L'intelligence artificielle, le nouveau couteau suisse de la spectroscopie proche infrarouge ?

Gregory Beurier,
Lauriane Rouan,
Denis Cornet

24èmes Rencontres HélioSPIR
Thème animé par le GFC. Comment aller plus loin dans l'analyse des spectres ?
Session NIRS et Deep Learning, 15 juin 2023, Montpellier, France

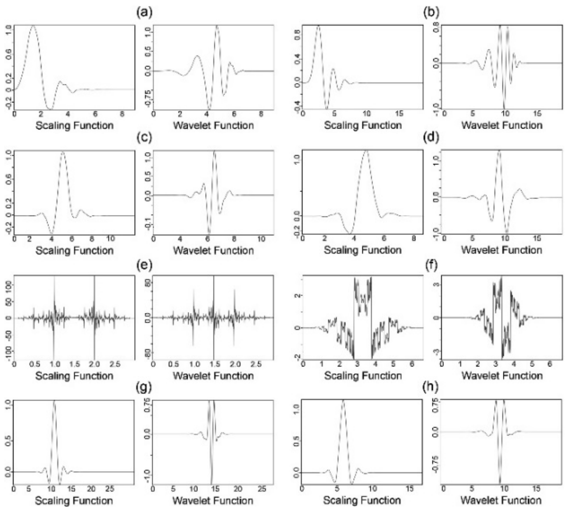
Avantages de l’IA

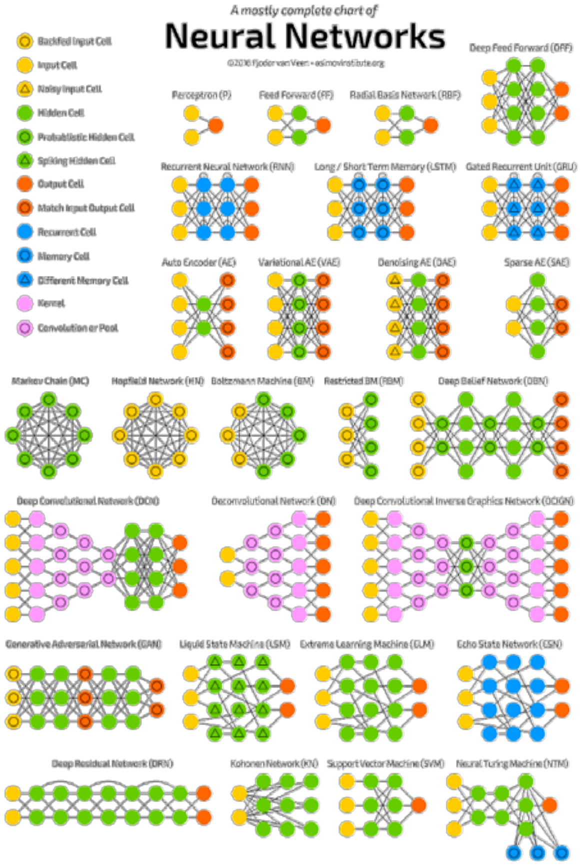
- Tenseur => gestion de multiples dimensions possible
  - => architecture friendly (multi-bloc and beyond !)
  - Données hétérogènes (preprocessing, repetition, sensors dimensions)
  - Interopérable avec l’ensemble des outils, régresseurs, filtres, denoiser, dropout,
- GPU => speedup
  - Parralélisation >16000 coeurs (RTX 4090)
  - Compatibilité versions tensorflow – cuda – GPU non trivial
- Outil de gestion du surapprentissage
- Deep => Multiscale learning: gestion des jeux de données structures (multi-espèce, multi-capteur…)
- Non linéaire
- Robuste aux outliers
- Grande communauté ultra réactive
- …

Generic pipeline

Why?

