

Scikit-learn

open, easy, yet versatile machine learning

Gaël Varoquaux

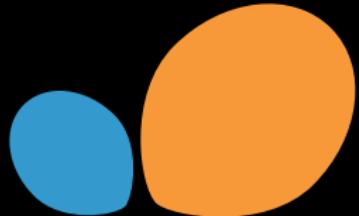
Inria



machine learning in Python

1 The vision

Goals and tradeoff



Outreach

across scientific fields,
applications, communities

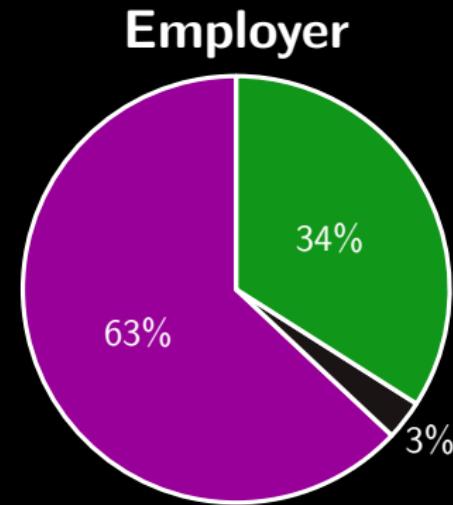
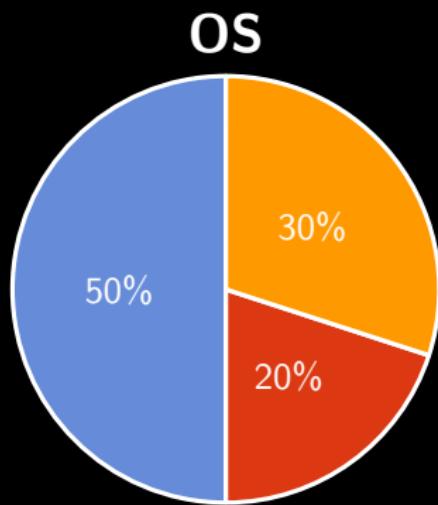


Enabling
foster innovation

1 scikit-learn user base

350 000 returning users

5 000 citations



■ Windows ■ Mac

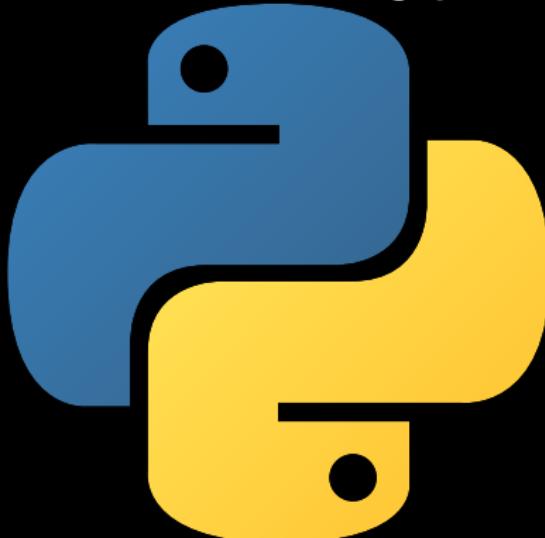
■ Linux

■ industry ■ academia ■ other

Python

- High-level language, for users and developers
- General-purpose: suitable for any application
- Excellent interactive use

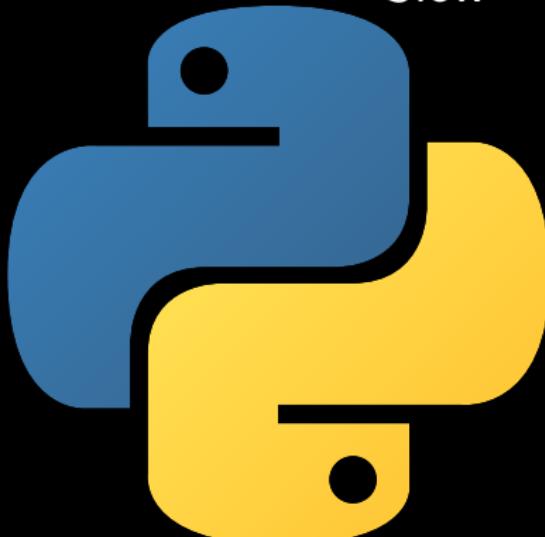
Slow \Rightarrow compiled code as a backend



