MOABB
Motivations and future

Sylvain Chevallier
LISV - Université de Versailles, France
sylvain.chevallier@uvsq.fr

Pedro Rodrigues & Marco Congedo
Gipsa-Lab - Grenoble, France
pedro.rodrigues@gipsa-lab.fr
marco.congedo@gmail.com

16 September 2019

What will this presentation be about?

- Less theoretical work
- Little bit more code explanation
- All you need to know for hands-on session

Outline

MOABB: Why and What

MNE for neuro

Scikit-learn for ML

Pyriemann

Why do we need MOABB?

Reproducible research in BCI has a long way to go...

- Unavailable code
- Exotic data format/language/toolboxes
- Preprocessed data (including errors)

⇒**No comprehensive benchmark of BCI algorithms**
⇒**Huge waste of time for everyone**

⇒**MOABB aims to be the standard benchmark for any new paper**

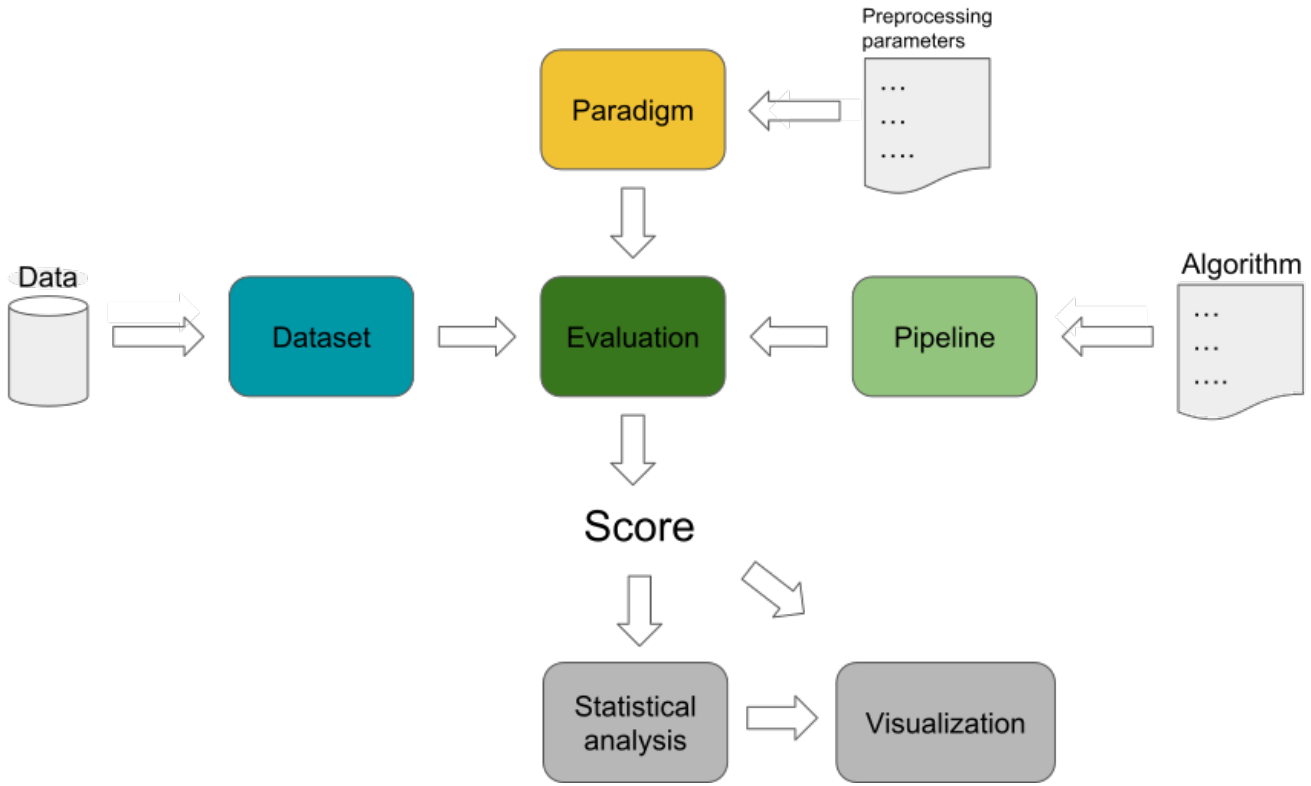
- Comprehensive benchmark of popular BCI algorithm
- Extensive list of freely available EEG datasets
- Ranking algorithms with fair evaluations

Who is behind MOABB?

