Enhancement and Maintenance of the scikit-learn project

1 Goals

The goal of this proposal is to add key features to the open-source machine learning library scikit-learn and aid in maintaining the project.

Scikit-learn powers many applications of machine learning and data science, and it is used in many bio-medical applications [KBFG18, RK19, HSP+16, Czo13]. Thanks to its ease of use, scikit-learn enables domain scientists without extensive machine learning knowledge to efficiently apply machine learning techniques to their core discipline; as a result, the original scikit-learn paper [PVG+11] has more than 17,000 citations. It is also a core upstream dependency of more specialized packages like QIIME [BRD+18, BKR+18, BDB+18], Nilearn [APE+14], MNE [GLL+14], Scikit-allel[1], Starfish[2] MSMBuilder [HSH+17], GALA [NIKP+14], TEES [BS15], Wyrm [VDH+15], MOABB [JB18], or PyOsirix [BCKL16].

This proposal aims to fund the work of Andreas Müller (PI) for 4.5 months, and that of Nicolas Hug (co-PI) for 12 months. Müller and Hug are both associate research scientists at Columbia University. Müller has been a core-developer of scikit-learn for over 7 years, and Hug has been a core developer for about six months. We plan to address the following tasks, detailed in the next subsections:

- Maintenance of the library and removal of technological debt
- Improvements to the new fast gradient boosting models
- Faster parameter searches, avoiding redundant computations

1.1 Maintenance of the library

As one of the most widely used machine learning libraries, scikit-learn has a tremendously large code-base, of which some parts are many years old. New contributions are submitted each day, along with many bug reports or feature requests. We propose to fund both Müller and Hug to ensure the continued improvement of the library, and to ensure the project evolves to meet the growing needs of the scientific community. Maintaining scikit-learn consists of:

a) Addressing bug reports, prioritizing bug fixes, and ensuring important bugs are fixed in a timely manner.

b) Reviewing new feature contributions and making sure they meet the high quality criteria of the project, in particular regarding code maintainability and user documentation.

c) Supporting and connecting with the community: this includes maintaining a high-quality documentation, reaching out to users to advertise new useful features, and decide what directions to take according to community needs. It also involves organizing sprints to engage the community and increase diversity among the contributors.

[1] https://github.com/cggh/scikit-allel
[2] https://github.com/spacetx/starfish

d) Making sure backward compatibility is preserved between versions, and that new features are consistent with the needs of the library. This is important for the health of the project and for the numerous downstream projects that rely on scikit-learn.

e) Reducing the current backlog. There are many issues and pull-requests that get stalled for various reasons. Reviewing these pull requests and closing irrelevant ones helps contributors and developers to focus on important matters.

These tasks require experience and a consistent involvement in the project to be carried out efficiently, making the team of Müller and Hug particularly well suited for this task.

1.2 Improvements to Gradient Boosting

Gradient Boosting Decision Trees (GBDTs) are a family of machine learning models used for classification and regression tasks. GBDTs are extremely efficient and often out-perform other models. They are used pervasively in machine learning, including biomedical applications [CWP19, SKB+19].

Recent implementations like LightGBM [KMF+17] and XGBoost [CG16] offer state-of-the-art results, both in terms of prediction accuracy and computation time. However, directly including an implementation in scikit-learn ensures compatibility with the core package, and prevents inconsistencies in the API. Moreover, due to scikit-learn's popularity, distributing an implementation with scikit-learn will increase the visibility and adoption of these highly effective algorithms by non specialists.

