

Scikit-learn: machine learning in Python

Gaël Varoquaux

inria



machine learning in Python



1 Scikit-learn

2 Better machine learning

1 Scikit-learn

A Python library for machine learning



©Theodore W. Gray

Outreach

across scientific fields,
applications, communities

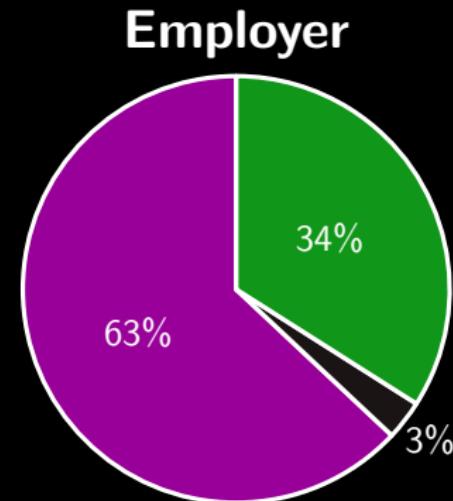
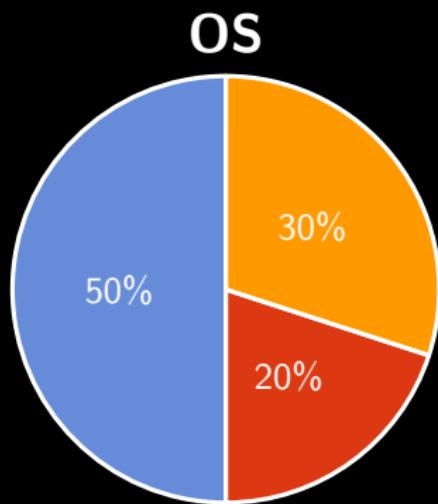
Enabling
foster innovation