Founder: **Alexandre Barachant**
Contributors:

- Vinay Jayaram
- Pedro Rodrigues
- Sylvain Chevallier
- Justin D. Harris
- Yannick Roy & the NeuroTechX community
- ...
- You!

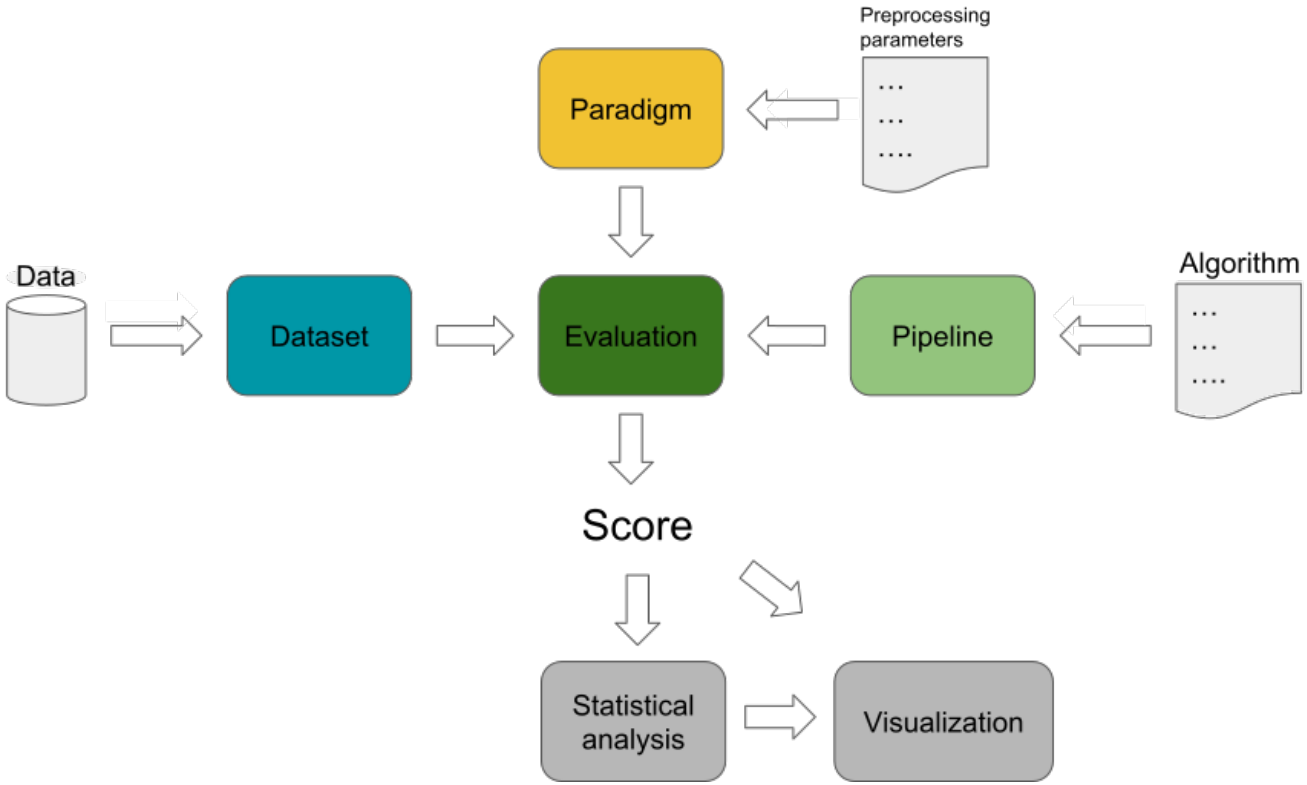
MOABB architecture



Dataset

- Stored locally, converted in MNE format
