Data Management Plan

1 Roles and responsibilities

Principal Investigator: Dr. Andreas Müller
PI Müller will oversee the data management, and ensure digital preservation of all produced data. Dr. Müller will also ensure that all resulting materials are deposited into a repository that performs digital preservation. He will also ensure that all developed code is continuously version-controlled to preserve development history.

Research Engineer:
The Research Engineer will use the GitHub version control platform to continuously and incrementally keep track of software development. While some part of the created software will immediately be integrated in to the `scikit-learn` project, other parts of the software will exist in a separate GitHub repository. The Research Engineer will also document all software and experimental protocols within the GitHub projects using the `sphinx` documentation generator, the standard tool in the Python development community. The Research Engineer will implement automatic compilation of the documentation into a website that will be publicly available for all materials outside the `scikit-learn` project. Materials contributed to the `scikit-learn` project are automatically published on the `scikit-learn` website via the `scikit-learn` infrastructure.

The research engineer will retrieve public machine learning data sets via the OpenML Python API, perform the experiments, and upload the experimental results to the OpenML platform. Large scale machine learning experiments will be performed using the NYU High Performance Computing clusters. The research engineer will also upload the collective results of all experiments periodically to the figshare platform.

2 Types of data

The main output of this project will be Python code, which will be recorded and shared via GitHub. Documentation of experimental procedures and documentation of the source code will be contained in the same GitHub repository as the code.

The project will make use of the public machine learning datasets hosted on the OpenML platform. We will use several hundred of the datasets hosted there, which will be downloaded via the Python API of OpenML. The datasets are stored as ARFF files, with the metadata stored as JSON. The Python API represents these together as Python objects.

This data will be processed using a variety of algorithms from the `scikit-learn` machine learning library, as well as methods that we will implement as part of the project. The result of the processing will be predictions made by machine learning algorithms, their accuracy and other evaluation metrics. The metadata of these results consists of the original dataset, machine learning task, machine learning model that was used, and the parameters of the model.

These results and the metadata will be captured and uploaded by the OpenML Python API to the OpenML platform, which will store it as JSON. We will also separately store the collected results across many datasets and algorithms on figshare as JSON file. This will provide a simple access mechanism as alternative to the OpenML interface for obtaining the results. OpenML has programming interfaces in Java, R, Python, C# and other languages that will allow processing of the results.

3 Policies for access and sharing

Source code for all software will be made available as an ongoing process during development. All code and accompanying documentation will be licensed under a BSD license.

Results will be pushed to OpenML immediately as part of the evaluation, whenever possible. The results will be archived and published to figshare after initial analysis confirmed them to be correct. All data will be made available under the CC-0 license.

4 Data storage and preservation of access

The code will be part of an open source project and thereby handed to the open source community. The initial archive will be GitHub, but the community is expected to curate and archive the project beyond the lifetime of GitHub. The results will be stored in the OpenML platform and on figshare, both of which have long-term preservation guarantees.

Project Summary

The open-source machine learning library `scikit-learn` has become a cornerstone of applied machine learning and data science in academic and industrial research. The PI Müller has been involved in the `scikit-learn` project as co-maintainer and core contributor for over 5 years. This proposal, if funded, will improve ease of use of the `scikit-learn` project, reduce the barrier to entry for no-experts to use the package, and add automation features that will allow more effective use by researchers trained in machine learning.

While the `scikit-learn` project has received wide recognition for its ease of use and extensive documentation, many areas for improvement remain. `scikit-learn` was developed for researchers in the machine learning domain, but was later adopted by researchers across many disciplines. Two of the main barriers to entry for domain scientists wanting to apply machine learning are finding a suitable representation of the data to apply machine learning, and choosing an algorithm and parameters for a particular dataset and task.

A large amount of research has recently been performed into automatic model selection, and systematic evaluation of machine learning system. However, advances in these areas have not made the transition from computer science research to being applied by domain scientists to solve practical problems. By providing guidance on model selection and data preprocessing via systematic evaluation, and integrating robust implementations of automatic algorithm selection in the established `scikit-learn` ecosystem, this proposal will allow a more effective use of machine learning by researchers from other domains.

Intellectual Merit

This proposal advances knowledge in two ways:

1. By further lowering the barrier of entry for applying machine learning, and improving the existing tools provided by `scikit-learn`, the proposed project will enable more researchers to adopt machine-learning solutions to data-driven problems.
2. By conducting a large-scale experimental survey of existing methods, the project will provide guidance for machine learning practitioners and pointers to future directions for machine learning researchers.

Broader Impacts

Machine learning has become a core part of many data driven research projects, and is often implemented via open source software. In particular the Python ecosystem of data science tools has been widely adopted in the scientific community, both for research and in teaching. With `scikit-learn` being the primary resource for machine learning inside the Python ecosystem, improvements to `scikit-learn` will benefit all researcher and teachers using this set of tools. With the enhancements described in this proposal, even more people will be able to easily apply machine learning to their problems, without requiring extensive training in machine learning.

As a major scientific open source software package with a wide contributor and user base, `scikit-learn` is in a great position to change the composition and values of the scientific open source ecosystem. This proposal includes plans to host "coding sprints" to enlarge the number of contributors even further, with a focus on attracting women to contributing to open source. These events will be held in collaboration with local and national organizations to promote women in programming and science.

Project Description

1 Introduction

As sciences move towards more data driven research, data analysis has become a main building block in a wide array of research disciplines. Many advances in recent research have been driven by algorithmic improvements in data analysis, in particular in predictive analytics and machine learning. As machine learning becomes a tool for scientists across disciplines, it is important that the neccessary analytical tools are freely available, and easy to use for domain scientists outside of the field of machine learning.

The `scikit-learn` machine learning library [44], provides machine learning functionality within the established—and growing—scientific Python ecosystem [29, 1]. The `scikit-learn` project is an open source project written in Python, implementing state-of-the-art machine learning algorithm and utilities to apply these algorithms to real-world data analysis and prediction problems. The distinguishing features of `scikit-learn` are its generic and intuitive interface, its comprehensiveness and its documentation [59, 29].

Interface `scikit-learn` provides a generic interface for machine learning, mainly consisting of only three methods: `fit`, to build models, `predict`, to make predictions using models, and `transform`, to change the representation of the input data [10]. Table 1 illustrates this interface. The simple and consistent interface helps to abstract away the algorithm, and let users focus on their particular problem. The `scikit-learn` project makes use of the well-established NumPy library to represent data and predictions. NumPy is a standard library for representing numeric data in Python, and integrating this generic representation into `scikit-learn` minimizes the friction of applying machine learning within an existing project.

Comprehensiveness `scikit-learn` implements a wide variety of models for classification, regression, clustering and dimensionality reduction, as well as methods for feature selection and feature extraction. The library contains most of the algorithms included in standard textbooks like Bishop [8] and Friedman, Hastie, and Tibshirani [19], while providing competitive implementation of state-of-the-art algorithms like Gradient Boosting [18], Random Forests [9] and SAG [48]. In addition to a large selection of algorithms, `scikit-learn` also contains a suite of evaluation metrics and tools for parameter selection.

Documentation Documentation is a key ingredient to usability, and the documentation of `scikit-learn` has been widely recognized as an excellent learning resource [60, 29, 31, 13]. The `scikit-learn` project has strict rules on documentation and requires examples and extensive descriptions for all algorithms.

The simple API together with the wide variety of algorithms that are implemented in `scikit-learn`, the package allows for very fast prototyping of machine learning applications. These key features are the ingredients leading to the wide-spread use of `scikit-learn`, and the creation of a large ecosystem of users, contributors, maintainers and dependent packages.

2 Previous Impact of the `scikit-learn` Package

The `scikit-learn` project has been widely used in academic and industrial research, and has made its way into multiple commercial products. The paper [44] describing `scikit-learn` has been cited

Table 1: Main `scikit-learn` interface. All models have a `fit` method with the training data as parameter. Supervised models are also provided with the training targets or labels. New (usually multivariate) representations of the input data are produced by the `transform` method, while prediction of target variables (usually discrete labels or single continuous variables) or cluster memberships are produced with the `predict` method.

