Gaël Varoquaux

scikit learn

machine learning in Python

1. **Scikit-learn**

2. **Better machine learning**

1 Scikit-learn

A Python library for machine learning

scikit learn

machine learning in Python

©Theodore W. Gray

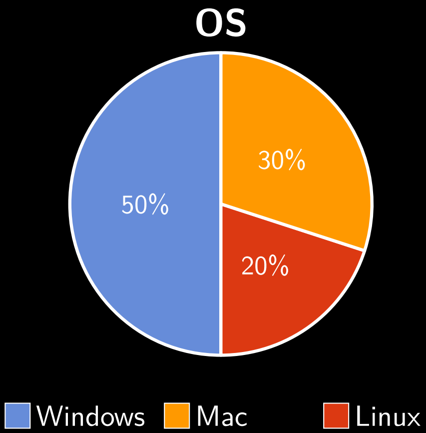
Outreach

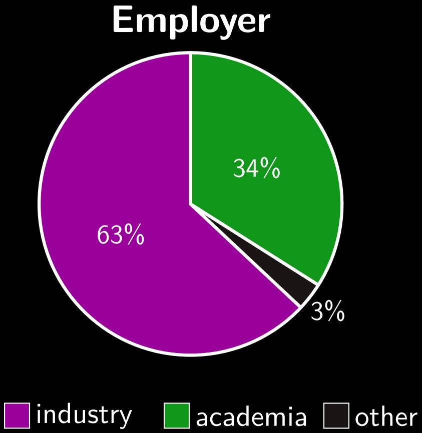
across scientific fields,
applications, communities

Enabling

foster innovation

350 000 returning users **5 000 citations**





1 A Python library

Python

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

Python's virtual machine is rudimentary

Enables low-level computation

and coupling to numerical libraries

Python

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

Great scientific libraries

- `numpy` arrays = wrappers on C pointers

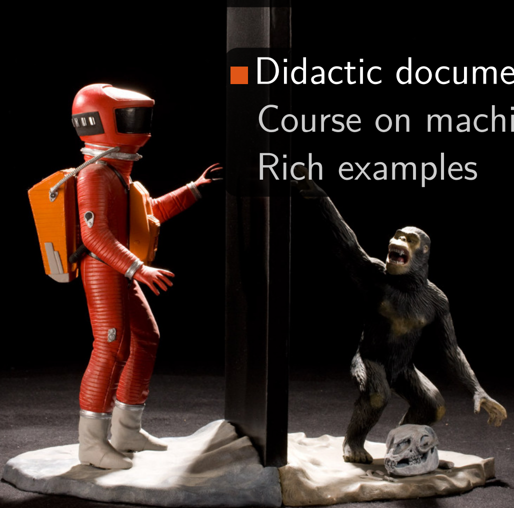
  Reshaping with minimal copies

  Semantics of operations
- `scipy`: numerical methods and fortran packs
- `pandas`: columnar data

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

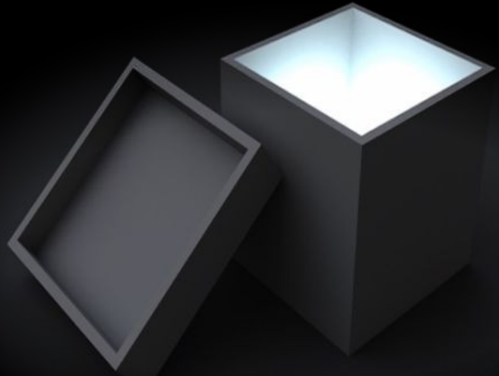
- **Didactic documentation**
  Course on machine learning
  Rich examples



The greybox model

Building bricks

- to combine with domain-specific knowledge
- interchangeable (mostly)



The greybox model

```python
from sklearn import svm
classifier = svm.SVC()
classifier.fit(X_train, Y_train)
Y_test = classifier.predict(X_test)
# or
X_red = classifier.transform(X_test)
```

Access to the model's inner parameters

```python
coef = classifier.coef_
```

1 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
- ...

