Google Summer of Code 2016 Project Proposal: Addition of various enhancements to the `tree` module by completing stalled pull requests.

Nelson Liu
University of Washington
nfliu@uw.edu

March 18, 2016

1 Introduction

Scikit-learn (as of this writing, 3/17/2015) currently has 435 opened pull requests and 699 issues. This project is an attempt to take some of the stalled work in the review pipeline in the theme of the `tree` module and complete them to merging quality.
To find pull requests to work on, I ran a GitHub search with the keywords `'tree' is:pr is:open`. My search filters were extremely loose and they literally captured any reference of tree; this resulted in a lot of code regarding nearest neighbors / the source tree in general to pop up, but I figured that its better to cast a wider net to ensure that I don't miss any potential work to be done. The pull requests fetched are listed below in order from oldest to newest. I used this method of ordering because I figured that older pull requests are also more likely to be stale, which would make them good candidates for a Google Summer of Code project. A list of all of the pull requests, as well as my commentary on them, can be found in this document: **A survey of scikit-learn tree-related pull requests**.

2 Project Goals

This project seeks to improve the scikit-learn codebase specifically through the addressing of the outstanding `tree` module-related pull requests. The potential benefits of this project are many; the `tree` module is improved by merging several stalled pull requests, and it would cut down on the amount of outstanding pull requests and issues.

2.1 List of Deliverables

By the end of this project, the following will have been completed:

1. Finish up and merge stale pull request **#941 Adding a pruning method to the tree**.
   (a) Implement decision tree pruning algorithm
   (b) Write tests
   (c) Write documentation explaining what it is and how to use it.
   (d) Create an example illustrating its use.

2. Finish up and merge stale pull request **#6039 Add mae regression tree**
   (a) Implement MAE regression tree
   (b) Write tests
   (c) Write documentation explaining what it is and how to use it.
3. Finish up and merge stale pull request **#6169 BestFirstTreeBuilder should ignore tree.max_depth**
   Finish up and merge stale pull request **#5181 Balanced Random Forest**
   (a) Implement Balanced Random Forest
   (b) Write tests
   (c) Write documentation explaining what it is and how to use it.
   (d) Create an example illustrating its use, specifically comparing performance vs a regular tree on unbalanced data.
4. Participate in project-wide code review/bug-fixes/discussion.

3 Implementation Plans

1. **#941 Adding a pruning method to the tree**.
   Motivation:
   Decision Trees are often used for their ease of interpretation. Pruning methods are especially useful in this regard as they aid both interpretability and generalization performance.
   The demand for pruning methods is clearly there; a pull request was first opened in July 2012, but stalled without being merged into the codebase. In February of 2016, two scikit-learn users commented on this original pull request, expressing their desire for decision tree pruning to be added to the codebase. Additionally, @amueller remarks that he "frequently get[s] asked about post-pruning". With this combination of potential for practical application and high community demand, I think that it would be a great idea to dust off **#941** and implement post-pruning for decision trees.
   Implementation Plan:
   Unfortunately, **#941** was written with regards to the old decision tree implementation; everything has since been ported to Cython. As a result, it will not be possible to reuse most of the code, although the ideas within its implementation may prove useful to study.
   I plan on reviewing the literature regarding decision tree pruning to ascertain the best practices, and then replicate the algorithm in the context of scikit-learn. After this is completed, I will write unit tests to ensure proper functioning of the method. Lastly, I will create add details about the method to the docs, and devise an example clearly illustrating how pruning works and its effects.

2. **#6039 Add mae regression tree**
   Motivation:
   Currently, the DecisionTreeRegression estimator only supports mean squared error (MSE). There have been several requests for new criterion, such as mean absolute error (MAE) and mean square of percentage error (MSPE) (for example, see **#5368 Request more criterion for random forest regression**). Adding new criterion would enable their use in both the decision tree classes and tree based estimators such as `RandomForest`. Additionally, various criterion are better suited for different applications.
   Implementation Plan:
   #6039 has some work already done, but it seems like a fair amount is still missing from the patch. Reusing code from the existing PR may be the easiest option, but I am not afraid to rewrite the code

/ reanalyze the best way to implement this feature if necessary. After adding the new criterion, I will write appropriate unit tests and add clarifying information to the docs about the new criterion, how they work, and what their applications are.

3. **#6169 BestFirstTreeBuilder should ignore tree.max_depth**
   Motivation:
   It is not quite clear in the docs how `tree.max_depth` and `tree.max_leaf_nodes` interact, which leads to user confusion. I want to clarify the docs in this area regarding how the trees are built, or make code changes if necessary.
   Implementation Plan:
   This should probably be a trivial fix, as developer opinion seems to be leaning toward just changing the documentation to avoid creating a situation where users need to change their code.
   If the choice is made to not make any changes in the codebase, I will simply update the appropriate docstrings (`BestFirstTreebuilder`, `DecisionTreeClassifier`, and `DecisionTreeRegressor`). If the choice is made to change the codebase to match the docstring as it currently is, then I will update the code, write some specific tests to ensure that these edge cases work as intended, and perhaps devise and example demonstrating the proper behavior.