`fit(features_train, [labels_train])`

`predict(features_test)`	`transform(features_test)`
Classification	Preprocessing
Regression	Dimensionality Reduction
Clustering	Feature Selection
	Feature extraction

over 5.000 times according to Google scholar[1]. Applications of `scikit-learn` spread a multitude of research areas, including Physics [3, 61], Astronomy [45, 4], Biology [33, 47], Medicine [25, 39], Psychology [43, 15], Cyber Security [49], Oceanograpy [55], Social Sciences [16, 14] and more.

Other ways to measure the use and impact of `scikit-learn` is the engagement with the project via code contributions, downloads, support requests and other community interactions. According to the email addresses used for code contributions (a conservative estimate), 40 researchers and students from at least 23 US universities *contributed code* to `scikit-learn`, indicating wide-spread academic use. These institutes include University of California Berkeley, Brown University, Carnegie Mellon University, Columbia University, Duke University, Harvard University, Massachusetts Institute of Technology, New York University, Stanford University, and University of Washington. The total number of unique contributors to the project is about 700. The `scikit-learn` mailing list has 1.666 subscribers, including 70 email addresses from "edu" domains, across 43 institutes. On the question-answering site Stack Overflow, there are around 7.000 questions tagged as related to `scikit-learn`.

For the last 5 years, the scientific Python conference SciPy has had a `scikit-learn` tutorial, showing the great demand for education in using machine learning and `scikit-learn` in particular.

The `scikit-learn` project has become the center of the machine learning ecosystem in the scientific Python community, with several domain-specific packages relying on and extending its functionality. Prominent examples of scientific software packages depending on `scikit-learn` for machine learning include `astroML` [57] for astronomical data, `nilearn` [2] for neuroimaging, `MNE-Python` for MEG and EEG data, `librosa` [32] for audio and musical data, `nltk` [7] and `rosetta` for Natural language processing, `bcbio` for RNA sequence analysis, `scikit-allel` for genetic variation data, and `rootpy` for integration with the `ROOT` scientific software framework. The main use of `scikit-learn` is outside of these major open source packages, though, as standalone library for data analysis. Using the GitHub repository, we found about 120.000 Jupyter notebooks—a format for interactive computing with Python—that use `scikit-learn` at the time of writing[2].

[1]Nearly half of these citations have been within the last year, demonstrating the accelerating growth of the ecosystem.

[2]this number tripplied within the last year

The scikit-learn website containing the documentation was visited by over 930.000 users in February 2017, nearly doubling within the last year.

The scikit-learn project has become so popular in teaching and data science applications that several books have been written about the use of scikit-learn including “Introduction to machine learning with Python” by PI Müller. [20, 22, 23, 46]

3 Proposed Work

Despite the extensive documentation, applying machine learning using scikit-learn still requires expert knowledge about:

- What kind of models to use for a given task.
- What kind of feature extraction and preprocessing is required for a model.
- What parameters to tune, and in which ranges.

While it is easy for a scientist to create a working model, this model might not be optimum, and they could obtain a much better model using more expert knowledge in machine learning to answer the questions listed above. However, the need for this expert knowledge can be eliminated at least partially using recent developments in model selection and meta-learning [5, 17, 54, 53]. While there exist attempts at open source implementation for automatic model and parameter selection [6, 26, 53, 24] and meta-learning [17], these implementations are in the form of research prototypes that lack in terms of usability, documentation, testing and integration into the existing ecosystem.

Funding from this proposal would enable a concentrated effort to include more automatic model selection in scikit-learn, and therefore lower the barrier to entry for applying machine learning even further. The components of the proposed work are:

- Improving the existing pipelining facilities for more automatic feature extraction.
- Recommend explicit sets of parameters to tune for each algorithm, together with recommended ranges, based on large-scale experiments.
- Integrate methods for Bayesian optimization for parameter selection.
- Integrate meta-learning for automatic model selection.

Together, these additions to the scikit-learn ecosystem will enable more widespread use, in particular by scientists outside of the field of machine learning, and improve results of existing users. Our goal is to improve the existing infrastructure for model selection in scikit-learn, and develop more novel features like meta-learning outside of the main scikit-learn package, for possible later inclusion. To accomplish these goals, we will collaborate with the group of *Prof. Frank Hutter*, who are leaders in the current research on meta-learning, and will allow us to built on their expertise. We will also collaborate with the group of *Prof. Joaquin Candela* developing the OpenML platform, to make sure their platform will support the needs for large-scale model comparison, and will be able to provide easy access to our results. We also established collaboration with scientists in multiple fields who apply machine learning in their research, to ensure that the solutions we create are applicable to a wide spectrum of domains. These collaborations include *Prof. David Hogg* (Astrophysics and Astronomy), *Prof. Kyle Cranmer* (Experimental Particle Physics), *Prof. David Sontag* (Medical Data Analysis), and Prof. Chris Fonnesbeck (Biostatistics), as described in the letters of collaboration accompanying this proposal. Their groups will adopt the tools created in this project to provide early feedback on their usefulness and provide use-cases to guide development.

4 Intellectual Merit

Improving automation in model selection and preprocessing in `scikit-learn` will have far-reaching implications for existing and future applications of machine learning. Research projects that already use `scikit-learn` for machine learning will be able to adapt better models with minimal changes to their workflow, potentially improving their research outcomes. For research projects that are not relying on machine learning yet, including a model from `scikit-learn` will be easier and require less expert knowledge.

Developing more automated model selection requires extensive benchmarking, analysis and evaluation on a diverse array of machine learning problems. Such a large scale evaluation, in the spirit of Caruana, Karampatziakis, and Yessenalina [11] and Caruana and Niculescu-Mizil [12] and more recently Feurer et al. [17] can provide important insights into the state-of-the-art in machine learning. This proposal will not only provide results of extensive evaluation, it will also result in improved infrastructure for large scale studies of algorithms. To evaluate the importance of the improvements, we conducted a community survey among scikit-learn users, which found a large demand for this functionality. Details can be found in Section 8.

5 The `scikit-learn` Package and Ecosystem

The `scikit-learn` machine learning library is an open source library for the Python programming language, distributed under a BSD license. It is developed largely by a community of volunteers, with some support from INRIA, Telecom ParisTech, Paris-Saclay Center for Data Science and through Columbia University.. The package was first released in 2010, and new releases are made semi-annually. The development team has 38 members, with releases and project management being coordinated between PI Müller and Olivier Grisel at INRIA. There have been approximately 670 contributors to the project so far, demonstrating the wide community engagement, with each release typically including changes from 100 to 150 contributors.

5.1 Project Description

The `scikit-learn` project focusses on efficient and easy-to-use implementation of state-of-the-art machine learning algorithms that are useful for a wide audience of machine learning practitioners in research and commercial applications [44, 10]. Implemented algorithms include Support Vector Machines, Random Forests, Gradient Boosting, Non-Negative Matrix factorization, Independent Component Analysis, K-Means, DBSCAN, Isomap, t-SNE and many others. To enable easy evaluation and model selection, `scikit-learn` implements metrics like the R^2, AUC, Adjusted Rand score, Mutual information, Average precision and many more. Additionally, a framework for cross-validation and parameter selection is provided, allowing parameter tuning with very little effort.

Given that `scikit-learn` is mostly developed and maintained by volunteers, one of the core principles of `scikit-learn` is to lower the barrier for new developers, and keep the complexity of the code as low as possible, to simplify maintenance. The success of this approach can be seen in the large number of contributors to the project.

5.2 Interface and Extensibility

The simple interface of `scikit-learn` has received substantial recognition for its user friendly design [29, 60]. The main functionality of machine learning models can be summarized using just three functions (see Table 1):

`fit`	for building models

`predict`	for creating predictions
`transform`	for generating new representations.

Creating a custom model or preprocessing method using this interface is very simple, and a way in which many users extend the functionality. The `scikit-learn` documentation has comprehensive documentation on the conventions and interfaces in `scikit-learn`, to promote the creation of custom extensions.

5.3 Impacts in Research

The `scikit-learn` project has had an impact first in the field of machine learning, where it continues to provide a baseline for comparison, as well as a framework in which to develop and evaluate new machine learning algorithms. The much broader omni-disciplinary impact is in providing easy-to-use tools for solving machine learning tasks. By providing a collection of methods with a simple and consistent interface, researchers can easily explore different solutions to their machine learning problems, with a large collection of well-tested and well-established algorithms at their finger tips, as witnessed by the number of citations of the paper describing `scikit-learn`.