- Pick only subjects/sessions you need

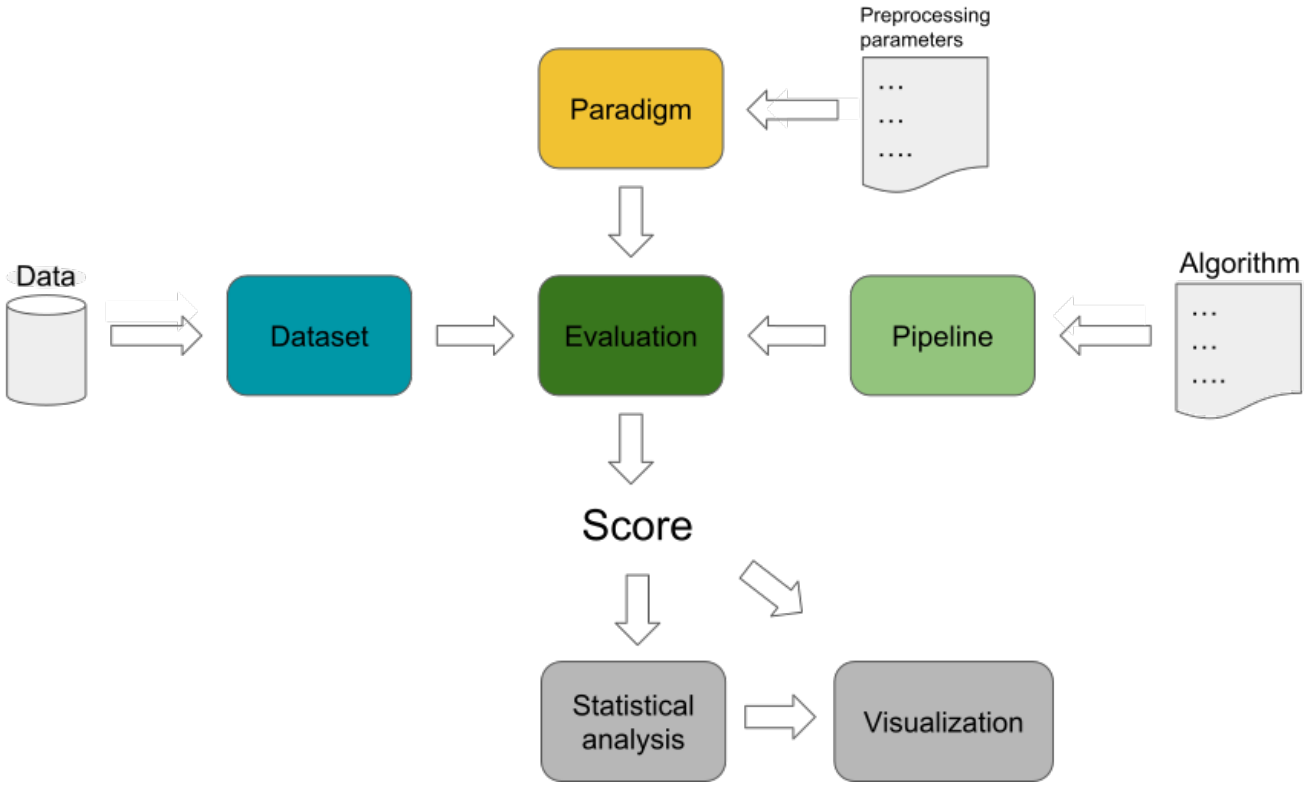
MOABB architecture



Paradigm

- Defines convertion to trial
- Preprocessing

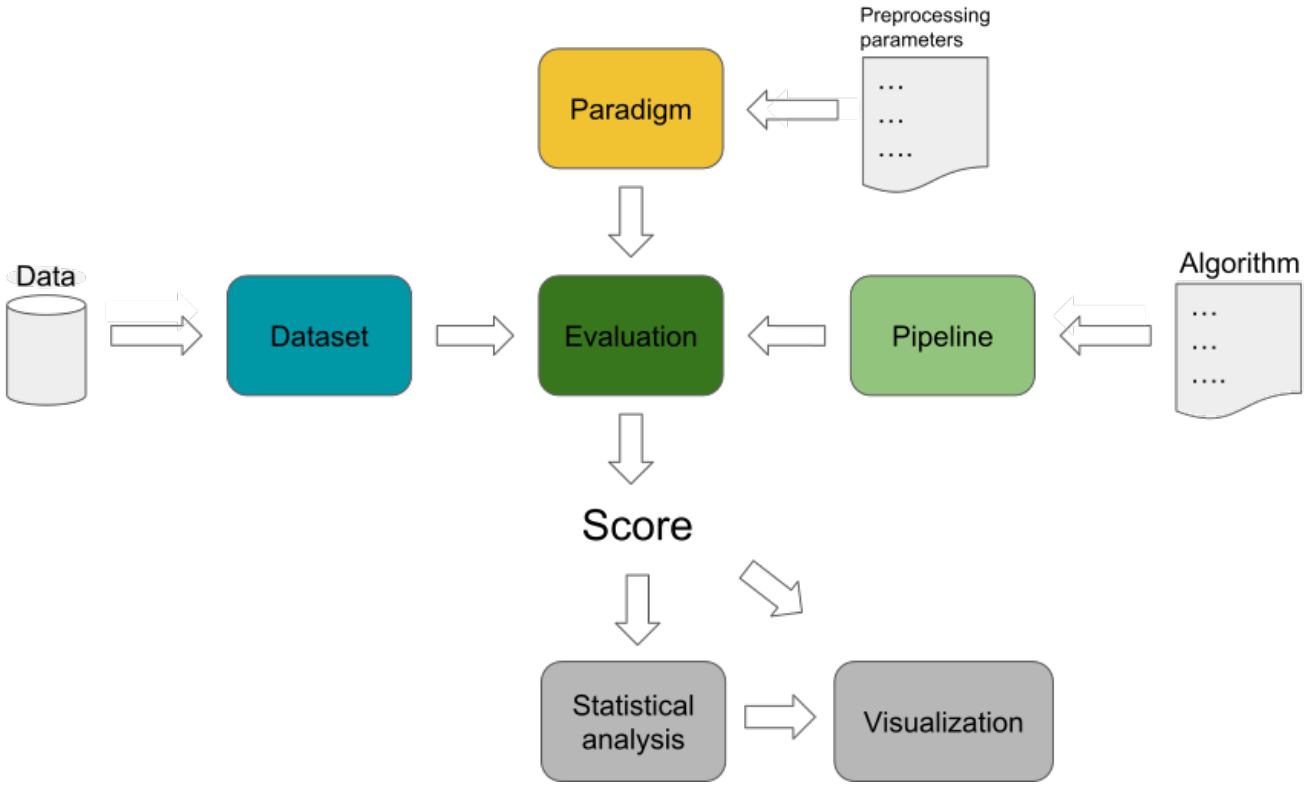
MOABB architecture



Evaluations

- Defines a scorig method (AUC, f-score, ...)
- within or across session, across-subject, ...

