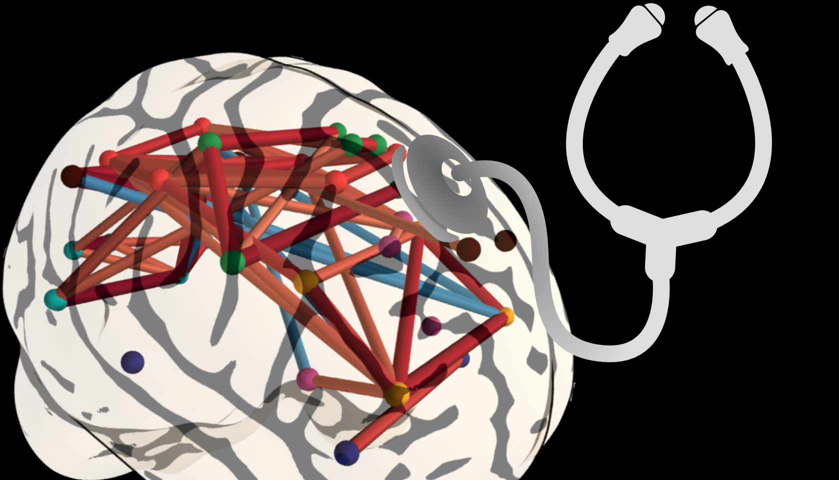
Advanced machine learning for neuroimaging

Gaël Varoquaux

McGill

Inria



1. Large scale

2. Some advanced estimators

3. Advanced learners on brain images

4. Machine learning principles

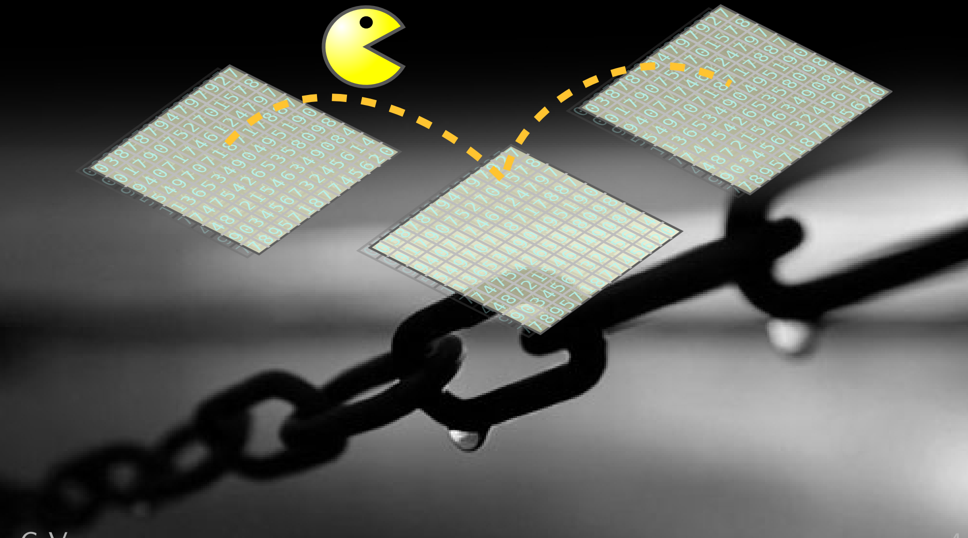
1 Large scale

Difficulty: the data do not fit in memory

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

1 On-line algorithms

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



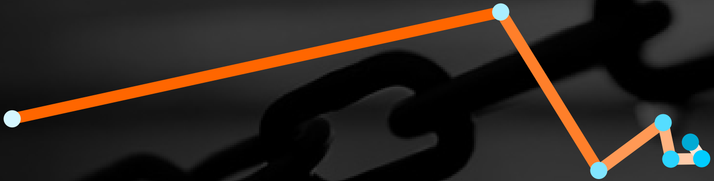
1 On-line algorithms

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

Linear models

```
    sklearn.linear_model.SGDRegressor
    sklearn.linear_model.SGDClassifier
```

SGD = Stochastic gradient descent

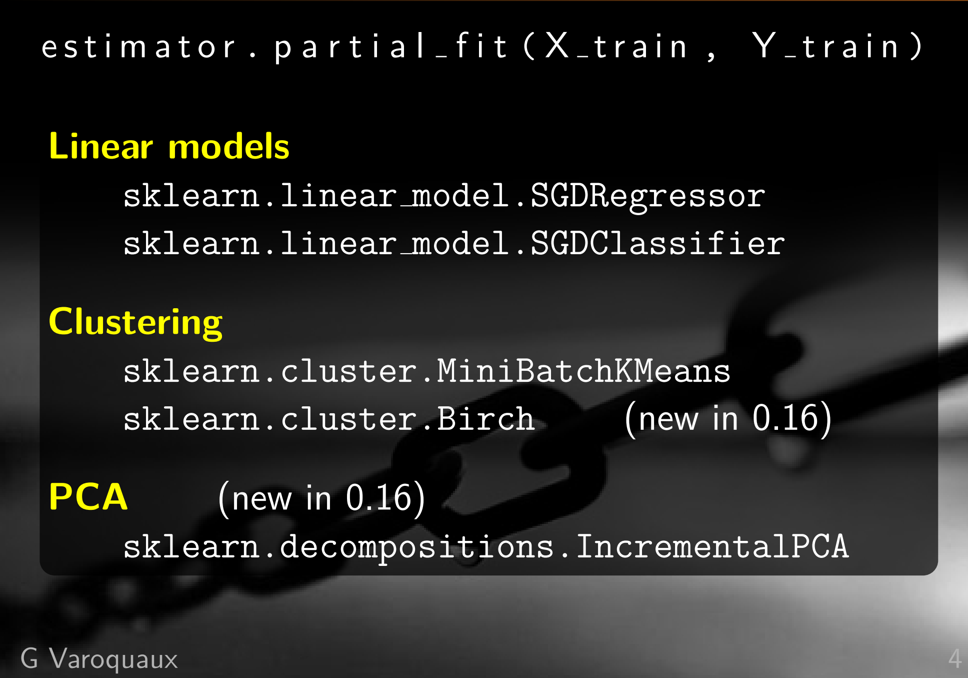


Different losses, different penalties

learning rate ☹

1 On-line algorithms

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



Linear models

```
sklearn.linear_model.SGDRegressor
sklearn.linear_model.SGDClassifier
```

