## Project Description

Scikit-learn is one of the most widely used machine-learning software packages in Python, which has emerged as the programming language of choice for AI and data science. It supports a wide range of algorithms, making it suitable for complex applications that integrate data cleaning, feature extraction, and prediction.

This project enhances scikit-learn for distributed deployments so that scikit-learn captures the entire lifecycle of a machine learning algorithm, from rapid prototyping and exploratory data analysis all the way to distributed deployments in cyber-physical systems or as Internet services. Currently, scikit-learn has limited scalability to massive data sets and lacks advanced features that support production deployments.

The  Sustain efforts will improve project management and governance. It will put in place more efficient, open, and fair processes for contributing and reviewing software contributions. These processes will be used to resolve existing technical debt of open issues and pull requests from the community's more than 1000 contributors and certify a stable release. Streamlined processes will be used to propose enhancements to the community.

scikit-learn is a community open source project with a formal governance and decision making process. As such, all contributions to the project must be reviewed and approved by the community. While our prominent position in the community puts us in an ideal position to contribute to the project, it does not guarantee that all proposed enhancements will be accepted. Features that are under review or not included will be provided through the scikit-learn-contrib organization, which contains third-party extensions to scikit-learn.

### Enhance: Ensemble Learning Deployments

The project will enhance tree-based ensemble learning of sklearn to be suitable for distributed deployments and optimized for inference latency. This includes straightforward throughput optimizations through parallelism for multicore. It also includes extensions using external memory algorithms to reduce latency for interactive services. We extend the external memory design to focus on two model deployments: (1) cloud microservices, and (2) Internet of Things (IoT). In both cases, one or few inputs are sent to a model that is not already loaded in the memory of a computing resource. The challenge is to minimally load parts of the model need to evaluate the input.

Many machine learning algorithms and libraries are designed for and used offline with batch processing of samples, do not require model storage, and are not optimized for computational efficiency. This is consistent with exploratory analysis; users train and test models on target data sets to determine its error properties, such as precision and recall. Notably, it is also the usage pattern needed for machine learning competitions, including Kaggle, in which contestants submit algorithms that run against training and test data sets and win typical based on precision/recall, F1 score, or other model performance metrics.

Deploying algorithms as part of AI systems has different requirements, specifically computational performance in terms of latency and throughput and the ability to persist, distribute and load trained models efficiently. Parallelism and throughput correspond directly with operational cost and deep learning frameworks have focused on these issues [1,2], [3]. The deep learning revolution has been tightly integrated with systems research, owing to the need to bring a large amount of computation to bear through GPUs [4]. Deep learning has also embraced latency requirements [5] in support of real-time applications, such as speech recognition and vision systems for driverless cars. Latency, specifically the

time to inference on a single sample, has been largely ignored in tree ensembles. XGBoost [6] and LightGBM [1] are gradient boosted forests for supervised learning that address issues of parallelism and model storage, but they operate in the batch model and are not optimized for latency.

Enhancement targets two important deployment scenarios that have common properties; they are distributed and they need to load models on demand to perform inference. The first provides inference as a cloud microservice on serverless compute, such as Amazon lambda. On a request, the cloud provisions a compute resource that loads the model, performs inference, and then disappears. Serverless compute is cheap, because you don't pay for resources when not in use, and scalable. It is the preferred way to deploy apps that are bursty or used sporadically. The second scenario deploys on IoT devices. Tree ensembles are particularly attractive because inference uses little power or compute when compared with deep networks. Because IoT devices have limited resources, forests are too big to fit in memory. Instead, they must be stored on non-volatile flash and loaded on demand.

The critical capability gap is a library of deployment ready ensemble learning algorithms that are well integrated into a rich data-science ecosystem. scikit-learn and related projects, such as NumPy, SciPy, scikit-image, matplotlib, are that ecosystem. This project will fill the capability gap by modernizing scikit-learn for parallelism, interactive use, and distributed deployments. The methods that we will develop will apply to all tree-based ensemble learning algorithms in scikit-learn, including random forest, gradient boosted trees, and their variants, such as sparse projection forests, extremely random trees, tree embedding, and outlier detection.

With the combination of a rich development ecosystem and performance optimizations, users can scale their models in data size and deployment scope from casual and exploratory use to distributed systems in the cloud and IoT without reimplementation. scikit-learn is a perfect environment for exploratory machine learning, it has a low barrier to entry and a wide variety of algorithms. The ecosystems makes it so that users can easily piece together complex pipelines of different algorithms, swap methods in and out, and analyze and visualize results. This project's  Enhance work adds the parallelism and performance optimizations that translates the exploratory models to scale for distributed deployments and parallel hardware. This avoids the tedious process of reimplementing code with different tools for performance and scale; our labs go through this process regularly, e.g. prototyping in sklearn and porting to optimized implementations of k-means [7] and random forests [8].

## Sustain: Future-proofing the scikit-learn community and development procedures

The scikit-learn project is a Python library for machine learning, including algorithm, metrics and model evaluation tools. It is a community driven open source project, with a broad user base and a wide reaching community of developers, including more than 76 researchers from at least 38 US institutions. Widespread use and a plurality of contributors from different backgrounds lead to large demands on project management, coordination, and support, none of which have been tackled systematically. In the last two years, the project has created a formal governance structure and established a roadmap. However, these did not immediately translate to improvements in project coordination and a sustainable project structure. Currently, the project has 1367 open issues and 687 open pull requests, many of them several years old.

To improve sustainability over the long term as a community open source project, we propose implementing several improvements to the project structure, as well as directly decreasing technical debt. A metric for project health that has been proposed by Eghbal [9] is the difference between opened and closed issues over time. We aim to triage existing issues to systematically improve maintainability. In collaboration with the rest of the community, we will create a maintenance plan to ensure the number of

issues and pull requests stays manageable, and issues and pull requests are answered in a timely manner.

