Scikit-learn

open, easy, yet versatile machine learning

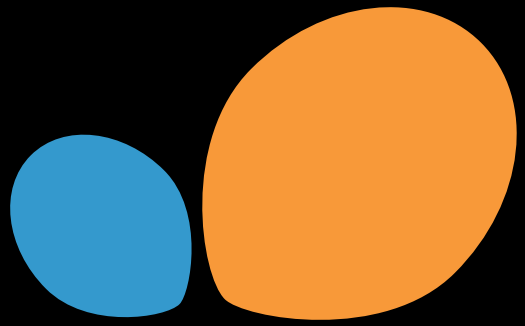
Gaël Varoquaux

Inria

machine learning in Python

1 The vision

Goals and tradeoff



Scikit-learn's vision: Machine learning for everyone

Outreach

across scientific fields,

applications, communities

Enabling

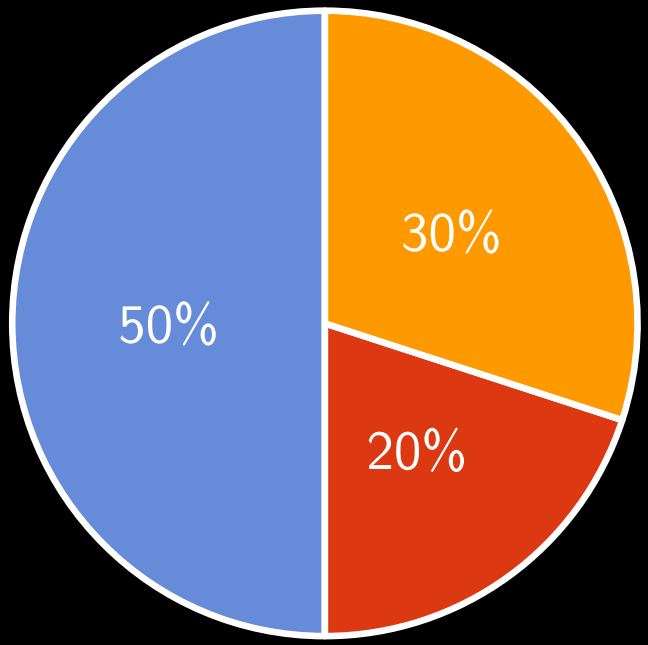
foster innovation