- Democratization/vulgarization of NIRS brings more and more "naive" users
- Most publications focus on demonstrating the superiority of one method
  - Applies 1 modeling strategy
  - Uses 1 specific combination of pretreatment
  - Optimizes calibration for 1 analyte
- While the user
  - Often has several traits (e.g. sugar, starch, protein)
  - Encounters a growing wall of possible pretreatment/model combinations
- If the optimal combination differs from one study to another, the way to identify it could be generalized





Generic pipeline

PiNARD: a Pipeline for Nirs Analysis ReloadeD

A NIRS data processing pipeline based on scikit-learn pipelines

Interoperable
(sklearn, tensorflow, pytorch, shap, etc.)

Parallel
(joblib)

Modular
(reuse scipy functions, sklearn transformers, etc.)

Reification

https://github.com/gbeurier/pinard

https://pypi.org/project/pinard/

Generic pipeline

PiNARD: a Pipeline for Nirs Analysis ReloadeD

A NIRS data processing pipeline based on scikit-learn pipelines

Interoperable
(sklearn, tensorflow, pytorch, shap, etc.)

Parallel
(joblib)

Modular
(reuse scipy functions, sklearn transformers, etc.)

Reification

https://github.com/gbeurier/pinard

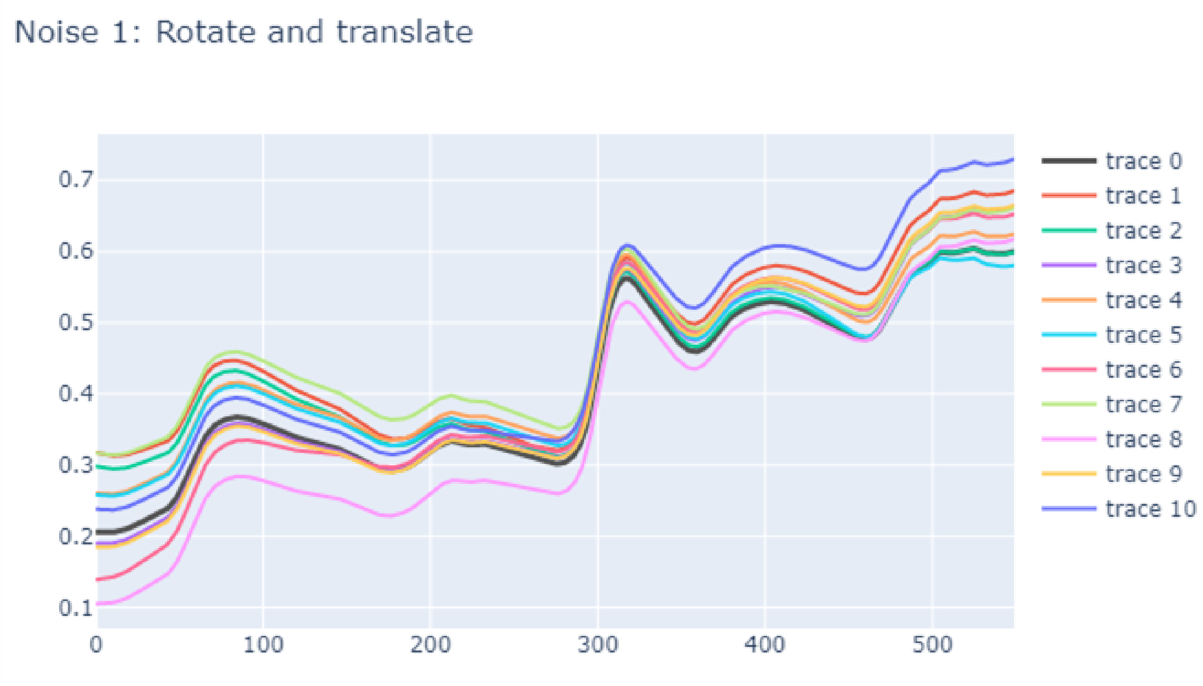
https://pypi.org/project/pinard/

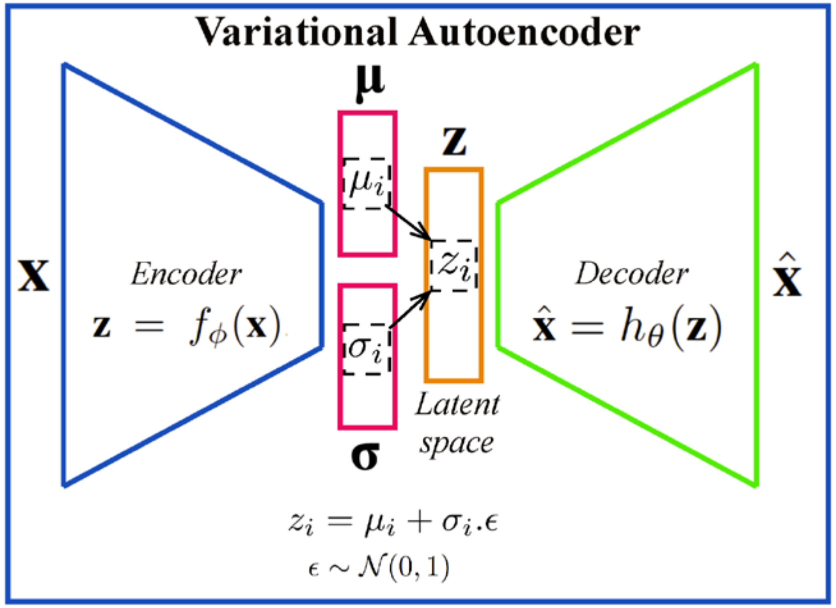
Generic pipeline

Data augmentation

- Known to improve robustness and learning of neural networks (A. Krizhevsky et al 2012)
- Used in various domains (image, NIRS)
- In the case of classification, allows to re-balance the dataset in case of underrepresentation of one or more classes