Python

- High-level language, for users and developers
- General-purpose: suitable for any application
- Excellent interactive use

Slow \Rightarrow compiled code as a backend

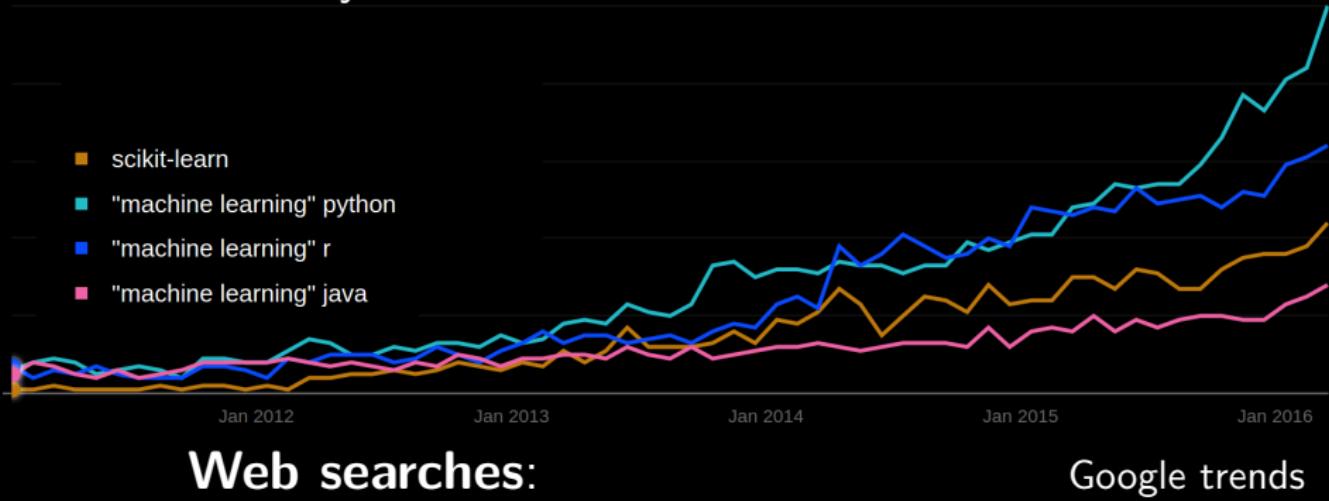


Scipy

- Vibrant scientific stack
- numpy arrays = wrappers on C pointers
- pandas for columnar data
- scikit-image for images