1 scikit-learn user base

350 000 returning users

5 000 citations



■ Windows ■ Mac

■ Linux

■ industry ■ academia ■ other

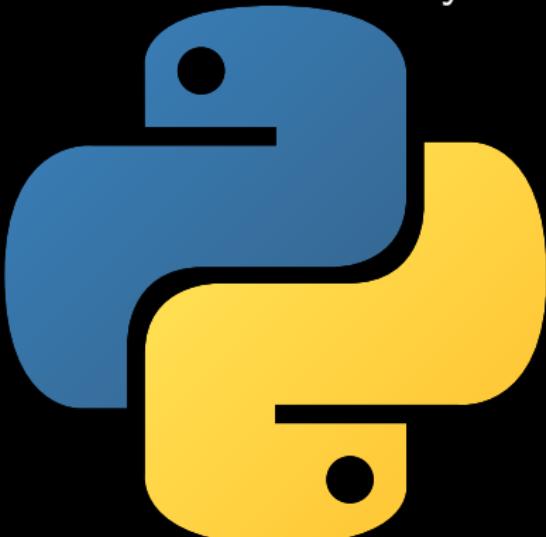
1 A Python library

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

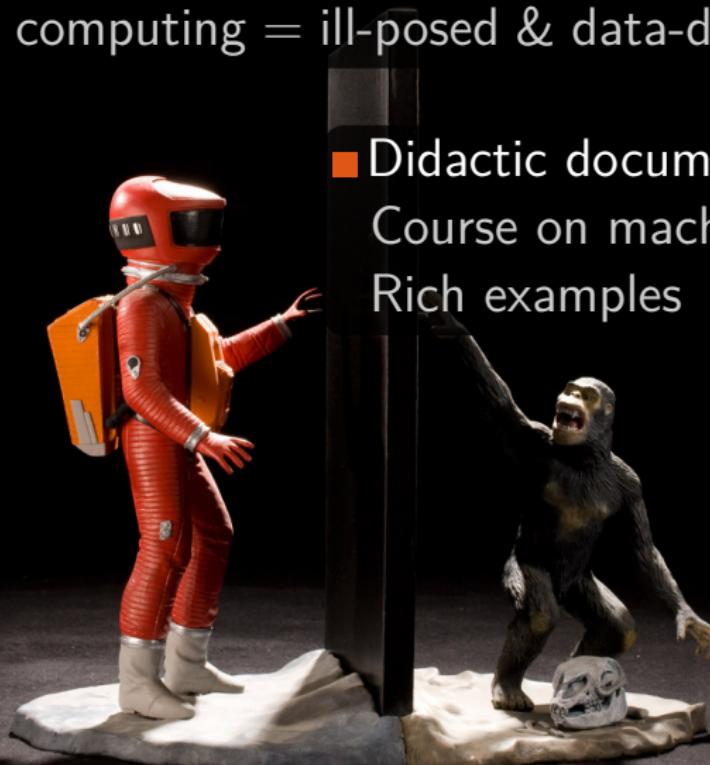


Great scientific libraries

- 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

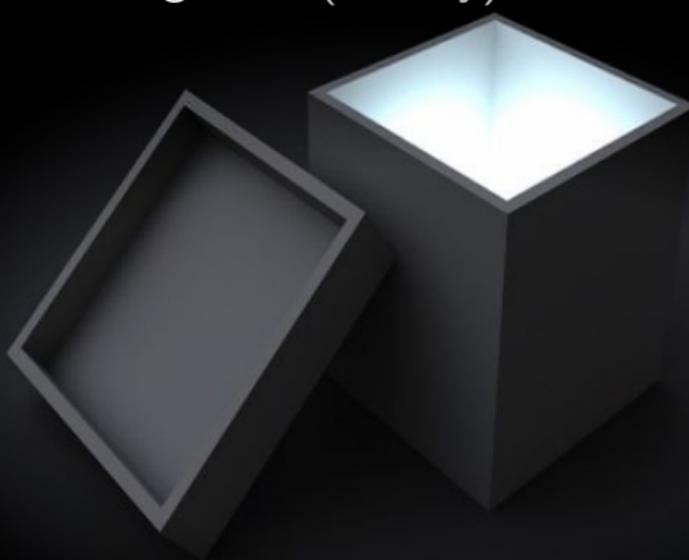
- Algorithms and models with good failure mode
Avoid parameters hard to set or fragile convergence
Statistical computing = ill-posed & data-dependent
- Didactic documentation
Course on machine learning
Rich examples



The greybox model

Building bricks

to combine with domain-specific knowledge
interchangeable (mostly)



The greybox model

```
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

```
coef = classifier.coef_
```

Supervised learning

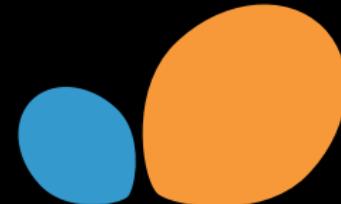
- Decision trees (Random-Forest, Boosted Tree)
- Linear models ■ SVM
- Gaussian processes ...

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
- ~ 20 core contributors



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

Community-driven project

1 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. At the top, there's a code diff between two versions of a file named `sklearn/cluster/_inertia.pyx`. The diff highlights changes in lines 24 and 39. Line 24 shows a subtraction operation being replaced by a division operation. Line 39 shows a division operation being replaced by a reciprocal operation. Below the code, there are two comments:

- A comment from **agramfort** (repo collab) stating: "i am afraid this is numerically less stable. it is justified by speed?"
- A reply from **jmetzen** stating: "you are right, I reverted it to the old implementation"

At the bottom, there's a button labeled "Add a line note".