## Intellectual Merit

As a CCRI enhance and sustain project, the key measure for intellectual merit is the CISE research enabled by this project and the new research opportunities created by the project's work and outreach.

The ease of use of scikit-learn and the availability of high-quality implementations of many popular machine learning algorithms has facilitated the use of machine learning across many CISE disciplines. Scikit-learn is used regularly and prominently in papers at top machine learning conferences. Scikit-learn is cited in 89 papers published at the highly competitive NeurIPS conference and in 71 papers published in the ICML conference or the JMLR journal, though usage is likely to be more pervasive than indicated by citations. More specifically, scikit-learn is central to research in automatic machine learning (AutoML) because it provides a consistent interface for many algorithms and a systematic way of composing multiple models. Most state-of-the-art research in AutoML relies on scikit-learn to provide the building blocks which are optimized or searched over [10–17]. In the field of cyber-security, several subfields have adopted machine learning methods. Scikit-learn has been widely used for this research, in particular for intrusion detection [18–24] and malware detection [25–32].

Going forward, the project will ease the integration of machine learning into other CISE disciplines for NSF initiatives. Specifically, the sustain tasks lower the barriers to entry for algorithm contributions, which will allow *Algorithmic Foundations* to be more easily incorporated. Reducing latency in  Enhance mission will support its use in human-computer interaction and *Cyber-Physical Systems*. The variety of algorithms and uniform interface makes scikit-learn critical in AutoML research in support of *Robust Intelligence*.

## Infrastructure Description

**ID.1 Existing Infrastructure:** The scikit-learn machine learning library [33] provides machine learning functionality within the established and growing scientific Python ecosystem. The scikit-learn project is an open source project for 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 [34].

**Interface:** scikit-learn provides a generic interface for machine learning, mainly consisting of only three methods: (1) fit, to build models, (2) predict, to make predictions using models, and (3) transform, to change the representation of the input data [35]. 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, minimizing the friction of applying machine learning within an existing project.

**Comprehensiveness:** scikit-learn implements a wide variety of models for classification, regression, clustering, dimensionality reduction, and model selection, as well as methods for feature scaling, transformation, normalization, selection, and extraction. The library contains most of the algorithms included in standard textbooks [36,37], while providing competitive implementation of state-of-the-art algorithms like Gradient Boosting [1,38] and Random Forests [39]. 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. 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.

The scikit-learn library has been widely adopted for machine learning in both scientific research and industry applications. It is a standard component of machine learning courses, and instructional materials are available in the form of documentation, online courses and several textbooks.

## ID.2 Plans for enhancement: Ensemble Learning Deployments

**Implement Parallel Execution:** We will parallelize the execution of both training and inference for tree ensembles in scikit-learn with the specific focus on *memory parallelism,* because tree traversals in random forests are memory (I/O) bound. Tree ensembles would seem to be “pleasantly” parallel in shared and distributed memory. Assigning each tree in a forest to a different core/processor gives each core/processor independent computation and data. Indeed, random forests have had parallel implementations since their inception [39,40]. In practice, memory contention limits the scalability of random forests and only a small fraction of the potential parallelism is realized. In most processors, many cores share a single memory bus and just a few cores saturate the bus with requests. Traversals perform many small random I/Os, one for each level of the tree. Batch operations improve throughput and are a critical optimization in state of the art implementations [1,6]. Batches of requests traverse multiple tree paths in the same node at once. If a batch is large enough, it will access the entire tree and can access all data sequentially, which maximizes throughput. For inference, batches increase latency at the expense of throughput.

*Task E.1:* Improve memory parallelism for tree ensembles. This task will enhance the existing primitive job/task parallelism support in scikit-learn.

**Optimize Tree Formats for Predicting:** Forest Packing, our recent result, drastically increases performance by packing multiple trees together into memory so that accessing a single cache line traverses paths in multiple trees concurrently [41]. At the top levels of the tree, we stripe the nodes of each level across many trees into one or few cache lines. We call the striped trees a *bin.* Figure 1 shows the layout across a bin of 4 trees (A,B,C,D). If 16 nodes fits in a cache line, one memory access results in 16 traversals at the root nodes, 8 in the second level, etc. The lower levels of the trees are packed statistically so that the most likely paths are sequential in memory and, thus, in a single cache line or spanning two cache lines. When evaluating trees, we evaluate a single level in all trees in at the same time and emit prefetch instructions to the next level as soon as possible. This maximizes independent work and allows for out-of-order execution by the processor.

Ours is the first system to make memory access more dense and coherent by optimizing across trees, whereas previous work has optimized within each tree. Treelite [42] and compiled trees [43,44] organizes individual trees based on branch prediction, similar to our statistical layout. VPred [45] adds out-of-order execution to batches. XGBoost [6] makes data structures smaller through sampling and discretization. LightGBM [1] bundles features and prunes small gradients. Many of these optimizations can be used together and we adopt best practices from all related work so long as they preserve inference accuracy. Forest Packing minimizes latency over unpacked forests by a factor of 4 or more on all inputs. It is five times faster than XGBoost and LightGBM on small batches and equally fast at any batch size.

*Task E.2:* Implement packed forests for gradient boosted trees and random forests in scikit-learn.