1 scikit-learn user base

350 000 returning users

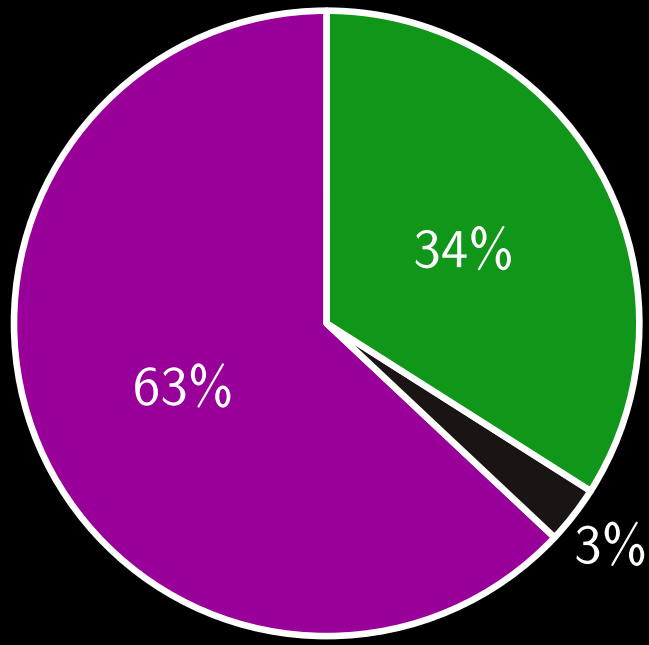
5 000 citations

OS



Windows Mac Linux

Employer

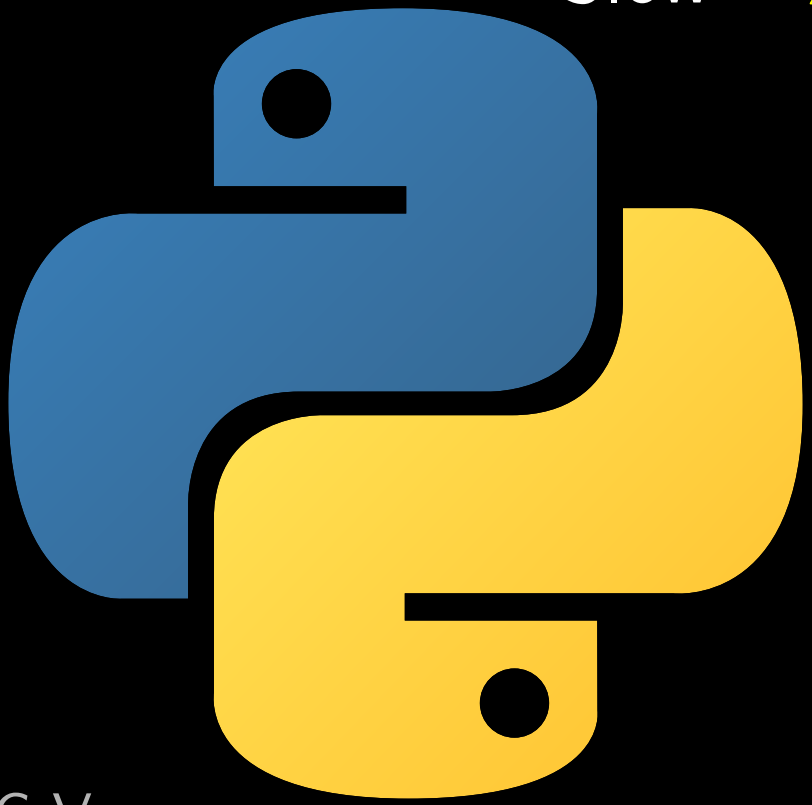


industry academia other

Python

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

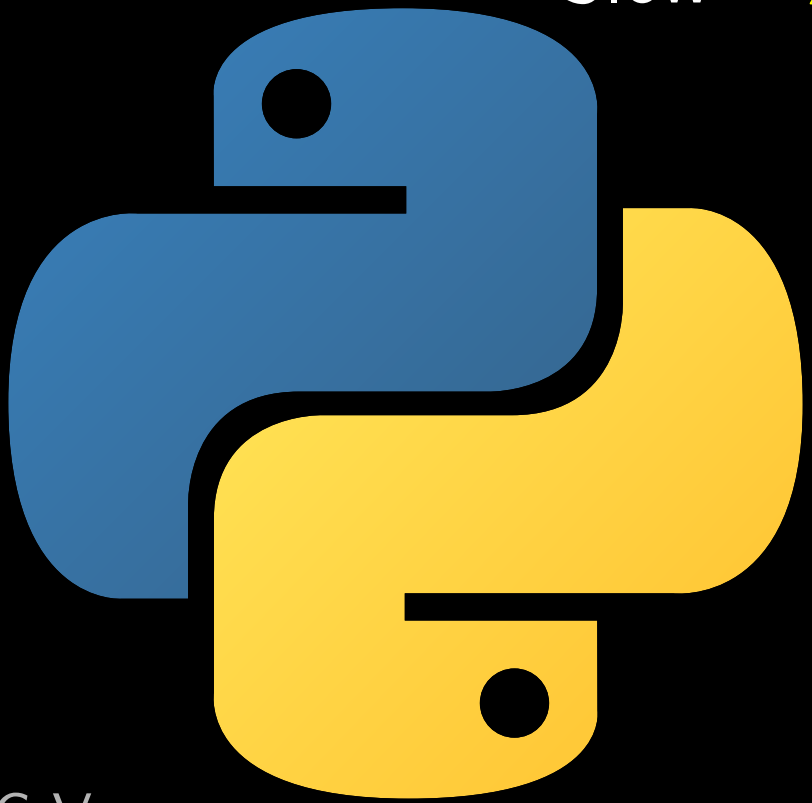
Slow $\Rightarrow$ compiled code as a backend