```
... ... @@ -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))
```

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
 - 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



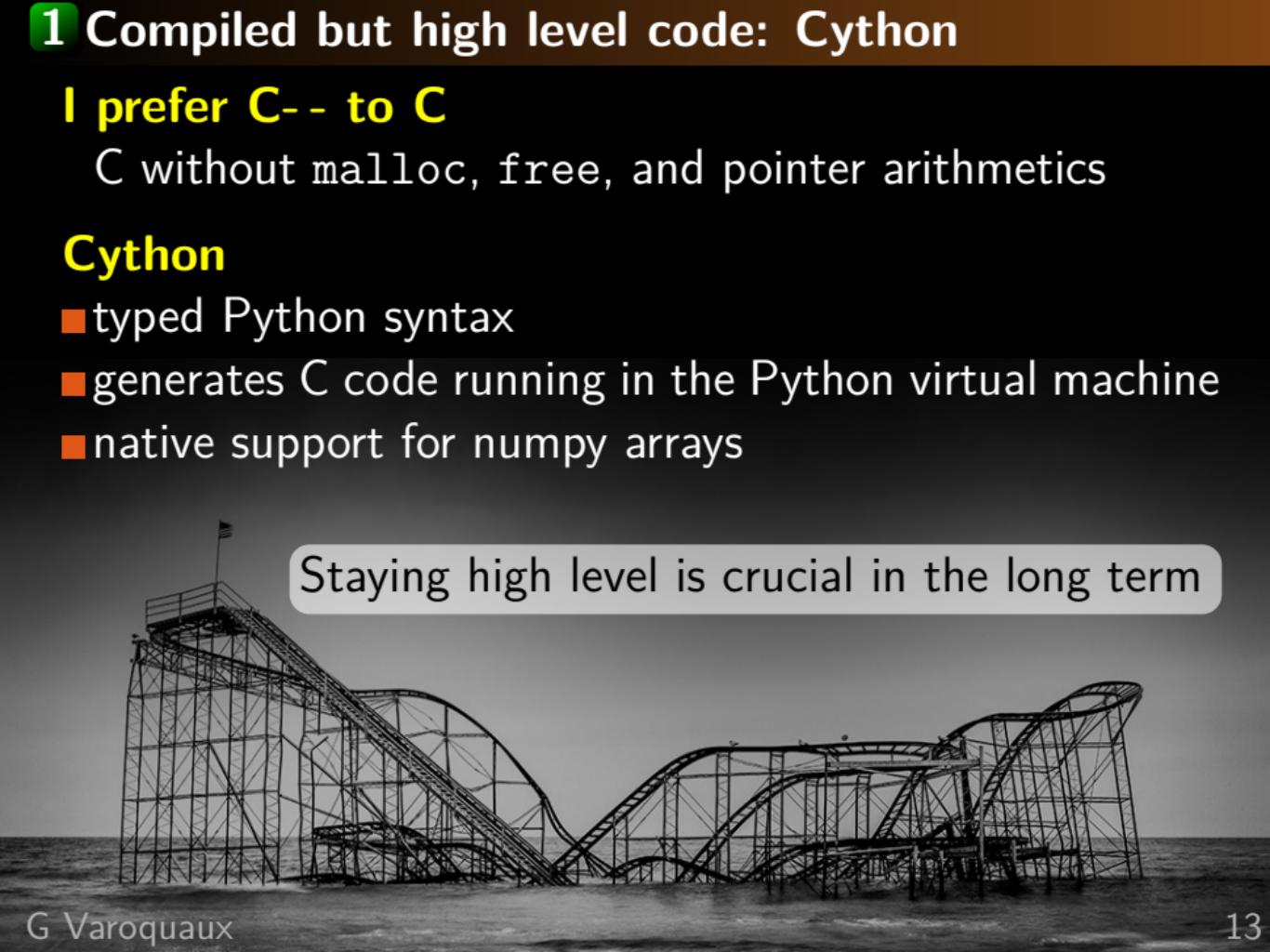
1 Compiled but high level code: Cython

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



Staying high level is crucial in the long term

“Big” data

Engineering efficient processing pipelines

Many samples

or Many features

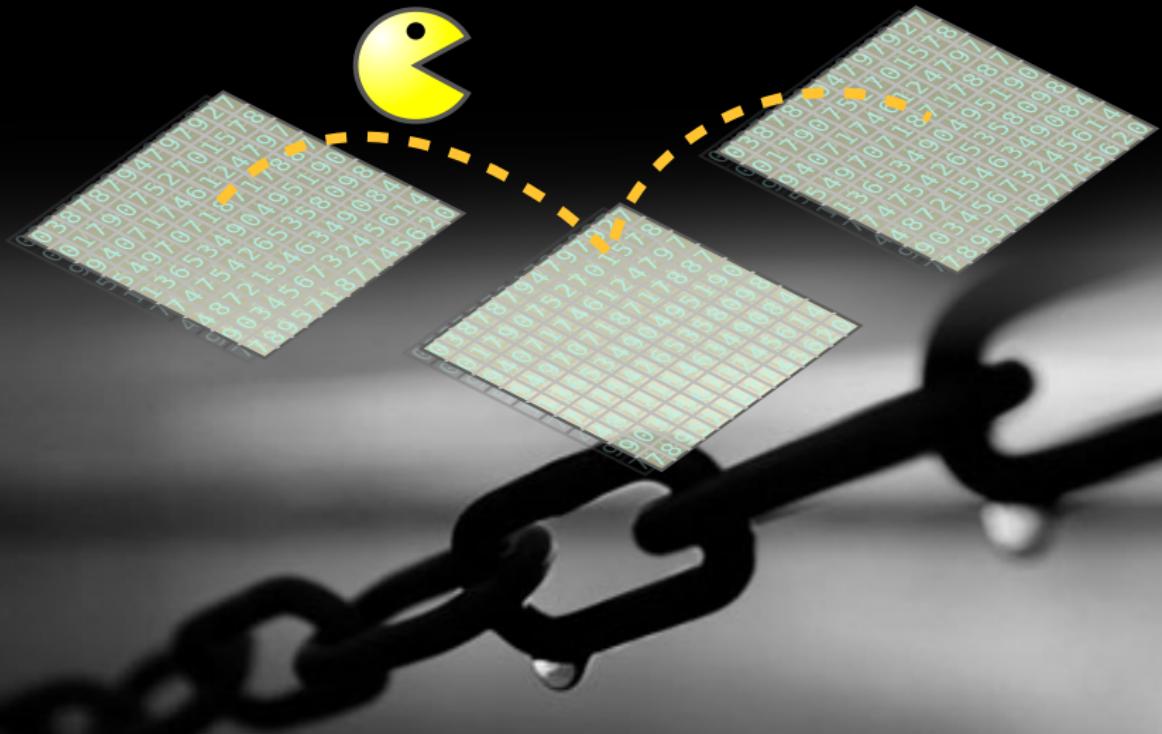
samples	features
	03078090707907
	00790752700578
	94071006000797
	00970008007000
	10000400400090
	00050205008000
	03078090707907
	00790752700578
	94071006000797
	00970008007000
	10000400400090
	00050205008000

samples	features
	0307809070790703078090707907
	00790752700578007907952700578
	9407100600079794071006000797
	009700080070000970008007000
	1000040040009010000400400090
	000502050080000050205008000

