Why you should learn Scikit-learn

By **Guest Contributor** - NOVEMBER 23, 2017 - 12:00 AM

7 min read

Today, machine learning in Python has become almost synonymous with scikit-learn.

The “Big Bang” moment for scikit-learn was in 2007 when a gentleman named David Cournapeau decided to write this project as part of Google Summer of Code 2007. Let’s take a moment to thank him. Matthieu Brucher later came on board and developed it further as part of his thesis. From that point on, sklearn never looked back. In 2010, the prestigious French research organization INRIA took ownership of the project with great developers like Gael Varoquaux, Alexandre Gramfort et al. starting work on it. Here’s the oldest pull request I could find in sklearn’s repository. The title says “we’re getting there”! Starting from there to today where sklearn receives funding and support from Google, Telecom ParisTech and Columbia University among others, it surely must’ve been quite a journey.

Sklearn is an open source library which uses the BSD license. It is widely used in industry as well as in academia. It is built on Numpy, Scipy and Matplotlib while also having wrappers around various popular libraries such LIBSVM. Sklearn can be used “out of the box” after installation.

Can I trust scikit-learn?

Scikit-learn, or sklearn, is a very active open source project having brilliant maintainers. It is used worldwide by top companies such as Spotify, booking.com and the like. That it is open source where anyone can contribute might make you question the integrity of the code, but from the little experience I have contributing to sklearn, let me tell you only very high-quality code gets merged. All pull requests have to be affirmed by at least two core maintainers of the project. Every code goes through multiple iterations. While this can be time-consuming for all the parties involved, such regulations ensure sklearn’s compliance with the industry standard at all times. You don’t just build a library that’s been awarded the “best open source library” overnight!

How can I use scikit-learn?

Sklearn can be used for a wide variety of use-cases ranging from image classification to music recommendation to classical data modeling.

Scikit-learn in various industries:

In the ***Image classification*** domain, Sklearn's implementation of K-Means along with PCA has been used for handwritten digit classification very successfully in the past.

Sklearn has also been used for **facial/ faces recognition** using SVM with PCA. Image segmentation tasks such as detecting Red Blood Corpuscles or segmenting the popular Lena image into sections can be done using sklearn.

A lot of us here use Spotify or Netflix and are awestruck by their recommendations. **Recommendation engines** started off with the collaborative filtering algorithm. It basically says "if people like me like something, I'll also most probably like that." To find out users with similar tastes, a KNN algorithm can be used which is available in sklearn. You can find a good demonstration of how it is used for music recommendation here.

Classical data modeling can be bolstered using sklearn. Most people generally start their kaggle competitive journeys with the titanic challenge. One of the better tutorials out there on starting out is by dataquest and generally acts as a good introduction on how to use pandas and sklearn (a lethal combination!) for data science. It uses the robust Logistic Regression, Random Forest and the Ensembling modules to guide the user. You will be able to experience the user-friendliness of sklearn first hand while completing this tutorial. Sklearn has made machine learning literally a matter of importing a package.

Sklearn also helps in Anomaly detection for highly imbalanced datasets (99.9% to 0.1% in credit card fraud detection) through a host of tools like EllipticEnvelope and OneClassSVM. In this regard, the recently merged IsolationForest algorithm especially works well in higher dimensional sets and has very high performance.

Other than that, sklearn has implementations of some widely used algorithms such as linear regression, decision trees, SVM and Multi Layer Perceptrons (Neural Networks) to name a few. It has around 39 models in the "linear models" module itself! Happy scrolling here! Most of these algorithms can run very fast compared to raw python code since they are implemented in Cython and use Numpy and Scipy (which in-turn use C) for low-level computations.

How is sklearn different from TensorFlow/MLlib?

TensorFlow is a popular library to implement deep learning algorithms (since it can utilize GPUs). But while it can also be used to implement machine learning algorithms, the process can be arduous. For implementing logistic regression in TensorFlow, you will first have to "build" the logistic regression algorithm using a computational graph approach. Scikit-learn, on the other hand, provides the same algorithm out of the box however with the limitation that it has to be done in memory. Here's a good example of how LogisticRegression is done in Tensorflow.

Apache Spark's MLlib, on the other hand, consists of algorithms which can be used out of the box just like in Sklearn, however, it is generally used when the ML task is to be performed in a distributed setting. If your dataset fits into RAM, Sklearn would be a better choice for the task.

If the dataset is massive, most people generally prototype on a small subset of the dataset locally using Sklearn. Once prototyping and experimentation are done, they deploy in the cluster using MLlib.

Some sklearn must-knows

Scikit-learn can be used for three different kinds of problems in machine learning namely supervised learning, unsupervised learning and reinforcement learning (ahem AlphaGo). Unsupervised learning happens when one doesn't have 'y' labels in their dataset. Dimensionality reduction and clustering are typical examples. Scikit-learn has implementations of variations of the Principal Component Analysis such as SparsePCA, KernelPCA, and IncrementalPCA among others.

Supervised learning covers problems such as spam detection, rent prediction etc. In these problems, the 'y' tag for the dataset is present. Models such as Linear regression, random forest, adaboost etc. are implemented in sklearn.

```
 From sklearn.linear_models import LogisticRegression
Clf = LogisticRegression().fit(train_X, train_y)
Preds = Clf.predict(test_X)
```

Model evaluation and analysis

Cross-validation, grid search for parameter selection and prediction evaluation can be done using the Model Selection and Metrics module which implements functions such as cross_val_score and f1_score respectively among others. They can be used as such:

```
 Import numpy as np
From model_selection import cross_val_score
From sklearn.metrics import f1_score

Cross_val_avg = np.mean(cross_val_score(clf, train_X, train_y, scoring='f1'))

# tune your parameters for better cross_val_score
# for model results on a certain classification problem

F_measure = f1_score(test_y, preds)
```

Model Saving

Simply pickle your model using pickle.save and it is ready to be distributed and deployed!

Hence a whole machine learning pipeline can be built easily using sklearn.

Finishing Remarks

There are many good books out there talking about machine learning, but in context to Python, Sebastian Raschka`s (one of the core developers on sklearn) recently released his book titled " Python Machine Learning" and it's in great demand. Another great blog you could follow is Erik Bernhardsson's blog. Along with writing about machine learning, he also discusses software development and other interesting ideas. Do subscribe to the scikit-learn mailing list as well. There are some very interesting questions posted there and a lot of learnings to take home. The machine learning subreddit also collates information from a lot of different sources and is thus a good place to find useful information.

Scikit-learn has revolutionized the machine learning world by making it accessible to everyone. Machine learning is not like black magic anymore. If you use scikit-learn and like it, do consider contributing to sklearn. There is a huge clutter of open issues and PRs on the sklearn GitHub page. Scikit-learn needs contributors! Have a look at this page to start contributing. Contributing to a library is easily the best way to learn it!

[author title="About the Author"]

Devashish Deshpande started his foray into data science and machine learning in 2015 with an online course when the question of how machines can learn started intriguing him. He pursued more online courses as well as courses in data science during his undergrad. In order to gain practical knowledge he started contributing to open source projects beginning with a small pull request in Scikit-Learn. He then did a summer project with Gensim and delivered workshops and talks at PyCon France and India in 2016. Currently, Devashish works in the data science team at belong.co, India. Here's the link to his GitHub profile.[/author]

Guest Contributor

PROGRAMMING

APPLICATION DEVELOPMENT

LANGUAGES

DESIGN PATTERNS

HIGH PERFORMANCE

MICROSERVICES

SUBSCRIBE TO OUR NEWSLETTER

Monthly digest of what's new and exciting from us.

Your email address

JOIN