A new implementation of GBDTs (by co-PI Hug) was recently released in scikit-learn version 0.21. This new implementation outperforms XGBoost in terms of speed (while matching it's accuracy), and its results (in terms of speed and accuracy) are on a par with LightGBM. While the current implementation is operational, several important features are not implemented yet. Therefore we propose the following directions related to this new GBDT implementation:

a) Implement native support for missing values. In real-world datasets, feature values might be missing due to measurement errors or incomplete measurements. In these cases, practitioners need to resort to imputation, which is often sub-optimal and not theoretically grounded [JPSV19]. Unlike most machine learning models, GBDTs are able to natively support missing data, without resorting to imputation of missing values. This makes them extremely handy to use, in particular for non-specialists. Both LightGBM and XGBoost natively support missing values.

b) Implement native support for categorical features. Categorical features are pervasive in biomedical science. For example sex, age, race, or geographic location are potentially relevant for predicting the cancer risk of a patient [RK19], and could be encoded as categorical variables. The current implementation of GBDTs only supports continuous features, but GBDTs are also able to handle categorical data. For now, users are required to preprocess categorical features using techniques like one-hot-encoding, which is often tedious and can lead to worse results. GBDTs have a native, more efficient way of dealing with categorical variables. Implementing this would free practitioners from having to worry about encoding categorical variables. This feature is implemented in LightGBM, but not in XGBoost.

c) Implement support for sample weights and class weights, which allow the user to specify the confidence they have in their measurements and deal with imbalanced datasets.

d) Document and explain the usage of the new implementation, with an emphasis on missing value support and support for categorical variables. Create detailed examples of typical use-cases.

e) Maintain the code, and fix the potential bugs. The current implementation already has about 5000 lines of Python (and Cython) code. Despite great efforts to provide a thorough test suite, some bugs or numerical instability issues may still be present, which is expected for a code base of that size and complexity.

Please also note that achieving a) and b) would make these models the first ones in scikit-learn to natively support missing data and categorical data. This will demand some careful design steps, and will increase the number of potential bugs and pitfalls. In addition, the new GBDT implementation is our first use of low-level parallelism (using OpenMP). This enables us to build extremely efficient estimators, but also adds new technical complexity that needs to be managed.

1.3 Faster parameter searches avoiding redundant computations

Supervised machine learning workflows typically consist of one or multiple preprocessing steps, and a final supervised model. The preprocessing steps, as well as the final model, commonly have hyper-parameters that need to be tuned for the overall procedure to perform well on a specific task: this is called a hyper-parameter tuning. Scikit-learn provides tools for hyper-parameter tuning: namely grid search (exhaustively try parameter combinations along a grid) and random search (parameter combinations are sampled at random).

Performing parameter searches in machine learning workflows often involves redundant computation, and previously computed results can potentially be re-used. As illustrated on Figure 1, when parameters of the final model change, it is unnecessary to recompute the previous (preprocessing) steps.

Currently scikit-learn unnecessarily performs redundant computations, wasting computation time.

Leveraging dask features, in particular dasks graph execution engine, the dask-ml package now provides equivalent implementations of parameter search logic that are able to re-use previously computed steps. This implementation can be much faster than the implementation available in scikit-learn. However, this package is not as popular as the scikit-learn library, and as a result it is much less likely to be widely used, despite its usefulness. It also relies on dask, adding a dependency and increasing maintenance burden. Having such a utility in scikit-learn would allow non-specialists to train complex pipelines much more efficiently.

We propose to implement parameter search utilities without re-computation within scikit-learn, taking advantage of experience from the dask-ml implementation. This is a complex task that may require intrusive changes to scikit-learn core utilities.

1.4 Expected outcomes, success evaluation and metrics

For open source projects, measuring impact is notoriously hard, as library authors don't have access to telemetry. Download counts are often spread over several distribution channels and highly skewed by automated downloads. The impact of specific additions to open source projects is even harder to measure, in particular since adoption of new features can only be measured with significant delay. Acknowledging this, we are suggesting proxy metrics that allow at least some quantitative evaluation of the work.

1.4.1 General maintenance

We consider number of solved issues and fixed bugs, and number of reviewed and merged pull requests, as well as the trend of open issue and pull requests. Even though the amount of time required to review a pull request varies from a few minutes to many months, depending on the activity of the contributor and the complexity of the task, we hope this metric provides some insight into project health.