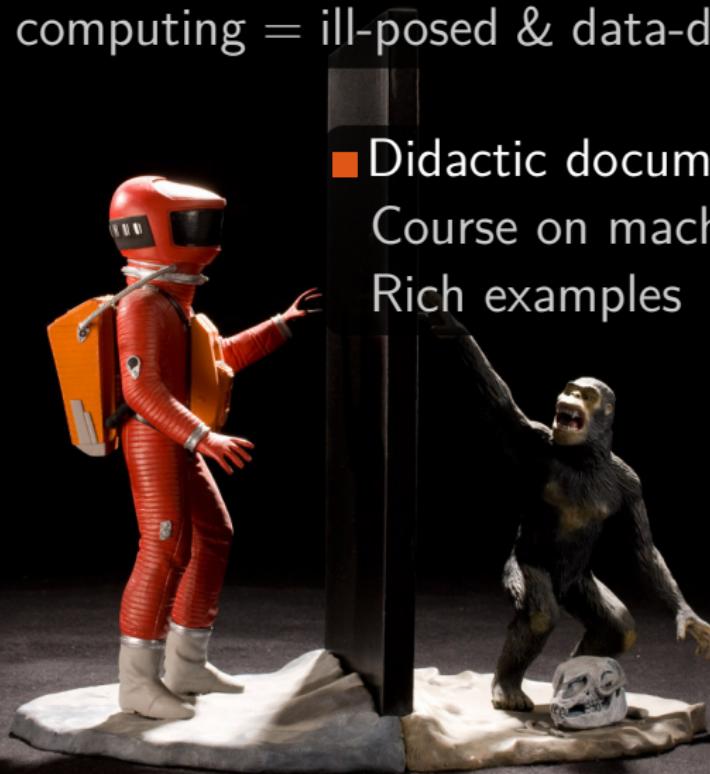
1 A Python library

Users like Python



1 Tradeoffs for outreach

- Algorithms and models with good failure mode
Avoid parameters hard to set or fragile convergence
Statistical computing = ill-posed & data-dependent
- Didactic documentation
Course on machine learning
Rich examples



2 The tool

A Python library for machine learning



©Theodore W. Gray

Simplify, but do not dumb down

2 API: specifying a model

A central concept: **the estimator**

- Instantiated without data
- But specifying the parameters

```
from sklearn.neighbors import  
KNearestNeighbors  
  
estimator = KNearestNeighbors(  
    n_neighbors=2)
```

Models often have hyperparameters

Finding good defaults is crucial, and hard

2 API: training a model

Training from data

```
estimator.fit(X_train, Y_train)
```

with:

- X a numpy array with shape

$$n_{\text{samples}} \times n_{\text{features}}$$

- y a numpy 1D array, of ints or float, with shape

$$n_{\text{samples}}$$

2 API: using a model

- Prediction: classification, regression

```
Y_test = estimator.predict(X_test)
```

- Transforming: dimension reduction, filter

```
X_new = estimator.transform(X_test)
```

- Test score, density estimation

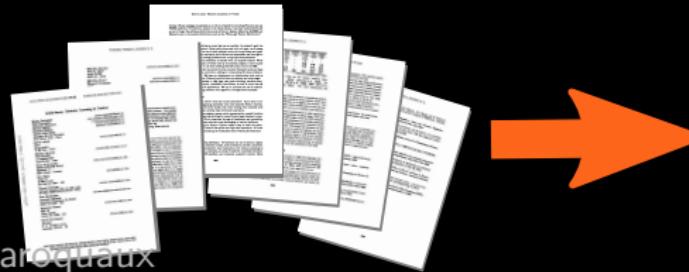
```
test_score = estimator.score(X_test)
```

2 Vectorizing: from raw data to a sample matrix X

- For text data: counting word occurrences
 - Input data: list of documents (string)
 - Output data: numerical matrix

```
from sklearn.feature_extraction.text
      import HashingVectorizer
hasher = HashingVectorizer()

X = hasher.fit_transform(documents)
```



data
algorithms
et
use
Python
supervised
efficient
some
method
shogun
other
libsvm
input
estimators
object
provide
score
PCA
libraries
high-level
Journal
example
algorithms
analysis
performance
scientific
language

more
implementation
While
BSD
Computing
pymvpa
large
license
objects
bindings
restrictions
numpy
array
development
data
interface
pybrain
memory
parameters
code
documentation
ecosystem
estimator
code
documentation
MDP
analysis
performance
scientific
language

Supervised learning

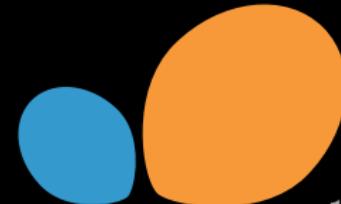
- Decision trees (Random-Forest, Boosted Tree)
- Linear models ■ SVM
- Gaussian processes ...

Unsupervised Learning

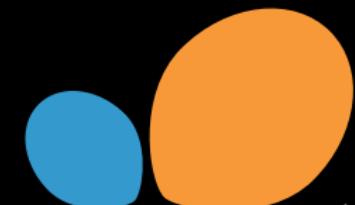
- Clustering ■ Mixture models
- Dictionary learning ■ ICA
- Outlier detection ...