4. **#5181 Balanced Random Forest**
   Motivation:
   This PR is tangentially related to the tree code, but I believe it is quite a useful feature and should be implemented into the main codebase. Balanced Random Forests are useful for learning unbalanced data, and is used by simply passing a parameter `balanced=True` to `RandomForestClassifier`.
   Implementation Plan:
   The original pull request **#5181 Balanced Random Forest** has a fair amount of work done, but I will rewrite portions of code if necessary to optimize for efficiency and clarity. Beyond that, I will be writing unit tests to verify proper function, documentation explaining the purpose of Balanced Random Forests, and an example illustrating its benefits over a normal Random Forest on imbalanced data.

5. **Participate in project-wide code review/bug-fixes/discussion**
   I am planning to implement at the least one (possibly trivial) pull request each week to bring about improvements in the code-base incrementally. I also plan on continuing my participation in discussion regarding issues and pull requests raised, and looking over the code of other contributors.

4 Timeline

During each of these weeks, I will be submitting at least one pull request unrelated to the `tree` module / my Google Summer of Code project. These weeks are not hard deadlines; if I finish a deadline early, I will move on to the next. In other words, this timeline is quite flexible and I will adjust it if necessary depending on my summer progress.

- **Community Bonding Period (April 22, 2016 - May 22, 2016)**
  During this period, I will continue contributing to scikit-learn by submitting high quality pull requests and issues, participating in associated discussion, and reviewing pull requests submitted by other contributors. Additionally, I plan to start investigation into various aspects of my Google Summer of Code project, such as literature reviews about tree pruning algorithms and criterion.

- **Week 1-3 (May 23, 2016 - June 12, 2016)**
  During these weeks, I plan on working on the tree pruning algorithm. I want to ensure that it is a

high quality addition to the API, and will thus spend three weeks on performing the literature review and implementing the efficient, clear, and clean code as well as verifying its correctness.

- **Week 4-7 (May June 13, 2016 - July 10, 2016)**
  During these weeks, I plan to add details about the tree pruning algorithm to the documentation, write a separate entry in the user guide about how the algorithm works, when to use it, and what it does. I will also be creating an example illustrating its benefits, and write appropriate unit tests to verify its functional correctness. Lastly, I will use this time to iron out any last kinks in the tree pruning code. I hope to have the tree pruning code and docs merged at the end of week 7.

- **Week 8-9 (July 11, 2016 - July 24, 2016)**
  During these weeks, I plan to work on the MAE regression tree. I plan to review literature regarding MAE and study implementations online to draw the best elements from these eclectic sources for use in a scikit-learn MAE regression tree. I plan on spending thse weeks and the next on implementing the tree code and verifying its correctness.

- **Week 10-11 (July 25, 2016 - August 7, 2016)**
  During this week, I will write unit tests to verify edge case coverage and add details about the criterion to the documentation. I will also use this time to iron out any last issues with the code, and I plan to have the MAE regression tree merged at the end of Week 11.

- **Week 12 (August 8, 2016 - August 14, 2016)**
  During this week, I want to address **#6169 BestFirstTreeBuilder should ignore tree.max_depth** and get it merged into master.

- **Week 13-14 (August 15, 2016 - August 23, 2016)**
  During this week, I plan on finishing up pull request **#5181 Balanced Random Forest** and merging the balanced Random Forest parameter into the codebase.

5 Student Information

5.1 Personal Information

Name: Nelson Liu
Email: nfliu@uw.edu
Telephone: +1 714 306 9521
Time Zone: GMT/UTC -08:00 (PST, I'm usually in Seattle, San Francisco, or Los Angeles)
IRC Nick: nelson-liu on freenode
Github handle: nelson-liu
Blog (Currently empty, I have a few unpublished drafts of posts to finalize): https://blog.nelsonliu.me

5.2 University Information

Name: University of Washington, Seattle
Degree: Bachelor of Science / Bachelor of Arts
Major: Computer Science, Statistics, Linguistics

5.3 About Me

My name is Nelson Liu, and I'm a current undergraduate at the University of Washington, Seattle, studying computer science, statistics, and linguistics.

I have done two summers of NLP research on automated natural language question-answering with MIT CSAIL's InfoLab Group. I'm currently a member of Noah Smith's ARK research group, where I work on natural language processing for social science. My current project involve building a model for prediction of Supreme Court case decisions based on submitted amicus briefs.

Beyond my participation in research, I'm an avid hacker – I love building personal side projects in my free time and participating in collegiate hackathons. For example, I was Top 20 / 200+ at HackPrinceton 2015, Top 30 / 300+ at PennApps Winter 2015, and a finalist at AngelHack Seattle 2015. I work quickly and efficiently, and love to learn new things.

I started contributing to scikit-learn in November of 2015, but I've been working with the library since early 2014. Scikit-learn is an awesome community to work with, and I believe that its mission of democratizing machine learning methods is an important one. I'd love to contribute my skills to the project as a Google Summer of Code student.

I think I'm a great fit for this project because of my experience with the library, machine learning experience, and familiarity with the contribution and review process. I'm committed to staying in constant contact with my project mentor to ensure that we reach optimal implementations in the allotted time.