We expect to have around 10 easy pull requests reviewed per month, and at least 10 complex pull requests reviewed (and merged) in the span of 12 months. We also expect simple and/or critical bugs to be fixed within a few weeks, that is before the next bug-fix release.

1.4.2 New features for GBDT models

For the GBDT models, our primary goal is merging the proposed features (in particular support for missing values and categorical data) into scikit-learn, and inclusion into a release of the package. We also expect our additions to increase usage of the HistGradientBoosting models, as can be measured by code search on Github. Currently a search for Jupyter Notebook returns about 50 hits, we expect this to rise to over 200.

1.4.3 Faster parameter searches

Given the complexity of the task, such an implementation would require some careful design steps. There are indeed many different approaches to solving this problem, each of which has its own technical constraints. As a result, clearly defining and assessing the pros and cons of each approach would already be an accomplishment and would serve as groundwork for future directions.

Our goal would be successfully met here if we can reach a consensus on implementation path. A release of the implementation would be a stretch goal.

2 Work plan

The funds would be used to:

- Fund Andreas Müller's position for 4.5 months. His long-term experience with the project is essential for evaluating changes in API design and dependencies that are necessary for the planned work.
- Fund Nicolas Hug's position for 12 months. Hug has been involved in maintaining scikit-learn for over a year, and has been a core developer for 6 months. He has contributed key features to the package, in particular the new GBDT implementation mentioned above.
- Fund participations in some key Python and scientific conferences, such as the SciPy conference [3].
- Fund participation and organization of sprints, in particular the WiMLDS sprints. These sprints aim at engaging more people from under-represented groups and encourage them to participate in the development of scikit-learn. During sprints, attendees usually address some easy issues, typically documentation or code cleaning. Despite their apparent simplicity, these contributions are tremendously useful for the general health of the project and provide an easy entry point for new developers to engage with the open source community.

[3] https://conference.scipy.org

We believe that mentoring other contributors is very important for the health of the project: the more contributors are familiar with the code base, the healthier the project. Therefore, a significant part of the effort in implementing the proposed features will go towards reviewing and mentoring other contributors, instead of implementing all of the features ourselves.

All of the proposed features in this proposal are part of the current project roadmap[4].

3 Existing support

Andreas Müller is a PI on the following grants related to scikit-learn:

- *Building blocks and Search Improvements for Automated Machine Learning Model Selection.* DARPA. $351k. 2018.
- SI2-SSE: *Improving Scikit-learn usability and automation.* NSF. $400k. 2017-2020.
- *Extension and Maintenance of Scikit-learn.* Alfred P. Sloan Foundation. $313k. 2017-2019.

The scikit-learn foundation at INRIA Paris [5] currently funds four full time employees working on scikit-learn and related projects:

- Olivier Grisel, technical lead.
- Jérémie du Boisberranger, research engineer.
- Guillaume Lemaître, research engineer.
- Chiara Marmo, community and operation manager (starting September 2019).

The University of Sydney funds Joel Nothman to work part-time on scikit-learn. Anaconda[6] funds Adrin Jalali as a full time employee to work on scikit-learn and related projects.

[4] https://scikit-learn.org/stable/roadmap.html
[5] https://scikit-learn.fondation-inria.fr/
[6] https://www.anaconda.com/

Figures

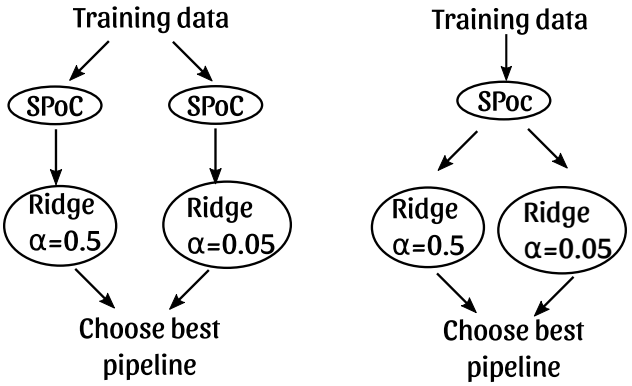

Figure 1: Parameter search for a pipeline predicting electromyography signal from magnetoencephalography activity. Inspired by the MNE package[GLL+14].
Left: Current parameter search in scikit-learn. The pre-processing SPoC step is repeated for each of the two pipelines, even though it never changes.
Right: Proposed implementation. The SPoc step is only computed once.