5.4 Impacts in Education

In addition to being widely used as a toolkit by practitioners, `scikit-learn` is also popular in teaching machine learning. The emphasis on accessibility, usability and documentation within the `scikit-learn` project makes it ideal for an introductory course in machine learning, and allows access to a wide variety of algorithms. The `scikit-learn` project is particularly popular in teaching Data Science courses that focus on making inferences about a particular data set, rather than the mathematics that go into particular ways to solve machine learning problems. Data Science courses are usually targeted at a broad spectrum of students with mixed backgrounds, providing them with the data analysis tools useful across domains.

Academic teaching has made use of `scikit-learn` at institutes including New York University, Brown University, Duke University, University of California Berkeley, Stanford University, Princeton University, Columbia University, University of Pennsylvania, Georgetown University, Cornell University and others.

5.5 Reproducibility, Democratization and Openness

An important contribution of `scikit-learn` has been in providing a common ground for scientists to base their research on. Reimplementing and algorithm from a text book or the description in a paper is often not straight-forward, and slight differences in implementation details can lead to different experimental outcomes. By providing shared, open and accessible infrastructure that is used by many research groups, `scikit-learn` facilitates reproducibility of algorithmic results. The open source nature of `scikit-learn` means that everybody can have access to advanced machine learning within the scientific Python ecosystem, without having to spend money on commercial analysis software like Matlab, SPSS or Stata.

5.6 Integration in Scientific Python Ecosystem

The `scikit-learn` project is firmly rooted in the scientific Python ecosystem, and has been one of the catalysts of its success [29, 1]. The `scikit-learn` project builds heavily on the foundations on `NumPy` and `SciPy`, and integrates easily with the popular `pandas` library for data analysis and the `matplotlib` library for plotting and visualization. The `scikit-learn` package has been included in several scientific Python distributions, such as the popular cross-platform ContinuumIO Anaconda and Enthought Canopy distributions, as well as the Python-xy distribution for Microsoft Windows.

```
# model fitting and prediction with fixed parameters:
from sklearn.ensembled import RandomForestClassifier
forest  = RandomForestClassifier(max_depth=10, max_features=3)
forest.fit(features_train, labels_train)
forest.predict(features_test)

# model fitting and prediction with grid-searched parameters
from sklearn.model_selection import GridSearchCV
parameter_grid = {'max_depth': [1, 3, 10], 'max_features': [1, 2, 3]}
forest_grid = GridSearchCV(RandomForestClassifier(), parameter_grid)
forest_grid.fit(features_train, labels_train)
forest_grid.predict(features_test)

# grid-search with pipeline of feature selection, pca and random forest
from sklearn.pipeline import Pipeline
pipeline = pipeline([('select', SelectPercentile()), ('pca', PCA()),
                     ('rf', RandomForestClassifier())])
parameter_grid = {'select__percentile': [10, 50, 90],
                  'rf__max_depth': [1, 3, 10], 'rf__max_features': [1, 2, 3]}
grid = GridSearchCV(pipeline, parameter_grid)
...
```

Figure 1: Current interface to `scikit-learn` classification and parameter selection. The user needs to select parameters of importance (here `max_depth` and `max_features`), and values for these parameters to test (here 1, 3, 10 for `max_depth` and 1, 2, 3 for `max_features`). Building more complex pipelines comprising multiple steps of feature processing complicates the process.

The succinct interface of `scikit-learn` also lends itself well to interactive data exploration and model building within the Jupyter Notebook environment.

5.7 Other Machine Learning Packages

The software tools used in the data science community can to a large degree be separated by language, with Python and R by far the most popular choices, in particular in research environments. Other languages include Java, Scala, C++, Lua and Julia.

While the R language has a rich ecosystem of tools for data analysis, statistical modeling and machine learning, it has two major shortcomings: Firstly, it is a domain specific language, specifically designed for statistic applications. This makes it hard to integrate into existing software, or to extend an R program to a web-service or a graphical user interface. Being a special purpose language, there are few R programmers outside the field of data analysis, making it harder to enter the field. Python on the other hand is a general purpose programming language, and is being widely adopted as the first programming language to be taught in schools and universities, owing to its simple syntax and powerful libraries. Python is widely used for user interfaces, web services and commercial systems. Secondly, the software packages available for R are highly fragmented, without common interfaces, often little documentation and lack of basic software development practices like unit tests. Because Python is rooted in the programming community rather than the statistics community, Python packages tend to have better software development practices, including version control, documentation and testing. As point of reference, the first unit testing framework in R was released in 2015, while a unit testing framework is included in the Python standard libraries since 2001.

There are two major collections of machine learning algorithms in R: MLR and caret. MLR is a general framework like scikit-learn, implementing many machine learning algorithms, as well as model selection and evaluation. However, MLR has not found wide adoption, with only 550 stars on the code sharing service GitHub and 35 contributors, compared to scikit-learns 15,000 stars and 769 contributors. [3] We speculate that this low adoption is based on the number of abstractions within MLR that create a higher barrier to entry. The caret package for R has a terser interface, but also does not have the level of adoption of scikit-learn, with only 460 stars and 41 contributors on GitHub. While the caret package includes tools for model selection, data analysis and preprocessing, it does not actually implement any machine learning methods. Instead, it wraps existing implementations, making it hard to triage bugs and provide a uniform interface. The adoption of third-party code also leads to a unit-test coverage of the code of only 17% (compared to 95% for scikit-learn).

In Python, the two most noteworthy related projects are the relatively recent tensor-flow, and the Orange toolbox. Tensor-flow is a framework for expressing computation graphs that can be executed efficiently on many compute infrastructures, including desktop PC CPUs, GPUs, mobile phones and clusters. It also supports automatic differentiation, that is the automatic computation of gradients in any computation graph. Tensor-flow was designed for deep learning research, but can be used more generically. While the tensor-flow library is a much lower level of abstraction than scikit-learn, more comparable to NumPy, tensor-flow also contains a module called tf.learn, which mimics the scikit-learn interface. Efforts on scikit-learn and tf.learn are mostly orthogonal, with tf.learn focusing solely on deep learning, with an emphasis on cutting edge research, while scikit-learn provides a wide array of time-tested methods for general machine learning applications. Most improvements suggested in this project also benefit the algorithms in tf.learn, as they focus on model-agnostic infrastructure, and therefore can be used with the scikit-learn compatible tf.learn models.

Outside the Python and R languages, the two most widely used software tools are the Weka environment writen in Java and machine learning software build on top of the spark distributed computing framework (written in Scala). Weka includes a graphical user interface (GUI), which is the primary way to interact with the software. Weka is meant for small-scale analysis and does not scale well to large datasets. Furthermore, working within the GUI severely limits the available tools compared to having the full flexibility of a programming language like Python or R.

In contrast, spark is an abstraction build on a cluster computing infrastructure and can scale elastically to datasets of nearly arbitrary size. Spark includes the machine learning library MLLib. While Spark with MLLib is the only of all the tools we mentioned that is capable of terrabyte-scale machine learning, only very few, highly specialize research projects have enough data to justify the overhead of using a distributed solution. On smaller datasets, solutions using a single machine are often faster, as they require less communication and can implement a wider variety of algorithms [42].

Weka and spark, as well as solutions in low-level languages like C and C++ share a common problem for use in research: they are not designed for interactivity with the user. While there are interactive environments to program in these languages (even for C), there are no integrated

[3] GitHub based metrics are likely to be skewed in favor of Python, as we suspect Python users are more likely to star a repository on GitHub. The number of contributors is likely accurate, but Python users might be more likely to contribute to a project they are using, so it might not be reflective of package use. We are not aware of less biased metrics.

ecosystems that would allow interactive exploration of data, which is central to data-driven research. There are no mature scientific plotting libraries for either Java or Scala, so generating something as simple as a bar chart or a confusion matrix can be challenging.