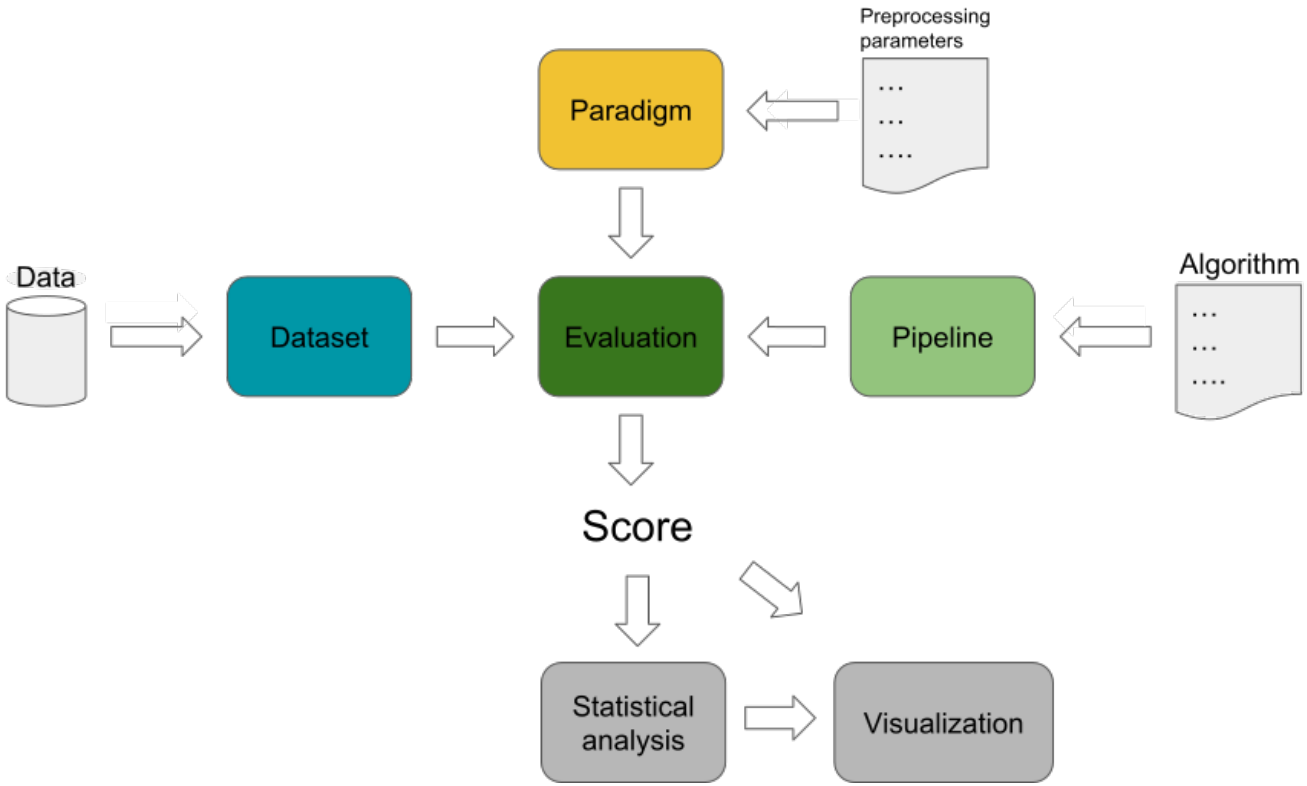
MOABB architecture



Pipelines

- All steps required for obtaining a prediction
- Scikit-learn style

MOABB architecture



Results

- Statistics & visualization
- Results are store in a DataFrame

Outline

MOABB: Why and What

MNE for neuro

Scikit-learn for ML

Pyriemann

Reproducible research with neurophysiological data

Freesurfer Popular software for extracting features from MRI
https://surfer.nmr.mgh.harvard.edu/
⇒Hardware and software differences can lead to different features/statistical results and scientific conclusions