Python

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

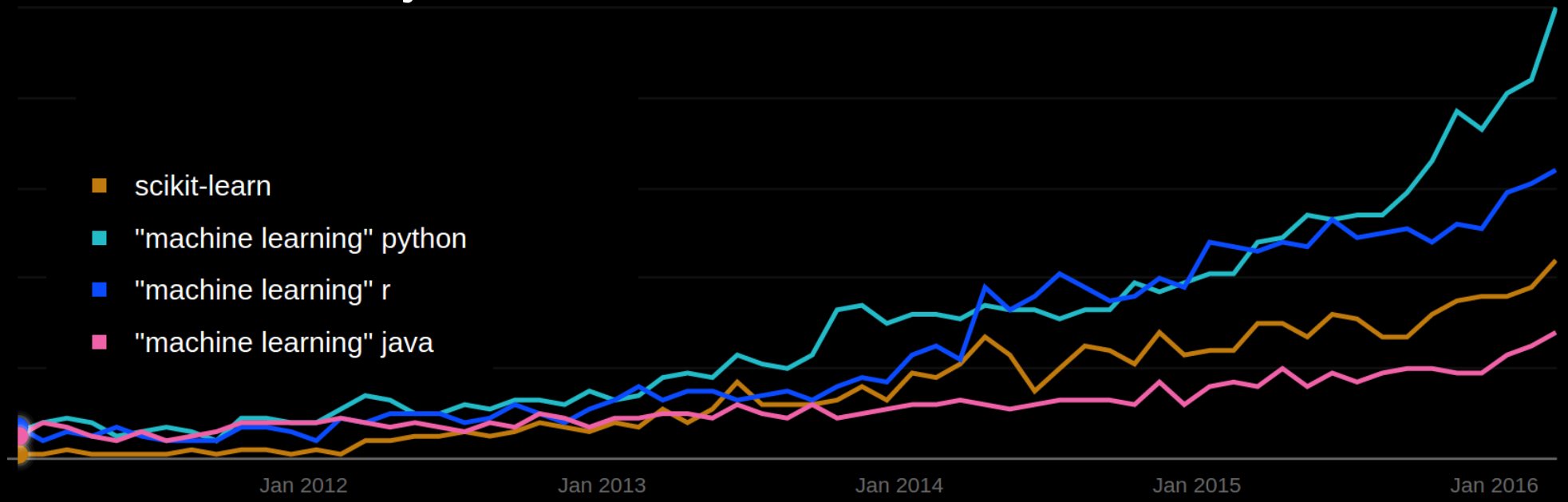
Slow $\Rightarrow$ compiled code as a backend



