Data Management Plan

1 Roles and responsibilities

Principle 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 and shared via github. Documentation of experimental procedures and 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 hundreds 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 and 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 an simple access mechanism as alternative to the OpenML interface for accessing 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 project 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 experts.

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 trained machine learning researchers, but was later adopted by researchers across many disciplines. We identified two main barriers to entry for domain scientists wanting to apply machine learning: The representation of the data needed to apply machine learning, and the choice of algorithm 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, advanced 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 across research domains.

Intellectual Merit

This proposal advances knowledge in two ways:

1. By lowering the barrier of entry for applying machine learning even further, and improving the existing tools provided by scikit-learn, 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 tools. 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 large amounts of machine learning training.

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 an 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 many 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 this 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 [29], for short `sklearn`, provides machine learning functunality within the established — but still growing — scientific python ecosystem. `scikit-learn` is an open source library 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 [42].

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 [8]. This simple and consistent interface helps to abstract away the algorithm, and let users focus on their particular problem. It also allows replacing an algorithm by another by changing a single line of code. `scikit-learn` utilizes the well-established NumPy library to represent data and predictions. NumPy is used across domains for numeric computations, 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 [6] and Friedman, Hastie, and Tibshirani [14], while providing competitive implementation of state-of-the-art algorithms like Gradient Boosting [13], Random Forests [7] and SAG [33]. 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 a useful resource. The `scikit-learn` project has strict rules on documentation and requires examples and extensive descriptions for all algorithms.

These features lead to a 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 [29] describing `scikit-learn` has been cited 2746 according to Google scholar. Applications of `scikit-learn` spread a multitude of research areas, including Physics, Astronomy [30, 2], Biology [22, 32], Medicine, Psychology [28, 11], Cyber Security [34], Oceanograpy [39], Sociology [**catalini2015incidence**] and more.

The number of 2746 is likely underestimating the use of `scikit-learn` in published research, as a search for the term "`scikit-learn`" yields 4850 results. 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 Berkeley, Brown, CMU, Columbia, Duke, Harvard, MIT, NYU, Stanford, and University of Washington. The total number of unique contributors to the project is about 700.

The `scikit-learn` mailing list has 1666 subscribers, including 70 email addresses from "edu" domains, across 43 institutes.

On the question-answering site Stackoverflow, there are around 4500 questions tagged as `scikit-learn` related, with 222 questions asked within the last 30 days of this writing.

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. In 2015, the tutorial was so overbooked that a second session was held.

`scikit-learn` 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` [40] for astronomical data, `nilearn` [1] for neuroimaging, `MNE-Python` for MEG and EEG data, `librosa` [21] for audio and musical data, `nltk` [5] and `rosetta` for Natural language processing, `bcbio` for RNA sequence analysis, `scikit-allel` for genetic variation data, and `rootpy` for the `ROOT` scientific software framework. The main use of `scikit-learn` is outside of these major open source packages, though, as stand alone library for data analysis. Using the GitHub repository, we found about 40.000 jupyter notebooks—a format for interactive computing with python—at the time of writing.

The `scikit-learn` website containing the documentation was visited by over 230.000 users in March 2016, upward from 200.000 users in February and 180.000 users in January, reflecting the growth of the `scikit-learn` user community.

`scikit-learn` has become so propular in teaching and data science applications that several books have been written about the use of `scikit-learn` [15, 17, 18, 31]. Another book, titled "Introduction to machine learning with Python" is currently in preparation by PI Müller.

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 and 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. However, the need for this expert knowledge can be eliminated at least partially using recent developments in model selection and meta-learning.

Funding form 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.
- Provide explicit sets of parameters to tune for each algorithm, together with recommended ranges.
- Integrate methods for Bayesian optimization for parameter selection.
- Integrate meta-learning for automatic model selection.

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 much easier and require much less expert knowledge.

Developing more automated model selection requires extensive benchmarking and evaluation on a diverse array of machine learning problems. Such a large scale evaluation, in the spirit of Caruana, Karampatziakis, and Yessenalina [9] and Caruana and Niculescu-Mizil [10] and more recently Feurer et al. [12] can provide important insights into the state of the art in machine learning, and will result in improved infrastructure for large scale studies of algorithms.