ICA Popular matrix factorization problem
https://github.com/mne-tools/mne-python/issues/4922
⇒Different results with different machines

eigs/eigsh Popular solver for eigenvalues decomposition
https://github.com/scikit-learn/scikit-learn/issues/5545
⇒Even on the same machine numerical solvers can lead to different outcome

MNE

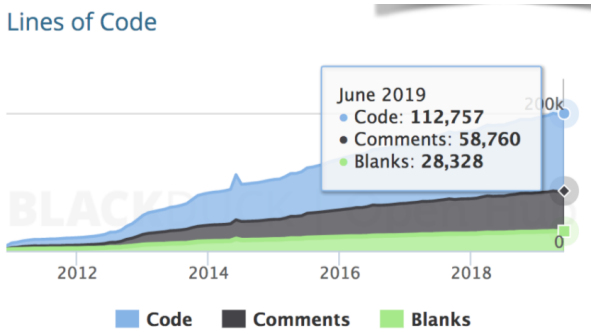
https://github.com/mne-tools/mne-python

History

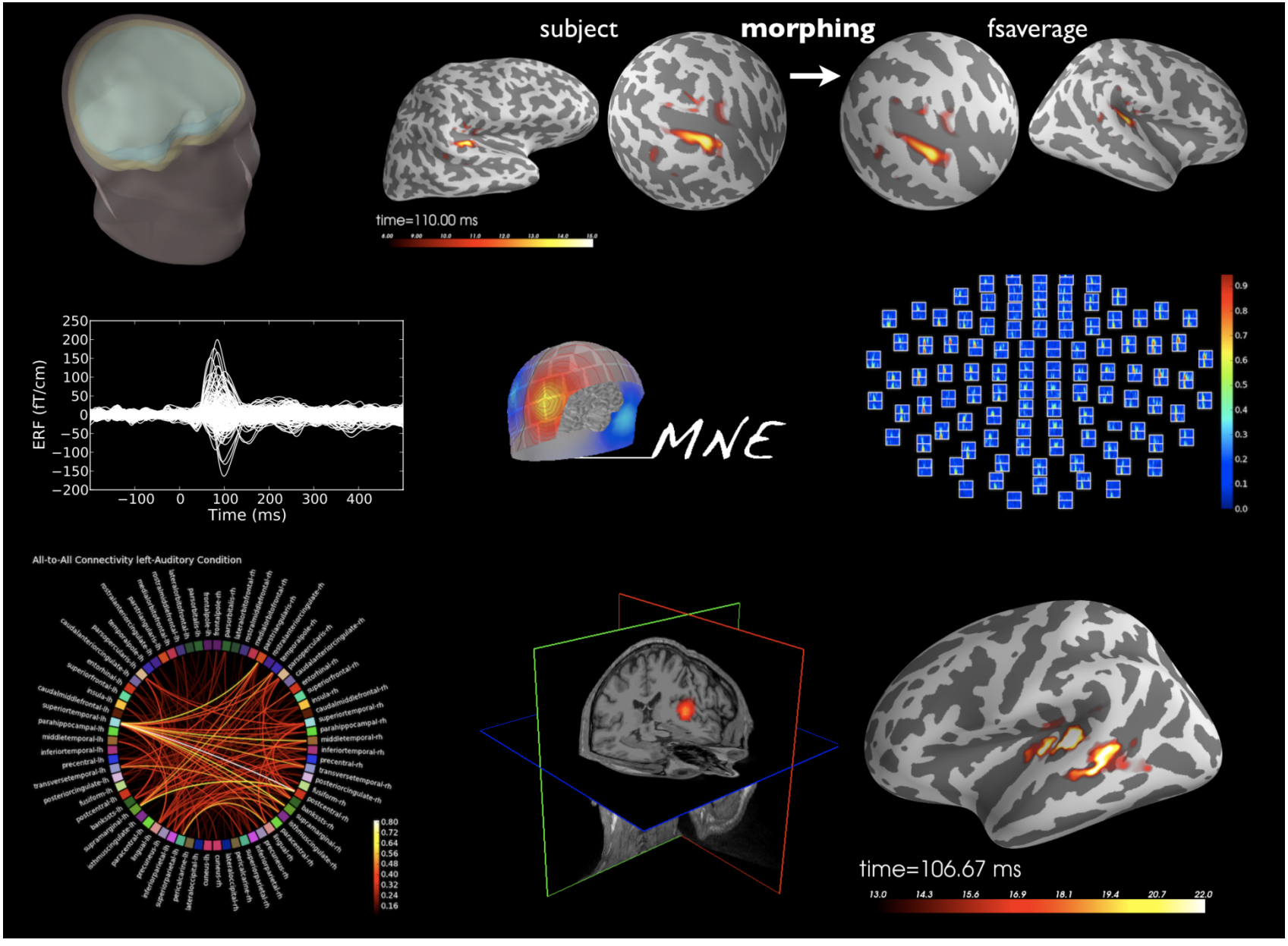
- based on C code developed for 18 years by Matti Hämäläinen
- Python started in 2010 at MGH, Boston