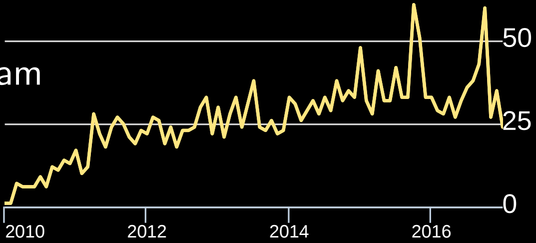
Model selection

- Cross-validation
- Parameter optimization

1 Community-based development in scikit-learn

Huge feature set:
benefits of a large team

Project growth:



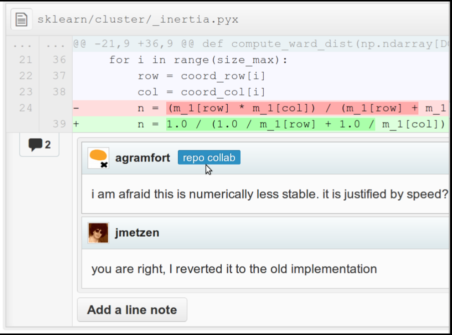
- More than 400 contributors
- $\sim$ 20 core contributors

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

Community-driven project

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

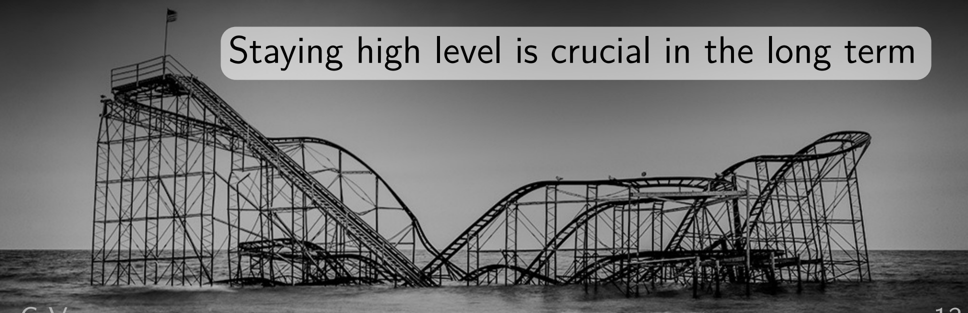
- Everything is tested
  Continuous integration
  If it's not tested, it's broken
- Test API
  Test as grey box
- Test numerics
  Check mathematical properties
  (*eg* decrease of energy)
- Tests should run fast
- Perfect control of randomness

I prefer C-- to C

C without `malloc`, `free`, and pointer arithmetics

Cython

- typed Python syntax
- generates C code running in the Python virtual machine
- native support for numpy arrays



“Big” data

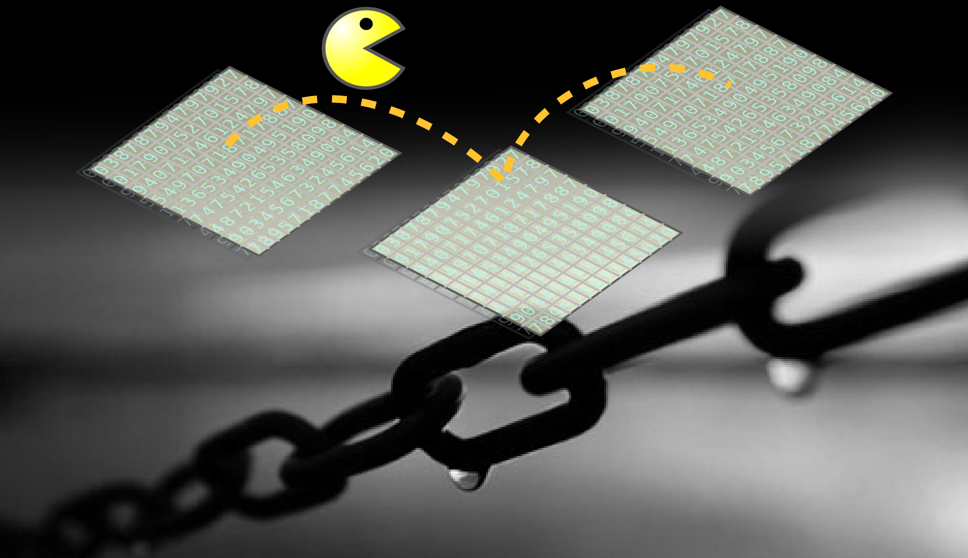
Engineering efficient processing pipelines

Many samples or **Many features**

See also: http://www.slideshare.net/GaelVaroquaux/processing-biggish-data-on-commodity-hardware-simple-python-patterns