Clustering

```
sklearn.cluster.MiniBatchKMeans
sklearn.cluster.Birch
```

(new in 0.16)

PCA (new in 0.16)

```
sklearn.decompositions.IncrementalPCA
```

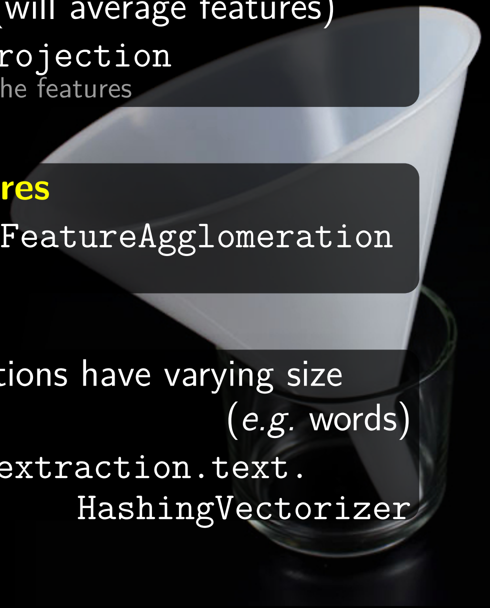
1 On-the-fly data reduction

Many features

⇒ Reduce the data as it is loaded

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

1 On-the-fly data reduction



Random projections (will average features)

```
sklearn.random_projection
```

random linear combinations of the features

Fast clustering of features

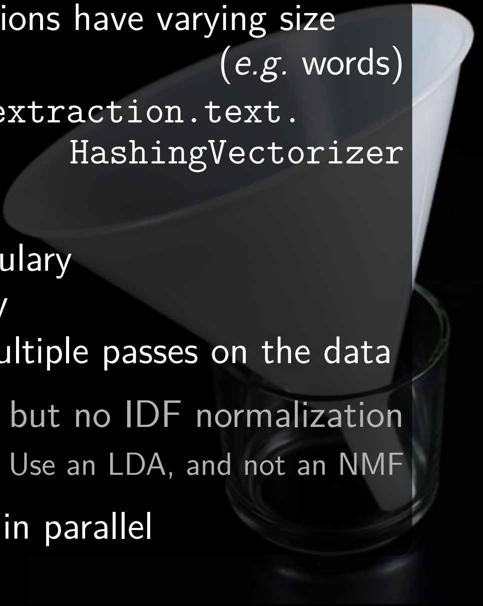
```
sklearn.cluster.FeatureAgglomeration
```

on images: super-pixel strategy

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

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

1 On-the-fly data reduction



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

`sklearn.feature_extraction.text.HashingVectorizer`

TF-IDF needs

- to know the vocabulary
- to count everybody

⇒ multiple passes on the data

Hashing avoids that

but no IDF normalization

Use an LDA, and not an NMF

+ stateless: can be used in parallel

2 Some advanced estimators

scikit learn

machine learning in Python

2 For text data mining publications

[Neurosynth,Neuroquery]

Linear estimators

- Can handle large number of features
- Typically a logistic regression

`sklearn.linear_model.LogisticRegression`
`sklearn.linear_model.LogisticRegressionCV`

'l2' and 'l1' penalties different solvers

`sklearn.linear_model.SGDClassifier`

For on-line estimator

Naive Bayes

- Very good for many classes
- On-line estimator

+ chi2 feature selection

2 For heterogeneous columnar data

Priceless for tabular data *eg* socio-demographics

- Tree methods are good:

