems, business rules will be tailor made to the question at hand. Even though business rules can be skipped in the autoML system, it is essential to automate the statistical evaluation techniques. Along with it, there can also be other processes like refining models, re-training, and deployment to be automated.

The following table gives an overview of few evaluation metrics that is used while training models.

Table 5. Training evaluation metrics

Description	
Advantage / Limitation	**Used if**
Hold-Out Validation: In a dataset with independently and identically distributed (IID) records, a small subset of random records is held out for validation. Model training is done with the large portion and the evaluation metrics are calculated with the smaller potion. A common practice is to subset 20% as validation set.	
Very easy to subset, but since validation is done on the smaller subset, generalization error can be less reliable and higher variance.	If the dataset is big enough to break into subsets.
Cross-Validation / Out-of-Sample Testing: In this method, dataset is first divided into k number of folds. Iteratively we consider each fold as the held-out validation set and training is done on the rest of the folds. The overall performance is the average of all k folds.	
Much better metric than hold out validation. Can even be used in hyperparameter tuning to calculate performance of tuples.	If the dataset is very small or computer power is limited.

The following table gives an overview of few evaluation metrics that is used while testing models in regression tasks.

Table 6. Regression evaluation metrics

Root Mean Square Error (RMSE / MSE): The most commonly used metric. It is defined as square root of the average squared distance between the actual score and the predicted score. In other words, sample standard deviation between predicted and observed values. It gives us the sense of how far the predictions were from actual values. Lower the RMSE better the model.	
Very common and widely used.	Always
Mean Absolute Error (MAE): Similar to RMSE, but the absolute values of distances are taken. Thus all the individual differences are weighted equally which makes it a linear score.	
Easy to interpret and understand than RMSE. More robust to outliers.	If interpretation is important.
R Squared (R^2): R2 tells how the selected independent variables explain variability in the dependent variables. Simply said, it tells how close the data are to the fitted regression line, which is also known as coefficient of determination. The higher R Squared means model fits well.	
Doesn't necessarily tell if the model is bad or good. Just gives the relationship.	In exploratory analysis.

The following table gives an overview of few evaluation metrics that is used while testing models in classification tasks.

Table 7. Classification evaluation metrics

Accuracy: Tells how often classier makes correct prediction. Calculated as the ratio of number of correct predictions among total predictions.	
Very simple and widely used.	For simple metric.
Confusion Matrix: Shows a detailed breakdown of correct classifications in classes. Presented as table of ground truth labels and predictions.	
More detailed than accuracy, thus can diagnose issues in dataset.	For metric breakdown of individual classes.
Logarithmic Loss: Used when the classifier gives numeric probability as output instead of class. It is considered as soft measurement as it contains details of how incorrect or how correct a prediction is and not just if it is correct.	
More tolerant to confidence values.	If output is numeric probability.
Area Under the Curve: Shows the sensitivity of classifier by plotting the rate of true positives to the rate of false positives. Used mainly in binary classification. Greater the AUC means it's a better model.	
Hard to interpret	For binary classifications

8 Benchmarking AutoML systems

Once an autoML system is developed, it will have to be benchmarked against existing systems and manual procedures. Though it won't be a component of the system itself and not required to be automated, it helps evaluate the built system. The following section discusses three researchers and their benchmarking methods.

Thronton et al [39] in their system Auto-WEKA, used 21 prominent bench mark datasets including 15 from the UCI repository with 70%-30% train-test splitting. Intel Xeon X5650 six-core processors, at 2.66GHz were used with RAM limit of 3GB for classification datasets to mimic typical data scientist settings. Bootstrap sampling and cross validation were used to choose best setting.

Feurer [27] in his project used 140 binary and multiclass classification sets with more than 1000 data points from *OpenML*. These datasets were of varied types like text, digit, gene, telescope and advertisement. He used balanced classification error rate instead of standard classification error since the data sets had imbalanced class distribution. Trainings were done on multiple controlled schemes like, with and without meta learning and, with and without ensembles. When testing a dataset, meta-data of only 139 other datasets were used according to leave-one-dataset-out method.

Allen et al [40] in their benchmarking project used mean squared error and weighted F1 score for regression and classification tasks. They choose 57 classification and 30 regression *OpenML* datasets. As the process was pretty extensive with as much as 10,440 compute hours, they opted for an amazon web service distributed setup. Average of several different pairwise comparisons with different seed values were considered as the final performance score.