1st approach: Simple mathematical transformation : translations, rotations
2nd approach (in dev.): Generation of purely synthetic spectra through Variational AutoEncodeur (VAE)

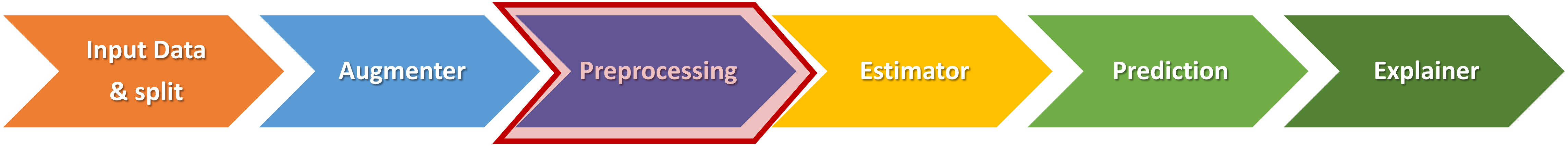




Generic pipeline

PiNARD: a Pipeline for Nirs Analysis ReloadeD

A NIRS data processing pipeline based on scikit-learn pipelines



Interoperable
(sklearn, tensorflow, pytorch, shap, etc.)

Parallel
(joblib)

Modular
(reuse scipy functions, sklearn transformers, etc.)

Reification

GitHub PINARD

https://github.com/gbeurier/pinard

python™ Package Index

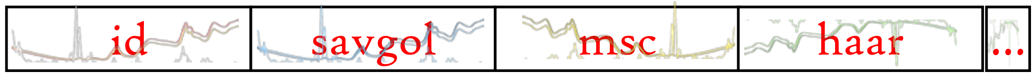
https://pypi.org/project/pinard/

PiNARD: spectra processing

```
preprocessing = [   ('id', pp.IdentityTransformer()),
                    ('savgol', pp.SavitzkyGolay()),
                    ('derivate', pp.Derivate()),
                    ('gaussian1', pp.Gaussian(order = 1, sigma = 2)),
                    ('gaussian2', pp.Gaussian(order = 2, sigma = 1)),
                    ('haar', pp.Wavelet('haar')),
                    ('savgol*savgol', Pipeline([('_sg1',pp.SavitzkyGolay()),('_sg2',pp.SavitzkyGolay())])),
                    ('gaussian1*savgol', Pipeline([('_g1',pp.Gaussian(order = 1, sigma = 2)),('_sg3',pp.SavitzkyGolay())])),
                    ('gaussian2*savgol', Pipeline([('_g2',pp.Gaussian(order = 1, sigma = 2)),('_sg4',pp.SavitzkyGolay())])),
                    ('haar*savgol', Pipeline([('_haar2',pp.Wavelet('haar')),('_sg5',pp.SavitzkyGolay())]))
                ]
```

Sklearn way

```
('Union', FeatureUnion(preprocessing))
```



CONCATENATION: For all models, PLS, SVM, Random Forest, xgboost, neural networks, etc.