  robust to strange data distributions

- Ensemble methods: need to combine many trees

Random forests

`sklearn.ensemble.RandomForestClassifier`

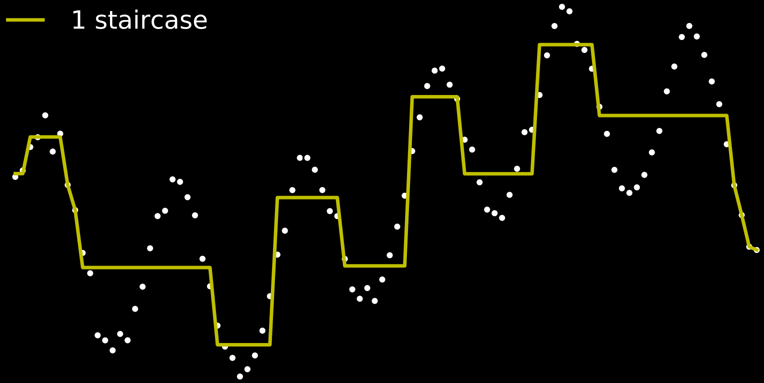
`sklearn.ensemble.ExtraTreesClassifier`

Boosted trees

`sklearn.ensemble.HistGradientBoostingClassifier`

Native support for missing values

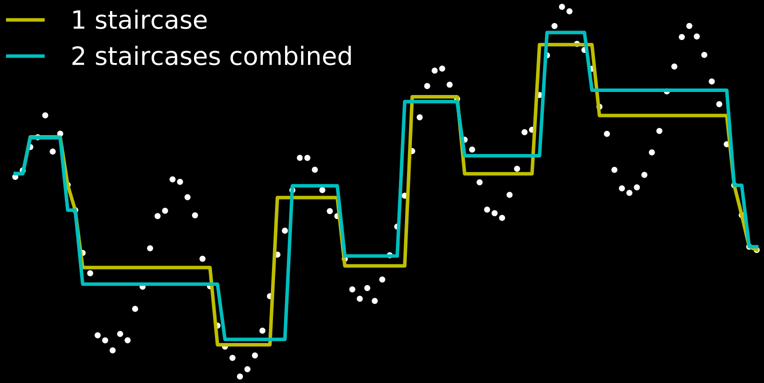
2 Gradient-boosted regression trees



- Fit with a tree of depth 10

staircase of 10 constant values

2 Gradient-boosted regression trees

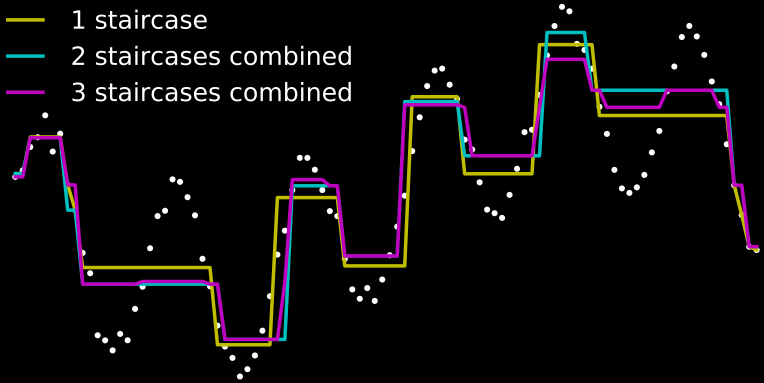


- Fit with a tree of depth 10

staircase of 10 constant values

- Fit a new tree on errors

2 Gradient-boosted regression trees

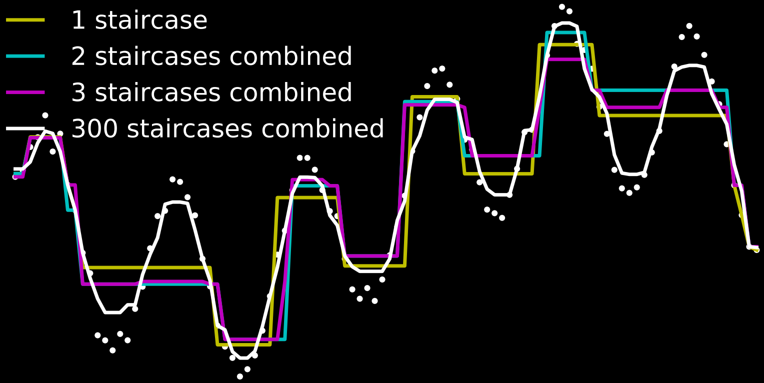


- Fit with a tree of depth 10

staircase of 10 constant values

- Fit a new tree on errors
- Keep going

2 Gradient-boosted regression trees



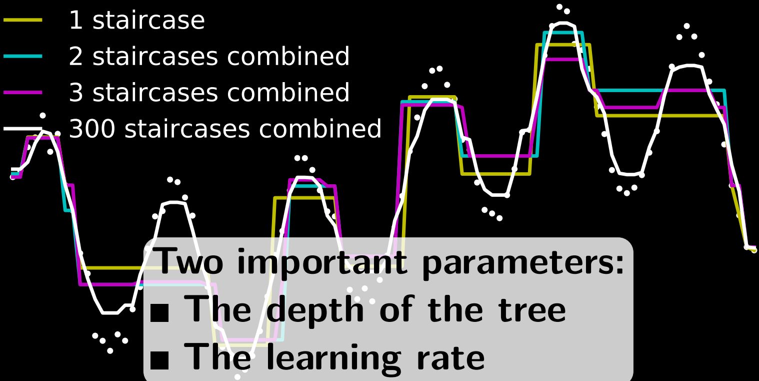
- Fit with a tree of depth 10

staircase of 10 constant values

- Fit a new tree on errors
- Keep going

Boosted regression trees

2 Gradient-boosted regression trees



Two important parameters:

- **The depth of the tree**
- **The learning rate**

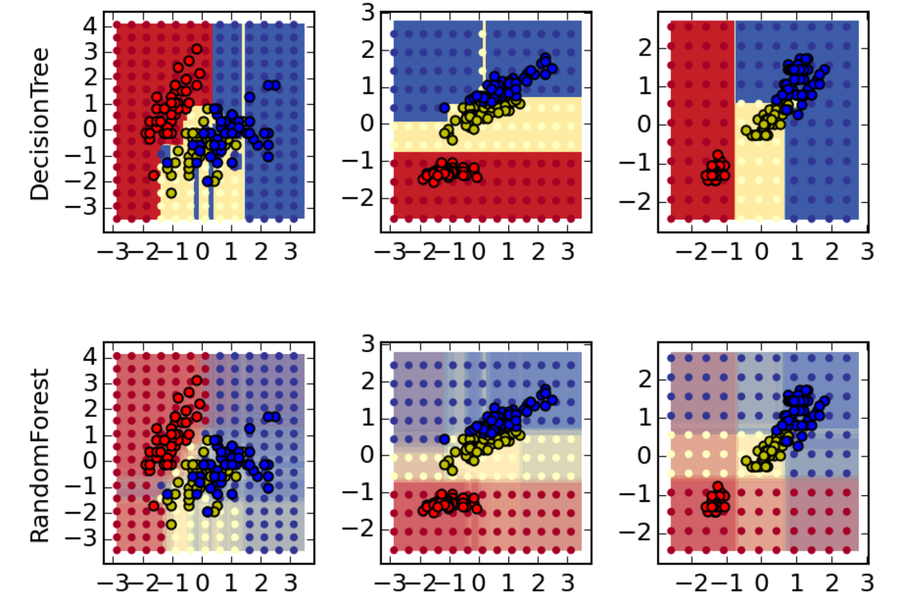
- Fit with a tree of depth 10

staircase of 10 constant values

- Fit a new tree on errors
- Keep going

Boosted regression trees

2D intuitions & model averaging



2 Model stacking

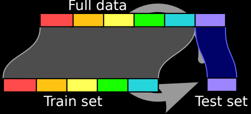
$$x \xrightarrow{\text{model}_1} z \xrightarrow{\text{model}_1} y$$

Learn model$_1$ separately

Directly supervising z:

$z = \hat{y}$ for a (simple) predictive model

Trick: "cross-fit" during training



obtain $\hat{y}$ by splitting the training data

Just use `sklearn.ensemble.StackingRegressor`

Useful to assemble non-linear models from simple ones

2 Missing values

Gender	Date Hired	Employee Position Title
M	09/12/1988	Master Police Officer
F	NA	Social Worker IV
M	07/16/2007	Police Officer III
M	01/13/2014	Electrician I
M	04/28/2002	Bus Operator
M	NA	Bus Operator
F	06/26/2006	Social Worker III
F	01/26/2000	Library Assistant I
M	NA 2014	Library Assistant I

2 Classic statistics on missing values

Model a) a complete data-generating process
b) a random process occluding entries

Missing at random situation (MAR)

Theorem [Rubin 1976], if for non-observed values, the probability of missingness does not depend on this non-observed value. maximizing likelihood for observed data while **ignoring** the unobserved values gives maximum likelihood of model a).