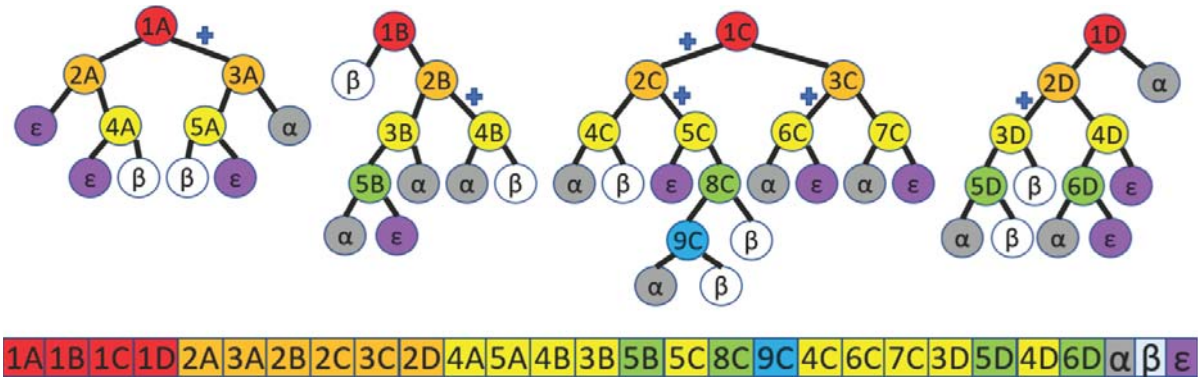



*Figure 1:* In-memory layout of a packed forest. The top two levels of the trees are interleaved. Residual paths are stored by frequency (e.g, 5C, 8C, 9C). + indicates a more popular path. Class labels (α, β, ε) are stored once for all trees.

Dynamically optimizing tree-layouts for forest packing avoids a post-processing step and will accelerate training time. Currently, forest packing and other layout optimizations [42,44], reorganize the forest after training. It is not expensive when compared with training large forest, but it is an extra step. We observe that the top levels of a packed forest are statically determined to conform with the cache line size. Training a packed bin of trees concurrently makes memory access sequential and coherent and supports out-of-order execution and prefetching, as it does during inference. This can improve performance by a lot or a little, depending on tree sizes. We see a factor of 2 improvement on typical data sets in preliminary work. The challenge lies in the statistical layout at the bottom of trees. We must defer the placement of nodes in memory until after training. We do this by storing leaf nodes in priority queues during training. All nodes have back pointers to recover the path from the leaf. We output paths by leaf priority to produce a statistical layout at training time.

*Task E.3:* Dynamically pack forests during training to improve training time and avoid post-processing.

**Serialization and Deserialization:** We now turn to storing (serialization), transferring, and loading models (deserialization), which is the central enhancement for distributed deployments. The previous tasks address the efficiency of training and inference when models are already in memory. For cloud microservices and IoT deployments, we wish to train the model on a data set, store the model in a compact and portable fashion, transfer the stored model over networks, and load the model on demand to perform inference.

scikit-learn has primitive support for storing and loading models that is functional, but not efficient. scikit-learn uses the python “pickling”: pickle() is a generic capability for all python objects that converts the object into a byte stream and unpickle() loads an object from a byte-stream. Because pickle is generic, the output is large. Because output is large and execution is serial, it is slow. We have prototyped a serialization capability that use semantic knowledge to minimize the stored data. The output stores no pointers, they are stored implicitly and reconstructed during deserialization. The process is parallel, serializing each tree independently into its own byte stream and writing the output sequentially. Deserialization reads the input sequentially and processes the reconstruction of trees in parallel.

*Task E.4:* Implement parallel serialization/deserialization in a compact format.

**I/O Optimized Deserialization:** For cloud services and IoT deployments, the model must be loaded into memory and the load time determines latency. IoT devices have minimal resources and cannot store the

entire model in memory. For cloud microservices, a compute instance is created on demand to serve one or a few requests and a model must be deserialized from remote storage. In both cases, we expect to transfer data from a solid-state storage device into memory on demand.

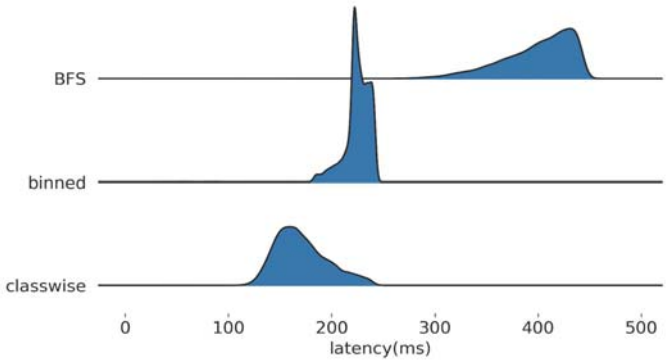



*Figure 2:* Inference latency for classwise layout compared with binned layout (no clases) and breadth-first search.

We observe that inference traverses select paths in the forest and that the combination of selective access and forest packing principles can reduce latency. Selective access means that only the parts of the model needed for classification are loaded from storage. Forest packing principles can reduce the number of blocks needed by selective access. As described earlier, forest packing minimized the number of 128-byte cache line loads. This task should minimize the number of 4KB-256KB block access; SSDs storage typically has a 4KB block size, but parallel access channels requires larger accesses to use hardware efficiently. This requires a very different design. We have developed an early prototype in which we stripe/bin many trees, up to 1024, in a single block and store the top 4 levels of the hierarchy. To isolate access to blocks in the lower parts of the tree, we partition paths based on the class they encode, e.g. in an image recognition task all paths that infer cats are placed in one block and all paths that infer dogs in another. While we have just started this investigation, early results indicate a 2.5 times reduction in latency versus an equally compact format that stores each tree independently (Figure 2) and that partitioning the forest ‘classwise’ improves latency by 33% with respect to a packed forest that does not organize by class (‘binned’).