5.8 Existing Solutions for Automatic Machine Learning

Within recent years, several implementations of automatic machine learning have been published. Many available solutins are implementations of global optimization algorithms, such as genetic programming and Bayesian optimization. These packages provide a black-box optimization method, but require additional work to define the search space and integrate with the actual machine learning algorithms. Furthermore, these packages do not provide any meta-learning capabilities. The hyperopt [6] and spearmint [54] packages are examples of this approach. Both of these packages have been discontinued as their authors have left the academic research community. We are aware of four packages that attempt to solve the more integrated end-to-end automatic machine learning problem: auto-sklearn [17], autoweka [56, 27], `tpot` [41, 40] and `hyperopt-sklearn` [26]. Out of these, only `auto-sklearn` performs meta-learning.

As Auto-WEKA is implemented as a component of WEKA, it inherits its limitations as outlined in Section 5.7. The `hyperopt-sklearn` packages is a catalog of parameters for several `scikit-learn` models that allows automatic machine learning with `hyperopt` and `scikit-learn`. `hyperopt-sklearn` is not actively developed any more. The `tpot` package uses genetic algorithms to build pipelines of `scikit-learn` models, but performs about as well as simpliy using a random forest without parameter tuning [40]. The `auto-sklearn` and related projects such as `hyperband` [28] represent the current state-of-the-art in automatic machine learning, by performing automatic parameter and model selection and meta-learning. However, `auto-sklearn` has major restrictions that limit its usability, such as no support for Windows or OS X, no support for the latest version of `scikit-learn`, and a complex installation process. Furthermore, `auto-sklearn` does not provide any feedback to the user before the whole search process is finished. By coordinating efforts within `scikit-learn` and `auto-sklearn` the proposed project will improve upon these limitations, as well as extend the very basic meta-learning algorithm implemented in `auto-sklearn`.

6 Description of Activities

The goal of this proposal is to implement major usability and automation features in `scikit-learn`, decreasing the domain expertise required to successfully implement machine learning models. We propose to improve three aspects of applying machine learning models that currently require substantial expert knowledge: parameter selection, model selection and data preprocessing.

6.1 Default Parameter Ranges

Nearly all machine learning models come with parameters to set or tune to achieve good predictive performance. The most wide-spread way to adjust these parameters is grid-search with nested cross-validation. Grid-search consists of an exhaustive search over all possible combinations of the parameters under consideration. Example code for the process is given in Figure 5.6. A major hurdle in applying grid-search in practice is that algorithms often have many tuning parameters, making exhaustive search over all of them infeasible. However, in practice only a small subset of the parameters is usually critical for good performance. This set, and good candidate values to try, are often not well-documented in research and not included in text books. The `scikit-learn` documentation tries to give guidance, but these can be hard to find and understand by people outside of machine learning. We propose the inclusion of a programmatic way to query for the parameters to adjust for each model, and what good parameter ranges are.

While there is some community consensus on this issue, we want to back up our choices by large scale experiments on existing benchmark libraries of datasets, like OpenML [57]. These enhancements will be included directly in the `scikit-learn` package.

Task 1 Analyze parameter ranges of commonly used supervised models.
Task 2 Provide default parameter ranges inside `scikit-learn`.

6.2 Bayesian Optimization Based Parameter Selection

An alternative to exhaustive grid-search in selecting parameters is using Bayesian optimization to iteratively improve the tuning parameters of a model [5, 52, 53]. This technique has been well-established in the machine learning literature, and there are several implementations available. [6, 17, 26, 54]. However, these algorithms have not made their way into the software used by domain scientists that use machine learning algorithms as tools in their research. By incorporating a robust and efficient implementation into `scikit-learn`, this proposal will bring the benefits of the recent advantages in model selection from the machine learning community to the scientific users of `scikit-learn`.

Task 3 Analyze and integrate Gaussian Process based parameter optimization.
Task 4 Analyze Random Forest based parameter optimization.
Task 5 Integrate `auto-sklearn` Bayesian optimization with `scikit-learn`.

6.3 Automatic Preprocessing Selection

The scikit-learn `scikit-learn` package has a build-in mechanism to construct "pipelines" which are complex machine learning workflows, consisting of operations like feature extraction, feature transformation, feature selection and predictive models. All evaluation and parameter selection mechanisms in `scikit-learn` can operate on these pipelines. However, selecting which steps to chain together, that is which preprocessing to use for which model, is left to the user. Currently there is no automatic process to compare different pipeline constructions, even though the right combination of methods is often crucial but not obvious in practice. We propose to extend the model selection and pipelining framework in `scikit-learn` to allow automatic selection of pipeline steps. To help automatic preprocessing selection, we will also programatically define input and output conditions of different processing steps, such as requirements for sparse data, dense data, non-negativity of features and others.

Task 6 Allow setting of pipeline steps in `scikit-learn` parameter searches.
Task 7 Implement tagging of pre-conditions and post-conditions for data transformations.
Task 8 Refactor `scikit-learn` testing to validate pre-condition and post-condition tags.
Task 9 Collect common encoding schemes for categorical and continuous data in a `scikit-learn` compatible way.

6.4 Meta-Learning

Going beyond brute-force search or even Bayesian optimization for parameter and model selection, it is possible to use machine learning to recommend suitable algorithms and parameters based on properties of a data set, such as number of samples, number of features, number of classes and statistical properties of the features [30, 17]. Given these meta-features and the best pipeline and parameters found using Bayesian optimization or grid-search, it is then possible to build a machine learning model to predict what the best classifier for a new dataset would be. This prediction

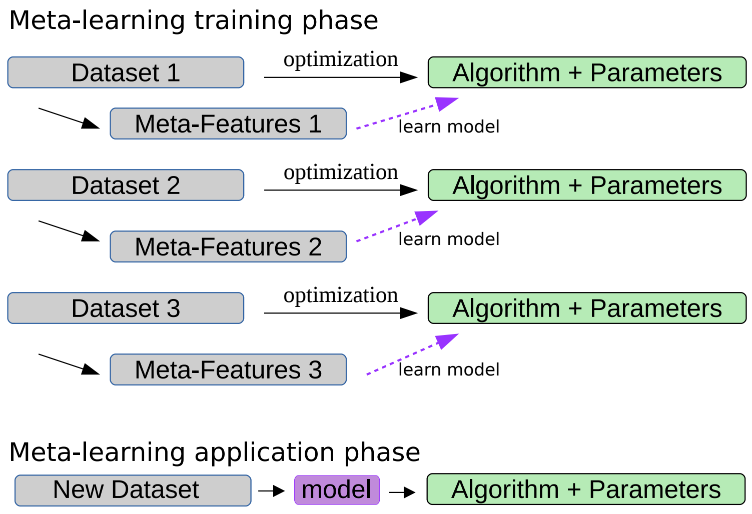

Figure 2: Illustration of meta-learning. First, good parameter values for each training dataset are found using optimization. Each dataset is then represented via meta-features. A Machine learning model is learned to predict parameter settings from the meta-features. For a new dataset, the machine learning model can be used to find good parameters, removing the costly optimization step.

based on meta-features is computationally much less demanding than searching for a model and parameters from scratch for each new dataset. Figure 2 illustrates the process. Meta-learning allows the principled incorporation of expert knowledge as encoded in the collection of datasets used for training. While there are working implementations of meta-learning available (`autoweka`, `auto-sklearn`), these projects are currently in a "research software" state. Making these methods available to the wider scientific audience will require substantial engineering efforts. The previously described steps of incorporating Bayesian optimization and automatic preprocessing selection will lay the foundation to enable meta-learning within the `scikit-learn` framework. These additions will likely not be included in the main `scikit-learn` package within the scope of this proposal, but will be made available as an additional package, compatible with and extending `scikit-learn` functionality. An illustration of the proposed additions is shown in Figure 6.4.

Task 10 Refactor `auto-sklearn` to make use of new pipeline and transformation conditions.
Task 11 Analyze meta-learning features on OpenML datasets.
Task 12 Create meta-learning package from the previously build components, with full user documentation and test coverage that is installable via the Python package manager.

7 Enabled Research Opportunities

The proposed project will benefit researchers inside the machine learning field, but more importantly will have an impact on new and existing applications of machine learning across many domains of science.