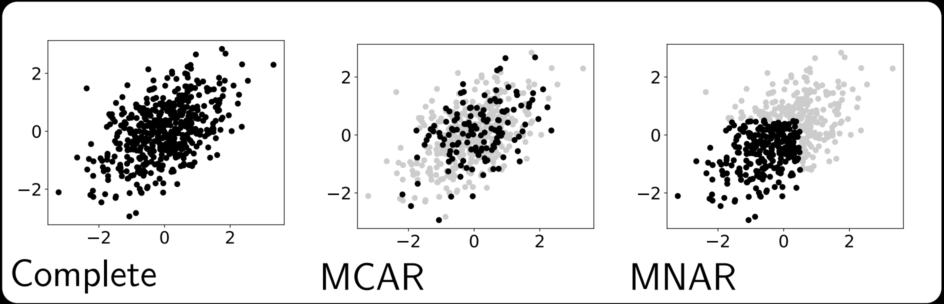
MCAR: Missing Completely At Random:
missingness independent of $\mathbf{X}$

Missing Not at Random situation (MNAR)
Missingness **not ignorable**

2 Classic statistics on missing values

Model a) a complete data-generating process

b) a random process occluding entries



Complete

MCAR

MNAR

MCAR: Missing Completely At Random:

missingness independent of **X**

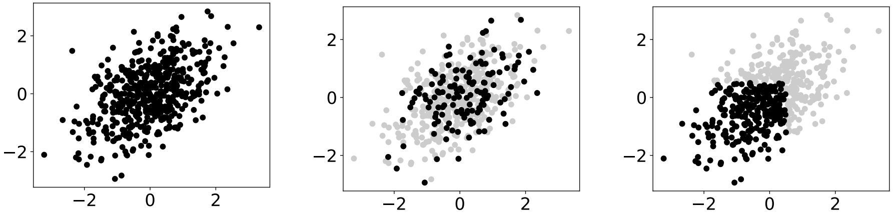
Missing Not at Random situation (MNAR)

Missingness **not ignorable**

2 Classic statistics on missing values

Model a) a complete data-generating process

b) a random process occluding entries



Complete MCAR MNAR

MCAR: Missing Completely At Random

But

- MAR is not frequent
- Machine learning is not about maximizing likelihoods

Missingness **not ignorable**

2 Classic statistics: Imputation

Fill in information

Gender	Date Hired	Employee Position Title
M	09/12/1988	Master Police Officer
F	~~NA~~ 2000	Social Worker IV
M	07/16/2007	Police Officer III
M	01/13/2014	Electrician I
M	04/28/2002	Bus Operator
M	~~NA~~ 2012	Bus Operator
F	06/26/2006	Social Worker III
F	01/26/2000	Library Assistant I
M	~~NA~~ 2014	Library Assistant I

2 **Imputation procedures** that work on test set

Mean imputation special case of univariate imputation

Replace `NA` by the mean of the feature

`sklearn.impute.SimpleImpute`

2 Imputation procedures that work on test set

Mean imputation special case of univariate imputation

Replace `NA` by the mean of the feature

`sklearn.impute.SimpleImpute`

Conditional imputation

- Modeling one feature as a function of others
- Possible implementation:
  iteratively predict one feature as a function of other
  `sklearn.impute.IterativeImputer`

Statistics: Conditional imputation considered richer

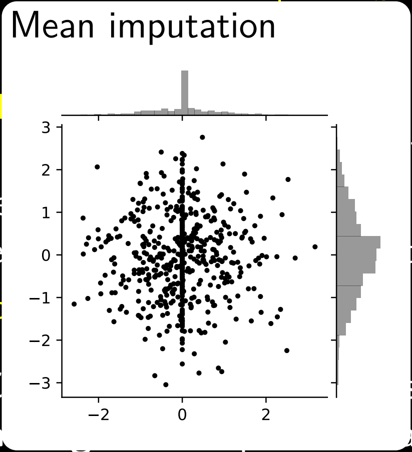
Machine learning mean imputation can be detected by non-linear learners [Josse... 2019]

2 **Imputation procedures** that work on test set

Mean imputation special case of univariate imputation

Replace `NA` by the mean of the feature

`sklearn.impute.SimpleImpute`



Mean imputation

Conditional im

- Modeling one … thers
- Possible imple

iterately pre … tion of other

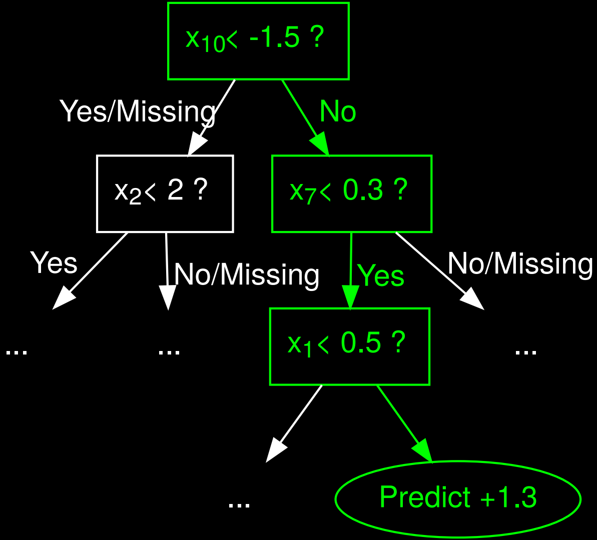
`sklearn.imp`

Statistics: C … sidered richer

Machine lea … an be detected by non-linear learners [Josse... 2019]

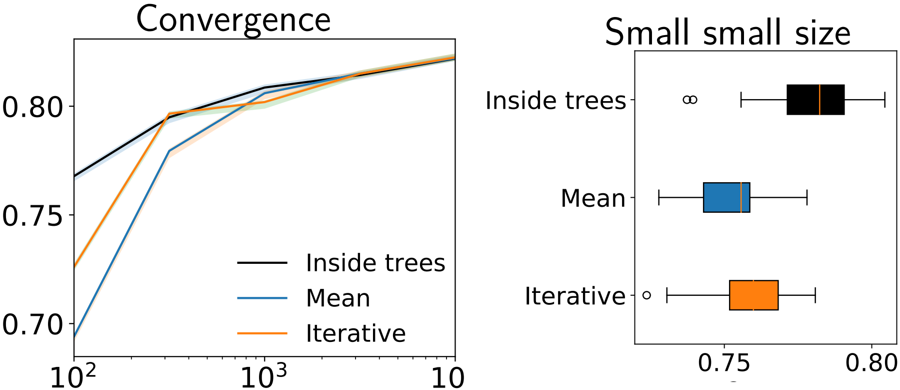
2 Missing attributes inside trees

MIA (Missing Incorporated Attribute) [Josse... 2019]