*Task E.5:* Implement a selective-access, block-packed serialization format for tree ensembles

Standard Formats: At present, the machine learning community has not converged on interchange formats for random forests and gradient boosted trees. scikit-learn depends on python pickle, which serializes arbitrary python objects and, thus, restricts it to python. LightGBM and xgboost have their own custom formats. This is in contrast to deep learning, which has widely-used standards, notably the Open Neural Network Exchange (ONNX) format. There are ONNX modules that load and store tree ensembles, but they are not widely used or well supported. Additionally, none of scikit-learn, xgboost, or LightGBM operate on these formats directly. It is an import/export process. A preliminary evaluation indicates that we could store models in ONNX format and support selective access, but the ultimate format will need to be chosen in a community process.

*Task E.6: Work with the machine learning community to define standard formats for model storage of ensemble learning methods, including random forests and gradient boosted trees.*

## ID.2 Sustain: Future-proofing the scikit-learn community and development procedures

**Establish sustainable maintenance procedure:** While the scikit-learn project has formal governance structure, it does not have a formal structure for performing maintenance tasks. Maintenance tasks, such as replying to issues of fixing bugs are often performed on an ad-hoc basis. Given the scale of the project, this often means that issues are not replied to (even for months of years), contributed code is not reviewed, or re-reviewed after modifications are made, and initial progress on bug fixes is sometimes abandoned. The result is an expanding list of issues and pull-requests, reaching back many years, and

consistently growing. To ensure the long-term feasibility of the project, a more principled approach for addressing maintenance tasks is required, including reviewing and issue triage.

We will work together with the rest of the scikit-learn community to develop and test a long-term maintenance procedure. Our first approach will be having a rotating schedule of a dedicated engagement person that will perform maintenance tasks. However, given the complexity of the project and the distributed organizational structure, we expect that a significant amount of time needs to be spent trying and evaluating different maintenance mechanisms. A successful mechanism will ensure a decrease of open issues and pull requests, as well as a decreased response time for new issues and pull requests.

*Task S.1: Establish a formal maintenance procedure for issue triage and reviewing in collaboration with the scikit-learn community.*

**Releasing Version 1.0**: The current version of scikit-learn is version 0.22.1. Despite the use of semantic versioning, the scikit-learn project has not released a version 1.0, which is usually the first version considered stable. Given the widespread use and maturity of the project, that has a series of negative consequences. Not having a version 1.0 means that some potential users will not trust in the stability or maturity of the project. On the other hand, not having a version 1.0 means that no release is allowed to make breaking changes (as all releases are “minor”), which somewhat slows down the growth of the project. Releasing version 1.0 represents a reference in terms of implementation and interface that is designated as being reliable and supported long-term, and that on the other hand can be changed deliberately by releasing a new major version 2.0 to introduce major innovations.

So far the community has been reluctant to release version 1.0, or even to define the scope of releasing a version 1.0. This grant will provide dedicated time to clearly plan the criteria for the release, and then systematically prioritize the required changes. Releasing 1.0 is a singular event, in that future versions will have a precedence for the requirements of a major version, and future releases will be possible without additional major support.

*Task S.2: Define requirements and scope for releasing version 1.0 and complete these requirements.*

**Issue triage and resolution**: While Task S.1 defines a procedure to ensure long-term maintenance of the project, there is also a need to directly resolve the current backlog. There are currently 1,358 open issues and 685 open pull requests. Establishing a more deliberate and sustainable maintenance procedure alone will likely not be enough to resolve the baggage that accumulated over the last 10 years of the project being active. Therefore we will spend dedicated effort on decreasing the amount of open issues and pull-requests directly.

*Task S.3: Triage and resolve existing issues, review and merge or close existing pull requests.*

**Downstream Developer Utilities and Documentation:** Scikit-learn provides the building blocks of many extensions and domain-specific machine learning tools and applications, see the scikit-learn-contrib github organization for a small selection. While there is documentation for developing scikit-learn compatible models that can integrate with the scikit-learn tooling, some aspects of the API are under-documented. In particular, the definition for being scikit-learn compatible is given by the implementation of the highly complex `check_estimator` function that is provided. The `check_estimator` function validates any scikit-learn-like machine learning model, and certifies the compatibility with the scikit-learn API. On the one hand, providing automatic testing to third-party developers is highly convenient. On the other hand, passing the test can sometimes be hard, and the reasons for failures are not always obvious.

There are many utility functions within scikit-learn that help fulfill the API contract, however their usage is often unclear, or might change drastically between versions because they are considered private.

We will define the exact API contract of scikit-learn in the documentation in a human readable format and clearly lay out the requirements made by the current API. This will enable third-party developers to adhere to the scikit-learn interface much more easily and debug incompatibilities more readily. We will also provide more in-depth documentation and examples of the existing developer utilities and how to use them to build models that extend scikit-learn. These developer utilities will be part of a *developers API*, which will be maintained and supported with requirements similar to those of the *user API*.

*Task S.4 Document the full scikit-learn API specification and provide clear guidance on how and when to use developer utilities.*

**Automatic optimization defaults:** As the project has grown, many standard machine learning models, such as logistic regression, have evolved to incorporate recent advances in machine learning research, such as the SAG and SAGA solvers [46]. However, this growth can lead to a complex user-interface that lacks transparency for new users, in particular, when it comes to different strategies for optimizing the same model. Logistic regression in scikit-learn currently supports 5 solvers in addition to path algorithms for 4 of these solvers. Choosing the right solver for a given problem is often a complex task for inexperienced users. The goal of automating solver choices is to find the one that reaches a numerically equivalent solution as fast as possible. Note that this problem is distinct from the problem of hyper-parameter optimization for the purpose of improving model accuracy.

