# Hyperparameter Importance Across Datasets

Jan N. van Rijn
Albert-Ludwigs-Universität Freiburg
Freiburg, Germany
vanrijn@cs.uni-freiburg.de

Frank Hutter
Albert-Ludwigs-Universität Freiburg
Freiburg, Germany
fh@cs.uni-freiburg.de

## ABSTRACT

With the advent of automated machine learning, automated hyperparameter optimization methods are by now routinely used in data mining. However, this progress is not yet matched by equal progress on automatic analyses that yield information beyond performance-optimizing hyperparameter settings. In this work, we aim to answer the following two questions: Given an algorithm, what are generally its most important hyperparameters, and what are typically good values for these? We present methodology and a framework to answer these questions based on meta-learning across many datasets. We apply this methodology using the experimental meta-data available on OpenML to determine the most important hyperparameters of support vector machines, random forests and Adaboost, and to infer priors for all their hyperparameters. The results, obtained fully automatically, provide a quantitative basis to focus efforts in both manual algorithm design and in automated hyperparameter optimization. The conducted experiments confirm that the hyperparameters selected by the proposed method are indeed the most important ones and that the obtained priors also lead to statistically significant improvements in hyperparameter optimization.

## CCS CONCEPTS

• **Computing methodologies** → *Supervised learning by classification*; *Batch learning*;

## KEYWORDS

Hyperparameter Optimization; Hyperparameter Importance; meta-learning

**ACM Reference Format:**
Jan N. van Rijn and Frank Hutter. 2018. Hyperparameter Importance Across Datasets. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220058

## 1 INTRODUCTION

The performance of modern machine learning and data mining methods highly depends on their hyperparameter settings. As a consequence, there has been a lot of recent work and progress on hyperparameter optimization, with methods including random search [2], Bayesian optimization [1, 20, 26, 34, 38], evolutionary optimization [29], meta-learning [6, 16, 30, 32, 40, 41] and bandit-based methods [23, 28].

Based on these methods, it is now possible to build reliable automatic machine learning (AutoML) systems [12, 39], which – given a new dataset $\mathcal{D}$ – determine a custom combination of algorithm and hyperparameters that performs well on $\mathcal{D}$. However, this recent rapid progress in hyperparameter optimization and AutoML carries a risk with it: if researchers and practitioners rely exclusively on automated methods for finding performance-optimizing configurations, they do not obtain any intuition or information beyond the single configuration chosen. To still provide such intuition in the age of automation, we advocate the development of automated methods that provide high-level insights into an algorithm's hyperparameters, based on a wide range of datasets.

When using a new algorithm on a given dataset, it is typically a priori unknown which hyperparameters should be tuned, what are good ranges for these, and which values in these ranges are most likely to yield high performance. Currently these decisions are typically made based on a combination of intuition about the algorithm and trial & error. While various post-hoc analysis techniques exist that, for a given dataset and algorithm, determine what were the most important hyperparameters and which of their values tended to yield good performance, in this work we study the same question across many datasets. For many well-known algorithms, there already exists some intuition about which hyperparameters impact performance most. For example, for support vector machines, it is commonly believed that the gamma and complexity hyperparameters are most important, and that a certain trade-off exists between these two. However, the empirical evidence for this is limited to a few datasets and therefore rather anecdotal.

In this work, given an algorithm, we aim to answer the following two questions:

(1) Which of the algorithm's hyperparameters matter most for empirical performance?
(2) Which values of these hyperparameters are likely to yield high performance?

We will introduce methods to answer these questions across datasets and demonstrate these methods for three commonly used classifiers: support vector machines (SVMs), random forests and Adaboost. Specifically, we apply the post-hoc analysis technique of functional ANOVA [22] to each of the aforementioned classifiers on a wide range of datasets, drawing on the experimental data available on OpenML [43]. Using the same available experimental data, we also infer prior distributions over which hyperparameter values work well. Several experiments demonstrate that the trends we find (about which hyperparameters tend to be important and which values tend to perform well) generalize to new datasets.

*KDD '18, August 19–23, 2018, London, United Kingdom*

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220058

Our contributions are as follows:

(1) We present a methodology and a framework that leverage functional ANOVA to study hyperparameter importance across datasets.
(2) We apply this to analyze the importance of SVMs, random forests and Adaboost on 100 datasets from OpenML, and confirm that the hyperparameters determined as the most important ones indeed are the most important ones to optimize.
(3) Using the same experimental data, we infer priors over which values of these hyperparameters perform well and confirm that these priors yield statistically significant improvements for a modern hyperparameter optimization method.
(4) In order to make this study reproducible, all experimental data is made available on OpenML. The results of all analyses are available in a separate Jupyter Notebook.
(5) Overall, this work is the first to provide quantitative evidence for which hyperparameters are important and which values should be considered, providing a better scientific basis for the field than previous knowledge based mainly on intuition.

The remainder of this paper is organized as follows. In Section 2 we position our contributions with respect to similar works in the field. Section 3 covers relevant background information about functional ANOVA. Section 4 formally introduces the methods that we propose, and Section 5 defines the algorithms and hyperparameters upon which we apply them. We then conduct two experiments: Section 6 covers the experiments that show which hyperparameters are important across datasets; and Section 7 covers the experiments that show how to use the experimental data on OpenML to infer good priors. Section 8 concludes.

## 2 RELATED WORK

We review related work on hyperparameter importance and priors.

**Hyperparameter Importance.** Various techniques exists that allow for the assessment of hyperparameter importance. Breiman [7] showed in his seminal paper how random forests can be used to assess attribute importance: if removing an attribute from the dataset yields a drop in performance, this is an indication that the attribute was important. *Forward selection* [21] is based on this principle. It predicts the performance of a classifier based on a subset of hyperparameters that is initialized empty and greedily filled with the next most important hyperparameter. *Ablation Analysis* [3, 11] requires a default setting and an optimized setting and calculates a so-called ablation trace, which embodies how much the hyperparameters contributed towards the difference in performance between the two settings. *Functional ANOVA* (as explained in detail in the next section) is a powerful framework that can detect the importance of both individual hyperparameters and interaction effects between arbitrary subsets of hyperparameters. Although all of these methods are very useful in their own right, none of these has yet been applied to analyze hyperparameters across datasets. We will base our work in this realm on functional ANOVA since it is computationally far more efficient than forward selection, can detect interaction effects, and (unlike ablation analysis) does not rely on a specific default configuration. The proposed methods are, however, by no means limited to functional ANOVA.

In a preliminary study, we already reported on important hyperparameters of random forests and Adaboost [42].