References

[APE+14] Alexandre Abraham, Fabian Pedregosa, Michael Eickenberg, Philippe Gervais, Andreas Mueller, Jean Kossaifi, Alexandre Gramfort, Bertrand Thirion, and Gael Varoquaux. Machine learning for neuroimaging with scikit-learn. *Frontiers in Neuroinformatics*, 8:14, 2014.

[BCKL16] Matthew D Blackledge, David J Collins, Dow-Mu Koh, and Martin O Leach. Rapid development of image analysis research tools: bridging the gap between researcher and clinician with pyosirix. *Computers in biology and medicine*, 69:203–212, 2016.

[BDB+18] Nicholas A. Bokulich, Matthew Dillon, Evan Bolyen, Benjamin D. Kaehler, Gavin A. Huttley, and J. Gregory Caporaso. q2-sample-classifier: machine-learning tools for microbiome classification and regression. *bioRxiv*, 2018.

[BKR+18] Nicholas A. Bokulich, Benjamin D. Kaehler, Jai Ram Rideout, Matthew Dillon, Evan Bolyen, Rob Knight, Gavin A. Huttley, and J. Gregory Caporaso. Optimizing taxonomic classification of marker-gene amplicon sequences with qiime 2's q2-feature-classifier plugin. *Microbiome*, 6(1):90, May 2018.

[BRD+18] Evan Bolyen, Jai Ram Rideout, Matthew R Dillon, Nicholas A Bokulich, Christian Abnet, Gabriel A Al-Ghalith, Harriet Alexander, Eric J Alm, Manimozhiyan Arumugam, Francesco Asnicar, Yang Bai, Jordan E Bisanz, Kyle Bittinger, Asker Brejnrod, Colin J Brislawn, C Titus Brown, Benjamin J Callahan, Andrs Mauricio Caraballo-Rodrguez, John Chase, Emily Cope, Ricardo Da Silva, Pieter C Dorrestein, Gavin M Douglas, Daniel M Durall, Claire Duvallet, Christian F Edwardson, Madeleine Ernst, Mehrbod Estaki, Jennifer Fouquier, Julia M Gauglitz, Deanna L Gibson, Antonio Gonzalez, Kestrel Gorlick, Jiarong Guo, Benjamin Hillmann, Susan Holmes, Hannes Holste, Curtis Huttenhower, Gavin Huttley, Stefan Janssen, Alan K Jarmusch, Lingjing Jiang, Benjamin Kaehler, Kyo Bin Kang, Christopher R Keefe, Paul Keim, Scott T Kelley, Dan Knights, Irina Koester, Tomasz Kosciolek, Jorden Kreps, Morgan GI Langille, Joslynn Lee, Ruth Ley, Yong-Xin Liu, Erikka Loftfield, Catherine Lozupone, Massoud Maher, Clarisse Marotz, Bryan D Martin, Daniel McDonald, Lauren J McIver, Alexey V Melnik, Jessica L Metcalf, Sydney C Morgan, Jamie Morton, Ahmad Turan Naimey, Jose A Navas-Molina, Louis Felix Nothias, Stephanie B Orchanian, Talima Pearson, Samuel L Peoples, Daniel Petras, Mary Lai Preuss, Elmar Pruesse, Lasse Buur Rasmussen, Adam Rivers, Michael S Robeson, II, Patrick Rosenthal, Nicola Segata, Michael Shaffer, Arron Shiffer, Rashmi Sinha, Se Jin Song, John R Spear, Austin D Swafford, Luke R Thompson, Pedro J Torres, Pauline Trinh, Anupriya Tripathi, Peter J Turnbaugh, Sabah Ul-Hasan, Justin JJ van der Hooft, Fernando Vargas, Yoshiki Vzquez-Baeza, Emily Vogtmann, Max von Hippel, William Walters, Yunhu Wan, Mingxun Wang, Jonathan Warren, Kyle C Weber, Chase HD Williamson, Amy D Willis, Zhenjiang Zech Xu, Jesse R Zaneveld, Yilong Zhang, Qiyun Zhu, Rob Knight, and J Gregory Caporaso. Qiime 2: Reproducible, interactive, scalable, and extensible microbiome data science. *PeerJ Preprints*, 6:e27295v2, December 2018.