Scipy

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

Users like Python

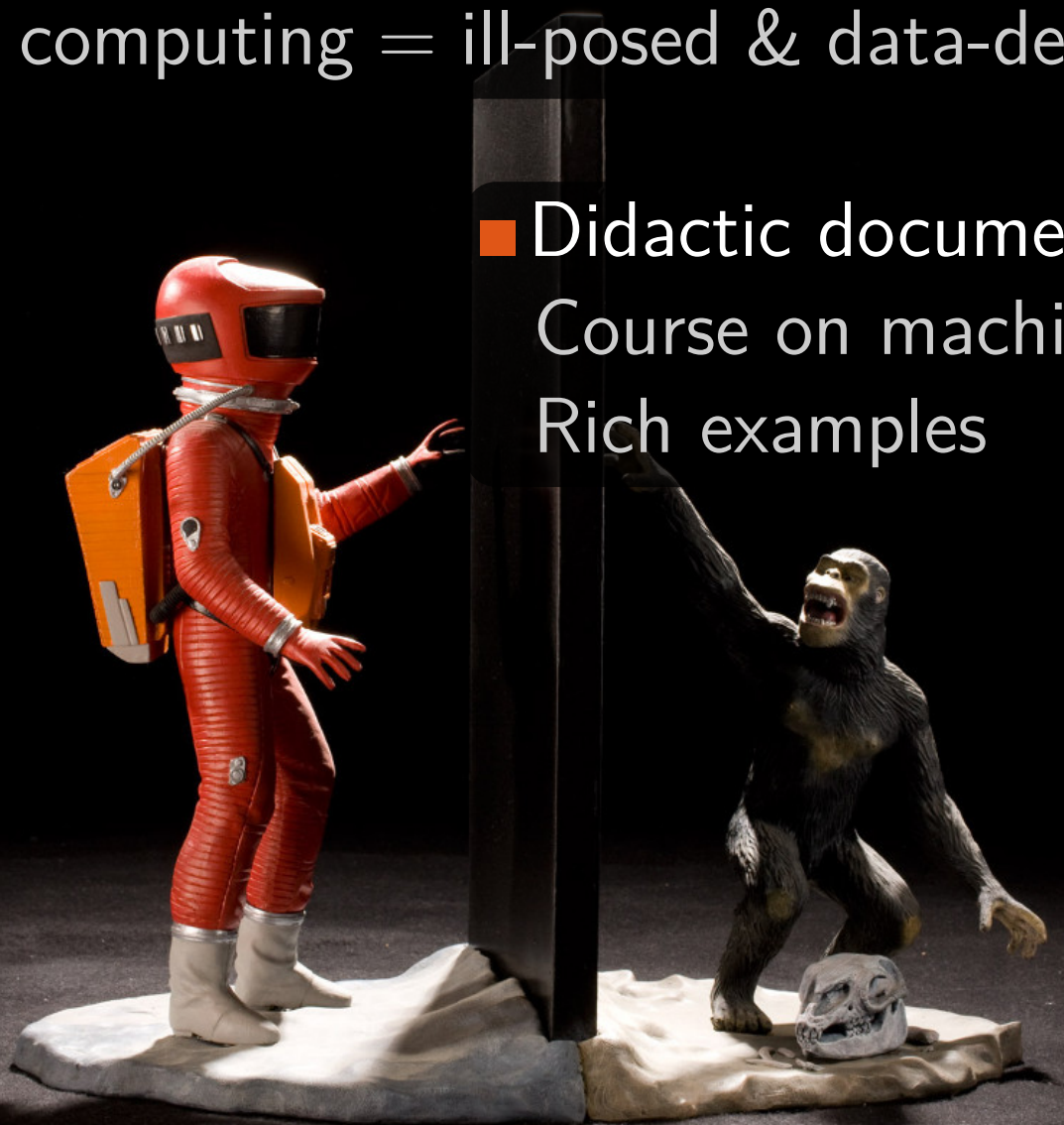


Web searches: Google trends

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**

- Instanciated 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

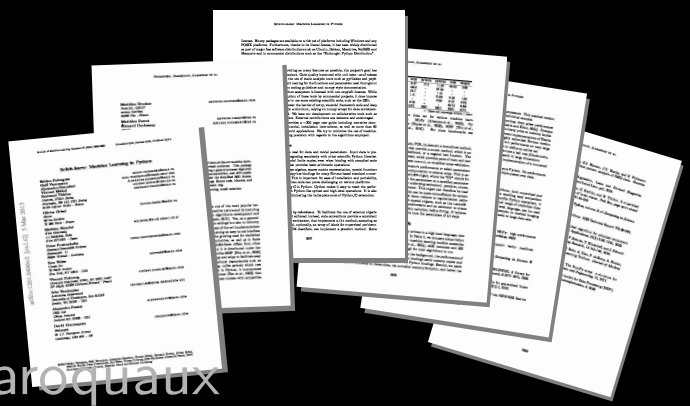
```
test_score = estimator.score(X_test)
```

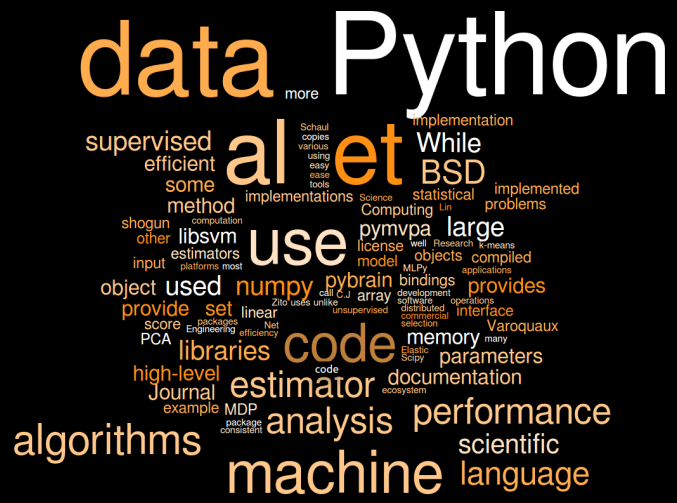
2 Vectorizing: from raw data to a sample matrix X