`sklearn.ensemble.HistGradientBoostingClassifier`

Simulation: MCAR + Gradient boosting



Notebook: github – @nprost / supervised_missing

2 Imputation is not enough

Pathological case [Josse... 2019]

y depends only on wether data is missing or not

eg tax fraud detection

theory: MNAR = “Missing Not At Random”

⚠ Imputing makes prediction impossible ⚠

Solution

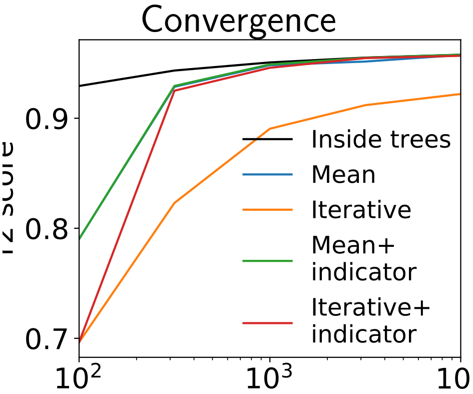
Add a missingness indicator: extra feature to predict

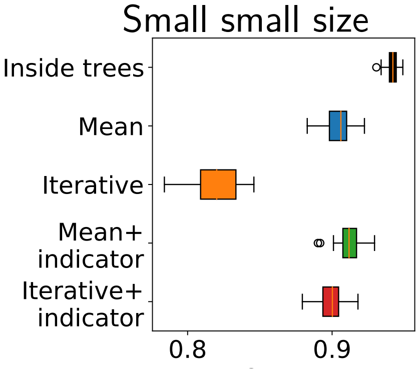
```
  ...SimpleImpute(add_indicator=True)
...IterativeImputer(add_indicator=True)
```

2 Imputation is not enough

Simulation: **y** depends *indirectly* on missingness

censoring in the data


Convergence


Small small size

Notebook: github – @nprost / supervised_missing

- **Adding a mask is crucial**
- **Iterative imputation can be detrimental**

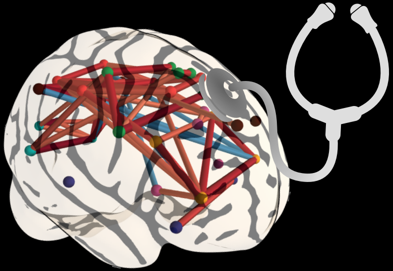
Recommendations

- High-dimensional settings ($p > 1\,000$):
  use linear models
- Lower dimensions, large n ($n > 1\,000$):
  use gradient-boosted trees

- Ensembling reduces variance

- Missing values with linear models: iterative imputer
- Missing values with trees: MIA (native support)

3 Advanced learners on brain images



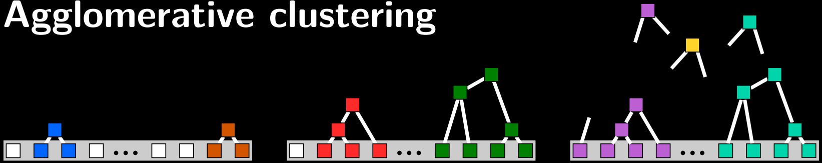
3 Feature clustering to reduce dimension

Challenge: many features

Learn feature groups by clustering

- Fast clustering for large k

Agglomerative clustering

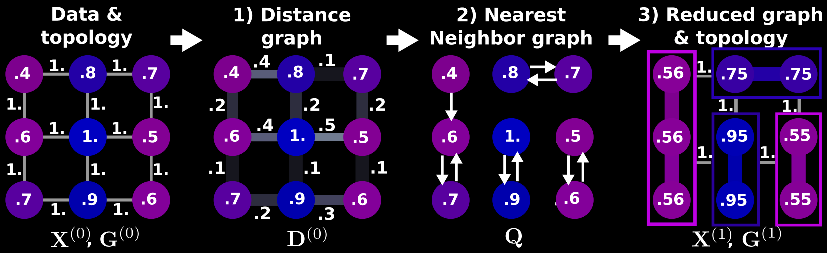


sklearn.cluster.FeatureAgglomeration

Choose Ward clustering for best results

[Michel... 2012]

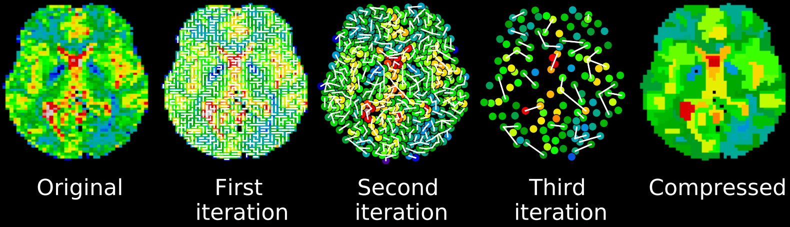
3 ReNA: fast spatial clustering



Very fast with spatial constraints

[Hoyos-Idrobo... 2018a]

3 ReNA: fast spatial clustering



1. Compute distance on neighborhood graph
2. Assign each vertex to its nearest neighbor on the graph
3. Connect components of graph are next features

Rinse and repeat

`nilearn.regions.Parcellations`

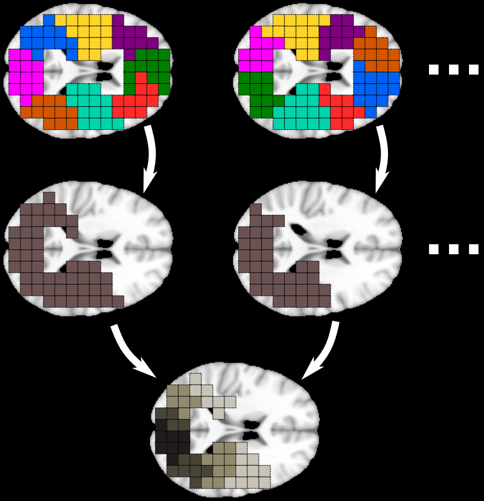
[Hoyos-Idrobo... 2018a]