**Priors.** The field of meta-learning (e.g., Brazdil et al. [6]) is implicitly based on priors: a model is trained on data characteristics (so-called meta-features) and performance data from similar datasets, and the resulting predictions are used to recommend a configuration for the dataset at hand. These techniques have been successfully used to recommend good hyperparameter settings [30, 35], to warm-start optimization procedures [13] or prune search spaces [44]. However, it is hard to select an adequate set of meta-features. Moreover, obtaining good meta-features comes at the cost of run time. This work can be seen as an alternative approach to meta-learning that does not require the aforementioned meta-features.

Multi-task Bayesian optimization [38] offers a different approach to meta-learning that alleviates meta-features. A multi-task model (typically a Gaussian Process [5]) is fitted on the outcome of classifiers to determine correlations between tasks, which can be exploited for hyperparameter optimization on a new task. However, this approach suffers from the cubic complexity of Gaussian processes. While a recent more scalable alternative for multi-task Bayesian optimization is to use Bayesian neural networks [37], to the best of our knowledge, this approach has not been evaluated at large scale yet.

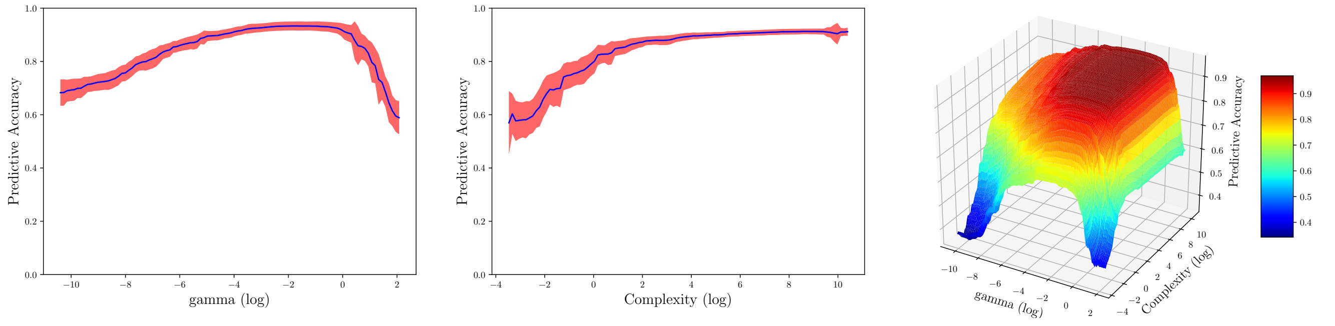
The class of Estimation of Distribution (EDA) algorithms (e.g. Larraanaga and Lozano [27]) optimizes a given function by iteratively fitting a probability distribution to points in the input space with high performance and using this probability distribution as a prior to sample new points from. Drawing on this, the method we propose determines priors over good hyperparameter values by using hyperparameter performance data on different datasets.

## 3 BACKGROUND: FUNCTIONAL ANOVA

The functional ANOVA framework for analyzing the importance of hyperparameters introduced by Hutter et al. [22] is based on a regression model that yields predictions $\hat{y}$ for the performance of arbitrary hyperparameter settings. It determines how much each hyperparameter (and each combination of hyperparameters) contributes to the variance of $\hat{y}$ across the algorithm's hyperparameter space $\mathbf{\Theta}$. Since we will use this technique as part of the proposed method, we now discuss it in more detail.

**Notation.** Algorithm $A$ has $n$ hyperparameters with domains $\Theta_1, \ldots, \Theta_n$ and *configuration space* $\Theta = \Theta_1 \times \ldots \times \Theta_n$. Let $N = \{1, \ldots, n\}$ be the set of all hyperparameters of $A$. An instantiation of $A$ is a vector $\boldsymbol{\theta} = \langle \theta_1, \ldots, \theta_n \rangle$ with $\theta_i \in \Theta_i$ (this is also called a *configuration* of $A$). A partial instantiation of $A$ is a vector $\boldsymbol{\theta}_U = \langle \theta_i, \ldots, \theta_j \rangle$ with a subset $U \subseteq N$ of the hyperparameters fixed, and the values for other hyperparameters unspecified. (Note that from this it follows that $\boldsymbol{\theta}_N = \boldsymbol{\theta}$).

**Efficient marginal predictions.** The *marginal performance* $\hat{a}_U(\boldsymbol{\theta}_U)$ is defined as the average performance of all complete instantiations $\boldsymbol{\theta}$ that agree with $\boldsymbol{\theta}_U$ in the instantiations of hyperparameters $U$. To illustrate the concept of marginal predictions, Figure 1 shows marginal predictions for two hyperparameters of SVMs and their union. We note that such marginals average over *all*



(a) Gamma

(b) Complexity

(c) Gamma vs. Complexity

**Figure 1: Marginal predictions for a SVM with RBF kernel on the letter dataset. The hyperparameter values are on a log scale.**

instantiations of the hyperparameters not in $U$, and as such depend on a very large number of terms (even for finite hyperparameter ranges, this number of terms is exponential in the remaining number of hyperparameters $N \setminus U$). However, for the predictions $\hat{y}$ of a tree-based model, the average over these terms can be computed exactly by a procedure that is linear in the number of leaves in the model [22].

**Functional ANOVA..** Functional ANOVA [18, 19, 25, 36] decomposes a function $\hat{y} : \Theta_1 \times \cdots \times \Theta_n \rightarrow \mathbb{R}$ into additive components that only depend on subsets of the hyperparameters $N$:

$$\hat{y}(\boldsymbol{\theta}) = \sum_{U \subseteq N} \hat{f}_U(\boldsymbol{\theta}_U) \tag{1}$$

The components $\hat{f}_U(\boldsymbol{\theta}_U)$ are defined as follows:

$$\hat{f}_U(\boldsymbol{\theta}_U) = \begin{cases} \hat{f}_\emptyset & \text{if } U = \emptyset. \\ \hat{a}_U(\boldsymbol{\theta}_U) - \sum_{W \subsetneq U} \hat{f}_W(\boldsymbol{\theta}_W) & \text{otherwise,} \end{cases} \tag{2}$$

where the constant $\hat{f}_\emptyset$ is the mean value of the function over its domain. Our main interest is the result of the unary functions $\hat{f}_{\{j\}}(\boldsymbol{\theta}_{\{j\}})$, which capture the effect of varying hyperparameter $j$, averaging across all possible values of all other hyperparameters. Additionally, the functions $\hat{f}_U(\boldsymbol{\theta}_U)$ for $|U| > 1$ capture the interaction effects between all variables in $U$ (excluding effects of subsets $W \subsetneq U$).

Given the individual components, functional ANOVA decomposes the variance $\mathbb{V}$ of $\hat{y}$ into the contributions by all subsets of hyperparameters $\mathbb{V}_U$:

$$\mathbb{V} = \sum_{U \subset N} \mathbb{V}_U, \quad \text{with} \quad \mathbb{V}_U = \frac{1}{||\Theta_U||} \int \hat{f}_U(\boldsymbol{\theta}_U)^2 d\boldsymbol{\theta}_U, \tag{3}$$

where $\frac{1}{||\Theta_U||}$ is the probability density of the uniform distribution across $\Theta_U$.