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

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

X = hasher.fit_transform(documents)
```





2 Very rich feature set: 160 estimators

Supervised learning

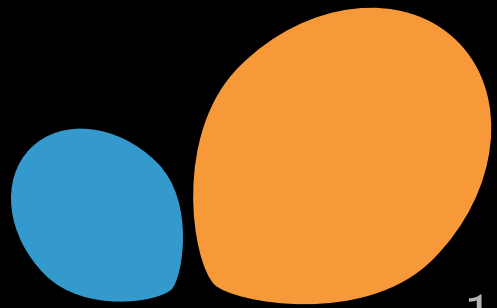
- Decision trees (Random-Forest, Boosted Tree)
- Linear models
- SVM
- Gaussian processes
- ...

Unsupervised Learning

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

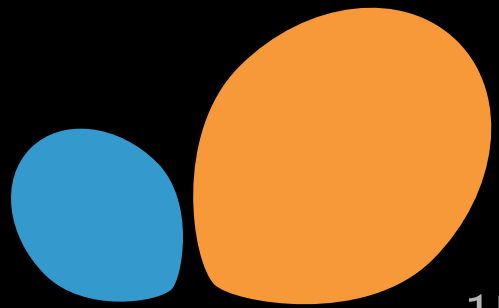
Model selection

- Cross-validation
- Parameter optimization



2 Models most used in scikit-learn

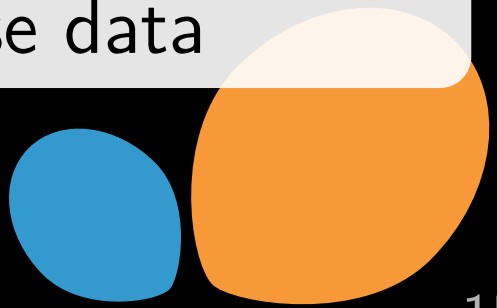
1. Logistic regression, SVM
   SAG/SAGA solver