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 PI Müller as part of the Moore-Sloan Data Science Environment at NYU. The package was first released in 2010, and new releases are made semi-annually. The development team has 38 members, and releases and project management are 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 and 150 contributors.

5.1 Project Description

The `scikit-learn` project focusses on effective 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. 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 facilitate 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 numerous praise for its design and user-friendlyness. The main functionality of machine learning models can be summarized using just three functions:

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. The `scikit-learn` library even has a generic test framework that allows users to test their own implementation against the behavior expected by `scikit-learn`.

5.3 Impacts in research

`scikit-learn` 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 methods. 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. The adoption of `scikit-learn` in research can be seen in the large number of citations (over 2700 according to Google scholar) as well as in the development of more domain specific solutions build on `scikit-learn`. According to the email addresses used for contributions, researchers and students from at least 23 US universities *contributed* to `scikit-learn`, indicating wide-spread use.

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. `scikit-learn` 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 Univserity, University of Pensilvania, 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 learning 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

`scikit-learn` is firmly rooted in the scientific Python ecosystem, and has been one of the catalysts of its success. `scikit-learn` 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. `scikit-learn` 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.

The succinct interface of `scikit-learn` also lends itself well to interactive data exploration and model building within the Jupyter Notebook environment. As mentioned previously, searching for Jupyter notebooks containing `scikit-learn` code on the GitHub code hosting platform yields about 40000 results.

6 Proposed Enhancements

While the `scikit-learn` package is under constant development by a large community of volunteers, the size and widespread adoption of the package result in much of this time being spend on maintenance and usability improvement, leaving little room for larger scale efforts to include major changes. 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 describes the exhaustive search over all possible combinations of the parameters under consideration. A major hurdle in applying grid-search in practice is that algorithms often have many different tuning parameters, making exhausive 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 candiate features are not usually well known and not included in text books. The `scikit-learn` documentation tries to give guidelines, 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 [40].

Task 1 Benchmark 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 [3, 36, 37]. This technique has been well-established in the machine learning literature, and there are several implementations available. [4, 12, 19, 38]. However, these algorithms have not made its way into the software used by domain scientists that use machine 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 Benchmark and integrate Gaussian Process based parameter optimization.

Task 4 Benchmark Random Forest based parameter optimization.

Task 5 Integrate `auto-sklearn` Bayesian optimization with `scikit-learn`.

6.3 Automatic Preprocessing Selection

`scikit-learn` 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 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 facilitate 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 [20, 12]. Given the 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 based on meta-features is computationally much less demanding than searching for a model and parameters from scratch for each new dataset. 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 effort. The above steps of incorporating Bayesian optimization and automatic preprocessing selection will lay the foundation to enable meta-learning within the `scikit-learn` framework.

Task 10 Refactor `auto-sklearn` to make use of new pipeline and transformation conditions.

Task 11 Benchmark 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. `scikit-learn` 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 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. An often laborious and time-consuming part of

analysis is data preprocessing and model tuning. 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 simply replace the models in their existing projects by an automatic model search. This will lead to better predictive results by simply 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 [10] and Caruana, Karampatziakis, and Yessenalina [9], we will identify strengths and weaknesses of existing models, to allow dissemination by the wider machine learning community.

8 Community Engagement, Outreach, and Sustainability

8.1 Community integration

As part of the core team and co-maintainer of `scikit-learn` developers, 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 anticipate contributions to the proposed project from the open source community from day 1 of this project. The close connection of the 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.

8.2 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.

`scikit-learn` 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. There is automated testing performed on all algorithms in `scikit-learn` that ensures 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.

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.

8.3 Documentation and Distribution

As for review processes and testing, the proposed project will be able to leverage 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 set up 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. As for testing, `scikit-learn` 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. This culture of comprehensive and accessible documentation will carry over to any additions made as part of this proposal.

8.4 Sustainability Plan

The `scikit-learn` community has grown substantially over the years, and volunteer efforts are by far the biggest part of work contributed to the project. While 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. Due to substantial efforts to ease the barrier of entry, 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 to greatly accelerate the progress in terms of usability and automation. Once these new features are included into the project, maintaining them will be possible using the resources provided by the community.