To apply functional ANOVA, we first collect performance data $\langle \boldsymbol{\theta}_i, y_i \rangle_{k=1}^K$ that captures the performance $y_i$ (e.g., accuracy or AUC score) of an algorithm $A$ with hyperparameter settings $\boldsymbol{\theta}_i$. We then fit a random forest model to this data and use functional ANOVA to decompose the variance of each of the forest's trees $\hat{y}$ into contributions due to each subset of hyperparameters. Importantly, based on the fast prediction of marginal performance available for tree-based models, this is an efficient operation requiring only seconds in the experiments for this paper. Overall, based on the performance data $\langle \boldsymbol{\theta}_i, y_i \rangle_{k=1}^K$, functional ANOVA thus provides us with the relative variance contributions of each individual hyperparameter (with the relative variance contributions of all subsets of hyperparameters summing to one).

This leads to the notion of hyperparameter importance. When a hyperparameter is responsible for a large fraction of the variance, setting this hyperparameter correctly is important for obtaining good performance, and it should be tuned properly. When a hyperparameter is not responsible for a lot of variance, it is deemed less important.

Besides attributing the variance to single hyperparameters, functional ANOVA also determines the interaction effects of sets of hyperparameters. This potentially gives insights in which hyperparameters can be tuned independently and which are dependent on each other and should thus be tuned together. In the hypothetical case where there are no interaction effects between any of the hyperparameters, all hyperparameters could be tuned individually by means of a simple hill-climbing algorithm.

By design, functional ANOVA operates on the result of a single hyperparameter optimization procedure on a single dataset. This leaves room for questions, such as: (i) Which hyperparameters are important in general? (ii) Are the same hyperparameters often important, or does this vary per dataset? (iii) Given a new dataset, on which a hyperparameter procedure is to be ran, which hyperparameters should be optimized and what are sensible ranges? We will investigate these questions in the next section.

## 4 METHODS

We address the following problem. Given

- an algorithm with configuration space $\boldsymbol{\Theta}$
- a large number of datasets $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(M)}$, with $M$ being the number of datasets
- for each of the datasets, a set of empirical performance measurements $\langle \boldsymbol{\theta}_i, y_i \rangle_{i=1}^K$ for different hyperparameter settings $\boldsymbol{\theta}_i \in \boldsymbol{\Theta}$,

we aim to determine which hyperparameters affect the algorithm's empirical performance most, and which values are likely to yield good performance.

## 4.1 Important Hyperparameters

Current knowledge about hyperparameter importance is mainly based on a combination of intuition, own experience and folklore knowledge. To instead provide a data-driven quantitative basis for this knowledge, in this section we introduce methodology for determining which hyperparameters are generally important, measured across datasets.

**Determining Important Hyperparameters.** For a given algorithm $A$ and a given dataset, we use the performance data $\langle \boldsymbol{\theta}_i, y_i \rangle_{i=1}^{K}$ collected for $A$ on this dataset to fit functional ANOVA's random forests to. Functional ANOVA then returns the variance contribution $\mathbb{V}_j/\mathbb{V}$ of every hyperparameter $j \in N$, with high values indicating high importance. We then study the distribution of these variance contributions across datasets to obtain empirical data regarding which hyperparameters tend to be most important.

It is possible that a given set of hyperparameters is responsible for a high variance on many datasets, but the best performance is typically achieved with the same set of values. We note that this method will flag such hyperparameters as important, although it could be argued that they have appropriate defaults and do not need to be tuned. Whether this is the case can be determined by various procedures, for example the one introduced in Section 4.2.

For some datasets, the measured performance values $y_i$ are constant, indicating that none of the hyperparameters are important; we therefore removed these datasets from the respective experiments.

**Verification.** Functional ANOVA uses a mathematically clearly defined quantity ($\mathbb{V}_j/\mathbb{V}$) to define a hyperparameter's importance, but it is important to verify whether this agrees with other, potentially more intuitive, notions of hyperparameter importance. To confirm the results of functional ANOVA, we therefore propose to verify in an expensive, post-hoc analysis to what extent its results align with an intuitive notion of how important a hyperparameter is in hyperparameter optimization.

One intuitive way to measure the importance of a hyperparameter $\theta$ is to assess the performance obtained in an optimization process that leaves $\theta$ fixed. However, similar to ablation analysis [11], the outcome of this approach depends strongly on the value that $\theta$ is fixed to; e.g., fixing a very important hyperparameter to a good default value would result in labelling it not important (this indeed happened in various cases in our preliminary experiments). To avoid this problem and to instead quantify the importance of setting $\theta$ to a good value in its range, we perform $k$ runs of the optimization process of all hyperparameters but $\theta$, each time fixing $\theta$ to a different value spread uniformly over its range; in the end, we average the results of these $k$ runs. Leaving out an important hyperparameter $\theta$ is then expected to yield worse results than leaving out an unimportant hyperparameter $\theta'$. As a hyperparameter optimization procedure for this verification procedure, we simply use random search, to avoid any biases.

Formally, for each hyperparameter $\theta_j$ we measure $y^*_{j,f}$ as the result of a random search for maximizing accuracy, fixing $\theta_j$ to a given value $f \in F_j$. (For categorical $\theta_j$ with domain $\Theta_j$, we used $F_j = \Theta_j$; for numeric $\theta_j$, we set $F_j$ to a set of $k = 10$ values spread uniformly over $\theta_j$'s range.) We then compute $y^*_j = \frac{1}{|F_j|}\sum_{f \in F_j} y^*_{j,f}$, representing the score when not optimizing hyperparameter $\theta_j$, averaged over fixing $\theta_j$ to various values it can take. Hyperparameters with lower $y^*_j$ are then judged to be more important, since performance deteriorates more when they are set sub-optimally.

## 4.2 Priors for Good Hyperparameter Values

Knowing what are the important hyperparameters, an obvious next question is what are good values for these hyperparameters. These values can be used to define defaults, or to sample from in hyperparameter optimization.

**Determining Useful Priors.** We aim to build priors based on the performance data observed across datasets. There are several existing methods for achieving this on a single dataset that we drew inspiration from. In hyperparameter optimization, the Tree-structured Parzen Estimator (TPE) by Bergstra et al. [1] keeps track of an algorithm's best observed hyperparameter configurations $\Theta_{best}$ on a given dataset and for each hyperparameter fits a 1-dimensional Parzen Estimator to the values it took in $\Theta_{best}$. Similarly, as an analysis technique to study which values of a hyperparameter perform well, Loshchilov and Hutter [29] proposed to fit kernel density estimators (see, e.g., [33]) to these values. Here, we follow this latter procedure, but instead of using the hyperparameter configurations that performed well on a given dataset, we used the top $n$ configurations observed for each of the datasets; in our experiments, we set $n = 10$. We only used 1-dimensional density estimators in this work, because the amount of data required to adequately fit these is known to be reasonable. We note that this is merely one possible choice, and that future work could focus on fitting other types of distributions to this data.