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:

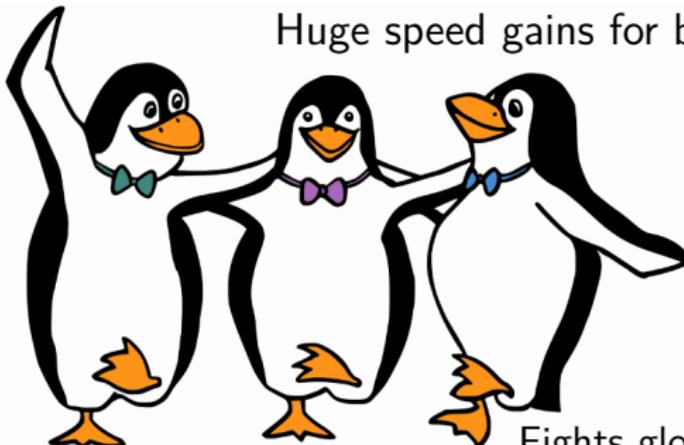
```
linear_model.LogisticRegression(solver='sag')
```

Fast linear model on biggish data

PCA == RandomizedPCA: (0.18)

Heuristic to switch PCA to random linear algebra

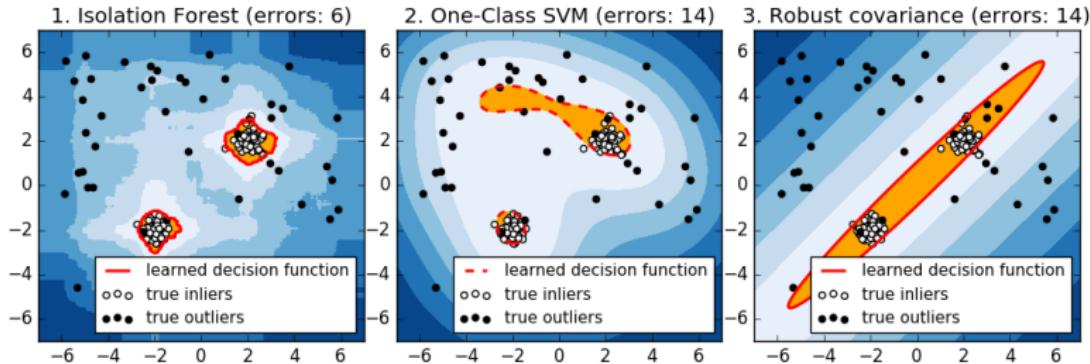
Huge speed gains for biggish data



Fights global warming

More gems in scikit-learn

Outlier detection and isolation forests (0.18)



2 Better machine learning

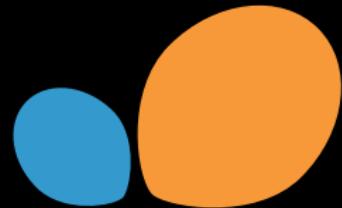
Thoughts on the future

Usability and engineering
of machine learning



2 Models most used in scikit-learn

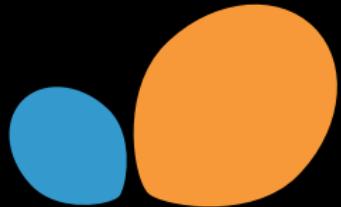
1. Logistic regression, SVM
2. Random forests
3. PCA
4. Kmeans
5. Naive Bayes
6. Nearest neighbors



From access statistics on the website

2 Addressing the needs of our users

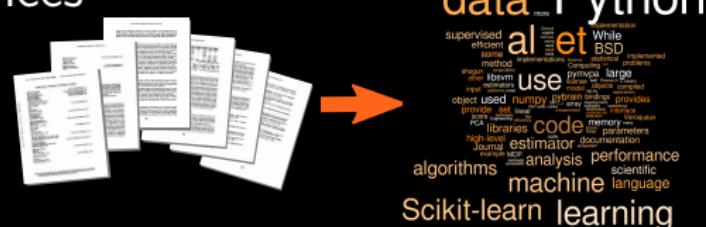
- Easier data integration