3 Fast spatial penalties (FREM) [Hoyos-Idrobo... 2018b]

- Very fast sub optimal models
- Average many of them

3 Fast spatial penalties (FREM) [Hoyos-Idrobo... 2018b]

- Very fast sub optimal models
- Average many of them

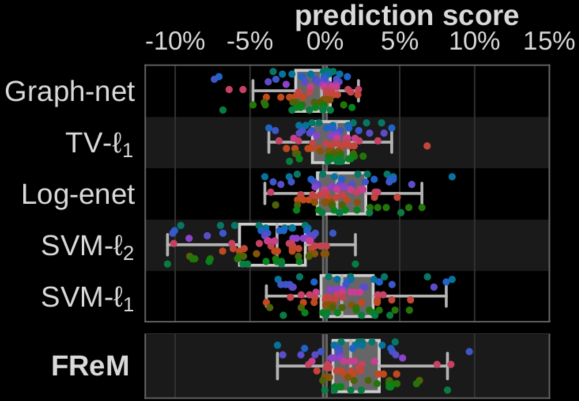


- Learn parcellation on perturbed data
- Estimate linear models

Average the results

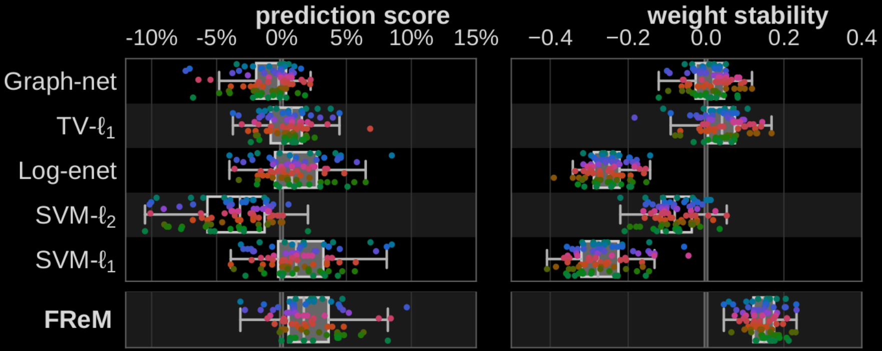
3 Fast spatial penalties (FREM) [Hoyos-Idrobo... 2018b]

- Very fast sub optimal models
- Average many of them



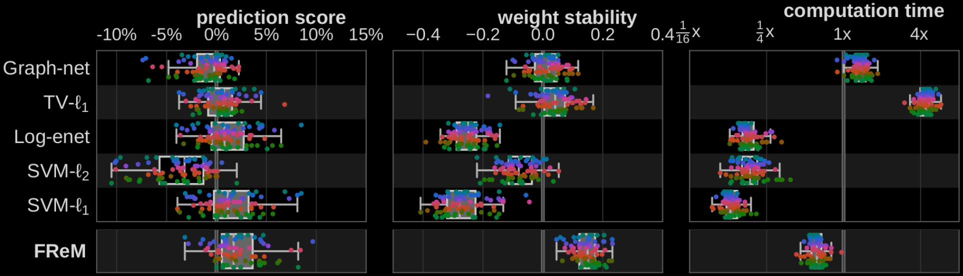
3 Fast spatial penalties (FREM) [Hoyos-Idrobo... 2018b]

- Very fast sub optimal models
- Average many of them



3 Fast spatial penalties (FREM) [Hoyos-Idrobo... 2018b]

- Very fast sub optimal models
- Average many of them



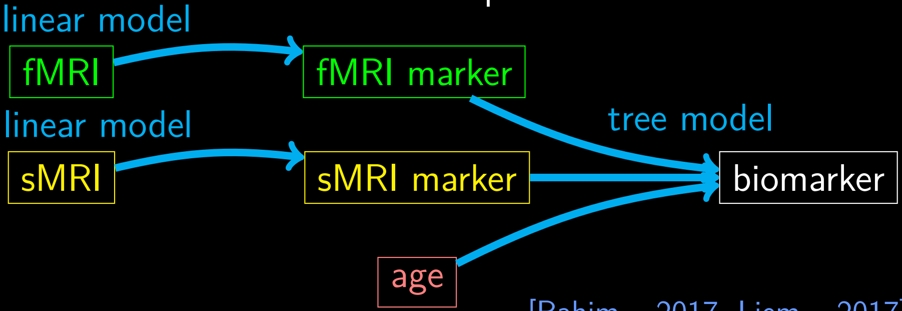
3 Stacking, multimodal non-linear models

Modality-specific linear models

- On each imaging modality fit a linear model

Non-linear model stacking

- Combine the **predicted** outcome values with other clinical variables as the input of tree-based model



[Rahim... 2017, Liem... 2017]

3 Stacking, multimodal non-linear models

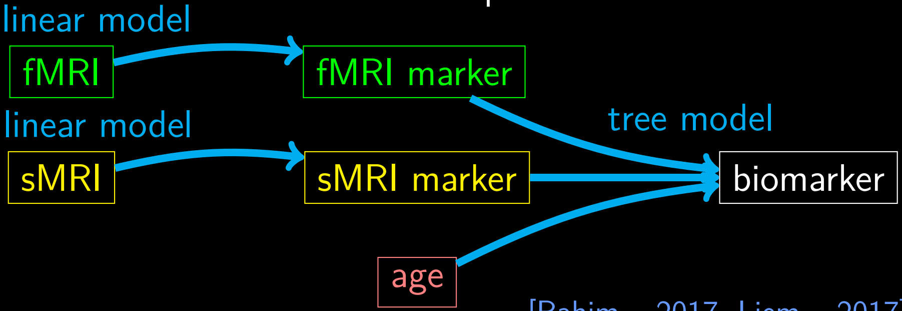
Modality-specific linear models

- On each imaging modality fit a linear model

Non-linear model stacking

[Engemann... 2020]: missing-value support in trees for subjects with only part of the modalities.

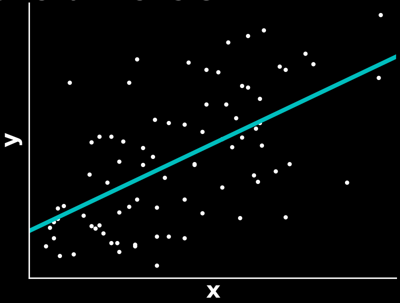
as the input of tree-based model



[Rahim... 2017, Liem... 2017]