For each model, a wealth of optimization methods have been published in the literature. Implementing the model requires to pick the best optimization method for a particular data set given by the user. To systematically address this problem, a framework for simplifying the selection of optimization algorithms and options is required. We will create a well-defined benchmark-suite, together with protocols for systematic evaluation of optimization algorithms and parameters to improve usability and performance. While the problem is not restricted to supervised models, those are the most prominent and widely used algorithms in scikit-learn, and we will focus on these. As a test-case, we will use our benchmark-suite and protocols to define an automatic strategy to choose solvers for logistic regression and linear regression.

*Task S.5 Create a benchmark for choosing solvers for classification algorithms and use it to automate the solver choices in logistic regression and linear regression.*

**Inclusion criteria for new implementations:** Scikit-learn has a well-defined set of inclusion criteria for new machine learning models. The package only accepts models that have been proven to be widely useful, with a strict threshold for citation count and publication date.

Similarly to a criteria for inclusion of new *models*, we will define a set of criteria for inclusion of new *implementations* of existing models. Such new implementations may, for example, greatly improve memory usage while slightly degrading computation time, preserving the same accuracy.

For huge code bases like that of scikit-learn, the inclusion of a new component is never a trivial decision since many aspects must be considered, including user satisfaction but also complexity and cost of maintenance. The criteria that we will define will be useful not only for scikit-learn but also for other packages, as it will serve as a standard for such decisions. These inclusion criteria will also steer the development in the Enhance part of this project, as it will provide guidelines for what changes have a high chance of inclusion with scikit-learn.

*Task S.6 Define a clear criteria for inclusion for new implementations of existing models.*

**ID.3 Tools, resources and data sets:** Scikit-learn is an open source machine learning library written for use in the Python language. It provides robust and efficient implementations of many standard machine learning algorithms, including classification, regression, clustering, dimensionality reduction, outlier detection and others, as well as tools for preprocessing, model evaluation and model selection.
Given that scikit-learn is a library, it is most commonly used by writing Python scripts or programs that use the library, or by interactive use in a Jupyter Notebook; in other words, there is no standard graphical user interface, and programming experience in Python is required.

Examples for the various capabilities of scikit-learn are models like Random Forests and Support Vector machines, stratified cross-validation for model evaluation, metrics like average precision or area under the ROC curve for model comparison, and randomized search for adjusting hyper-parameters.

**ID.4 User services:** Scikit-learn is a community-driven project, meaning that services currently are provided by the community of users and developers. These services include workshops and tutorials on using the software, such as the annual workshops at the SciPy conference (around 40 attendees annually since 2014, did not happen in 2019) and Open Data Science Conference (around 100 attendees each, 2-3 times a year since 2015), usage help given on Stack Overflow (under the scikit-learn tag, 17,522 questions[1]) and the mailing list (1,969 subscribers), and debugging help and feature prioritization via the issue tracker (1,358 open and 5,595 closed issues). Many conferences also include formal or informal office hours to directly engage with the core developers.

**ID.5 Community Engagement**: Scikit-learn is a community-driven project, with most members coming from CISE disciplines. Developers engage with the scikit-learn primarily with the GitHub issue tracker, by posting a “pull request” (i.e. code contributions, 8,328 of which have been accepted and 685 of which are pending) or “issues” (i.e. feature requests, bug reports or other suggestions). The project decision making is formalized in a governance document. Decisions are made by the “core developers”, which are those that have been active contributors to the project in the past. Core developers can be nominated by anyone (though are usually nominated by existing core developers), and are confirmed by the existing core developers. All decisions are made by consensus among core developers, with the option to resolve conflicts by vote. There are currently 20 core developers, including Mueller and Hug. Issues that can not be resolved by the core developers will move to the Technical Committee, consisting of seven senior core developers, including co-PI Mueller. Any major changes to the package have to be proposed using a formal document (“scikit-learn enhancement proposal”) modelled after the decision making process used for the Python language.

**Developer Workshops:** We will hold workshops for scikit-learn developers at Columbia in year 1 and year 2. These workshops will take the form of “coding sprints” and bring together existing developers, but also serve to on-board new contributors. Coding sprints and hackathons are well-established methods to increase collaboration within the community and broaden the contributor base [47]. The workshops will span two days consisting of a simultaneous hackathon and an unconference (participant driven meeting). The hackathon addresses current development needs, helps new contributors make their first contribution, facilitates finalizing pending contributions, and allows the rapid creation of prototypes of new ideas. The simultaneous unconference provides opportunity for long-term planning and in-depth technical discussions. We will invite scikit-learn core developers from around the world, as well as interested prospective contributors from the CISE community with a focus on under-represented groups.

[1] The StackOverflow platform doesn’t provide a way to count unanswered questions, but there are only 153 questions without either an upvote or an accepted answer.

PI Mueller has hosted similar workshops in the past. In particular, he hosted an event in collaboration with the Women in Machine Learning and Data Science, a non-profit organization, with the goal of increasing the diversity of the scikit-learn developer community. Over the last three years, this collaboration has led to five open source sprints in New York, San Francisco and Nairobi, Kenya. These day-long events have been attended by over 35 participants each, most of them women or from other gender minorities. Most attendees were able to successfully contribute to the project during the day, or finalize their contribution in the days following the event. PI Vogelstein has also hosted similar workshops, including a hackathon dedicated to ensemble learning methods which PI's Mueller and Hug attended.