- Bigger data
- Faster models



2 Data integration and feature engineering

Vectorizing: create a numerical matrix

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



2 Data integration and feature engineering

Vectorizing: create a numerical matrix

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



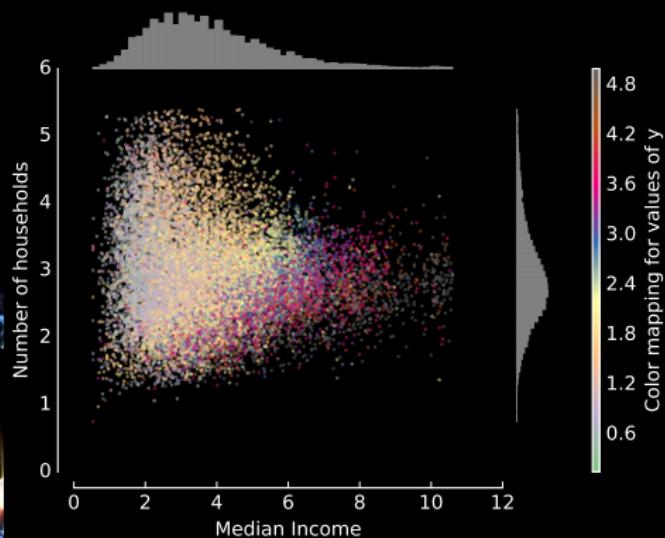
- word embeddings



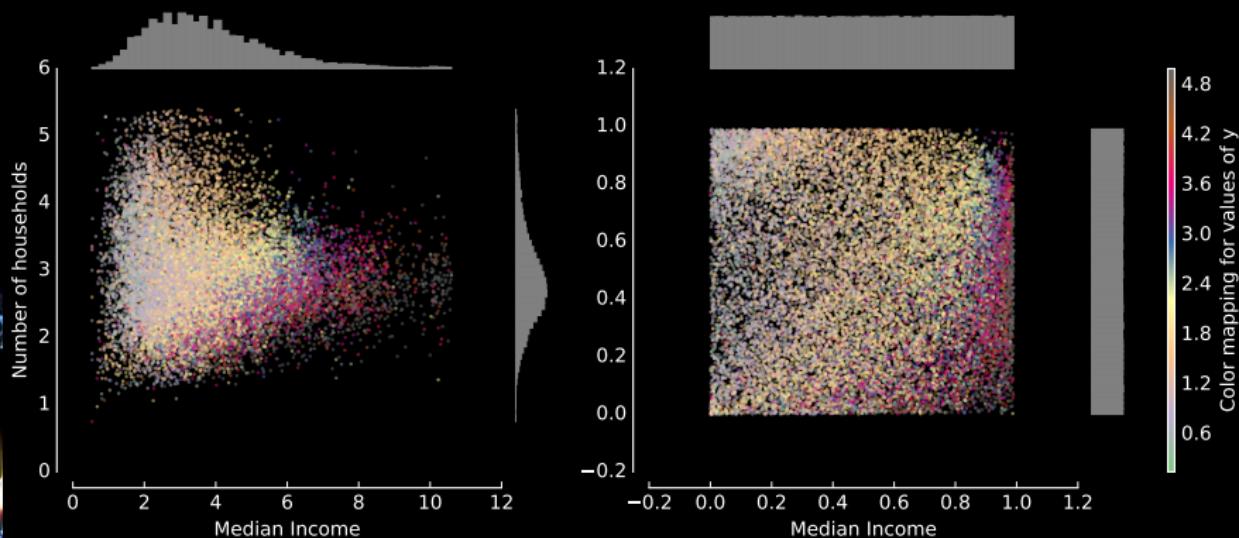
data more Python
supervised efficient While
some implementation BSD
method implemented problems
use al et
use code
object used numpy array
provide memory
Pandas estimator
high-level parameters
Journal analysis performance
algorithms machine scientific language
Scikit-learn learning

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

Quantile transformer:

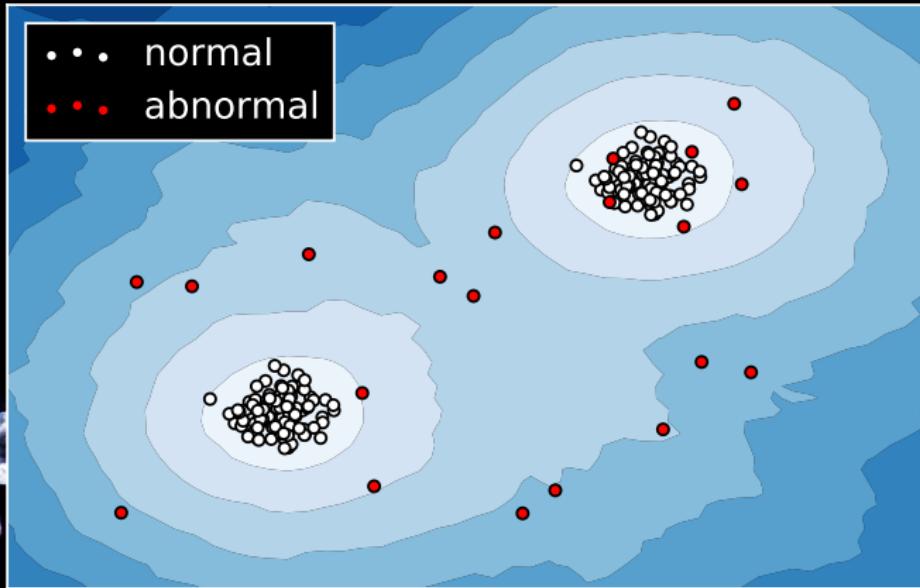


■ Quantile transformer:



- 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)
```