7.1 Reduce Barrier to Entry

One of the premier goals of this project is to lower the barrier to entry to applying machine learning in scientific applications even further. The `scikit-learn` package with its intuitive interface and comprehensive documentation has made machine learning algorithms available to a much wider audience. However, selecting and tuning models still requires machine learning expertise. The wealth of algorithmic choices for solving a particular research problem can be overwhelming to researchers from other disciplines. By building more automated abstractions on top of the existing machine learning algorithms in `scikit-learn`, we will enable researchers to apply powerful models

```python
# model selection with default ranges
from sklearn.model_selection import GridSearchCV, default_grid
forest = RandomForestClassifier()
forest_grid = GridSearchCV(RandomForestClassifier(), default_grid[forest])
...
# Bayesian optimization is an alternative to grid-search:
from sklearn.model_selection import BayesianSearchCV
BayesianSearchCV(RandomForestClassifier())
...
# allowing searches over pipeline steps
from sklearn.model_selection import AutoPipeline
pipe = AutoPipeline(RandomForestClassifier())
BayesianSearchCV(pipe)
...
# meta-learning removes need to specify model or ranges:
from metalearning import AutoClassifier
classifier = AutoClassifier()
classifier.fit(features_train, labels_train)
...
```

Figure 3: Illustration of some of the proposed additions to the current `scikit-learn` API. The `default_grid` contains important parameters and ranges. `BayesianSearchCV` replaces the exhaustive search in `GridSearchCV` with Bayesian optimization. `AutoPipeline` searches over preprocessing steps, removes the need for users to specify them explicitly. Finally, `AutoClassifier` takes care of preprocessing and model selection via meta-learning.

without learning the characteristics and particularities of specific methods. This will ease the adaption of machine learning for many researchers that have not yet made use of machine learning.

7.2 Improved Rapid Prototyping

In data science, exploration is often limited by the human interactions needed to analyze data. Being able to rapidly ask many research questions about a dataset or task of interest allows quick exploration of hypotheses and speeds up research. Data preprocessing and model tuning is often a laborious and time-consuming part of analysis. More automation in applying machine learning means that a researcher can ask a scientific question in terms of a machine learning problem, start the automated machinery to search for a model, and then continue exploring the data, without having to closely monitor the process on the model. This frees up research time to investigate other questions, instead of trying to find the right model to answer the first.

7.3 Plug-In Replacements

Many researchers are already using `scikit-learn` models in their projects, as witnessed by the citations and other usage statistics we reported above. As the automation features in this project will provide the same interface as the existing models, researchers can replace the models in their existing projects by an automatic model search. This will lead to better predictive results by only changing two lines of code.

7.4 Large-Scale Comparisons

Lastly, by providing a reproducible and open large-scale comparison of machine learning methods on a wide variety of datasets, we provide guidance for future research in machine learning itself. In the tradition of Caruana and Niculescu-Mizil [12] and Caruana, Karampatziakis, and Yessenalina

[11], we will identify strengths and weaknesses of existing models, to allow dissemination by the wider machine learning community. By using the OpenML platform for model evaluation, we are not only facilitating reproducibility of results, as all parameters and results are stored on the platform, we are also enriching the information available on OpenML by populating the database with our results.

8 Community Engagement, Outreach, and Sustainability

8.1 Community Survey

To evaluate the importance of the proposed additions to the `scikit-learn` user community and to identify other possible aread of improvement we conducted a self-selected survey of the scikit-learn community. Using the `scikit-learn` mailing list and social media, we collected 372 responses, including 184 responses from users involved in academic research [4]. When asked how likely they would make use of automated machine learning functionality if it became available, 88% of respondents replied they are "likely" or "very likely" to use automated hyper-parameter search and 72% replied they are "likely" or "very likely" to use automatic model selection. When asked to pick the top 5 out of 14 possible areas of improvement, 38% ranked automatic parameter selection within the top five, behind "Working with out-of-memory datasets" (57%), "Better pandas integration" (43%) and "Easier visualization of data and models" (47%).

8.2 Community Integration

As co-maintainer and member of the core team of developers of `scikit-learn`, PI Müller is well integrated into the development process of `scikit-learn`. This will enable the direct incorporation of many of the proposed enhancements into the `scikit-learn` main package. It will also provide a wide exposure of the proposed activities to the `scikit-learn` community. We expect contributions to the proposed project from the open source community from day 1 of this project. The close connection of PI Müller to the `scikit-learn` user community will also enable us to closely interact with users to ensure covering common use cases, instead of creating software solutions in the vacuum.

8.3 Software Quality and Testing

The `scikit-learn` project has a history of high standards on code quality, reviews and testing. Each new contribution to `scikit-learn` needs to be reviewed by at least two senior team members in addition to the contributor. Often, many more reviews are performed. The pull-requests (contributions) made to the project have on average 16 comments made by developers and maintainers on improving code quality and algorithms.

The `scikit-learn` package has an extensive testing suite, covering 94% of lines of code in the project. The test suite consists of unit tests, integration tests and algorithmic tests. Automated testing is performed on all algorithms in `scikit-learn` to ensure a common interface and user experience. All tests are run as continuous integration tests on Microsoft Windows and Linux, and using multiple versions of Python as well as multiple versions of `NumPy` and `SciPy`, ensuring compatibility with a wide array of end-user systems.

In adding to the `scikit-learn` project, this proposal will leverage the existing infrastructure and quality standards inside the `scikit-learn` community, ensuring high quality, well-tested code.

[4]The complete results can be found at https://www.surveymonkey.com/results/SM-RHGZVZ73/

8.4 Documentation and Distribution

Similar to the processes for reviews and testing, the proposed project will also be able to take advantage of the well-tested documentation and distribution infrastructure of `scikit-learn`. As mentioned above, there is continuous integration testing on multiple operating systems, ensuring compatibility and seamless installation across platforms. The integration services are also set up to build binary releases of the `scikit-learn` package for distribution, so that making a release that can be installed on any platform is as easy as tagging a commit as a release. The continuous integration servers are also configured to rebuild the documentation on a per-change basis, so that the documentation website (for the development version) is always up-to-date with the current code. The `scikit-learn` project has a history of increasingly high standards for documentation, requiring a description of algorithms, use cases, important parameters and theory, as well as compelling examples [13, 29]. This culture of comprehensive and accessible documentation will carry over to any additions made as part of this proposal.

8.5 Sustainability Plan

The `scikit-learn` community has grown substantially over the years, and volunteer efforts are by far the largest component of work contributed to the project. When people do leave for personal reasons, often time commitments made as part of more senior academic positions, the project has managed to smoothly integrate new contributors into the core team. Due to substantial efforts to ease the barrier of entry for contributors, the `scikit-learn` team is able to attract new volunteer core developers on a regular basis, and has successfully transitioned from one “generation” to the next multiple times.

While there is an active community around `scikit-learn`, it is very hard to make major changes based on volunteer efforts alone. New models that are added have often been the product of internships or the Google Summer of Code program. We therefore propose to hire a full-time developer as part of this proposal to greatly accelerate the progress in terms of usability and automation. Once the proposed features are included into the project, maintaining them will be possible using the resources provided by the community.

8.6 Diversified Outreach

As part of this proposal, we plan to hold an annual workshop or coding sprint in the New York area to engage scientists, and introduce them to contributing to open source. Experience has shown that many people are interested in contributing to open source, but are often intimidated by the process. By hosting an open and interactive workshop, we aim to broaden the base of contributors to the `scikit-learn` project. Unfortunately open source projects often lack diversity in their core team, and most core contributors of scientific Python packages being men. We hope to improve the situation by partnering with local and national organizations for women in machine learning and women in the Python community, to engage more female developers. We plan a weekend-long workshop, which will give a short introduction to contributing to open source, followed by two days of hands-on contributions to `scikit-learn`. From prior experience [51], it is possible for new contributors to make meaningful changes within one or two days, so that attendants will go home with a contribution to the project. After pioneering this model with `scikit-learn`, our goal it to engage with other open source projects with core developers local to New York (`pandas`, `matplotlib`) to increase the impact of the workshop.