The project will form an advisory board that is seperate from the scikit-learn governing bodies. We see three major roles in this board. We propose a CISE data science and machine learning expert to provide input for the  Enhance mission and providing external input on which features will truly improve scikit-learn. We have asked Chris White to serve in this role. White is Partner Director at Microsoft working on AI and data analytics and was formerly DARPA program director of the XDATA and MEMEX programs. We envision the need for an open-source expert with a track record of community building to steer the  Sustain mission. We plan to ask Fernando Peres who is the author of IPython and co-founder or Jupyter. We also require guidance in outreach and education. Brian Caffo has agreed to serve on the board. Caffo has developed a dozen online courses in data science and statistics, is a leading instructor on Coursera, and an influencer and through leader for MOOCs.

**ID.6 Community Outreach:** The grant will provide funds for an outreach coordinator (OC) for the scikit-learn project. The OC will be responsible for community engagement and increasing the diversity of the user and developer base. The OC will engage with users through existing community infrastructure such as the issue tracker, StackOverflow and social media, as well as through office hours online and at conferences. The goal of the role of the OC is to have dedicated resources for the social interactions required to sustain an open source community [48].

More specifically, the OC will respond to user questions and concerns on these platforms, and engage relevant developers where appropriate. The OC will also promote educational materials already available, and those newly created by this grant. Through close interaction with users, the OC will be in an ideal position to identify unmet needs in the community, in terms of functionality, documentation or user engagement.

**User Workshops:** To broaden the user base and increase diversity, we will hold one-day workshops located at JHU in years 2 and 3. The participants will be selected among CISE researchers. We will target 100 participants and use project funds to support 10-12 travel allowances for under-represented minorities. These workshops will serve three purposes: (1) Provide instruction to new users by giving an introduction into applying machine learning with scikit-learn. (2) Promote new features and inform about changes in scikit-learn, both from the community and this project in particular. And, (3) Solicit feedback on existing features, prioritization of new features and unmet needs in the user community.

We have found that both for instruction and soliciting feedback, in-person communication is invaluable. Focusing on attendees from underrepresented groups within CISE means that priority is given to issues that might otherwise be overlooked given the broad reach of social media and other online platforms. The workshop will have two parallel instructional tracks: an introductory track aimed at new users and an advanced track for those already using scikit-learn in their research. The workshops will also contain focus-groups to discuss best-practices for particular applications and subfields and a round-table that allows direct discussion and feedback to developers.

**Online Courses and Tutorials:** The project includes substantial support for the development of online materials. These will be produced at JHU, using course recording studios and professional video editing.

The content will be developed by PIs, senior personnel, and the outreach coordinator. We plan for a variety of content formats, targeted at different audiences. All courses will be free; we will deliver courses in LeanPub with videos hosted on YouTube. Based on our experience with MOOCs, we do not feel that there is potential for monetizing education. The early courses in the JHU coursera data science program were profitable, but none in the last five years. We feel that trying to monetize education would be a poor path to attempt to achieve sustainability for scikit-learn.

Short lectures in mini-course format teach new users about machine learning concepts in the context of scikit-learn. This is traditional recording for content that requires high-production value. Lectures are recorded in front of a green screen with slides and visuals added in later.

For developers, we will short boot-camp style lectures that introduce advanced features and demonstrate algorithm and data use cases in a text-to-speech (TTS) engine called ARI (unpublished prototype) being used at JHU. The TTS engine reads written slides in the speakers (e.g. PI Burns') voice. The huge benefit of this approach is that content that is complex or rapidly changing can be updated/modified without going back into the studio. Developers are experienced online learners and we find that they listen to video at 2x speed and jump around, e.g. go to the quiz and backfill select knowledge to complete the task. They are used to distortion and interruptions. TTS will allow us to adapt to changing content rapidly and tackle high-concept material. Plus, we like the notion of using 'AI to teach AI'.

We will also produce videos that are part of scikit-learn's documentation. In the description of our Sustain tasks, we argued extensively that documentation must be more comprehensive and include examples. We will produce videos that teach the usage of an algorithm in scikit-learn through a running example on data with results and interpretation.

We discuss content development in the Workplan section. We commit to creating a one hour short course and 6-8 tutorials for scikit-learn documentation each year.

## Broader Impacts

As its central goal, the execution of this proposal will grow and diversify the scikit-learn community. The Enhance efforts harden scikit-learn for new uses cases. Enhance will make tree ensembles suitable for new deployments that are scalable, performance oriented, and latency sensitive. The Sustain tasks create a more open community of developers with a clear path to contributing software and with confidence that efforts to report problems, fix bugs, and add features will be reviewed promptly, fairly, and transparently. The education and outreach effort reduces barriers to entry for new users and provides support for developers and users by giving them access to project personnel both online and in person.

The project goals can be divided into those that grow the community by improving scikit-learn and those that directly engage people. Focusing on improvements, we will spend a majority of PI effort and funding on implementing processes, developing documentation, and servicing the technical debt of bug fixes, pull requests, and open issues. All of these have a direct and positive broader impact, making the scikit-learn community easier to enter and more rewarding. The software engineering in Enhance tasks are designed to grow the community to encompass users that have had to leave scikit-learn as they deploy.

We will provide face-to-face human interaction both user and developer communities. We offer opportunities that are casual and some that are immersive. On the casual side, we will provide office hours both online and in-person. A contributor or user may connect to ask a single question and a positive experience may serve as an on ramp. On the immersive side, user and developer workshops allow members of the community to connect and define the goals of longer-term virtual collaborations.

Outreach and education activities were designed to enhance diversity. Mueller will continue to offer workshops in conjunction with Women in Machine Learning and Data Science. In the past, these events have successfully allowed URMs to contribute to the scikit-learn source code. This on ramp to participation will be improved with increased documentation and improved governance. The Outreach Coordinator will hold office hours, providing follow-on support for workshop attendees.