Pinard way

```
('Augmentation', FeatureAugmentation(preprocessing))
```

LAYERS : For Neural networks only

Matrices merging can also be applied to different signal sources (NIRS, MIRS, Raman), sample state (raw, mixed), organ (seed, leaf) or stage

Generic pipeline

PiNARD: a Pipeline for Nirs Analysis ReloadeD

A NIRS data processing pipeline based on scikit-learn pipelines

Input Data & split → Augmenter → Preprocessing → Estimator → Prediction → Explainer

Interoperable
(sklearn, tensorflow, pytorch, shap, etc.)

Parallel
(joblib)

Modular
(reuse scipy functions, sklearn transformers, etc.)

Reification

GitHub

PINARD

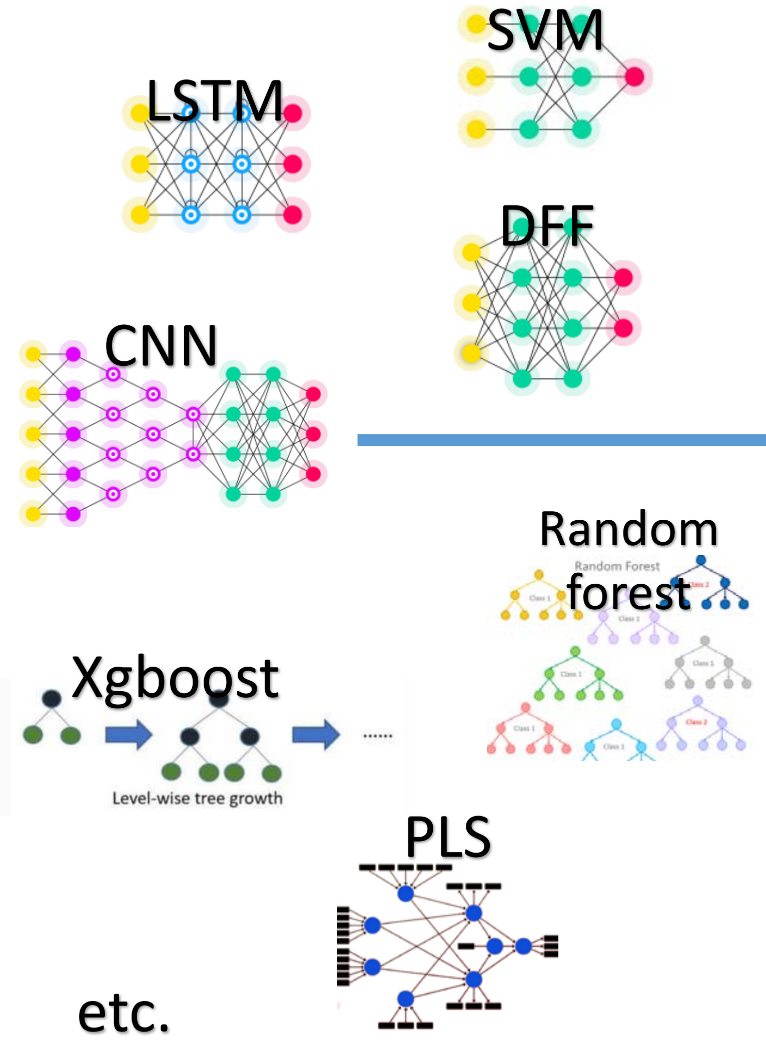
https://github.com/gbeurier/pinard