**Verification.** As in the case of hyperparameter importance, we propose an expensive post-hoc analysis to verify whether the priors over good hyperparameter values identified above are useful and generalize across datasets. Specifically, as a quantifiable notion of usefulness, we propose to evaluate the impact of using the prior distributions defined above within a hyperparameter optimization procedure. For this, we use the popular bandit-based hyperparameter optimization method Hyperband [28]. Hyperband is based on the procedure of successive halving [23], which evaluates a large number of randomly-chosen configurations using only a small budget, and iteratively increases this budget, at each step only retaining a fraction of configurations that are best so far. For each dataset, we propose to run two versions of this optimization procedure: one sampling uniformly from the hyperparameter space and one sampling from the obtained priors. If the priors are indeed useful and generalize across datasets, the optimizer that uses them should obtain better results on the majority of the datasets. Of course, for each dataset on which this experiment is performed, the priors should be obtained on empirical performance data that was not obtained from this dataset.

We note that – due to differences between datasets – there are bound to be datasets for which using priors from other datasets deteriorates performance. However, since human engineers have successfully used prior knowledge to define typical ranges to consider, our hypothesis is that the data-driven priors will improve the optimization procedure's results on most datasets.

## 4.3 Algorithm Performance Data

The proposed methods do not crucially rely on how exactly the training performance data was obtained. We note, however, that for all training datasets the data should be gathered with a wide range of hyperparameter configurations (to allow the construction of predictive performance models for functional ANOVA) and should contain close-to-optimal configurations (to allow the construction of good priors).

We note that for many common algorithms, the open machine learning environment OpenML [43] already contains very comprehensive performance data for different hyperparameter configurations on a wide range of datasets. OpenML also defines curated benchmarking suites, such as the OpenML100 [4]. We therefore believe that the proposed methods can in principle be used directly on top of OpenML to automatically provide and refine insights as more data becomes available.

In our experiments, which involve classifiers with up to six hyperparameters, we indeed used data from OpenML. We ensured that for each dataset at least 150 runs with different hyperparameters were available to make functional ANOVA's model reliable enough. We generated additional runs for classifiers that did not meet this requirement by executing random configurations on a large compute cluster. We note that for larger hyperparameter spaces, more sophisticated data gathering strategies are likely required to accurately model the performance of the best configurations.

## 4.4 Computational Complexity of Analysis Techniques

While we also propose the use of expensive, post-hoc verification methods to confirm the results of our analysis, we would like to emphasize that the proposed analysis techniques themselves are computationally very efficient. Their complexity is dominated by the cost of fitting functional ANOVA's random forest to the performance data observed for each of the datasets. The cost of the remainder of functional ANOVA, and of fitting the Gaussian kernel density estimator is negligible. In the experiments we conducted, given an algorithm's performance data, performing the analyses required only a few seconds.

## 5 ALGORITHMS AND HYPERPARAMETERS

We analyze the hyperparameters of three classifiers implemented in scikit-learn [8, 31]: random forests [7], Adaboost (using decision trees as base-classifier) [14] and SVMs [9]. The SVMs are analysed with two different kernel types: radial basis function (RBF) and sigmoid.

For each of these, to not incur any bias from our choice of hyperparameters and ranges, we used exactly the same hyperparameters and ranges as the automatic machine learning system Auto-sklearn [12].[1] The hyperparameters, ranges and scales are listed in Tables 1–3.

**Preprocessing.** We used the same data preprocessing steps for all algorithms. Missing values are imputed (categorical features with the mode; for numerical features, the imputation strategy was one of the hyperparameters), categorical hyperparameters are one-hot-encoded, and constant features are removed. As support vector machine's are sensitive to the scale of the input variables, the input variables for the SVM's are scaled to have unit variance. Of course, all these operations are performed based on information obtained from the training data.

**Datasets.** We performed all experiments on the datasets from the OpenML100 [4]. The OpenML100 is a curated benchmark suite, containing 100 datasets from various domains. The datasets contain between 500 and 100,000 data points, are generally well-balanced and are all linked to a scientific publication. These criteria ensure that the datasets pose a challenging and meaningful classification task, and the results are comparable to earlier studies.

## 6 HYPERPARAMETER IMPORTANCE

We now discuss the results of the experiment for determining the most important hyperparameters per classifier. All together, this analysis is based on the performance data of 250,195 algorithm runs over the 100 datasets using 3,184 CPU days to generate. All performance data we used is publicly available on OpenML[2].

We show the results for each classifier as a set of three figures. The top figure (e.g., Figure 2(a)) shows violinplots of each hyperparameter's variance contribution, across all datasets. The $x$-axis shows the hyperparameter $j$ under investigation, and each data point represents $\mathbb{V}_j/\mathbb{V}$ for one dataset; a high value implies that this hyperparameter accounted for a large fraction of variance on this dataset, and therefore would account for high accuracy-loss if not set properly. We also show for each classifier the three most important interaction effects between groups of hyperparameters.

The middle figure (e.g., Figure 2(b)) shows the results of the verification experiment. It shows the average rank of each run of random search, labeled with the hyperparameter whose value was fixed to a default value. A high rank implies poor performance compared to the other configurations, meaning that tuning this hyperparameter would have been important.

The bottom figure (e.g., Figure 2(c)) shows the result of a Nemenyi test over the average ranks of the hyperparameters (for details, see [10]). A statistically significant difference was measured for every pair of classifiers that are not connected by the horizontal black line. The interaction effects are left out to meet the independent input assumptions of the Nemenyi test.

**SVM Results.** We analyze SVMs with RBF and sigmoid kernels in Figures 2 and 3, respectively.

[1] There was one exception: For technical reasons, in random forests, we modelled the maximal number of features for a split as a fraction of the number of available features (with range [0.1, 0.9]).