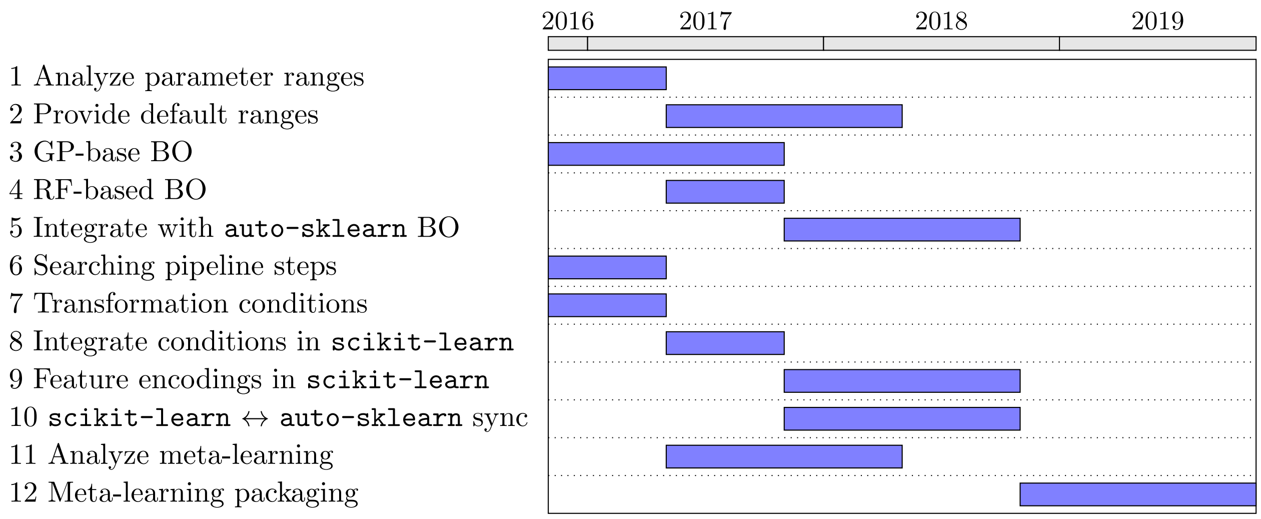

Figure 4: Project timeline

9 Project Coordination and Evaluation Plan

9.1 Project Coordination and Timeline

PI Müller is a core developer and co-maintainer of `scikit-learn` and deeply integrated into the community around it. The PI will lead the execution of the project and the integration into `scikit-learn` and the `scikit-learn` ecosystem. The developer will implement the new features and review changes proposed by the `scikit-learn` community. The developer will also perform and analyze large-scale benchmarking experiments to validate the effectiveness of the parameter recommendations and the automation features. The PI and developer will both interact with the greater community via issue trackers, mailing lists and chat rooms. The developer and PI will share working space to aid close collaboration. Figure 4 illustrates the timeline of the project.

9.2 Need for a Senior Developer

As many of the contributors of `scikit-learn` work in academic environments, often the time they have available to do volunteer work on open source is inversely proportional to their seniority. Consequently, there are many junior contributors with good coding skills, but less developed project management and timing skills. Therefore, it is paramount to have a senior developer to lead the efforts on major new features, to ensure roadmap and scoping are useful and realistic. The `scikit-learn` project has around 430 open contributions (GitHub pull requests) at the moment, which require code review and oversight from a senior developer to be integrated into the project. This exemplifies the size of the community of contributors, but also points out the bottleneck in terms of senior developers that have enough machine learning and software development expertise to judge the usefulness, efficiency and correctness of the proposed additions.

9.3 Evaluation

The success of an open source project is notoriously hard to measure. The success and impact of an addition to an existing open source project even more so. Open source software has many paths of distribution, few of which can be tracked reliably. The number of citations of the relevant paper [44] could be used instead, but is likely to underestimate the number of papers using the `scikit-learn` library in their research, as software is often not cited.

To get a more comprehensive picture of the adoption of an open source software package, we can look at the citation and download counts together with other statistics, like the number of discussions on the question answering site Stack Overflow, the number of contributors, the number of people writing to the mailing list, the number of projects depending on the package etc. These numbers are particularly meaningful when compared to other projects with a similar scope.

We will develop the more novel features outside of the `scikit-learn` package, which will ease measuring the impact of the proposed additions. The impact of contributions to the `scikit-learn` package, however, is harder to measure. One simple measure of impact is that some the proposed features will actually be integrated into `scikit-learn`. Even though the PI is tied into the core developer team, there is no guarantee a contribution will be accepted unless it meets the very high standars on code quality, usability and usefulness set by the community.

Another direct measure of impact is to count use of a particular function in code available on GitHub. As more and more research groups use version control for their experimental code, and publish it as open source, a growing number of groups and individual researchers have a presence on GitHub.com. Counting the use of the added functionality, in particular inside Jupyter Notebooks, provides great insight into the use of the proposed additions in research and data analysis.

In addition to measuring the usage of the project outcome, we will also measure the effect of using the implemented improvements on ease of use and efficiency of implementing machine learning algorithms. We will do so by conducting user studies with students at the Columbia University Data Sciencie Institute. The study will measure time spend to arrive at a solution for supervised machine learning problem, accuracy of the produced solution, and a qualitative evaluation of user experience.

10 Collaborations

10.1 Diversity in Open Source in Collaboration with WiMLDS

While the `scikit-learn` community is quite successful in finding new contributors, unfortunately only one of the 38 `scikit-learn` core developers is a woman. Similar issues can be found in related packages, with `NumPy` having no woman among the 13 core developers and `SciPy` having one woman out of 22 core developers.

To increase diversity, this proposal includes an annual workshop to increase contributions to the open source community, in particular targeted at women. To this end, we collaborate with the "Women in Machine Learning and Data Science" meetup group located in New York, a community of over 1.500 (mostly) female data science and machine learning experts.

10.2 Collaboration with `auto-sklearn`

The `auto-sklearn` project [17] currently provides a research prototype of meta-learning with Bayesian optimization for parameter selection. One of the main goals of this proposal is to transfer the research within the `auto-sklearn` project to an easy-to-use and well-documented library, integrated within the `scikit-learn` ecosystem. To this end, we will collaborate with the `auto-sklearn` team, building upon their insights and technologies. *Prof. Frank Hutter* committed to providing the support of his group to integrate our additions to `scikit-learn`, such as the default parameter ranges and automated preprocessing selection into their software, while in turn providing the necessary domain knowledge to reproduce their research results in a user-friendly library.

10.3 Collaboration with OpenML

Evaluation and benchmarking on real-world datasets are essential to this proposal, in providing guidance for good parameter ranges, and assessing the success of automatic model selection methods. The OpenML project [57] provides a quickly growing database of machine learning datasets with associated tasks, including classification and regression [58]. At the time of this writing, there were nearly 20.000 datasets hosted on OpenML, ranging from classical datasets like MNIST and small toy datasets like the iris and wine datasets, to large scale datasets with millions or even tens of millions of samples, from domains including as biology, medicine and physical sciences and commercial applications. This growing collection of research datasets provides the basis of our assessments, and ensure relevance of our effort across research domains. *Prof. Joaquin Candela* committed to improving and maintaining the Python interface for the OpenML platform, and improving the support for `scikit-learn` based models.

10.4 Early Adopters

The following researchers have committed to being *early adopters* of our software products. They will use the improvements and packages in their research projects, and provide valuable feedback for ensuring usefulness of the software for a variety of research applications:

Experimental Particle Physics *Prof. Kyle Cranmer* (NYU) committed to have his group be an early adopter of the proposed project for their work on searching new phenomena at the Large Hardron Collider.

Astronomy *Prof. David Hogg* (NYU) committed to have his group be an early adopter of the proposed project their projects in astronomy and astrophysics. In particular, they will apply the developed methods in their work to calibrate imaging spacecraft (like the GALEX and Kepler missions) and in their data-driven models of stars (exemplified by our project called The Cannon), as well as related projects.

Medical Data Analysis *Prof. David Sontag* (NYU) commited to have his group be an early adopter of the proposed project for their work on building risk stratification algorithms for early detection of type 2 diabetes. Building their models involves a significant amount of iteration over approaches to regularization, the type of data to include in the model, and different ways of deriving features from the data, which could potentially be automated using the software described in this proposal.

Biostatistics *Prof. Christopher Fonnesbeck* (Vanderbilt University) committed to be an early adopter of the proposed project in his work in statistical of biomedical data in epidemiological applications, such as the control of infectious disease. This work includes building of classification and regression models for meta-analysis, with the goal if supporting evidence-based medicine.