## Current User Population

The scikit-learn project has been widely used in academic and industrial research, and has made its way into multiple commercial products. Because of the prevalence of AI and ML in all scientific endeavors, scikit-learn has spread to a multitude of research areas beyond CISE, including Physics [49,50], Astronomy [51,52], Biology [53,54], Medicine [55,56], Psychology [57,58], Cyber Security [25], Oceanography [59], and Social Sciences [60,61].

The paper describing scikit-learn [33] has been cited over 22,276 times according to Google scholar. Given the many distribution channels and open nature of the project, accurately measuring the academic user-base is quite challenging. An easier measure for engagement is contributions, which imply active engagement of the community with the project. According to the email addresses used for code contributions (a conservative estimate since many users use non-institutional email addresses), 76 researchers and students from at least 38 US universities contributed code to scikit-learn. This indicates widespread academic engagement with the development process of the project. The total number of unique code contributors to the project is about 1,500.

## Community Satisfaction

The project has about 38,600 stars on GitHub, and is used by 2,761 packages and 81,281 repositories on GitHub (according to GitHub's automated metric). The project is included in the Anaconda Python distribution, as well as in popular Linux distributions. The scikit-learn mailing list has 1,969 subscribers, including 70 email addresses from "edu" domains, across 43 institutes. 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 [62–64], including one by PI Mueller [65]. The use of the scikit-learn in teaching is so common now that it is hard to obtain useful statistics, but we expect that most university courses in machine learning or data science will use the scikit-learn to some degree. Similarly, adoption by industry has become so wide-spread that collecting meaningful metrics becomes hard. Several companies, including Microsoft, Intel and NVidia fund the scikit-learn-foundation at INRIA in France to improve their scikit-learn use-cases. Internal data from Microsoft shows that about 50% of data scientists at Microsoft use scikit-learn, an adoption rate similar to their in-house tools.

## Commitment to Share Resources

The project team is particularly interested in participating in the CCRI community the real and virtual world with the goal of connecting CISE researchers with AI and ML tools to enhance their infrastructure. A central theme in the AI revolution is that intelligence and learning has the potential to enhance all aspects of computing and human computer interaction. We expect that many CCRI teams will want to deploy ML in the context of their applications, e.g. as part of traffic classification in a software-defined networking application. scikit-learn is (in our opinion) the best toolkit to do just that: because it's implemented in Python, it integrates well with many other languages and runtimes and the uniform interfaces allow for many algorithms to be tested in a single implementation. We find the other CCRI teams a particularly attractive class of users. They are technically sophisticated and work on CISE related problems.

In addition to standard PI meetings, we plan to hold office hours both online and in person at CCRI events to answer questions and do rapid prototyping in scikit-learn. This engagement is modeled after the success of office hours held at open-source and ML conferences by Mueller, which have been wildly successful. Mueller regularly engages more than 100 people at the open data science conferences. We believe that this will offer an opportunity for a more substantial and technical interaction that will add to conference-style meetings.

## PI Qualifications

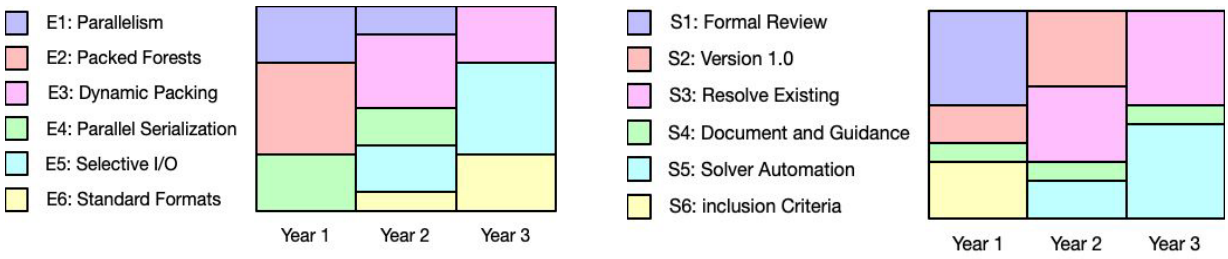
**Burns**: The last decade of PI Burns' research has built systems that accelerate machine learning and data science for large datasets, specifically graph processing [66], sparse linear algebra [67,68], clustering [7], and decision forests [41]. All of these projects have open-source implementations that continue to be supported. He has expertise in translating theory to practice, reimplementing algorithms so that they run massively parallel and for data sets that exceed the bounds of memory. He also has experience implementing commercial software, as founder and chief scientist of Gigantum, LLC.

**Mueller:** PI Mueller has been involved in the scikit-learn project as co-maintainer and core contributor for over 7 years. He is the top contributor to scikit-learn in terms of commits reported on GitHub. He is also a member of the scikit-learn Technical Committee, which is the steering body of the scikit-learn project, as well as co-author of the governance document. Mueller has co-authored several peer-reviewed papers on the design and implementation of scikit-learn [33,34] and a text book on machine learning in Python with scikit-learn that is widely used in graduate and undergraduate education [65].

**Vogelstein:** PI Vogelstein is both a developer and user of open source software, with specific expertise in ensemble methods. Vogelstein has written several papers pushing the limits of what ensemble methods can do, including in classification [69,70], unsupervised learning [71], quantifying uncertainty [72], and manifold learning [73]. Moreover, PI Vogelstein's team has released several open source software products, including GrasPy [74], and a decision forests library [41]. Finally, PI Vogelstein's team has extensively contributed to widely used Free and Open Source Software, including SciPy, and NeuroGlancer (a Google tool for visualizing large three-dimensional arrays in the Cloud).