[2] Full details: https://www.openml.org/s/71

**Table 1: SVM Hyperparameters.**

| hyperparameter | values | description |
|---|---|---|
| complexity (or: ‘C’) | $[2^{-5}, 2^{15}]$ (log-scale) | Soft-margin constant, controlling the trade-off between model simplicity and model fit. |
| coef0 | $[-1, 1]$ | Additional coefficient used by the kernel (sigmoid kernel only). |
| gamma | $[2^{-15}, 2^{3}]$ (log-scale) | Length-scale of the kernel function, determining its locality. |
| imputation | {mean, median, mode} | Strategy for imputing missing numeric variables. |
| shrinking | {true, false} | Determines whether to use the shrinking heuristic (introduced in [24]). |
| tolerance | $[10^{-5}, 10^{-1}]$ (log-scale) | Determines the tolerance for the stopping criterion. |

**Table 2: Random Forest Hyperparameters.**

| hyperparameter | values | description |
|---|---|---|
| bootstrap | {true, false} | Whether to train on bootstrap samples or on the full train set. |
| max. features | $[0.1, 0.9]$ | Fraction of random features sampled per node. |
| min. samples leaf | $[1, 20]$ | The minimal number of data points required in order to create a leaf. |
| min. samples split | $[2, 20]$ | The minimal number of data points required to split an internal node. |
| imputation | {mean, median, mode} | Strategy for imputing missing numeric variables. |
| split criterion | {entropy, gini} | Function to determine the quality of a possible split. |

**Table 3: Adaboost Hyperparameters.**

| hyperparameter | values | description |
|---|---|---|
| algorithm | {SAMME, SAMME.R} | Determines which boosting algorithm to use. |
| imputation | {mean, median, mode} | Strategy for imputing missing numeric variables. |
| iterations | $[50, 500]$ | Number of estimators to build. |
| learning rate | $[0.01, 2.0]$ (log-scale) | Learning rate shrinks the contribution of each classifier. |
| max. depth | $[1, 10]$ | The maximal depth of the decision trees. |

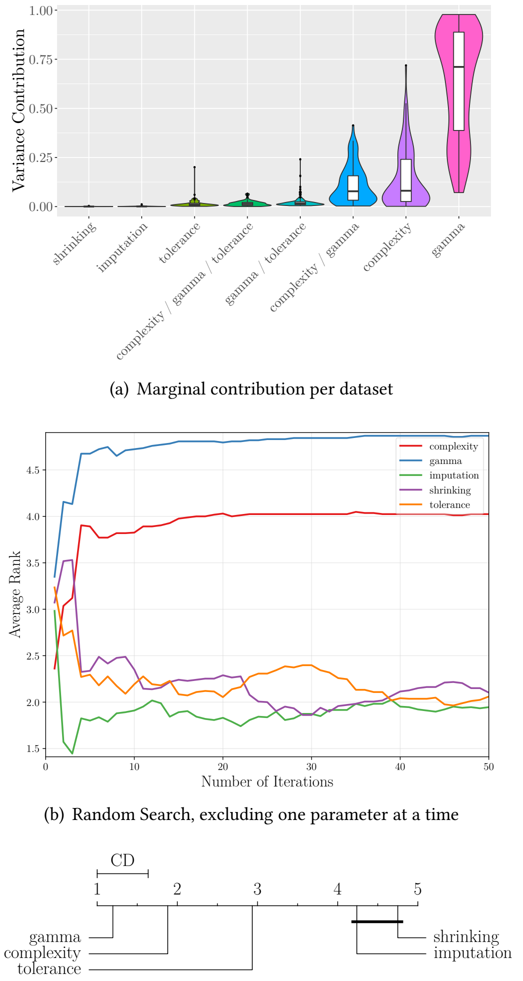
The results show a clear picture: The most important hyperparameter to tune in both cases was gamma, followed by complexity. Both of these hyperparameters were significantly more important than the others according to the Nemenyi test. This conclusion is supported by the random search experiment: not optimizing the gamma parameter obtained the worst performance, making it the most important hyperparameter, followed by the complexity hyperparameter. Interestingly, according to Figure 3(a), when using the sigmoid kernel, the interaction effect between gamma and complexity was even more important than the complexity parameter by itself.

We note that while it is well-known that gamma and complexity are important SVM hyperparameters, to the best of our knowledge, this is the first study that provides systematic empirical evidence for their importance on a wide range of datasets. The fact that the proposed methods recovered these known most important hyperparameters also acts as additional verification that the proposed methodology works as expected. The least important hyperparameter for the accuracy of SVMs was whether to use the shrinking heuristic. As this heuristic is intended to decrease computational resources rather than improve predictive performance, our data suggests that it is safe to enable this feature.

**Random Forest Results.** Figure 4 shows the results for random forests. The results reveal that most of the variance could be attributed to a small set of hyperparameters: the minimum samples per leaf and maximal number of features for determining the split were most important. Both of these hyperparameters were significantly more important than the others according to the Nemenyi test. Only in a few cases, bootstrap was the most important hyperparameter (datasets ‘balance-scale’, ‘credit-a’, ‘kc1’, ‘Australian’, ‘profb’ and ‘climate-model-simulation-crashes’) and the split criterion only once (dataset ‘scene’). Again, the results from functional ANOVA agree with the results from the random search experiment and our intuition. It is well-known that ensembles perform well when two conditions are met [7, 17]: (i) the individual models perform better than random guessing, and (ii) the errors of the individual models are uncorrelated. Both hyperparameters influence the variance among trees, uncorrelating their predictions.

At first sight, the minimal samples per split and minimal samples per leaf hyperparameters seem quite similar, but at closer inspection they are not: logically, minimal samples per split is overshadowed by minimal samples per leaf.

**Adaboost Results.** Figure 5 shows the results for Adaboost. Again, most of the variance can be explained by a small set of hyperparameters, in this case the maximal depth of the decision tree and, to a lesser degree, the learning rate. Both of these hyperparameters were significantly more important than the others according to the Nemenyi test. There were only a few exceptions, in which the boosting algorithm was the most important hyperparameter (datasets ‘madelon’, ‘diabetes’ and ‘hill-valey’). The results were again confirmed by the verification experiment.

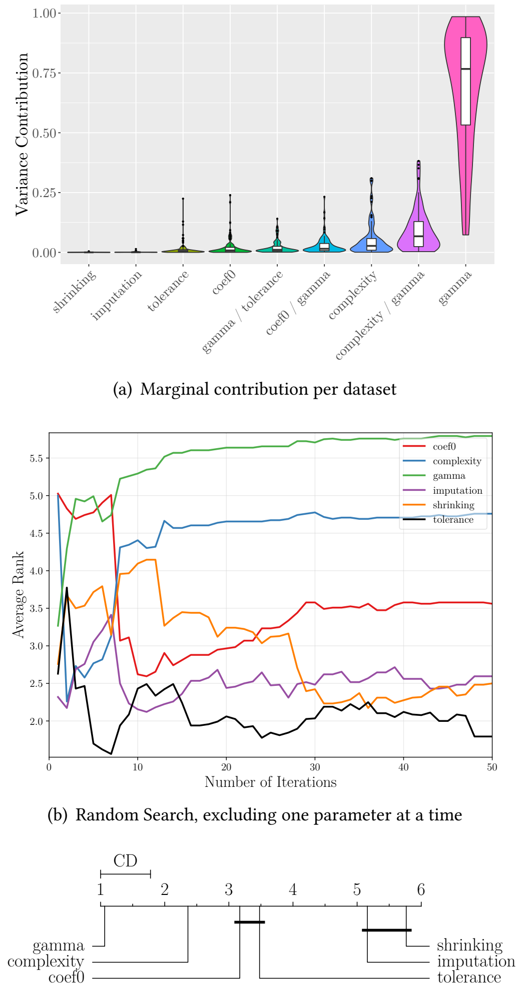


(a) Marginal contribution per dataset

(b) Random Search, excluding one parameter at a time

(c) Ranked hyperparameter importance, $\alpha = 0.05$.