11 Broader Impacts

The proposed project has a wide-reaching impact on the practical use of machine learning in research, and on how machine learning can be taught to scientists from other domains. Providing more automatic model selection will drastically lower the barrier to entry to using machine learning for people without domain expertise in machine learning. Additionally, it will save time and effort spend by researchers doing model selection by hand, replacing their effort by an automated process. This will make researchers more productive, and will allow them to focus on their area of study.

Automation, coupled with publishing the results of large scale experiments will also provide help in education. Currently, often parameters and preprocessing are seen as undocumented expert knowledge, derived from personal experience. Formalizing this knowledge, and providing a database of experiments, will allow students to quickly master the necessary steps to apply machine learning in practice.

The proposed project will also enable us to grow the open source community within the data science and machine learning community. In particular, the position of `scikit-learn` as a popular and valued research library enables us to reach out to a broad community of users. The prominent status of `scikit-learn` enables us to influence the current make-up and values of the scientific open source ecosystem. We proposed to host “coding sprints” to introduce outsiders to contributing to open source and enlarging the number of contributors even further. This event will be held collaboration with the New York based Women in Machine Learning and Data Science group, to grow the number of female contributors in particular.

12 Results From Prior NSF Support

Dr. Andreas Müller has not been a PI or co-PI on an NSF grant.

References

[1] *11 open source tools to make the most of machine learning.* http://www.infoworld.com/article/2853707/machine-learning/11-open-source-tools-machine-learning.html#slide2. Accessed: 2016-04-19. 2014.

[2] Alexandre Abraham, Fabian Pedregosa, Michael Eickenberg, Philippe Gervais, Andreas Mueller, Jean Kossaifi, Alexandre Gramfort, Bertrand Thirion, and Gaël Varoquaux. "Machine learning for neuroimaging with scikit-learn". In: *Frontiers in Neuroinformatics* 8 (2014).

[3] Pierre Baldi, Kevin Bauer, Clara Eng, Peter Sadowski, and Daniel Whiteson. *Jet Substructure Classification in High-Energy Physics with Deep Neural Networks.* 2016. arXiv: 1603.09349 [hep-ex].

[4] CL Bennett, D Larson, JL Weiland, and G Hinshaw. "The 1% concordance Hubble constant". In: *The Astrophysical Journal* 794.2 (2014), p. 135.

[5] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for Hyper-Parameter Optimization". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. Curran Associates, Inc., 2011, pp. 2546–2554. URL: http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.

[6] James Bergstra, Dan Yamins, and David D Cox. "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms". In: *Proceedings of the 12th Python in Science Conference*. 2013, pp. 13–20.

[7] Steven Bird. "NLTK: the natural language toolkit". In: *Proceedings of the COLING/ACL on Interactive presentation sessions*. Association for Computational Linguistics. 2006, pp. 69–72.

[8] CM Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2001.

[9] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[10] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Müller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake Vanderplas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. "API design for machine learning software: experiences from the scikit-learn project". In: *European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases*. Prague, Czech Republic, Sept. 2013. URL: https://hal.inria.fr/hal-00856511.

[11] Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. "An empirical evaluation of supervised learning in high dimensions". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 96–103.

[12] Rich Caruana and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 161–168.

[13] Daoud Clarke. *Why I Love Scikit-learn.* http://daoudclarke.github.io/machine learning in practice/2013/09/18/why-i-love-scikit-learn. Accessed: 2016-04-19. 2013.

[14] Mihai Croicu and Nils B Weidmann. "Improving the selection of news reports for event coding using ensemble classification". In: *Research & Politics* 2.4 (2015), p. 2053168015615596.

[15] Oliver Doehrmann, Satrajit S Ghosh, Frida E Polli, Gretchen O Reynolds, Franziska Horn, Anisha Keshavan, Christina Triantafyllou, Zeynep M Saygin, Susan Whitfield-Gabrieli, Stefan G Hofmann, et al. "Predicting treatment response in social anxiety disorder from functional magnetic resonance imaging". In: *JAMA psychiatry* 70.1 (2013), pp. 87–97.

[16] Kevin Driscoll and Kjerstin Thorson. "Searching and Clustering Methodologies Connecting Political Communication Content across Platforms". In: *The ANNALS of the American Academy of Political and Social Science* 659.1 (2015), pp. 134–148.

[17] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. "Efficient and Robust Automated Machine Leraning". In: *Advances in Neural Information Processing Systems 28*. Dec. 2015, pp. 2944–2952.

[18] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[19] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin, 2001.

[20] Raúl Garreta and Guillermo Moncecchi. *Learning scikit-learn: Machine Learning in Python*. Packt Publishing Ltd, 2013.

[21] Olivier Grisel, Andreas Müller, Fabian Pedregosa, Lars Buitinck, Alexandre Gramfort, Gilles Louppe, Peter Prettenhofer, Mathieu Blondel, Vlad Niculae, Arnaud Joly, Joel Nothman, Jake Vanderplas, Manoj Kumar, Robert Layton, Nelle Varoquaux, Noel Dawe, Johannes Schönberger, Denis A. Engemann, Wei Li, Raghav R V, Clay Woolam, Kemal Eren, Eustache, Alexander Fabisch, Alexandre Passos, and Virgile Fritsch. *scikit-learn 0.17.1*. Nov. 2015. DOI: 10.5281/zenodo.49910. URL: http://dx.doi.org/10.5281/zenodo.49910.

[22] Gavin Hackeling. *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd, 2014.

[23] Trent Hauck. *scikit-learn Cookbook*. Packt Publishing Ltd, 2014.

[24] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. "Sequential model-based optimization for general algorithm configuration". In: *Learning and Intelligent Optimization*. Springer, 2011, pp. 507–523.

[25] M Kamalov, V Dobrynin, J Balykina, A Kolbin, E Verbitskaya, and M Kasimova. "Improving data retrieval quality: Evidence based medicine perspective". In: *International Journal of Risk & Safety in Medicine* 27.s1 (2015), S106–S107.

[26] Brent Komer, James Bergstra, and Chris Eliasmith. "Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn". In: *ICML workshop on AutoML*. 2014.

[27] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. "Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA". In: *Journal of Machine Learning Research* 17 (2016), pp. 1–5.

[28] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *arXiv preprint arXiv:1603.06560* (2016).

[29] Ben Lorica. *Six reasons why I recommend scikit-learn*. http://radar.oreilly.com/2013/12/six-reasons-why-i-recommend-scikit-learn.html. Accessed: 2016-04-19. 2013.

[30] Gang Luo. *A review of automatic selection methods for machine learning algorithms and hyper-parameter values.* 2015.

[31] Matthew Mayo. *Top 10 Machine Learning Projects on Github.* http://www.kdnuggets.com/2015/12/top-10-machine-learning-github.html. Accessed: 2016-04-19. 2015.

[32] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th Python in Science Conference.* 2015.

[33] Bernhard Misof, Shanlin Liu, Karen Meusemann, Ralph S Peters, Alexander Donath, Christoph Mayer, Paul B Frandsen, Jessica Ware, Tomáš Flouri, Rolf G Beutel, et al. "Phylogenomics resolves the timing and pattern of insect evolution". In: *Science* 346.6210 (2014), pp. 763–767.

[34] Andreas C Müller and Sarah Guido. *Introduction to machine learning with Python.* O'Reilly Media, 2016.

[35] Andreas Müller and Sven Behnke. "Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on.* IEEE. 2014, pp. 6232–6237.

[36] Andreas Müller and Sven Behnke. "PyStruct: learning structured prediction in python". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 2055–2060.

[37] Andreas Müller, Forest Gregg, Vlad Niculae, Lars, Thorsten B., Shuyang Sheng, Dmitry Kondrashkin, Joel Nothman, Eduardo Zamudio, and Bart Janssen. *pystruct: 0.2.5.1.* Apr. 2016. DOI: 10.5281/zenodo.49909. URL: http://dx.doi.org/10.5281/zenodo.49909.

[38] Andreas Müller, Sebastian Nowozin, and Christoph Lampert. "Information Theoretic Clustering Using Minimum Spanning Trees". In: *Pattern Recognition* (2012), pp. 205–215.