1 Many samples: on-line algorithms

```
estimator.partial_fit(X, y)
```



1 Many samples: on-line algorithms

```
estimator.partial_fit(X, y)
```

Supervised models: predicting

`sklearn.naive_bayes...`

`sklearn.linear_model.SGDRegressor`

`sklearn.linear_model.SGDClassifier`

Clustering: grouping samples

`sklearn.cluster.MiniBatchKMeans`

`sklearn.cluster.Birch`

Linear decompositions: finding new representations

`sklearn.decompositions.IncrementalPCA`

`sklearn.decompositions.MiniBatchDictionaryLearning`

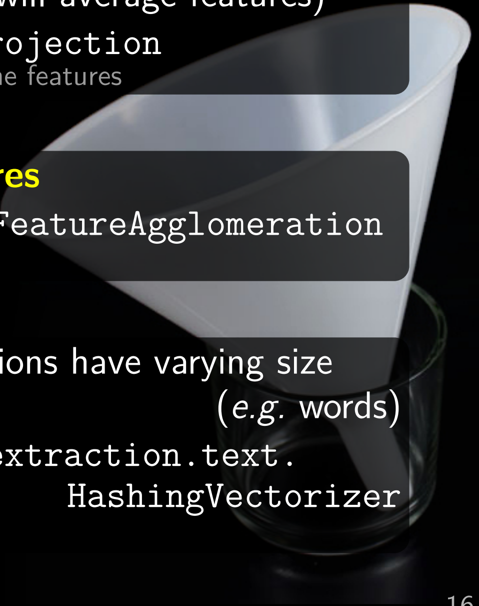
`sklearn.decompositions.LatentDirichletAllocation`

1 Many features: on-the-fly data reduction

⇒ Reduce the data as it is loaded

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

1 Many features: on-the-fly data reduction



Random projections (will average features)

`sklearn.random_projection`

random linear combinations of the features

Fast clustering of features

`sklearn.cluster.FeatureAgglomeration`

on images: super-pixel strategy

Hashing when observations have varying size (*e.g.* words)

`sklearn.feature_extraction.text.HashingVectorizer`

stateless: can be used in parallel

More gems in scikit-learn

SAG:

`linear_model.LogisticRegression(solver='sag')`

Fast linear model on biggish data

More gems in scikit-learn

SAG:

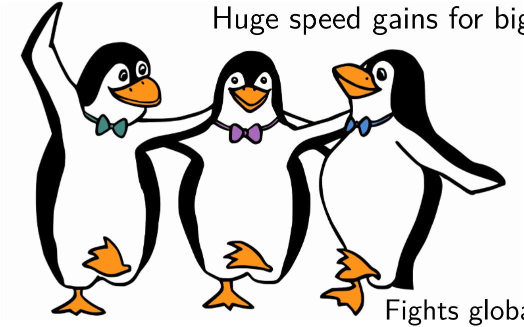
```
linear_model.LogisticRegression(solver='sag')
```

Fast linear model on biggish data

PCA == RandomizedPCA: (0.18)

Heuristic to switch PCA to random linear algebra

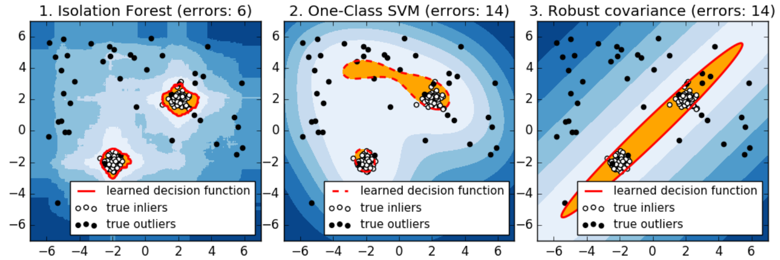
Huge speed gains for biggish data



Fights global warming

More gems in scikit-learn

Outlier detection and isolation forests (0.18)



2 Better machine learning

Thoughts on the future

Usability and engineering of machine learning

2 Models most used in scikit-learn

1. Logistic regression, SVM
2. Random forests
3. PCA
4. Kmeans
5. Naive Bayes
6. Nearest neighbors

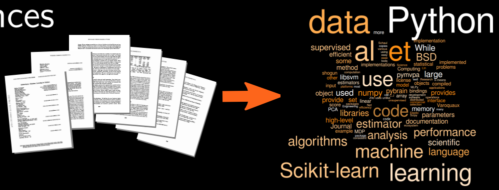
From access statistics on the website

2 Addressing the needs of our users

- Easier data integration
- Bigger data
- Faster models

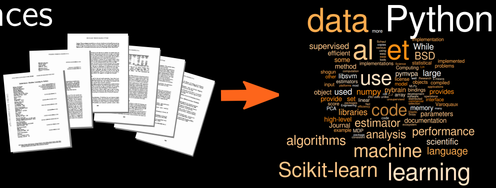
Vectorizing: create a numerical matrix

- For text data: list of strings
  - counting word occurences



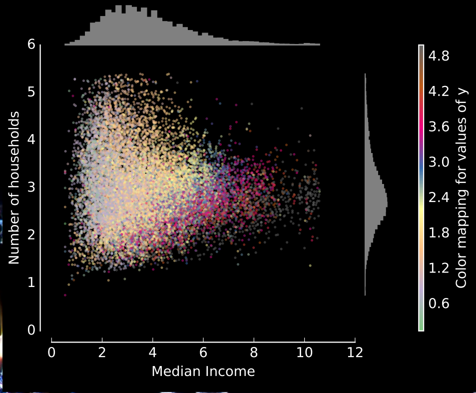
Vectorizing: create a numerical matrix

- For text data: list of strings
  - counting word occurences
  - word embeddings