2. Random forests
   Parallel computing, feature importance
3. PCA
   Randomized linear algebra, online solver
4. Kmeans
   KMeans++ initialization, Elkan solver, online solver
5. Naive Bayes
   On-line solver
6. Nearest neighbor
   KD-tree / ball-tree



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
6. Nearest neighbor
   KD-tree / ball-tree

- Multi-class / multi-output
- Dense or sparse data

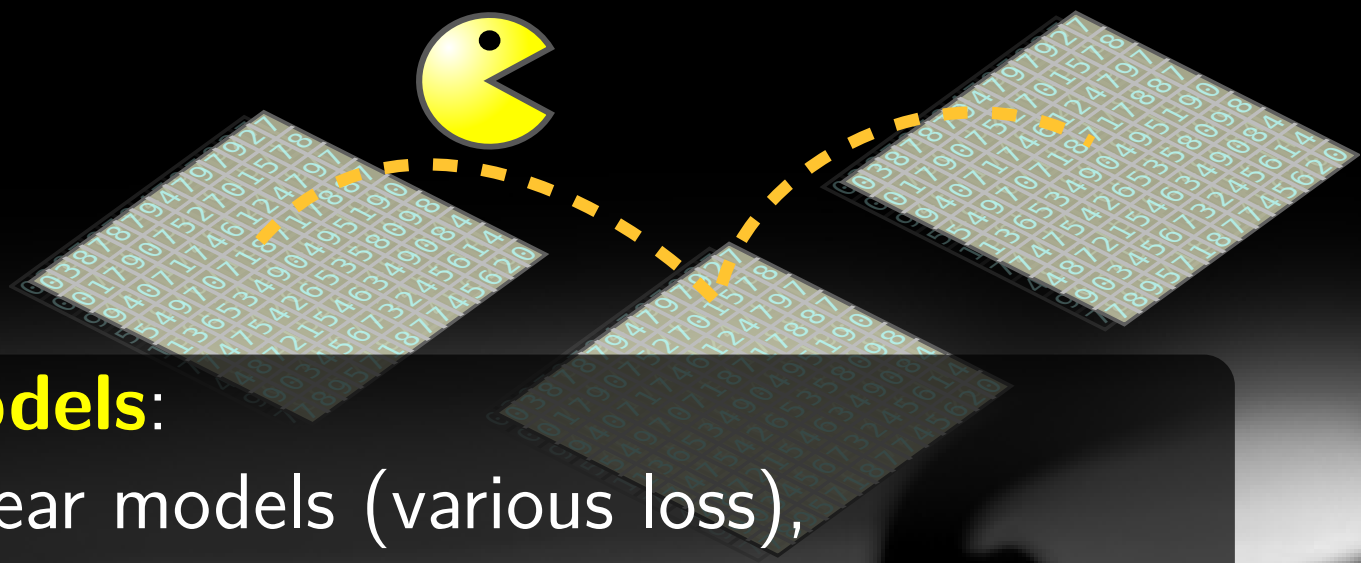


“Big” data

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, Dictionnary 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

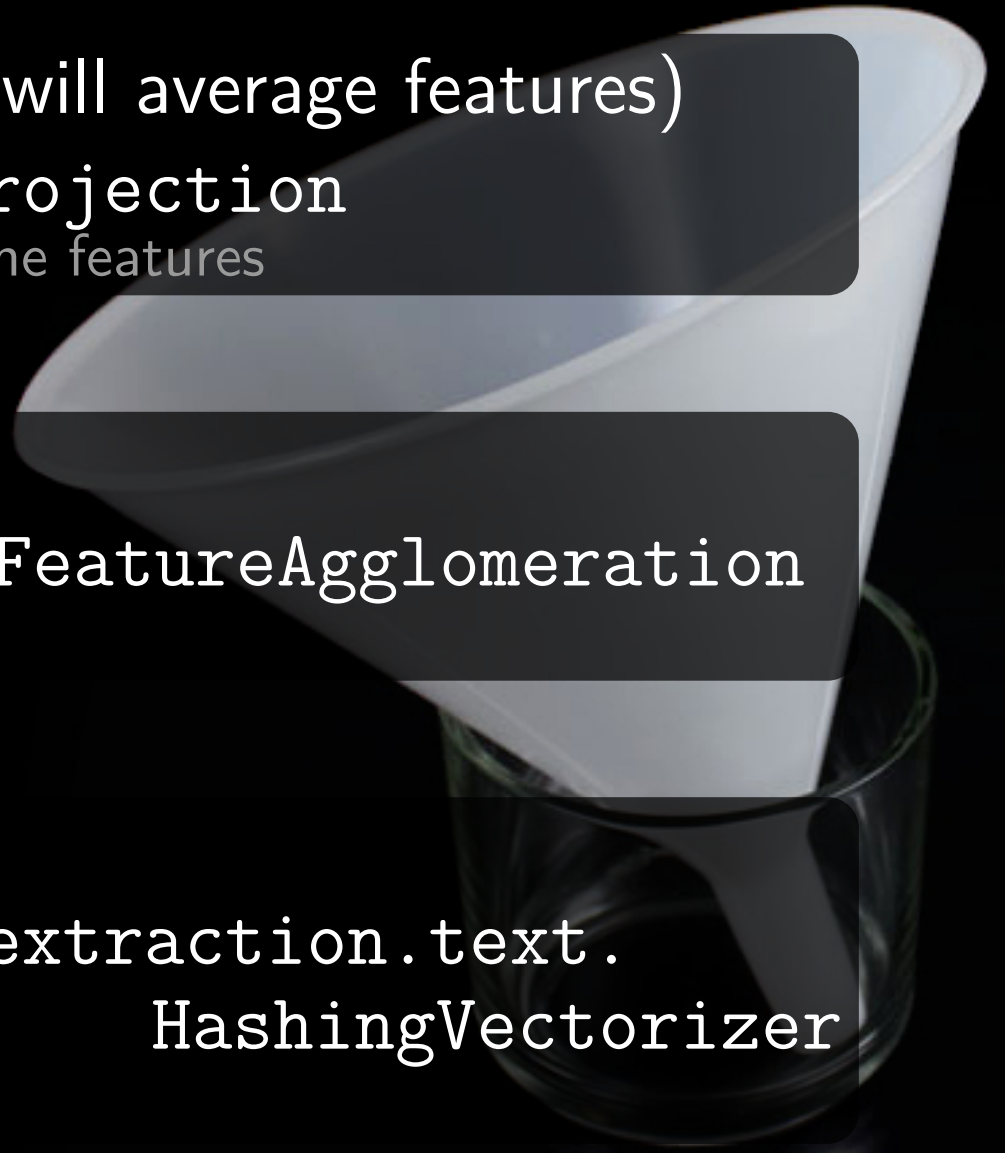
`sklearn.cluster.FeatureAgglomeration`