python™ Package Index

https://pypi.org/project/pinard/

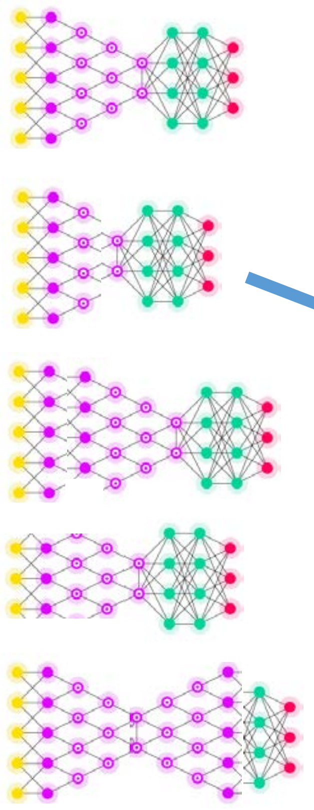
Model choice

BACON Hyperparametrization



Model

CNN



etc.

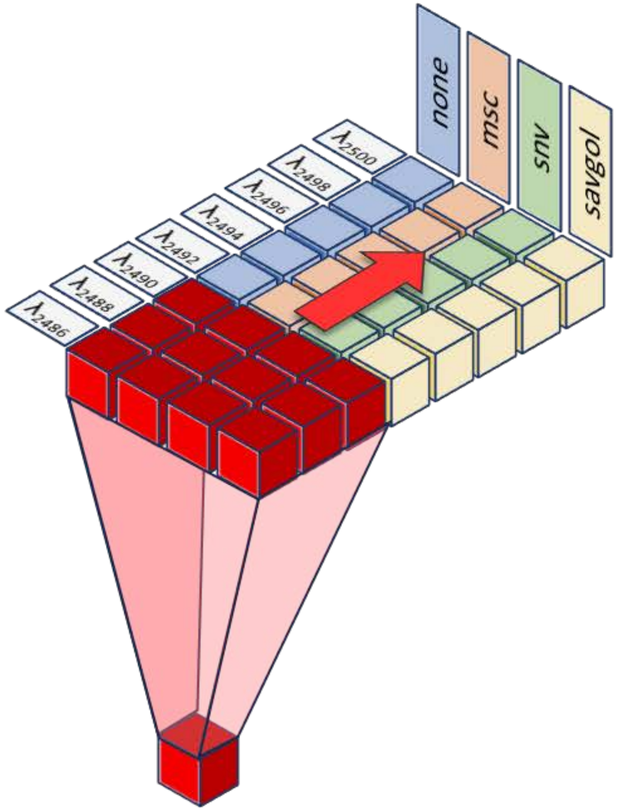
Architecture

BACON : BAsic COnvolutional neural network on Nirs

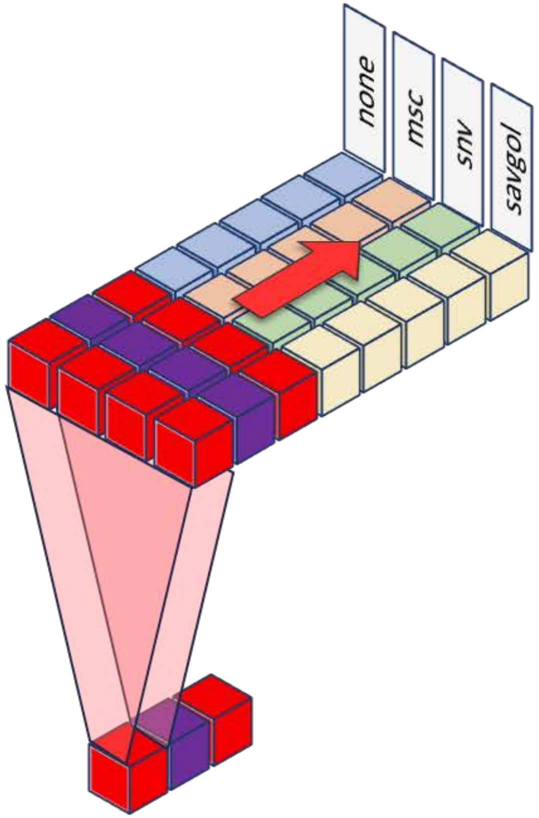
```
Sequential()
Input(shape=input_shape)
SpatialDropout1D(0.08)
Conv1D (filters=8, kernel_size=15, strides=5, activation='selu')
Dropout(0.2))
Conv1D (filters=64, kernel_size=21, strides=3,
activation='relu')
BatchNormalization())
Conv1D (filters=32, kernel_size=5, strides=3, activation='elu')
BatchNormalization()
Flatten()
Dense(16, activation='sigmoid')
Dense(1, activation='sigmoid')
```