In a nutshell

- 236 contributors, 100k LOC
- mature codebase, large dev team
- 29 years of efforts (COCOMO)



⇒BSD licensed (commercial use ok)
⇒Mac / Linux / Windows


subject
morphing
fsaverage
time=110.00 ms
6.00 9.00 10.0 11.0 12.0 13.0 14.0 15.0
ERF (fT/cm)
Time (ms)
MNE
All-to-All Connectivity left-Auditory Condition
time=106.67 ms
13.0 14.3 15.6 16.9 18.1 19.4 20.7 22.0

Outline

MOABB: Why and What

MNE for neuro

Scikit-learn for ML

Pyriemann

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
université PARIS-SACLAY

Scikit-learn vision

http://scikit-learn.org

- **Machine learning for all**
  ⇒No specific application domain
  ⇒No requirements in machine learning
- **High-quality Pythonic software library**
  ⇒Interfaces designed for users
- **Community-driven development**
  ⇒BSD licensed, very diverse contributors

Easy as py:

```
from sklearn import svm
classifier = svm.SVC()
classifier.fit(X train, Y_train)
Y_test = classifier.predict(X_test)
```

Specifying a model

A central concept: **the estimator**

- Instanciated without data
- But specifying the parameters

Example:

```
from sklearn.neighbors import KNearestNeighbors

estimator = KNearestNeighbors( n_neighbors=2)
```

Training a model

Training from data

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

with:

- `X` a numpy array with shap `n_samples` $\times$ `n_features`
- `y` a numpy 1D array, of ints or float, with shape `n_samples`