9 Future Avenues and Conclusion

This paper presented the steps to build an automated machine learning system starting from data preprocessing to model deployment. The different methods and technologies available to develop these systems and their review were also discussed. From the findings, it is clear the algorithm selection and feature preprocessing components require more refinement from research community while hyperparameter tuning and meta learning spaces are explored actively. More technologies and statistical concepts unexplored in the autoML systems will make up the majority of future efforts while the knowledge of previous efforts need to be accumulated as knowledge hubs or meta databases. So far most of the researches are in the Python and Java languages while statistical languages like R can open up more options to explore. Since execution of multiple model trainings can take up high computational power, autoML systems needs to explore distributed task offloading mechanisms actively.

With this knowledge of the autoML components, and how they can be improved, we hope to come up with an architectural style in near future, towards an efficient automated machine learning system.

References

1. Guyon, I., Bennett, K., Cawley, G., Escalante, H.J., Escalera, S., Tin Kam Ho, Macia, N., Ray, B., Saeed, M., Statnikov, A., Viegas, E.: Design of the 2015 ChaLearn AutoML challenge. Presented at the July (2015).
2. Bardenet, R., Brendel, M., Kégl, B., Sebag, M.: Collaborative hyperparameter tuning. Proceedings of the 30th International Conference on International Conference on Machine Learning. 28, II–199--II–207 (2013).
3. Kim, J., Lee, S., Kim, S., Cha, M., Lee, J.K., Choi, Y., Choi, Y., Cho, D.-Y., Kim, J.: Auto-Meta: Automated Gradient Based Meta Learner Search. 32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada. (2018).
4. Sun-Hosoya, L., Guyon, I., Sebag, M.: Algorithm Recommendation with Active Meta Learning. IAL 2018 workshop, ECML PKDD, Sep 2018, Dublin, Ireland. 12 (2018).
5. Yang, C., Akimoto, Y., Kim, D.W., Udell, M.: OBOE: Collaborative Filtering for AutoML Initialization. arXiv:1808.03233 [cs, stat]. (2018).
6. Guyon, I., Sun-Hosoya, L., Boulle, M., Es, H.J.: Analysis of the AutoML Challenge series 2015-2018. 46 (2018).
7. Costa, V.O., Rodrigues, C.R.: Hierarchical Ant Colony for Simultaneous Classifier Selection and Hyperparameter Optimization. In: 2018 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE, Rio de Janeiro (2018).
8. de Sá, A.G.C., Freitas, A.A., Pappa, G.L.: Automated Selection and Configuration of Multi-Label Classification Algorithms with Grammar-Based Genetic Programming. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., and Whitley, D. (eds.) Parallel Problem Solving from Nature – PPSN XV. pp. 308–320. Springer International Publishing, Cham (2018).
9. Wever, M., Mohr, F., Hüllermeier, E.: Automated Multi-Label Classification based on ML-Plan. arXiv:1811.04060 [cs, stat]. (2018).
10. Gil, Y., Yao, K.-T., Ratnakar, V., Garijo, D., Steeg, G.V.: P4ML: A Phased Performance-Based Pipeline Planner for Automated Machine Learning. Proceedings of Machine Learning Research 1:1–8, 2018 ICML 2018 AutoML Workshop. 8 (2018).
11. Mohr, F., Wever, M., Hüllermeier, E.: ML-Plan: Automated machine learning via hierarchical planning. Machine Learning. 107, 1495–1515 (2018).
12. Wever, M., Mohr, F., Hullermeier, E.: ML-Plan for Unlimited-Length Machine Learning Pipelines. ICML 2018 AutoML Workshop. 8 (2018).
13. Le, Q., Zoph, B.: Using Machine Learning to Explore Neural Network Architecture, https://ai.googleblog.com/2017/05/using-machine-learning-to-explore.html, (2017).
14. AutoML: Automatic Machine Learning, http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html, (2018).
15. Bogatinovski, J.: Automating machine learning for structured output prediction. 25 (2018).

16. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian Optimization of Machine Learning Algorithms. arXiv:1206.2944 [cs, stat]. (2012).
17. Liu, B.: A Very Brief and Critical Discussion on AutoML. arXiv:1811.03822 [cs]. (2018).
18. Das, S., Cakmak, U.M.: Hands-on automated machine learning: a beginner's guide to building automated machine learning systems using AutoML and Python. (2018).
19. Scikit-learn: Choosing the right estimator, https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html.
20. Weiss, S.M., Indurkhya, N.: Rule-based Machine Learning Methods for Functional Prediction. arXiv:cs/9512107. (1995).
21. Urbanowicz, R.J., Moore, J.H.: Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. Journal of Artificial Evolution and Applications. 2009, 1–25 (2009).
22. Agnar, A., Enric, P.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. 39–59 (1994).
23. Feurer, M., Springenberg, J.T., Hutter, F.: Initializing Bayesian Hyperparameter Optimization via Meta-Learning. AAAI'15 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. 1128–1135 (2015).
24. Lemke, C., Budka, M., Gabrys, B.: Metalearning: a survey of trends and technologies. Artificial Intelligence Review. 44, 117–130 (2015).
25. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. ACM SIGKDD Explorations Newsletter. 15, 49–60 (2014).
26. Komer, B., Bergstra, J., Eliasmith, C.: Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn. PROC. OF THE 13th PYTHON IN SCIENCE CONF. (SCIPY 2014). 7 (2014).
27. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and Robust Automated Machine Learning. NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems. 2, 2755–2763 (2015).
28. Tague, N.R.: The quality toolbox. ASQ Quality Press, Milwaukee, Wis (2005).
29. Burnham, K.P., Anderson, D.R.: Multimodel Inference: Understanding AIC and BIC in Model Selection. Sociological Methods & Research. 33, 261–304 (2004).
30. Claeskens, G., Hjort, N.L.: The Focused Information Criterion. Journal of the American Statistical Association. 98, 900–916 (2003).
31. Bergstra, J., Bengio, Y.: Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research. 13, 281–305 (2012).
32. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential Model-Based Optimization for General Algorithm Configuration. In: Coello, C.A.C. (ed.) Learning and Intelligent Optimization. pp. 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg (2011).
33. Daning, C., Hanping, Z., Fen, X., Shigang, L., Yunquan, Z.: Using Known Information to Accelerate HyperParameters Optimization Based on SMBO. arXiv:1811.03322 [cs, stat]. (2018).

34. Bergstra, J., Bardenet, R., Bengio, Y., Kegl, B.: Algorithms for Hyper-Parameter Optimization. Curran Associates Inc. , USA ©2011. 2546–2554 (2011).
35. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. MIT Press, Cambridge, Mass (2006).
36. Paris, L.: Genetic Programming/Auto-ML for One-Shot Learning. 5 (2018).
37. Quanming, Y., Mengshuo, W., Hugo, J.E., Isabelle, G., Yi-Qi, H., Yu-Feng, L., Wei-Wei, T., Qiang, Y., Yang, Y.: Taking Human out of Learning Applications: A Survey on Automated Machine Learning. arXiv:1810.13306 [cs, stat]. (2018).
38. Li, Y.-F., Wang, H., Wei, T., Tu, W.-W.: Towards Automated Semi-Supervised Learning. Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI-19). 33, 8 (2019).
39. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. arXiv:1208.3719 [cs]. (2012).
40. Balaji, A., Allen, A.: Benchmarking Automatic Machine Learning Frameworks. arXiv:1808.06492 [cs, stat]. (2018).
41. Cashman, D., Humayoun, S.R., Heimerl, F., Park, K., Das, S., Thompson, J., Saket, B., Mosca, A., Stasko, J., Endert, A., Gleicher, M., Chang, R.: Visual Analytics for Automated Model Discovery. arXiv:1809.10782 [cs]. (2018).
42. Shen, W.: DARPA's Data Driven Discovery of Models (D3M) and Software Defined Hardware (SDH) Programs. In: Proceedings of the 2018 on Great Lakes Symposium on VLSI - GLSVLSI '18. pp. 1–1. ACM Press, Chicago, IL, USA (2018).
43. Bergstra, J., Yamins, D., Cox, D.D.: Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. Proceedings of the 30 th International Conference on Ma- chine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP. 28, 9 (2013).
44. Liu, Z., Bousquet, O., Elisseeff, A., Escalera, S., Guyon, I., Jacques, J., Pavao, A., Silver, D., Sun-Hosoya, L., Treguer, S., Tu, W.-W., Wang, J., Yao, Q.: AutoDL Challenge Design and Beta Tests-Towards automatic deep learning. CiML workshop @ NIPS2018, Dec 2018, Montreal, Canada. 7 (2018).
45. Mahpod, S., Keller, Y.: Auto-ML Deep Learning for Rashi Scripts OCR. arXiv:1811.01290 [cs]. (2018).
46. Gijsbers, P.: Automatic construction of machine learning pipelines. 65 (2017).
47. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., Han, S.: AMC: AutoML for Model Compression and Acceleration on Mobile Devices. 17 (2018).
48. Olson, R.S., Sipper, M., La Cava, W., Tartarone, S., Vitale, S., Fu, W., Orzechowski, P., Urbanowicz, R.J., Holmes, J.H., Moore, J.H.: A System for Accessible Artificial Intelligence. arXiv:1705.00594 [cs]. (2017).