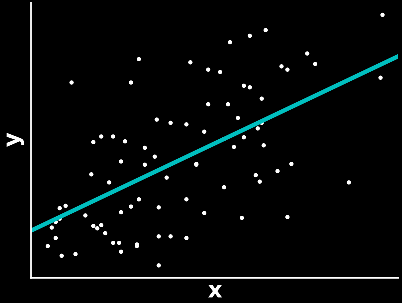
4 Machine learning principles

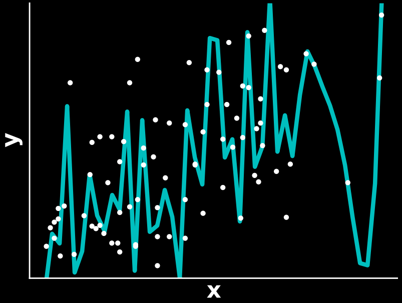
A single descriptor: one dimension



Machine learning in a nutshell: regression

A single descriptor:
one dimension

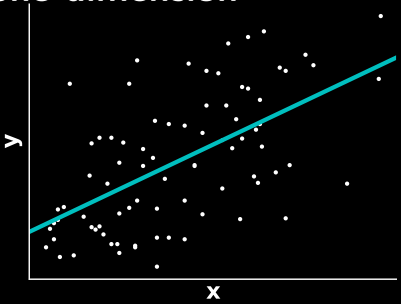


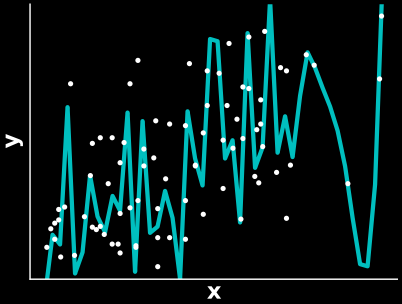


Which model to prefer?

4 Machine learning in a nutshell: regression

A single descriptor:
one dimension



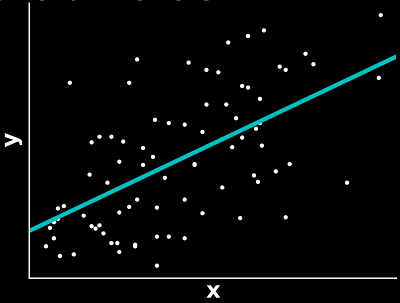


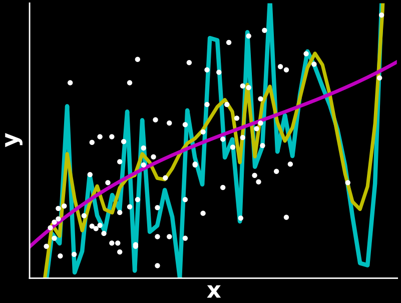
Problem of *"over-fitting"*

- Minimizing error is not always the best strategy (learning noise)
- Test data $\neq$ train data

4 Machine learning in a nutshell: regression

A single descriptor: one dimension





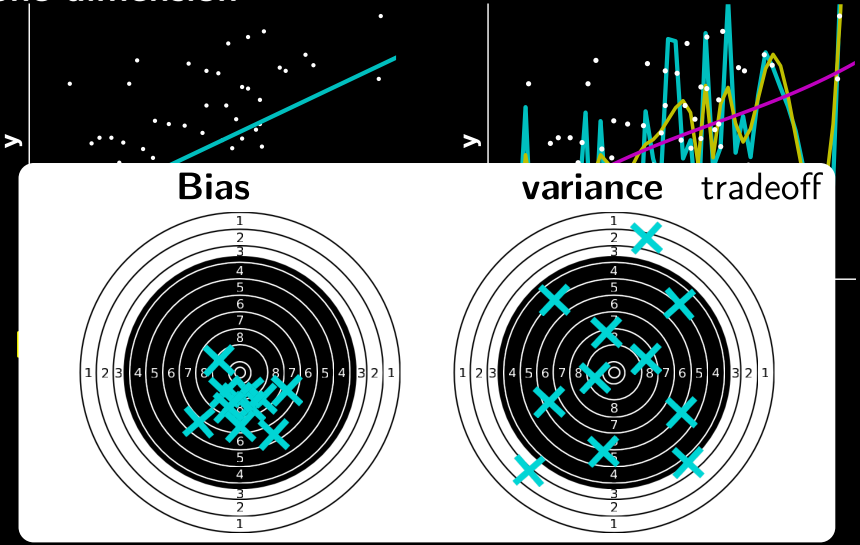
Prefer simple models

= concept of *"regularization"*

Balance the number of parameters to learn with the amount of data

4 Machine learning in a nutshell: regression

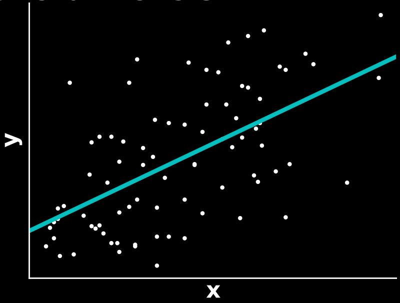
A single descriptor: one dimension



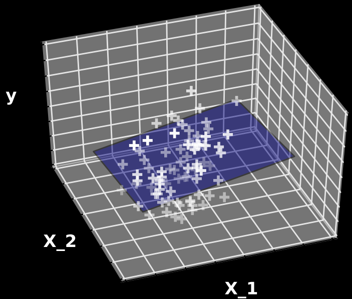
Bias variance tradeoff

4 Machine learning in a nutshell: regression

A single descriptor: one dimension



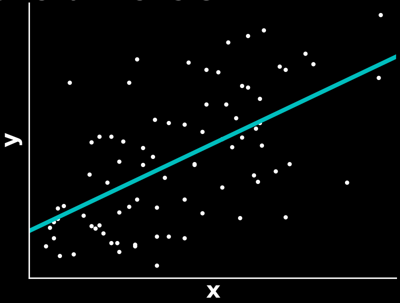
Two descriptors: 2 dimensions



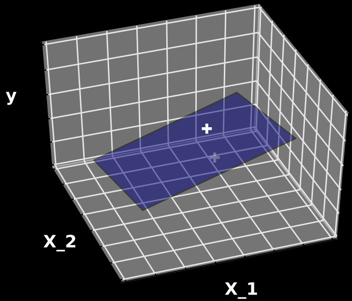
More parameters

4 Machine learning in a nutshell: regression

A single descriptor: one dimension



Two descriptors: 2 dimensions



More parameters

⇒ Model with more parameters need much more data

"curse of dimensionality"

4 A statistical formulation

- Given n pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ drawn *i.i.d.* find a function $f : \mathcal{X} \to \mathcal{Y}$ such that $f(x) \approx y$

Notation: $\hat{y} \overset{\text{def}}{=} f(x)$

- Given n pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ drawn *i.i.d.*
  find a function $f : \mathcal{X} \to \mathcal{Y}$ such that $f(x) \approx y$