[BS15] Jari Björne and Tapio Salakoski. Tees 2.2: biomedical event extraction for diverse corpora. *BMC bioinformatics*, 16(16):S4, 2015.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[CWP19] Xiang Chen, Zhi-Xin Wang, and Xian-Ming Pan. Hiv-1 tropism prediction by the xgboost and hmm methods. *Scientific Reports*, 9(1):9997, 2019.

[Czo13] Paul Czodrowski. herg me out. *Journal of chemical information and modeling*, 53(9):2240–2251, 2013.

[GLL+14] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Lauri Parkkonen, and Matti S. Hmlinen. Mne software for processing meg and eeg data. *NeuroImage*, 86:446 – 460, 2014.

[HSH+17] Matthew P Harrigan, Mohammad M Sultan, Carlos X Hernández, Brooke E Husic, Peter Eastman, Christian R Schwantes, Kyle A Beauchamp, Robert T McGibbon, and Vijay S Pande. Msmbuilder: statistical models for biomolecular dynamics. *Biophysical journal*, 112(1):10–15, 2017.

[HSP+16] Gerrit Hilgen, Martino Sorbaro, Sahar Pirmoradian, Jens-Oliver Muthmann, Ibolya E. Kepiro, Simona Ullo, Cesar Juarez Ramirez, Albert Puente Encinas, Alessandro Maccione, Luca Berdondini, Vittorio Murino, Diego Sona, Francesca Cella Zanacchi, Evelyne Sernagor, and Matthias H. Hennig. Unsupervised spike sorting for large scale, high density multielectrode arrays. *bioRxiv*, 2016.

[JB18] Vinay Jayaram and Alexandre Barachant. Moabb: trustworthy algorithm benchmarking for bcis. *Journal of neural engineering*, 15(6):066011, 2018.

[JPSV19] Julie Josse, Nicolas Prost, Erwan Scornet, and Gaël Varoquaux. On the consistency of supervised learning with missing values. *arXiv preprint arXiv:1902.06931*, 2019.

[KBFG18] Konrad Paul Kording, A Benjamin, Roozbeh Farhoodi, and Joshua I Glaser. The roles of machine learning in biomedical science. In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2017 Symposium*. National Academies Press, 2018.

[KMF+17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

[NIKP+14] Juan Nunez-Iglesias, Ryan Kennedy, Stephen M Plaza, Anirban Chakraborty, and William T Katz. Graph-based active learning of agglomeration (gala): a python library to segment 2d and 3d neuroimages. *Frontiers in neuroinformatics*, 8:34, 2014.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RK19] Aaron N. Richter and Taghi M. Khoshgoftaar. Efficient learning from big data for cancer risk modeling: A case study with melanoma. *Computers in Biology and Medicine*, 110:29 – 39, 2019.

[SKB+19] Akash A Shah, Aditya V Karhade, Christopher M Bono, Mitchel B Harris, Sandra B Nelson, and Joseph H Schwab. Development of a machine learning algorithm for prediction of failure of nonoperative management in spinal epidural abscess. *The Spine Journal*, 2019.

[VDH+15] Bastian Venthur, Sven Dähne, Johannes Höhne, Hendrik Heller, and Benjamin Blankertz. Wyrm: A brain-computer interface toolbox in python. *Neuroinformatics*, 13(4):471–486, 2015.