**Figure 2: SVM (RBF kernel).**



(a) Marginal contribution per dataset

(b) Random Search, excluding one parameter at a time

(c) Ranked hyperparameter importance, $\alpha = 0.05$.

**Figure 3: SVM (sigmoid kernel).**

One interesting observation is that, in contrast to other ensemble techniques, the number of iterations did not seem to influence performance too much. The minimum value (50) appears to already be large enough to ensure good performance, and increasing it does not lead to significantly better results.

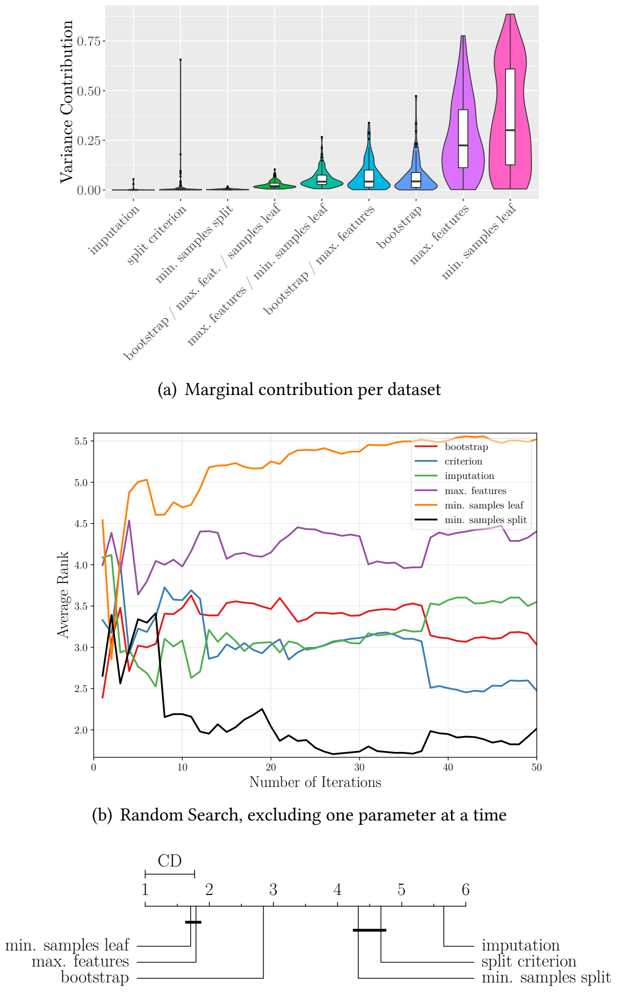
**General Conclusions.** For all classifiers, it appears that a small set of hyperparameters are responsible for most variation in performance. In many cases, this is the same set of hyperparameters across datasets. Knowing which hyperparameters are important is relevant in a variety of contexts, ranging from experimental setups to automated hyperparameter optimization procedures. Furthermore, knowing which hyperparameters are important is interesting as a scientific endeavor in itself, and can provide guidance for algorithm developers.

Interestingly, the hyperparameter determining the imputation strategy did not seem to matter for any of the classifiers, even though the selected benchmarking suite contains datasets such as 'KDDCup09 upselling', 'sick' and 'profb', all of which have many missing values. Imputation is clearly important (as classifiers do not function on undefined data), but which strategy to use for the imputation does not matter much according to the data.

We note that the results presented in this section do by no means imply that it suffices to tune just the set of most important hyperparameters. While the results by Hutter et al. [22] showed that this can indeed lead to faster improvements, they also indicated that it is still advisable to tune all hyperparameters when enough budget is available. In the next experiment, as a complementary analysis, we will study which values are likely to yield good performance.

## 7 GOOD HYPERPARAMETER VALUES

Now that we know which hyperparameters are important, the next natural question is which values they should be set to in order to likely obtain good performance. We now discuss the results of the experiment for answering this question.

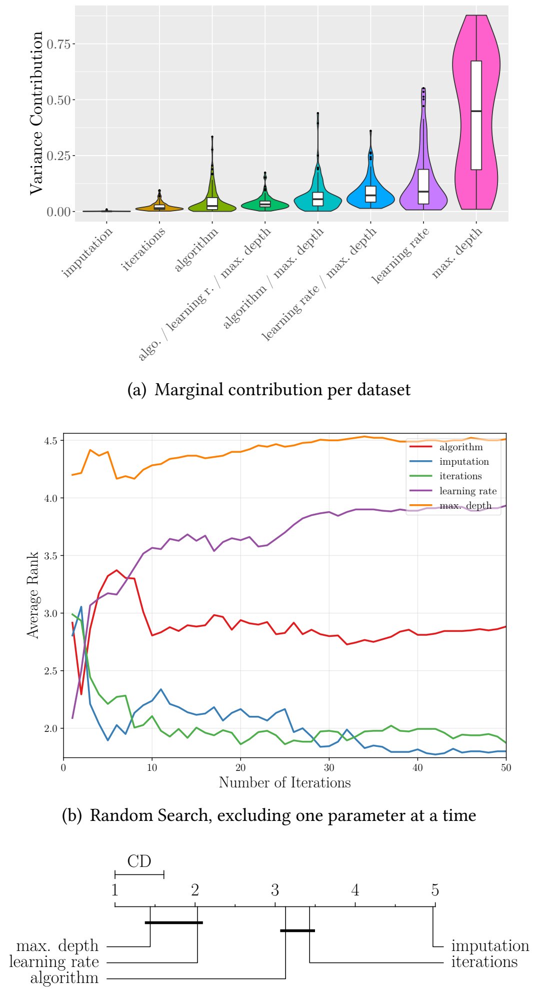


(a) Marginal contribution per dataset

(b) Random Search, excluding one parameter at a time

(c) Ranked hyperparameter importance, $\alpha = 0.05$.

**Figure 4: Random Forest.**



(a) Marginal contribution per dataset

(b) Random Search, excluding one parameter at a time

(c) Ranked hyperparameter importance, $\alpha = 0.05$.