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 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 facilitate collaboration. The timeline of the project is illustrated in Figure 1

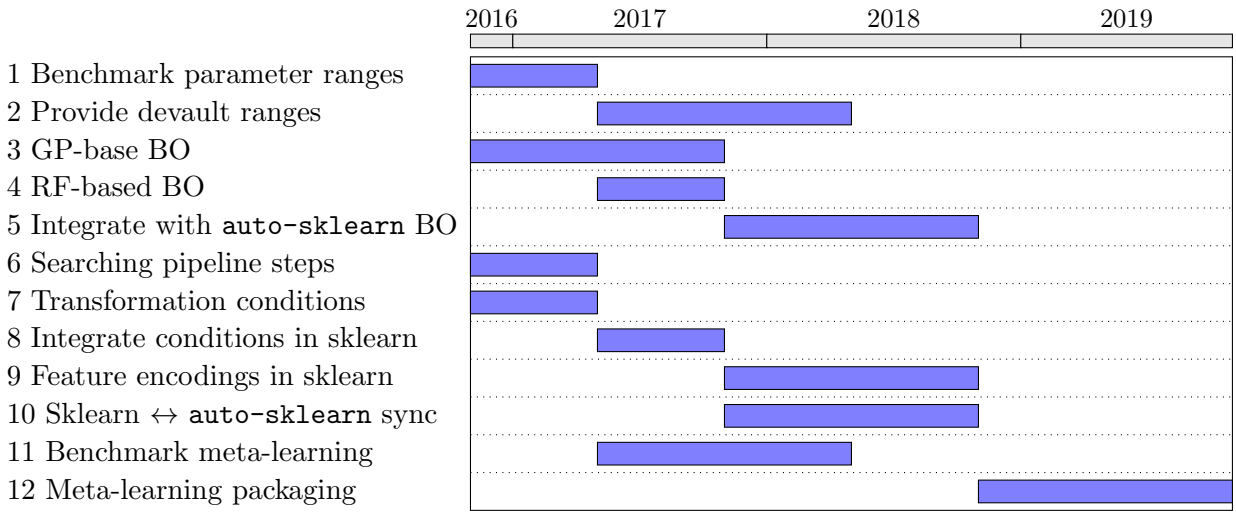

Figure 1: Project timeline

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. `scikit-learn` currently has around 430 open contributions (GitHub pull requests) that 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 bottle neck 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; in case of `scikit-learn`, these are the python package manager (pypi), pre-packaged distributions like anaconda, canopy and Python-xy, the package managers of Linux distributions (like dpkg and rpm) and downloads directly from the GitHub repository. Most of these distributions paths are hard or impossible to track. The python package manger for example reports over 500.000 downloads of the last release, which is likely to be inflated when taken as a measure of users. On the other hand, the number of citations of the relevant paper[29], around 2700, is likely to be much lower than the number of papers actually using the `scikit-learn` library in their research.

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 stackoverflow, 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 facilitate 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 developers, a contribution being accepted is not guaranteed, and the guidelines on quality, usability and usefulness are very high.

Another direct measure of impact is to count use of a particular function in code available on GitHub.com. As more and more research groups use version control fore 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.

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 women. Similar issues can be found in related packages, with `NumPy` having no women among the 13 core developers and `SciPy` having one women 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 1500 (mostly) female data science and machine learning experts. The weekend-long workshop will give a short introduction to contributing to open source, followed by two days of hands-on contributions to `scikit-learn`. From prior experience, it is possible for new contributors to make meaningful changes within one or two days, so that attendants will go home with a contribution in the project. After pioneering the project 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.

10.2 Collaboration with `auto-sklearn`

The `auto-sklearn` project [12] 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 a 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. 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 [40] provides a quickly growing database of machine learning datasets with associated tasks, including classification and regression [41]. 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, ranging from biological, medical and physical datasets to commercial applications. This growing collection or research datasets provides the basis of our assessments, and ensure relevance of our effort across research domains. 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:

Kyle Cranmer David Hogg

11 Broader Impacts of the Proposed Work

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 domain experts. 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 NSF grants.