  Notation: $\hat{y} \overset{\text{def}}{=} f(x)$

Empirical risk minimization

- Given a "loss" function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
- Estimation of f: $$f^\star = \operatorname*{argmin}_{f \in \mathcal{F}} \mathbb{E}\big[l(\hat{y}, y)\big]$$

  Can create f such that $\hat{y} = \mathbb{E}[\mathbf{y}|\mathbf{X}]$

- Given n pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ drawn *i.i.d.*
  find a function $f : \mathcal{X} \to \mathcal{Y}$ such that $f(x) \approx y$

  Notation: $\hat{y} \overset{\text{def}}{=} f(x)$

Empirical risk minimization

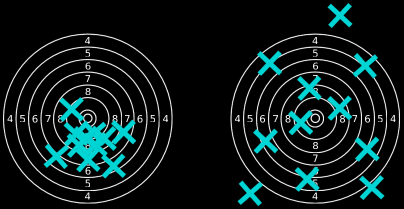
- Given a "loss" function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
- Estimation of f: $$f^\star = \underset{f \in \mathcal{F}}{\text{argmin}}\, \mathbb{E}\big[l(\hat{y}, y)\big]$$

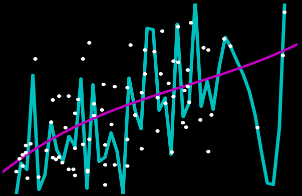
Can create f such that $\hat{y} = \mathbb{E}[\mathbf{y}|\mathbf{X}]$

The inference & control is on f, not parameters

In general, f can be anything (choice of $\mathcal{F}$)

4 Some formalism: bias and regularization

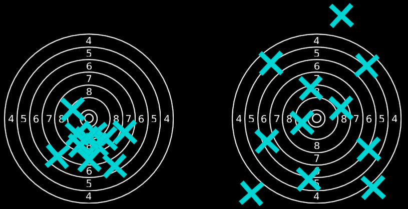


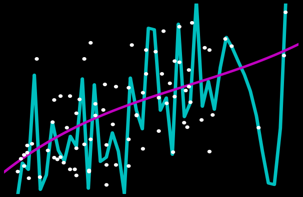


Settings: data $(\mathbf{X}, \mathbf{y})$, prediction $\mathbf{y} \sim f(\mathbf{X}, \mathbf{w})$

Our goal: $\underset{\mathbf{w}}{\text{minimize}} \quad \|\mathbf{y} - f(\mathbf{X}, \mathbf{w})\|$

4 **Some formalism**: bias and regularization





Settings: data $(\mathbf{X}, \mathbf{y})$, prediction $\mathbf{y} \sim f(\mathbf{X}, \mathbf{w})$

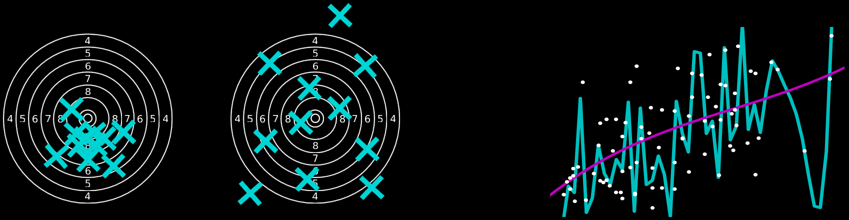
Our goal: $\underset{\mathbf{w}}{\text{minimize}}\ \mathbb{E}\big[\|\mathbf{y} - f(\mathbf{X}, \mathbf{w})\|\ \big]$

We only can measure $\|\mathbf{y} - f(\mathbf{X}, \mathbf{w})\|$

Prediction is very difficult, especially about the future.

Niels Bohr

4 **Some formalism**: bias and regularization



Settings: data $(\mathbf{X}, \mathbf{y})$, prediction $\mathbf{y} \sim f(\mathbf{X}, \mathbf{w})$

Our goal: $\underset{\mathbf{w}}{\text{minimize}}\ \mathbb{E}\big[\|\mathbf{y} - f(\mathbf{X}, \mathbf{w})\|\ \big]$

We only can measure $\|\mathbf{y} - f(\mathbf{X}, \mathbf{w})\|$

Solution: bias $\mathbf{w}$ to push toward a plausible solution

In a minimization framework:

$$\underset{\mathbf{w}}{\text{minimize}}\ \|\mathbf{y} - f(\mathbf{X}, \mathbf{w})\| + p(\mathbf{w})$$

Going further

- Scipy lecture notes:
  `http://www.scipy-lectures.org`

  In particular chapter on statistics

- The scikit-learn documentation:
  `http://scikit-learn.org`

  It's a reference on machine learning

- `nilearn`

@GaelVaroquaux

5 References I

D. A. Engemann, O. Kozynets, D. Sabbagh, G. Lemaitre, G. Varoquaux, F. Liem, and A. Gramfort. Combining electrophysiology with MRI enhances learning of surrogate-biomarkers. *bioRxiv*, 2020. doi: 10.1101/856336. URL https://www.biorxiv.org/content/early/2019/11/26/856336.

A. Hoyos-Idrobo, G. Varoquaux, J. Kahn, and B. Thirion. Recursive nearest agglomeration (rena): fast clustering for approximation of structured signals. *IEEE transactions on pattern analysis and machine intelligence*, 41(3):669–681, 2018a.

A. Hoyos-Idrobo, G. Varoquaux, Y. Schwartz, and B. Thirion. Frem–scalable and stable decoding with fast regularized ensemble of models. *NeuroImage*, 180:160–172, 2018b.

5 References II

J. Josse, N. Prost, E. Scornet, and G. Varoquaux. On the consistency of supervised learning with missing values. *arXiv preprint arXiv:1902.06931*, 2019.

F. Liem, G. Varoquaux, J. Kynast, F. Beyer, S. K. Masouleh, J. M. Huntenburg, L. Lampe, M. Rahim, A. Abraham, R. C. Craddock, ... Predicting brain-age from multimodal imaging data captures cognitive impairment. *NeuroImage*, 2017.

V. Michel, A. Gramfort, G. Varoquaux, E. Eger, C. Keribin, and B. Thirion. A supervised clustering approach for fMRI-based inference of brain states. *Pattern Recognition*, 45:2041, 2012.

M. Rahim, B. Thirion, D. Bzdok, I. Buvat, and G. Varoquaux. Joint prediction of multiple scores captures better individual traits from brain images. *Neuroimage*, in rev, 2017.

D. B. Rubin. Inference and missing data. *Biometrika*, 63(3): 581–592, 1976.