■ Memory in pipeline:

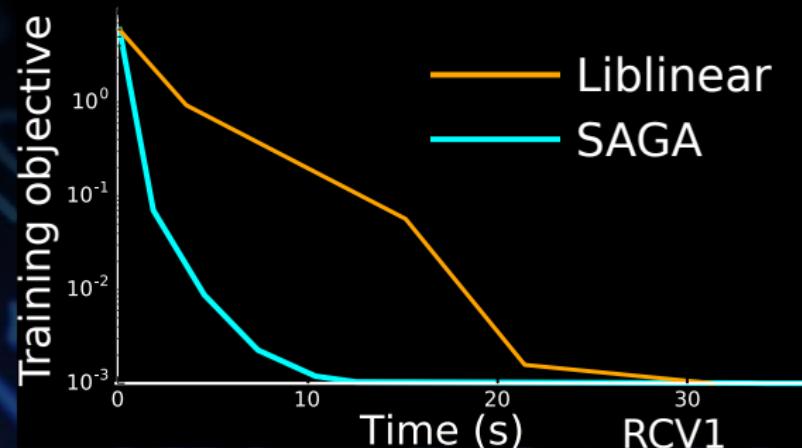
```
make_pipeline(PCA(), LinearSVC(), memory='/tmp/joe')  
    Limits recompilation (eg in grid search)
```

- Memory in pipeline

- New solver for logistic regression: SAGA

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

Fast linear model on biggish data



- 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