on images: super-pixel strategy

Feature hashing

`sklearn.feature_extraction.text.HashingVectorizer`

stateless: can be used in parallel

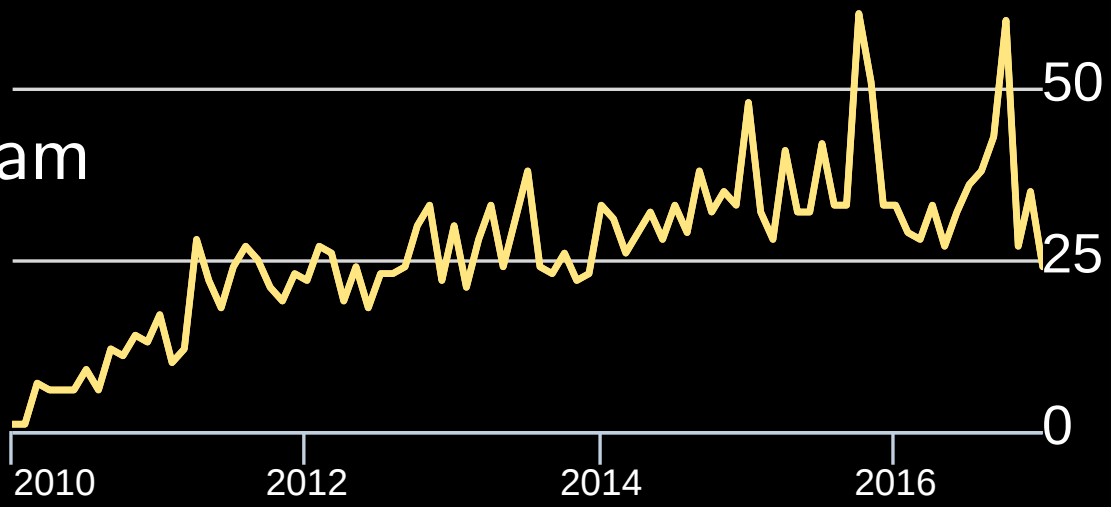


3 The development

3 Community-based development in scikit-learn

Huge feature set:
benefits of a large team

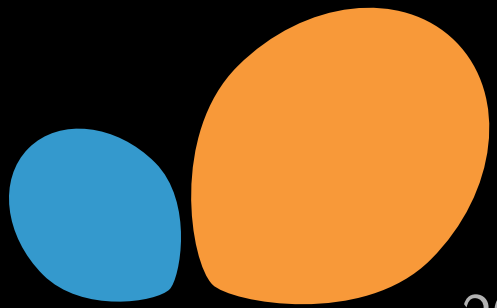
Project growth:



- More than 400 contributors
- $\sim$ 20 core contributors

`https://www.openhub.net/p/scikit-learn`

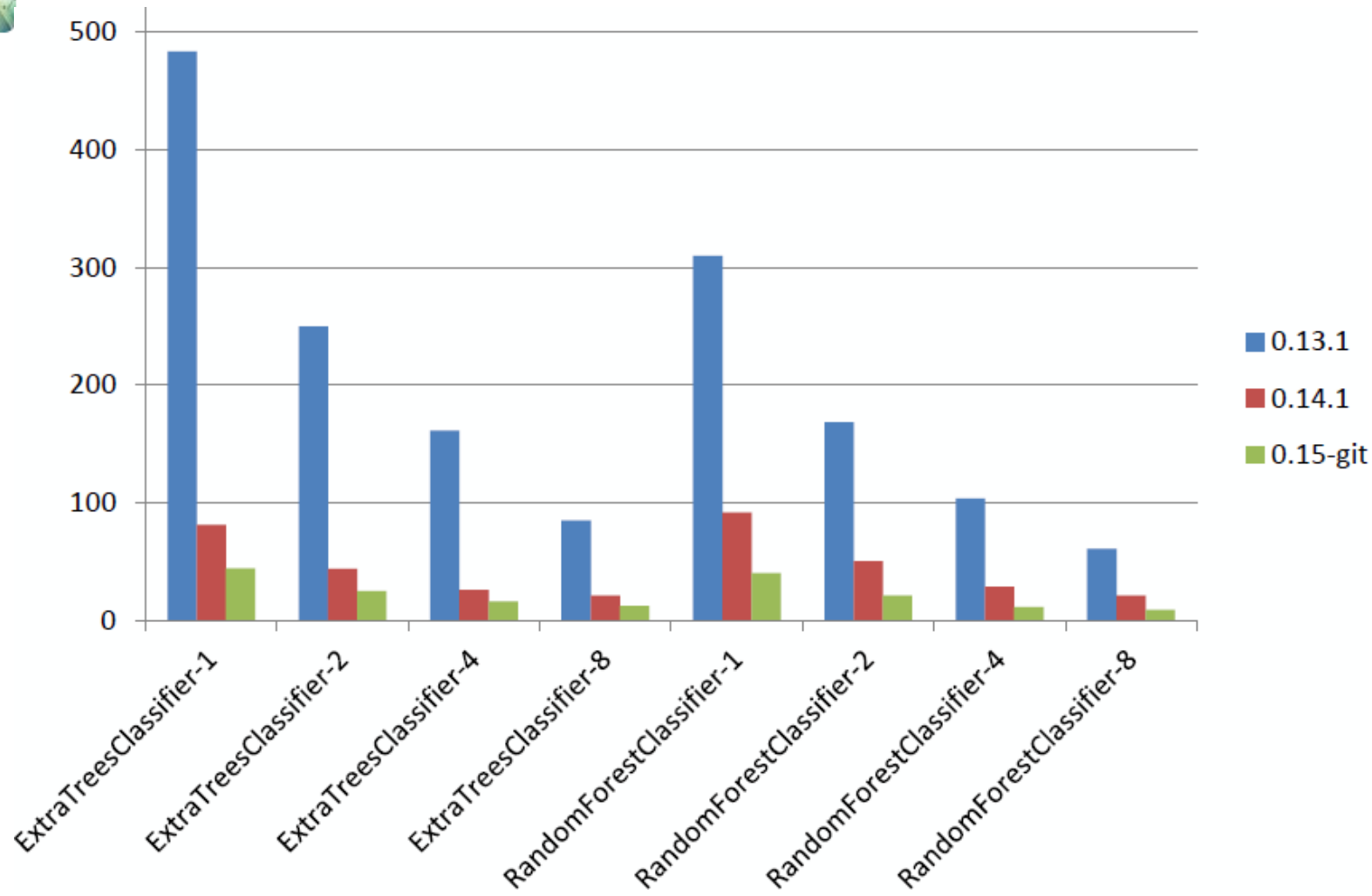
Community-driven project



3 Many eyes makes code fast

Gilles Louppe @glouppe · Feb 18

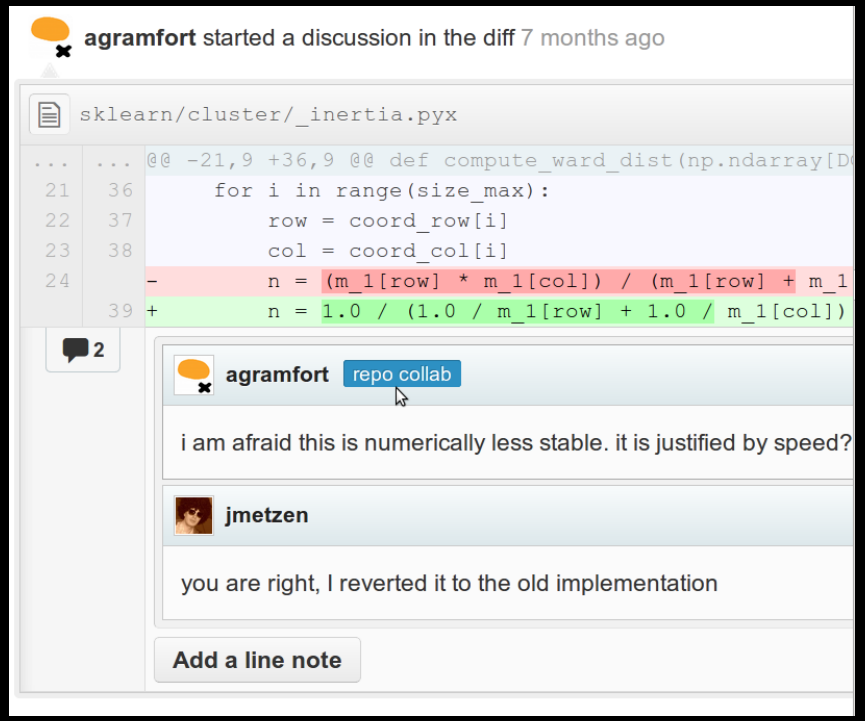