[39] David P Ng, David Wu, Brent L Wood, and Jonathan R Fromm. "Computer-Aided Detection of Rare Tumor Populations in Flow Cytometry". In: *American journal of clinical pathology* 144.3 (2015), pp. 517–524.

[40] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. "Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science". In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016.* GECCO '16. Denver, Colorado, USA: ACM, 2016, pp. 485–492. ISBN: 978-1-4503-4206-3. DOI: 10.1145/2908812.2908918. URL: http://doi.acm.org/10.1145/2908812.2908918.

[41] Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore. "Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I". In: ed. by Giovanni Squillero and Paolo Burelli. Springer International Publishing, 2016. Chap. Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pp. 123–137. ISBN: 978-3-319-31204-0. DOI: 10.1007/978-3-319-31204-0_9. URL: http://dx.doi.org/10.1007/978-3-319-31204-0_9.

[42] Szilard Pafka. *A minimal benchmark for scalability, speed and accuracy.* May 2015. URL: https://github.com/szilard/benchm-ml.

[43] Gregory Park, H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Michal Kosinski, David J Stillwell, Lyle H Ungar, and Martin EP Seligman. “Automatic personality assessment through social media language.” In: *Journal of personality and social psychology* 108.6 (2015), p. 934.

[44] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. “Scikit-learn: Machine learning in Python”. In: *The Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[45] R Pereira, RC Thomas, G Aldering, P Antilogus, C Baltay, S Benitez-Herrera, S Bongard, C Buton, A Canto, F Cellier-Holzem, et al. “Spectrophotometric time series of SN 2011fe from the Nearby Supernova Factory”. In: *Astronomy & Astrophysics* 554 (2013), A27.

[46] Sebastian Raschka. *Python Machine Learning*. Packt Publishing, 2015.

[47] Graham RS Ritchie, Ian Dunham, Eleftheria Zeggini, and Paul Flicek. “Functional annotation of noncoding sequence variants”. In: *Nature methods* 11.3 (2014), pp. 294–296.

[48] Nicolas L Roux, Mark Schmidt, and Francis R Bach. “A stochastic gradient method with an exponential convergence _rate for finite training sets”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 2663–2671.

[49] Justin Sahs and Latifur Khan. “A machine learning approach to android malware detection”. In: *Intelligence and Security Informatics Conference (EISIC), 2012 European*. IEEE. 2012, pp. 141–147.

[50] Dominik Scherer, Andreas Müller, and Sven Behnke. “Evaluation of pooling operations in convolutional architectures for object recognition”. In: *Artificial Neural Networks–ICANN 2010*. Springer, 2010, pp. 92–101.

[51] *scikit-learn Code Sprint with Andreas Mueller*. http://www.meetup.com/San-Francisco-ODSC/events/225950983/. Accessed: 2016-04-19.

[52] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. “Taking the human out of the loop: A review of bayesian optimization”. In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175.

[53] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 2951–2959. URL: http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf.

[54] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md Patwary, Mostofa Ali, Ryan P Adams, et al. “Scalable Bayesian Optimization Using Deep Neural Networks”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 2015, pp. 2171–2180.

[55] Shinichi Sunagawa, Luis Pedro Coelho, Samuel Chaffron, Jens Roat Kultima, Karine Labadie, Guillem Salazar, Bardya Djahanschiri, Georg Zeller, Daniel R Mende, Adriana Alberti, et al. “Structure and function of the global ocean microbiome”. In: *Science* 348.6237 (2015), p. 1261359.

[56] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms". In: *Proc. of KDD-2013*. 2013, pp. 847–855.

[57] Jan N Van Rijn, Bernd Bischl, Luis Torgo, Bo Gao, Venkatesh Umaashankar, Simon Fischer, Patrick Winter, Bernd Wiswedel, Michael R Berthold, and Joaquin Vanschoren. "OpenML: A collaborative science platform". In: *Machine learning and knowledge discovery in databases*. Springer, 2013, pp. 645–649.

[58] Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. "OpenML: networked science in machine learning". In: *ACM SIGKDD Explorations Newsletter* 15.2 (2014), pp. 49–60.

[59] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Müller. "Scikit-learn: Machine Learning Without Learning the Machinery". In: *GetMobile: Mobile Comp. and Comm.* 19.1 (June 2015), pp. 29–33. ISSN: 2375-0529. DOI: 10.1145/2786984.2786995. URL: http://doi.acm.org/10.1145/2786984.2786995.

[60] *Who is using scikit-learn?—scikit-learn 0.17.1 documentation.* http://scikit-learn.org/stable/testimonials/testimonials.html. Accessed: 2016-04-19.

[61] En-Bo Yang, Zhi-Bin Zhang, Chul-Sung Choi, and Heon-Young Chang. *Classifying Gamma-Ray Bursts with Gaussian Mixture Model.* 2016. arXiv: 1603.03680 [astro-ph.HE].

Biographical Sketch: Andreas C. Müller

Professional Preparation

University of Bonn	Germany	Mathematics	Vordiplom 2005
University of Bonn	Germany	Mathematics	Diplom 2009
University of Bonn	Germany	Computer Science	PhD 2014

Appointments

Since 2016	Lecturer in Data Science, Columbia University
2014–2016	Research Engineer, NYU Center for Data Science
2013–2014	Machine Learning Scientist, Amazon Germany

Five related products

- Andreas C Müller and Sarah Guido. *Introduction to machine learning with Python.* O'Reilly Media, 2016
- Olivier Grisel, Andreas Müller, Fabian Pedregosa, Lars Buitinck, Alexandre Gramfort, Gilles Louppe, Peter Prettenhofer, Mathieu Blondel, Vlad Niculae, Arnaud Joly, Joel Nothman, Jake Vanderplas, Manoj Kumar, Robert Layton, Nelle Varoquaux, Noel Dawe, Johannes Schönberger, Denis A. Engemann, Wei Li, Raghav R V, Clay Woolam, Kemal Eren, Eustache, Alexander Fabisch, Alexandre Passos, and Virgile Fritsch. *scikit-learn 0.17.1.* Nov. 2015. DOI: 10.5281/zenodo.49910. URL: http://dx.doi.org/10.5281/zenodo.49910
- Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Müller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake Vanderplas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. "API design for machine learning software: experiences from the scikit-learn project". In: *European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases.* Prague, Czech Republic, Sept. 2013. URL: https://hal.inria.fr/hal-00856511
- Andreas Müller, Forest Gregg, Vlad Niculae, Lars, Thorsten B., Shuyang Sheng, Dmitry Kondrashkin, Joel Nothman, Eduardo Zamudio, and Bart Janssen. *pystruct: 0.2.5.1.* Apr. 2016. DOI: 10.5281/zenodo.49909. URL: http://dx.doi.org/10.5281/zenodo.49909
- G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Müller. "Scikit-learn: Machine Learning Without Learning the Machinery". In: *GetMobile: Mobile Comp. and Comm.* 19.1 (June 2015), pp. 29–33. ISSN: 2375-0529. DOI: 10.1145/2786984.2786995. URL: http://doi.acm.org/10.1145/2786984.2786995

Five other significant products

- Alexandre Abraham, Fabian Pedregosa, Michael Eickenberg, Philippe Gervais, Andreas Mueller, Jean Kossaifi, Alexandre Gramfort, Bertrand Thirion, and Gaël Varoquaux. "Machine learning for neuroimaging with scikit-learn". In: *Frontiers in Neuroinformatics* 8 (2014)
- Andreas Müller and Sven Behnke. "PyStruct: learning structured prediction in python". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 2055–2060
- Andreas Müller, Sebastian Nowozin, and Christoph Lampert. "Information Theoretic Clustering Using Minimum Spanning Trees". In: *Pattern Recognition* (2012), pp. 205–215
- Dominik Scherer, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". In: *Artificial Neural Networks–ICANN 2010*. Springer, 2010, pp. 92–101
- Andreas Müller and Sven Behnke. "Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 6232–6237

Synergistic Activities

- Software Carpentry instructor, contributor to software carpentry and data carpentry teaching material.
- Regular tutorials on machine learning and scikit-learn, materials published under CC-0 license.
- Contributions to the OpenML open souce project.
- Contributions to the nbconvert open source project for publishing using Jupyter Notebooks.
- Organizer of regular "coding sprints" to broaden the contributor base of open source projects.