## Workplan

This proposal was built to leverage an existing, active collaboration. The successful execution will define a process that translates validated research into the scikit-learn ecosystem. PI Burns will direct the Enhance mission, managing a software developer and PhD student that will collaborate and team program to translate existing research in scikit-learn ready software artifacts. Co-PI Mueller will lead the Sustain mission and serve as the interface to the scikit-learn open-source community. This includes managing lead developer and senior personnel Nicolas Hug and the outreach coordinator. Co-PI Joshua Vogelstein will lead user engagement, running user workshops and working with the outreach coordinator to identify the tutorial and education content. This role reflects his experience of using scikit-learn to conduct CISE research.




We present a work plan for project tasks in a mosaic chart that indicates the project years and the fractional effort spent on each task represented by the height of each box. The Enhance tasks are software features that are developed and released using scikit-learn's engineering practices, continuously integrated and continuously released through scikit-learn-contrib. Most of the Sustain tasks have an organizational and community involvement component. Developing documentation and user guidance is ongoing at a small fraction of total effort. The first year of Sustain will focus intensively on building the structure and process (S1 and S6) needed to ease the transition of new software features into the scikit-learn ecosystem. The success of the Enhance mission depends on lowering the barriers to entry.

The specific tutorials and courses to be produced have not been specifically identified. During the first half of each year, a content proposal will be developed by Vogelstein, Mueller, and the Outreach Coordinator. Selections will be presented and reviewed at the Advisory Board meeting. Content will be produced in the back half of each year, with the goal of creating one hour short course and 6-8 documentation supporting tutorials each year.

The team works together regularly already. The project grew out of Burns' and Vogelstein's effort to translate their research into scikit-learn, which has informed the efforts to improve governance and submission process. We hold a weekly project meeting online, interact daily through software workflow tools, visit each others institutions each semester, and attend many of the same conferences.

## Sustainability Plan

Sustainability and long-term health of the scikit-learn project is the primary goal of this proposal. The Sustain tasks improve the ability of the community to maintain the scikit-learn project for the long term. Sustaining open source infrastructure is a much-studied subject [75–77], and there are many competing definitions of sustainability of open source software [78]. Given that the scikit-learn project has been maintained for nearly 10 years based mostly on volunteer labor is evidence to architectural sustainability [79]. Looking at the structure of the community, an open governance model is in place since February 2019, which has been argued to be an essential aspect of community sustainability [80]. However, the size of the user and contributor base of the project creates major challenges for volunteer-driven maintenance. Without dedicated resources, sustainability will always be a relative, not an absolute statement. Our Sustain effort consists primarily of activities relating to the sustainability and long-term health of the community and organizational structure of the project, and only minor aspects reflect direct technical issues. The funding provided by this grant will allow for focused interventions and long-term planning that will decrease technical debt and improve the organizational structure of the scikit-learn project which is critical for the long-term health of the project [81].

The Enhance aspect of the proposed work represents an effort to improve one of the most widely-used parts of the scikit-learn package. Such large-scale improvements are hard to support using volunteer labor, even though they provide a large benefit to the user-community. However, given the high level of activity within the scikit-learn community, it's clear that the community will maintain any contributions made as part of this project.

## Results from Prior NSF Support

(a) Burns was the PI with co-PI Vogelstein of NSF 1649880 (1/1/2017–12/31/2018, $147,299). (b) "Computational Infrastructure for Brain Research: EAGER: BrainLab CI: Collaborative, Community Experiments with Data-Quality Controls through Continuous Integration". (c) The project built prototype a cloud-based experimental management system for reproducible science that provides workflows, visualization, and analysis with applications to functional MRI and neurophysiology. (d) The award generated two refereed publications [41,67] and contributed to three submissions under review [82–84]. (e) The project contributed to the knor, brainlab-ci, and ndmg software packages available at https://github.com/neurodata/ (f) No renewed support was requested.

(a) Burns is also co-PI on OCE-1633124 (11/1/2016–10/31/2019, $952,570) (b) “BIGDATA: IA: Democratizing Massive Fluid Flow Simulations via Open Numerical Laboratories and Applications to Turbulent Flow and Geophysical Modeling”. (c) The project studies fundamental phenomena in fluid turbulence using datasets from massive computer simulations that are ingested into databases and accessible via Web-services. (d) 10 journal articles have been published on turbulence [85,86] and computer systems [87]. (e) The JHTDB open turbulence laboratory is being used by researchers worldwide, leading to many publications (e.g. 22 in 2018 alone). The facility is also used extensively for graduate education. (f) This proposal is not for renewed support.

(a) Mueller is PI of NSF 1740305 (9/1/2017-8/31/2020 $399,356.00). (b) “SI2-SSE: Improving Scikit-Learn Usability and Automation”. (c) The project created the dabl package that includes models for automatic supervised learning, with a very simple interface, requiring minimal user interaction, as well as automatic tools for visualization for supervised learning. (d) The award generated one refereed publication in the NeurIPS meta-learning workshop [88]. (e) The project funded contributions to scikit-learn, dabl and the openml-python packages. (f) This proposal is not for renewed support.

(a) Co-PI Vogelstein was funded by NSF 1707298 (9/1/2017–8/31/2020, $959,999) and NSF 1637376 (04/15/2016–04/14/2017, $97,950). (b) “A Scientific Planning Workshop for Coordinating Brain Research Around the Globe, Baltimore, Maryland, April 7-8, 2016”. (c) *Intellectual Merit:* At the workshop, the community defined the three grand challenges for global brain sciences and the unifying resource, The International Brain Station. *Broader Impacts:* That conference led to additional three additiona NSF workshops. (d) This workshop led to four manuscripts: on the grand challenges in global brain science [89], on The International Brain Station [90], our NeuroData Cloud platform [91], and the first to demonstrate “science in the cloud” [92]. (e) Cyber-infrastructure we built based on this is available from https://neurodata.io (f) This proposal is not for renewed support.