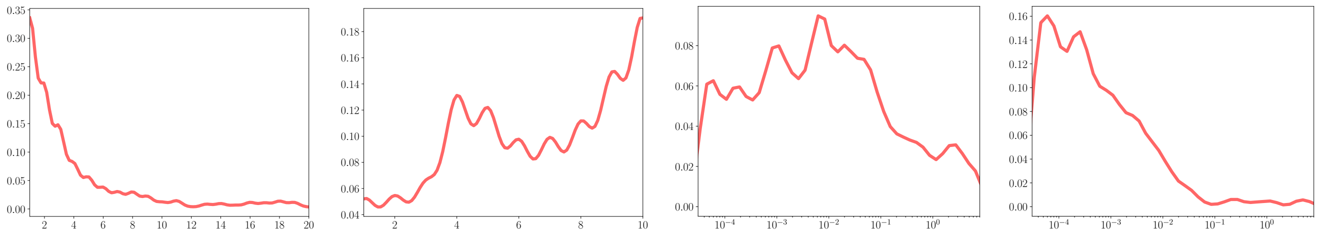
**Figure 5: Adaboost.**

Figure 6 shows the kernel density estimators for the most important hyperparameters per classifier. It becomes clear that for random forests the minimal number of data points per leaf has a good default and should typically be set to quite small values. This is in line with the results reported by Geurts et al. [15] (albeit for the variant of 'Extremely Randomized Trees'). Likewise, the maximum depth of the decision tree in Adaboost should typically be set to a large value. Both hyperparameters are commonly used for regularization, but the empirical data indicates that this should only be applied in moderation. For both types of SVMs, the best performance can typically be achieved with low values of the gamma hyperparameter.

Next, we report the results of the experiment for verifying the usefulness of these priors in hyperparameter optimization. We do this in a leave-one-out setting: for each dataset under investigation, we build the priors based on the empirical performance data from the 99 other datasets. Figure 7 and Table 4 report results comparing Hyperband with a uniform prior vs. the data-driven prior. Hyperband was ran with the following hyperparameters: 5 brackets, $s_{max} = 4$, $\eta = 2$ and $R = |\mathcal{D}^{(i)}|$ (the number of data points of dataset $\mathcal{D}^{(i)}$). Each optimizer was ran with 10 different random seeds, and we report the average of their results.

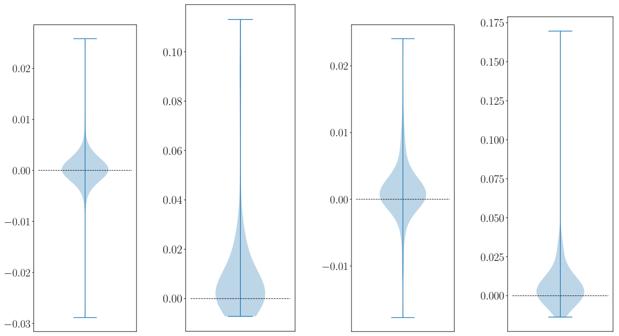
For each dataset, Figure 7 shows the difference in predictive accuracy between the two procedures: values greater than 0 indicate that sampling according to the data-driven priors was better by this amount, and vice versa. These per-dataset differences are aggregated using a violinplot. The results indicate that on many datasets the data-driven priors were indeed better, especially for random forests.

When evaluating experiments across a wide range of datasets, performance scales become a confounding factor. For example, for several datasets a performance improvement of 0.01 already makes a great difference, whereas for others an improvement of 0.05 is



(a) RF: min. samples per leaf (b) Adaboost: max. depth of tree (c) SVM (RBF kernel): gamma (d) SVM (sigmoid): gamma

**Figure 6: Obtained priors for the hyperparameter found to be most important for each classifier. The $x$-axis represents the value, the $y$-axis represents the probability that this value will be sampled (integer parameters will be rounded).**



(a) SVM (RBF) (b) Sigmoid (c) Adaboost (d) RF

**Figure 7: Difference in performance between two instances of Hyperband, one sampling based on the obtained priors and one using uniform sampling. Values bigger than zero indicate superior performance for the procedure sampling based on the priors, and vice-versa.**

**Table 4: Results of Nemenyi test ($\alpha = 0.05$, $CD \approx 0.20$). We report ranks across $M$ datasets (max. 100), boldface the better approach (lower rank) and show whether the improvement is significant.**

| Classifier | $M$ | Uniform | Priors | Sig. |
|---|---|---|---|---|
| random forest | 100 | 1.72 | **1.28** | yes |
| Adaboost | 92 | 1.71 | **1.29** | yes |
| SVM (sigmoid) | 86 | 1.73 | **1.27** | yes |
| SVM (RBF) | 89 | 1.60 | **1.40** | yes |

considered quite small. In order to alleviate this problem we conduct a statistical test, in this case the Nemenyi test, as recommended by Demšar [10]. For each dataset, the Hyperband procedures are ranked by their final performance on the test set (the best procedure obtaining the lower rank, and an equal rank in case of a draw). If the ranks averaged over all datasets differ by more than a critical distance $CD$, the procedure with the lower rank performs statistically significant better.

The results of this test are presented in Table 4. We observe that the data-driven priors significantly improved performance over using uniform priors for all classifiers.[3] The fact that the priors we obtained with a straightforward density estimator already yielded statistically significant improvements shows great promise. We see these simple estimators only as a first step and believe that better methods (e.g., based on traditional meta-learning and/or more sophisticated density estimators) are likely to yield even better results.

## 8 CONCLUSIONS AND FUTURE WORK

In this work we addressed the questions which of a classifier's hyperparameters are most important, and what tend to be good values for these hyperparameters. In order to identify important hyperparameters, we applied functional ANOVA to a collection of 100 datasets. The results indicate that the same hyperparameters are typically important for many datasets. For SVMs, the gamma and complexity hyperparameters are most important, for Adaboost the maximum depth and learning rate, and for random forests the minimum number of samples per leaf and maximum features available for a split. To the best of our knowledge, this is the first methodological attempt to demonstrate these findings across many datasets. In order to verify these findings, we conducted a large-scale optimization experiment, for each classifier optimizing all but one hyperparameter. The results of this experiment are in line with the functional ANOVA results and largely agree with popular belief (for example, confirming the common belief that the gamma and complexity hyperparameters are the most important hyperparameters for SVMs). One surprising outcome of this analysis is that the strategy of data imputation hardly influences performance; investigating this matter further could warrant a whole study on its own, ideally leading to additional data imputation techniques.

In order to determine which hyperparameter values tend to yield good performance, we fitted kernel density estimators to hyperparameter values that performed well on other datasets. This simple method already shows great promise based on the power of using data from many datasets: sampling from data-driven priors in hyperparameter optimization performed significantly better than sampling from a uniform prior. We strove to keep all aspects of this

[3] For the case of SVMs with RBF kernel, we note that the difference does not visually appear significant in Figure 7, but using priors was better in 60% of the datasets.

work reproducible by anyone; we uploaded all the algorithm performance data to OpenML, including a Notebook for reproducing all results and figures in this paper.