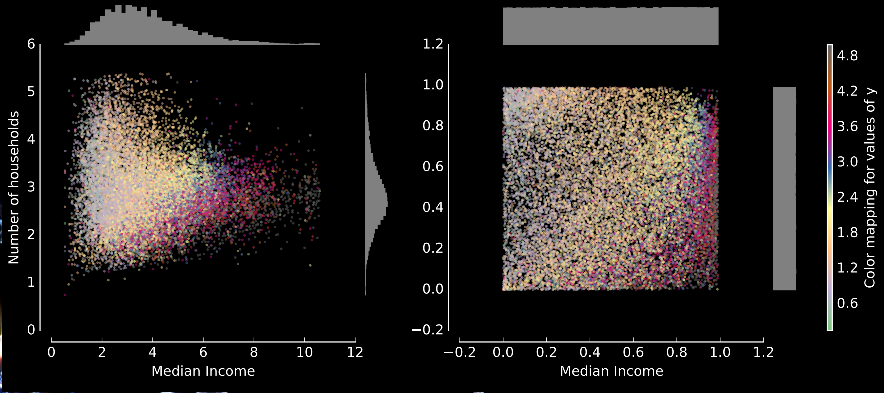


- For pandas dataframes
  - dealing with heterogeneous data types
  - one-hot encoding of categorical data

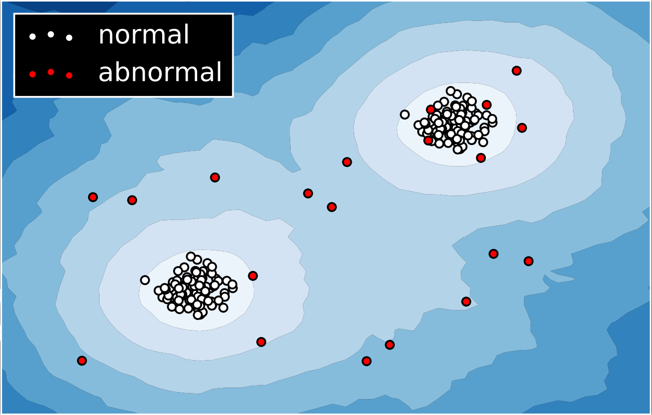
Quantile transformer:



Quantile transformer:



- **Quantile transformer**:
- **Local outlier factor**:



- **ColumnsTransformer**:
  Pandas in ... feature engineering ... array out

```
transformer = make_column_transformer({
                StandardScaler():  ['age'],
                OneHotEncoder():  ['company']
             })

array = transformer.fit_transform(data_frame)
```

2 Bigger faster models coming soon

- **Memory in pipeline**:

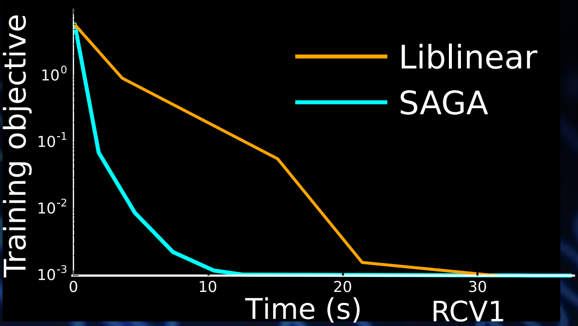
```
make_pipeline(PCA(), LinearSVC(), memory='/tmp/joe')
```

Limits recomputation (*eg* in grid search)

- **Memory in pipeline**
- **New solver for logistic regression**: SAGA

```
linear_model.LogisticRegression(solver='saga')
```

Fast linear model on biggish data



2 Bigger faster models coming soon

- **Memory in pipeline**
- **New solver for logistic regression**: SAGA
- **Memory savings**
  - Avoid casting (work with `float32`)
  - T-SNE (in progress)

- **Faster trees, forest& boosting**:

  Teaching from `XGBoost`, `lightgbm`:
  - bin features for discrete values
  - depth-first tree, for access locality

Implementing machine learning

Huge amount of engineering

- Minimizing memory copies
- Multiple data types (sparse, float32, float64...)
- Minimizing and understanding failure modes

...

Implementation quality
often matters more than algorithmics

Infrastructure: we want to use it and forget it
It needs maintenance and investment

@GaelVaroquaux

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