Model selection

- Cross-validation
- Parameter optimization



2 Models most used in scikit-learn

1. Logistic regression, SVM
SAG/SAGA solver
 2. Random forests
Parallel computing, feature importance
 3. PCA
Randomized linear algebra, online solver
 4. Kmeans
KMeans++ initialization, Elkan solver, online solver
 5. Naive Bayes
On-line solver
 6. Nearest neighbor
KD-tree / ball-tree
- 

2 Models most used in scikit-learn

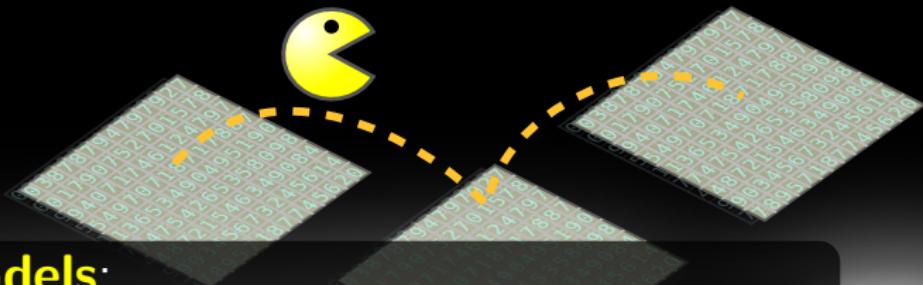
1. Logistic regression, SVM
SAG/SAGA solver
2. Random forests
Parallel computing, feature importance
3. PCA
Randomized linear algebra, online solver
4. Kmeans
KMeans++ initialization, Elkan solver, online solver
5. Naive Bayes
On-line solver
 - Multi-class / multi-output
 - Dense or sparse data
6. Nearest neighbor
KD-tree / ball-tree

“Big” data

Efficient processing on large datasets

2 Many samples: on-line algorithms

```
estimator.partial_fit(X_train, Y_train)
```



Supervised models:

Naive Bayes, linear models (various loss),
multi-layer perceptron

Clustering:

KMeans, Birch

Linear decompositions:

PCA, Dictionary learning, Latent Dirichlet Allocation

2 Many features: on-the-fly data reduction

```
X_small = estimator.transform(X_big, y)
```

Random projections (will average features)

```
sklearn.random_projection
```

random linear combinations of the features

Feature clustering

```
sklearn.cluster.FeatureAgglomeration
```

on images: super-pixel strategy

Feature hashing

```
sklearn.feature_extraction.text.
```

HashingVectorizer

stateless: can be used in parallel

3 The development



3 Community-based development in scikit-learn

Huge feature set:

benefits of a large team



Project growth:

- More than 400 contributors
- ~ 20 core contributors



<https://www.openhub.net/p/scikit-learn>

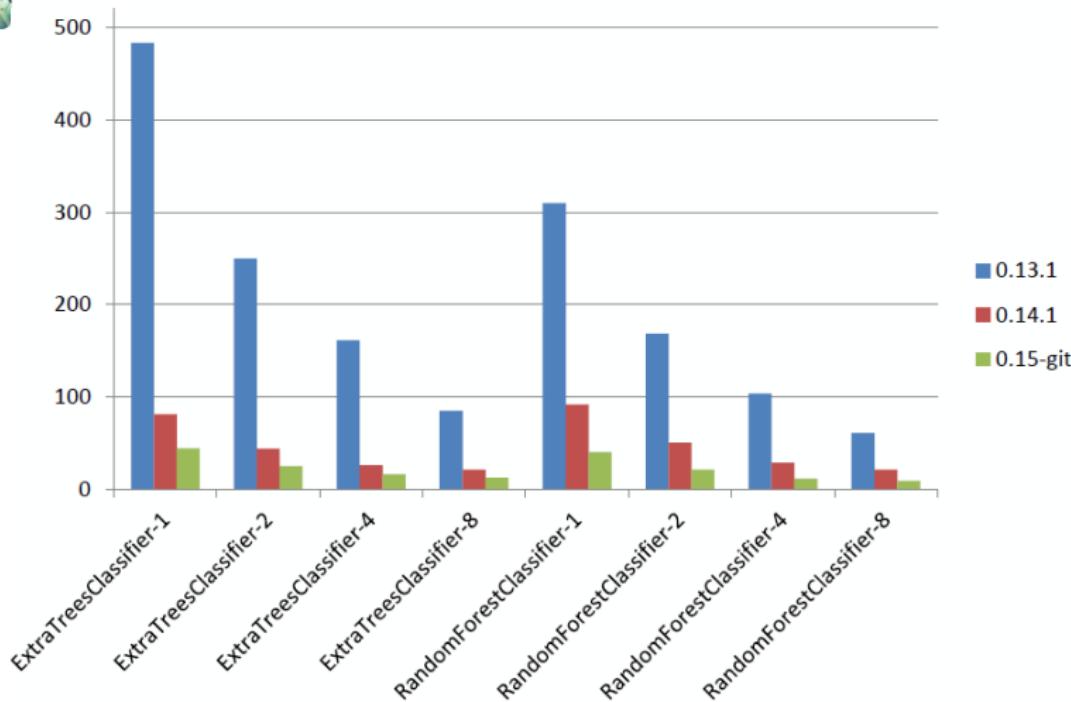
Community-driven project

3 Many eyes makes code fast



Gilles Louuppe @glouppe · Feb 18

Speed improvement from 0.13 to 0.15-git of Random Forests in Scikit-Learn:



L. Buitinck, O. Grisel, A. Joly, G. Louuppe, J. Nothman, P. Prettenhofer

3 Quality assurance

Code review: pull requests

- We read each others code
- Everything is discussed:
 - Should the algorithm go in?
 - Are there good defaults?
 - Are the numerics stable?
 - Could it be faster?

The screenshot shows a GitHub pull request interface. The code being reviewed is in `sklearn/cluster/_inertia.pyx`. A specific line of code is highlighted in red:

```
...     ... @@ -21,9 +36,9 @@ def compute_ward_dist(np.ndarray[D
21      36         for i in range(size_max):
22      37             row = coord_row[i]
23      38             col = coord_col[i]
24 -             n = (m_1[row] * m_1[col]) / (m_1[row] + m_1
25 +             n = 1.0 / (1.0 / m_1[row] + 1.0 / m_1[col])
```

A comment from user `agramfort` is visible, pointing out numerical instability:

i am afraid this is numerically less stable. it is justified by speed?

Another comment from user `jmetzen` indicates the issue was resolved:

you are right, I reverted it to the old implementation

An "Add a line note" button is also present.

3 Quality assurance

Code review: pull requests

- We read each others code
- Everything is discussed:
 - Should the algorithm go in?
 - Are there good defaults?
 - Are the numerics stable?
 - Could it be faster?

The screenshot shows a GitHub pull request interface. The code being reviewed is in `sklearn/cluster/_inertia.pyx`. A specific line of code is highlighted in red:

```
... 36     for i in range(size_max):  
37         row = coord_row[i]  
38         col = coord_col[i]  
39 -     n = (m_1[row] * m_1[col]) / (m_1[row] + m_1  
39 +     n = 1.0 / (1.0 / m_1[row] + 1.0 / m_1[col])
```

A comment from user `agramfort` is visible, followed by a reply from `jmetzen`:

agramfort started a discussion in the diff 7 months ago

2 agramfort repo collab
i am afraid this is numerically less stable. it is justified by speed?

jmetzen
you are right, I reverted it to the old implementation

Add a line note

Unit testing

- Everything is tested
- Great for numerics



Scikit-learn



Machine learning for everyone

– from beginner to expert

A design challenge: hiding complexity

A development challenge: keeping quality

A research challenge: robust methods without surprises

Sustainability (funding) is an issue



@GaelVaroquaux