In future work we plan to apply this analysis techniques to a wider range of classifiers. While in this work we focused on more established types of classifiers to develop the methodology, quantifying important hyperparameters and good hyperparameter ranges of modern techniques, such as deep neural networks and extreme gradient boosting classifiers, could provide a useful empirical foundation to the field. Furthermore, the developed methodology is by no means restricted to the classification setting; in future work, we plan to also apply it to regression and clustering algorithms. Finally, we aim to employ recent advances in meta-learning to identify similar datasets and base the priors only on these in order to yield dataset-specific priors for hyperparameter optimization.

**Acknowledgements.** We would like to thank Ilya Loshchilov for the valuable discussion that led to the methods we used for (i) the verification of hyperparameter importance and (ii) determining priors over good hyperparameter values. This work has been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant no. 716721, the German state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no. INST 39/963-1 FUGG.

## REFERENCES

[1] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2546–2554.

[2] J. Bergstra and Y. Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, Feb (2012), 281–305.

[3] A. Biedenkapp, M. Lindauer, K. Eggensperger, C. Fawcett, H. H. Hoos, and F. Hutter. 2017. Efficient Parameter Importance Analysis via Ablation with Surrogates. In *Proc. of AAAI 2017*. AAAI Press, 773–779.

[4] B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, and J. Vanschoren. 2017. OpenML Benchmarking Suites and the OpenML100. *ArXiv [stat.ML]* 1708.03731v1 (2017), 6 pages.

[5] E. V. Bonilla, K. M. Chai, and C. Williams. 2008. Multi-task Gaussian Process Prediction. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (Eds.). Curran Associates, Inc., 153–160.

[6] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. 2008. *Metalearning: Applications to Data Mining* (1 ed.). Springer Publishing Company, Incorporated.

[7] L. Breiman. 2001. Random Forests. *Machine learning* 45, 1 (2001), 5–32.

[8] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[9] C. Chang and C. Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2 (2011), 27:1–27:27. Issue 3.

[10] J. Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7 (2006), 1–30.

[11] C. Fawcett and H. H. Hoos. 2016. Analysing differences between algorithm configurations through ablation. *Journal of Heuristics* 22, 4 (2016), 431–458.

[12] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2962–2970.

[13] M. Feurer, J. T. Springenberg, and F. Hutter. 2015. Initializing Bayesian Hyperparameter Optimization via Meta-Learning. In *Proc. of AAAI 2015*. AAAI Press, 1128–1135.

[14] Y. Freund and R. E. Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*. Springer, 23–37.

[15] P. Geurts, D. Ernst, and L. Wehenkel. 2006. Extremely randomized trees. *Machine learning* 63, 1 (2006), 3–42.

[16] T. A. F. Gomes, R. B. C. Prudêncio, C. Soares, A. L. D. Rossi, and A. Carvalho. 2012. Combining meta-learning and search techniques to select parameters for support vector machines. *Neurocomputing* 75, 1 (2012), 3–13.

[17] L.K. Hansen and P. Salamon. 1990. Neural Network Ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 12, 10 (1990), 993–1001.

[18] G. Hooker. 2007. Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics* 16, 3 (2007), 709–732.

[19] J. Z. Huang. 1998. Projection estimation in multiple regression with application to functional ANOVA models. *The annals of statistics* 26, 1 (1998), 242–272.

[20] F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*. Springer, 507–523.

[21] F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2013. Identifying key algorithm parameters and instance features using forward selection. In *International Conference on Learning and Intelligent Optimization*. Springer, 364–381.

[22] F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2014. An efficient approach for assessing hyperparameter importance. In *Proc. of ICML 2014*. 754–762.

[23] K. Jamieson and A. Talwalkar. 2016. Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *Proc. of AISTATS 2016*, Vol. 51. PMLR, 240–248.

[24] T. Joachims. 1999. Making Large-scale Support Vector Machine Learning Practical. In *Advances in Kernel Methods*, Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola (Eds.). MIT Press, Cambridge, MA, USA, 169–184.

[25] D. R. Jones, M. Schonlau, and W. J. Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13, 4 (1998), 455–492.

[26] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. 2017. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proc. of AISTATS 2017*, Vol. 54. PMLR, 528–536.

[27] P. Larraanaga and J. A. Lozano. 2001. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Norwell, MA, USA.

[28] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. 2017. Hyperband: Bandit-Based Configuration Evaluation for Hyperparameter Optimization. In *Proc. of ICLR 2017*. 15 pages.

[29] I. Loshchilov and F. Hutter. 2016. CMA-ES for Hyperparameter Optimization of Deep Neural Networks. In *Proc. of ICLR 2016 Workshop*. 8 pages.

[30] P. B. C. Miranda, R. B. C. Prudêncio, A. P. L. F. De Carvalho, and C. Soares. 2014. A hybrid meta-learning architecture for multi-objective optimization of SVM parameters. *Neurocomputing* 143 (2014), 27–43.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[32] M. Reif, F. Shafait, and A. Dengel. 2012. Meta-learning for evolutionary parameter optimization of classifiers. *Machine learning* 87, 3 (2012), 357–380.

[33] D. W. Scott. 2015. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.

[34] J. Snoek, H. Larochelle, and R. P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems 25*. ACM, 2951–2959.

[35] C. Soares, P. Brazdil, and P. Kuba. 2004. A meta-learning method to select the kernel width in support vector regression. *Machine learning* 54, 3 (2004), 195–209.

[36] I. M. Sobol. 1993. Sensitivity estimates for nonlinear mathematical models. *Mathematical Modelling and Computational Experiments* 1, 4 (1993), 407–414.

[37] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. 2016. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 4134–4142.

[38] K. Swersky, J. Snoek, and R. Adams. 2013. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems 26*. 2004–2012.

[39] C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown. 2013. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In *Proc. of ACM SIGKDD conference on Knowledge Discovery and Data Mining (KDD)*. 847–855.

[40] J. N. van Rijn. 2016. *Massively Collaborative Machine Learning*. Ph.D. Dissertation. Leiden University.

[41] J. N. van Rijn, S. M. Abdulrahman, P. Brazdil, and J. Vanschoren. 2015. Fast Algorithm Selection using Learning Curves. In *Advances in Intelligent Data Analysis XIV*. Springer, 298–309.

[42] J. N. van Rijn and F. Hutter. 2017. An Empirical Study of Hyperparameter Importance Across Datasets. In *Proc. of AutoML 2017 @ ECML-PKDD*. CEUR-WS, 97–104.

[43] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. 2014. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter* 15, 2 (2014), 49–60.

[44] M. Wistuba, N. Schilling, and L. Schmidt-Thieme. 2015. Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization. In *Proc. of ECML/PKDD 2015*. Springer, 104–119.