Speed improvement from 0.13 to 0.15-git of Random Forests in Scikit-Learn:



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

Code review: pull requests

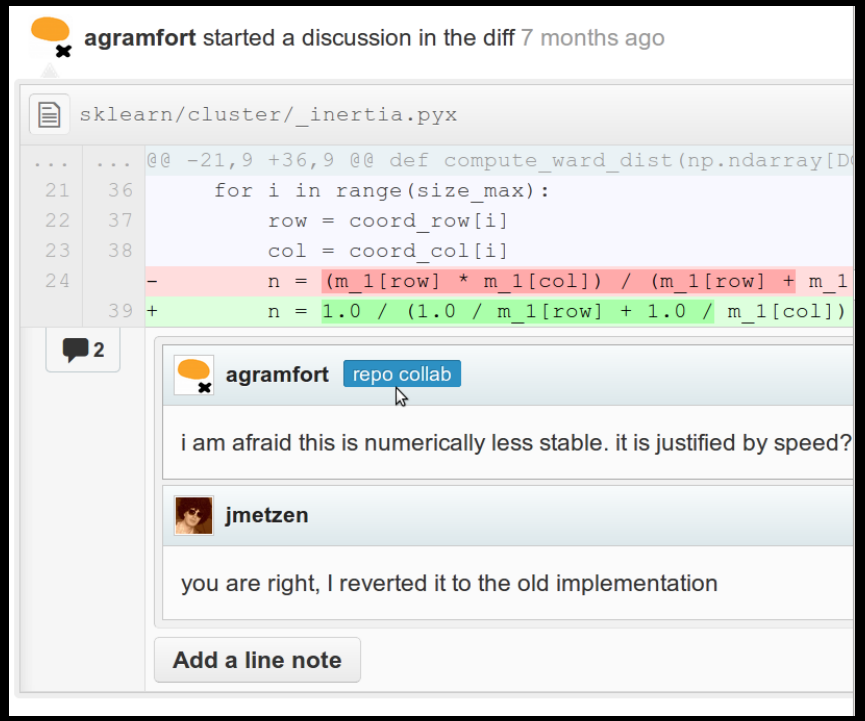
- We read each others code
- Everything is discussed:
  - Should the algorithm go in?
  - Are there good defaults?
  - Are the numerics stable?
  - Could it be faster?



3 Quality assurance

Code review: pull requests

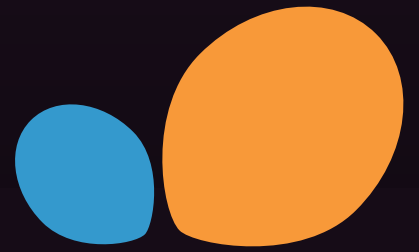
- We read each others code
- Everything is discussed:
  - Should the algorithm go in?
  - Are there good defaults?
  - Are the numerics stable?
  - Could it be faster?



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