Using a model

Prediction classification, regression

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

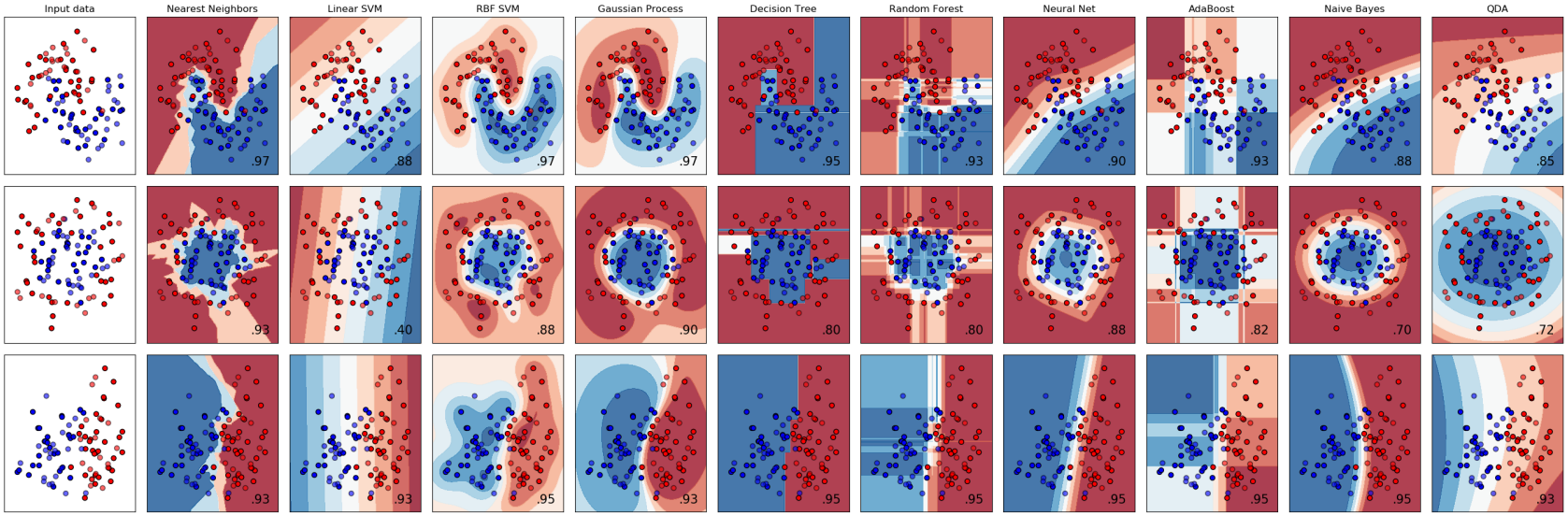
Transforming dimension reduction, filter