References

[1] Alexandre Abraham et al. "Machine learning for neuroimaging with scikit-learn". In: *arXiv preprint arXiv:1412.3919* (2014).

[2] CL Bennett et al. "The 1% concordance Hubble constant". In: *The Astrophysical Journal* 794.2 (2014), p. 135.

[3] James S. Bergstra et al. "Algorithms for Hyper-Parameter Optimization". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 2546–2554. URL: http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.

[4] 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.

[5] 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.

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

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

[8] Lars Buitinck et al. "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.

[9] 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.

[10] 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.

[11] Oliver Doehrmann et al. "Predicting treatment response in social anxiety disorder from functional magnetic resonance imaging". In: *JAMA psychiatry* 70.1 (2013), pp. 87–97.

[12] M. Feurer et al. "Efficient and Robust Automated Machine Leraning". In: *Advances in Neural Information Processing Systems 28*. Dec. 2015, pp. 2944–2952.

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

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

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

[16] Olivier Grisel et al. *scikit-learn 0.17.1*. Nov. 2015. DOI: 10.5281/zenodo.49910. URL: http://dx.doi.org/10.5281/zenodo.49910.

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

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

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

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

[21] Brian McFee et al. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th Python in Science Conference*. 2015.

[22] Bernhard Misof et al. "Phylogenomics resolves the timing and pattern of insect evolution". In: *Science* 346.6210 (2014), pp. 763–767.

[23] 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.

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

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

[26] Andreas Müller, Giorgio Patrini, and Alexander Ostrikov. *patsylearn: 0.1*. Apr. 2016. DOI: 10.5281/zenodo.49915. URL: http://dx.doi.org/10.5281/zenodo.49915.

[27] Andreas Müller et al. *pystruct: 0.2.5.1*. Apr. 2016. DOI: 10.5281/zenodo.49909. URL: http://dx.doi.org/10.5281/zenodo.49909.

[28] Gregory Park et al. "Automatic personality assessment through social media language." In: *Journal of personality and social psychology* 108.6 (2015), p. 934.

[29] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *The Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[30] R Pereira et al. "Spectrophotometric time series of SN 2011fe from the Nearby Supernova Factory". In: *Astronomy & Astrophysics* 554 (2013), A27.

[31] Sebastian Raschka. "Python Machine Learning". In: (2015).

[32] Graham RS Ritchie et al. "Functional annotation of noncoding sequence variants". In: *Nature methods* 11.3 (2014), pp. 294–296.

[33] 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.

[34] 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.

[35] 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.

[36] Bobak Shahriari et al. "Taking the human out of the loop: A review of bayesian optimization". In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175.

[37] 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 et al. Curran Associates, Inc., 2012, pp. 2951–2959. URL: http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf.

[38] Jasper Snoek et al. "Scalable Bayesian Optimization Using Deep Neural Networks". In: *arXiv preprint arXiv:1502.05700* (2015).

[39] Shinichi Sunagawa et al. "Structure and function of the global ocean microbiome". In: *Science* 348.6237 (2015), p. 1261359.

[40] Jan N Van Rijn et al. "OpenML: A collaborative science platform". In: *Machine learning and knowledge discovery in databases*. Springer, 2013, pp. 645–649.

[41] Joaquin Vanschoren et al. "OpenML: networked science in machine learning". In: *ACM SIGKDD Explorations Newsletter* 15.2 (2014), pp. 49–60.

[42] G. Varoquaux et al. "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.

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 2014	Research Engineer, NYU Center for Data Science
2013–2014	Machine Learning Scientist, Amazon Germany

Five related products

- 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

- Andreas Müller, Giorgio Patrini, and Alexander Ostrikov. *patsylearn: 0.1*. Apr. 2016. DOI: 10.5281/zenodo.49915. URL: http://dx.doi.org/10.5281/zenodo.49915

Five other significant products

- Alexandre Abraham, Fabian Pedregosa, Michael Eickenberg, Philippe Gervais, Andreas Müller, Jean Kossaifi, Alexandre Gramfort, Bertrand Thirion, and Gäel Varoquaux. "Machine learning for neuroimaging with scikit-learn". In: *arXiv preprint arXiv:1412.3919* (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.