Latest Articles / Data

Six reasons why I recommend scikit-learn

It's an extensive, well-documented, and accessible, curated library of machine-learning models

By Ben Lorica

December 29, 2013

I use a variety of tools for advanced analytics, most recently I've been using Spark (and MLlib), R, scikit-learn, and GraphLab. When I need to get something done quickly, I've been turning to scikit-learn for my first pass analysis. For access to high-quality, easy-to-use, implementations of popular algorithms, scikit-learn is a great place to start. So much so that I often encourage new seasoned data scientists to try it whenever they're faced with analy that have short deadlines.

I recently spent a few hours with one of scikit-lea Grisel. We had a free flowing discussion were we t learning, data science, programming languages, b learn! Along the way, I was reminded by why I've scikit-learn project.

documentation (which I hold up as an example for other communities and projects to emulate). Contributions to scikit-learn are required to include narrative examples along with sample scripts that run on small data sets. Besides good documentation there are other core tenets that guide the community's overall commitment to quality and usability: the *global* API is safeguarded, all public API's are well documented, and when appropriate contributors are encouraged to expand the coverage of unit tests.

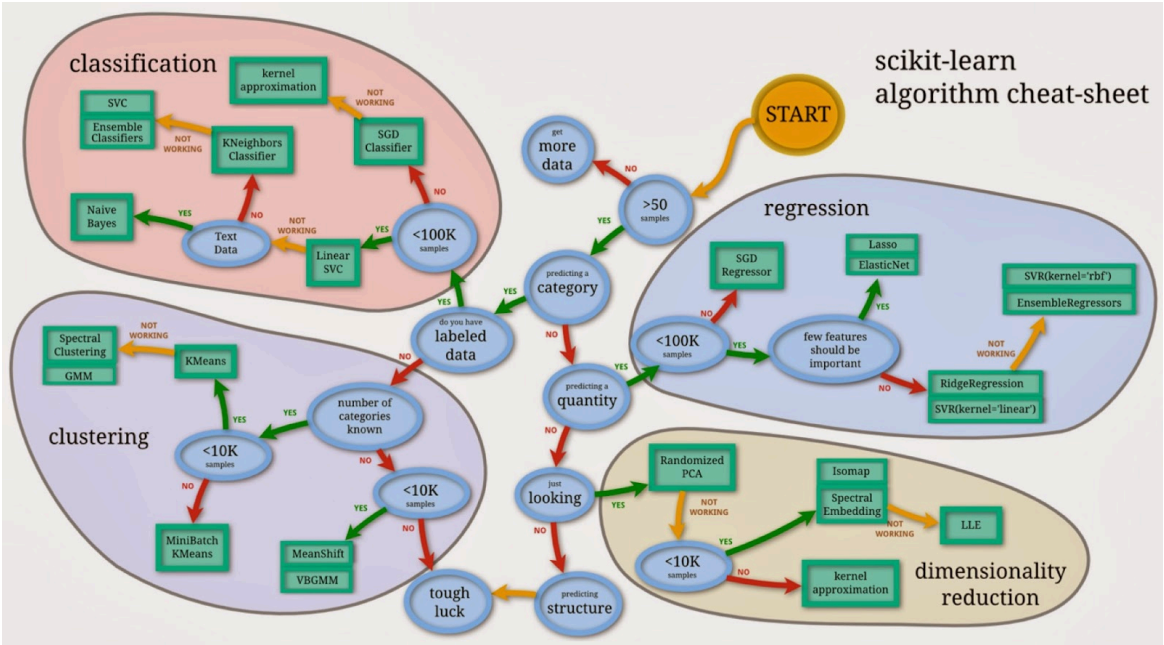
Models are chosen and implemented by a dedicated team of experts

scikit-learn's stable of contributors includes experts in machine-learning and software development. A few of them (including Olivier) are able to devote a portion of their professional working hours to the project.

Covers most machine-learning tasks

Scan the list of things available in scikit-learn and you quickly realize that it includes tools for many of the standard machine-learning *tasks* (such as clustering, classification, regression, etc.). And since scikit-learn is developed by a large community of developers and machine-learning experts, promising new techniques tend to be included in fairly short order.

As a *curated* library, users don't have to choose from multiple competing implementations of the same algorithm (a problem that R users often face). In order to assist users who struggle to *choose between different models*, Andreas Muller created a simple flowchart for users:



in the SF Bay Area, and it does appear to be the language preferred by many data scientists. Python's interpreter allows users to *interact and play* with data sets, and from the outset this made the language attractive to data analysts. More importantly an impressive set of Python data tools (*pydata*) have emerged over the last few years (I wrote about the pydata ecosystem early this year).

Many data scientists work regularly with several pydata tools including scikit-learn, IPython, and matplotlib. A common practice when using scikit-learn is to create matplotlib charts to evaluate data quality or debug a model. Users are also starting to share multi-step analytic projects, using IPython notebooks that embed results and outputs from different pydata components.

One other sign that Python has emerged as the preferred language of data scientists: new analytic tools like Spark (PySpark), GraphLab (GraphLab notebook), and Adatao all support Python.

Focus

scikit-learn is a machine-learning library. Its goal is to provide a set of common algorithms to Python users through a *consistent* interface. This means that hard choices have to be made as to what fits into the project. For example the community recently decided that Deep Learning had enough *specialized requirements* (large number of hyper-parameters; computation on GPU introduces new complex software dependencies) that it was best included in a new project. scikit-learn developers have instead opted to implement baseline neural networks as building blocks (Multilayer Perceptron and Restricted Boltzmann Machines).

scikit-learn scales to most data problems

The knock on Python is speed and scale. It turns out that while scale can be a problem, it may not come up as often as some detractors claim. Many problems can be tackled using a single (big memory) server, and well-designed software that runs on a single machine can blow away distributed systems. Other techniques like sampling or ensemble learning can also be used to train models on massive data sets.

But there are occasions when the combination of *raw data size* and workflow dictates my choice of tools. I sometimes turn to machine-learning tools that integrate with my data wrangling and ETL tool (e.g., Spark, MapReduce,

Post topics: Data