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

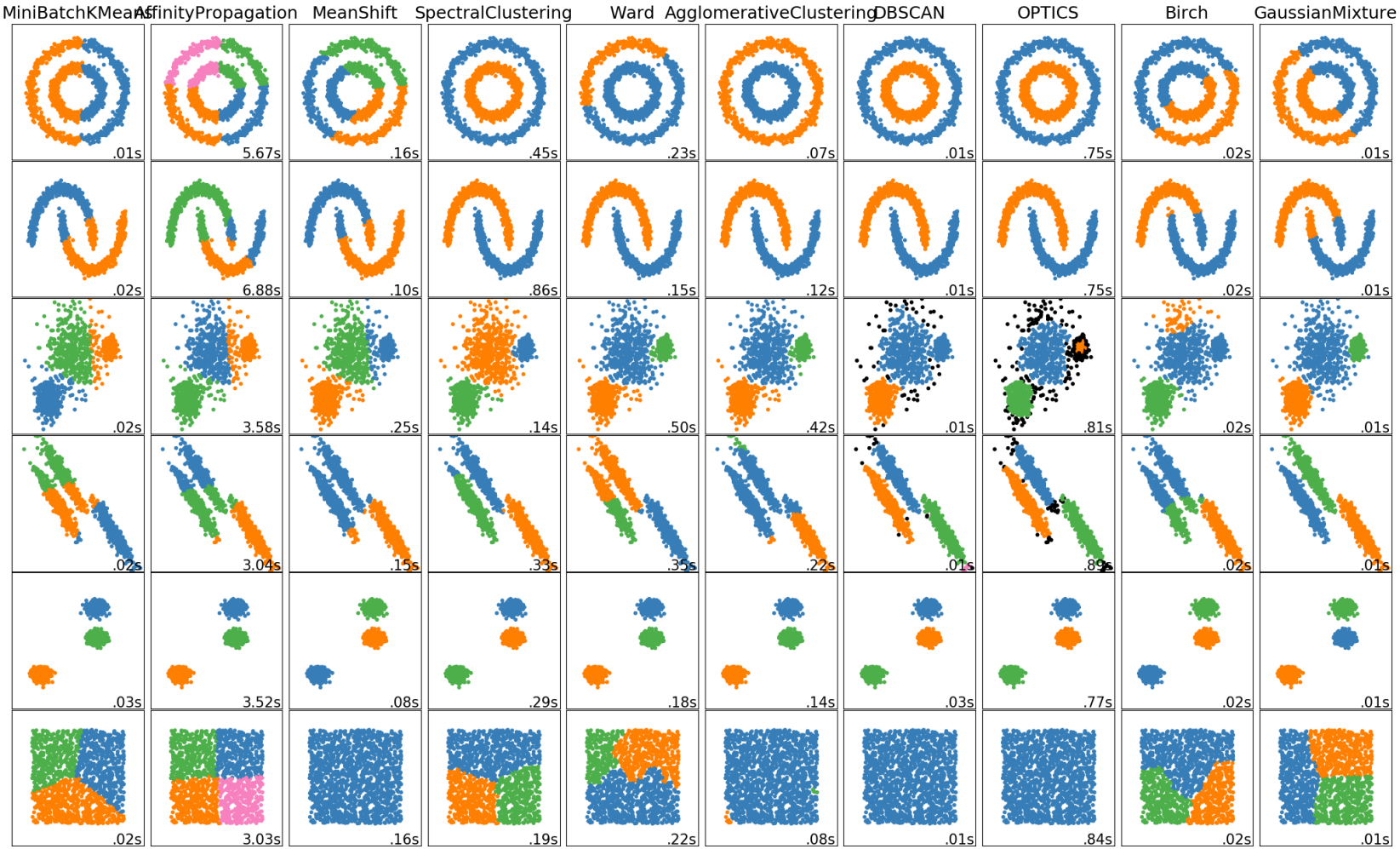
Test score, density estimation

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

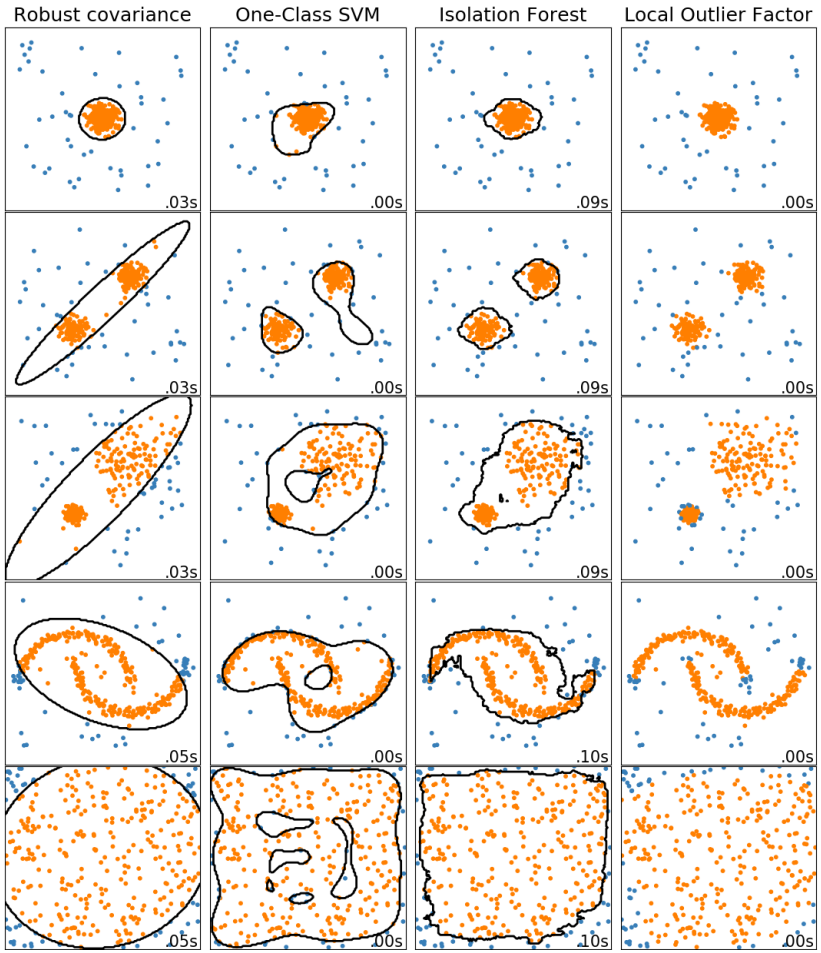
Classification models



Clustering models



Anomaly detection



Outline

MOABB: Why and What

MNE for neuro

Scikit-learn for ML

Pyriemann

Pyriemann

https://github.com/alexandrebarachant/pyRiemann

- Fully integrated in scikit
- Could use MNE output

All you need in one toolbox

- Covariance estimation
- Classification
  - MDM
  - Tangent Space
  - K-Nearest Neighbor

- Clustering
- Tangent Space projection
- Spatial filtering
  - CSP
  - xDawn

One example script

```
from scipy.io import loadmat
from pyriemann.estimation import Covariances
from pyriemann.classification import MDM
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

data = loadmat('data.mat')
X = data['X']
y = data['y'].squeeze()

covs = Covariances(estimator='scm').fit_transform(X)
covs_train, covs_test, y_train, y_test = \
                        train_test_split(covs, y, train_size=0.8)

clf = MDM(metric='riemann')

clf.fit(covs_train, y_train)
y_pred = clf.predict(covs_test)
print (accuracy_score(y_test, y_pred)
```

Thank you !