Hyperparameters

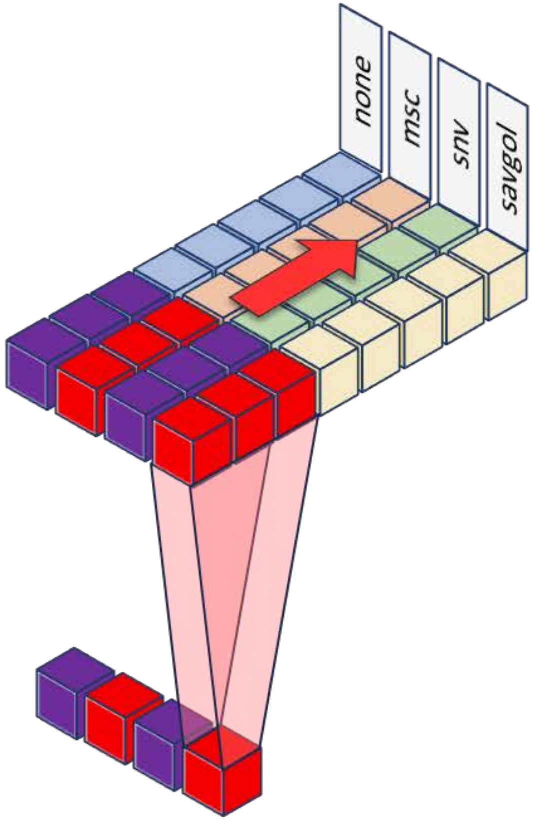
Réseaux de neurons convolutifs



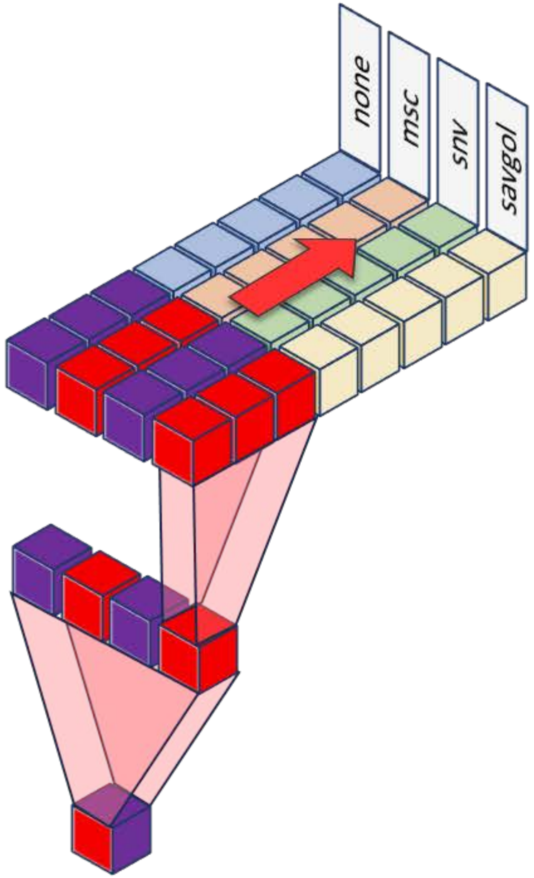
1D
Convolution



1D
Pointwise
Convolution



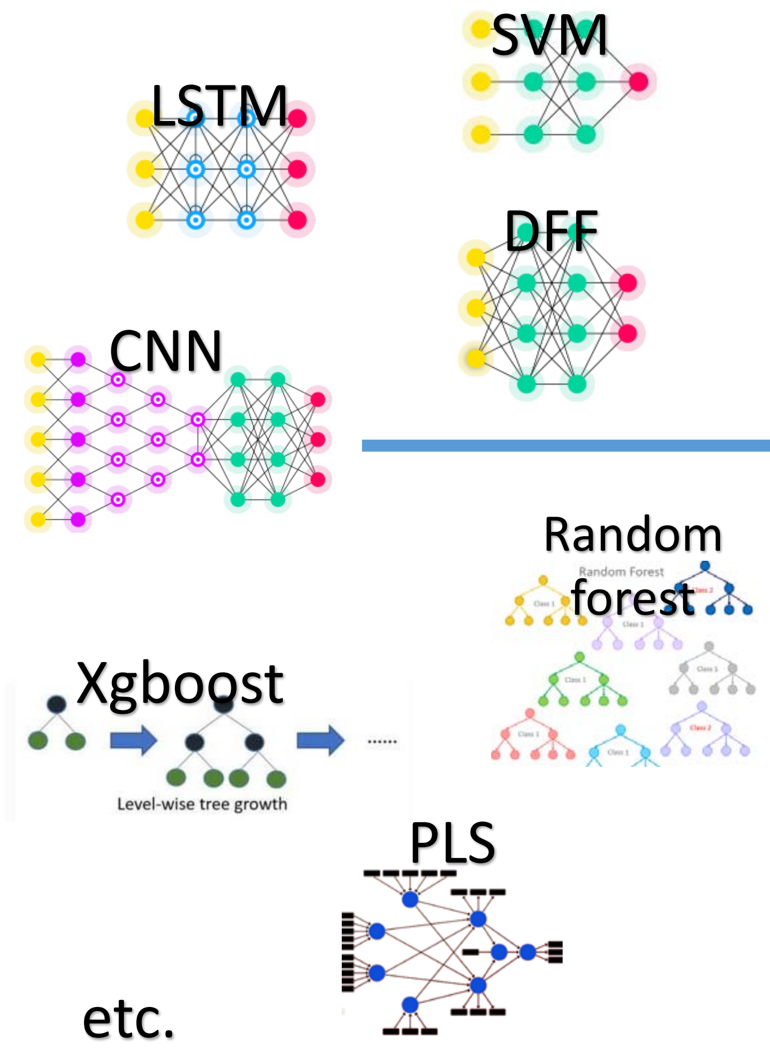
1D
Depthwise
Convolution

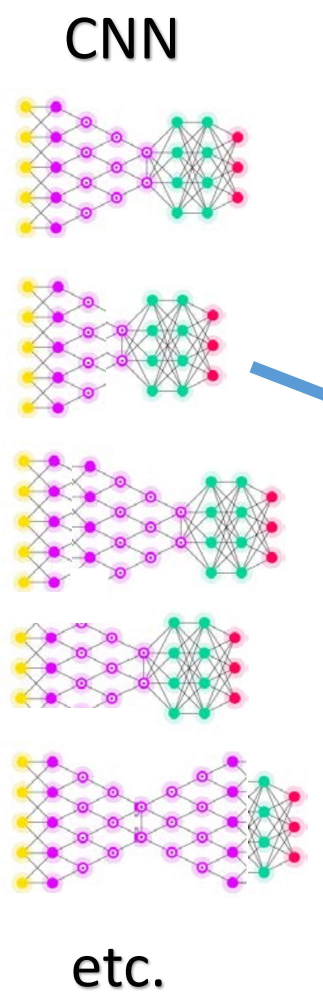


1D
Separable
Convolution

Model choice

BACON Hyperparametrization





BACON : BAsic COnvolutional neural network on Nirs

[2;4;8;16;32;64;128]

[0.01-0.9]

[3;5;7;9;11;13;15;17;21;31]

[sigmoid, tanh, elu, relu, swish, selu, softmax]

```
Sequential()
Input(shape=input_shape)
SpatialDropout1D(0.08)
Conv1D (filters=8, kernel_size=15, strides=5, activation='selu')
Dropout(0.2))
Conv1D (filters=64, kernel_size=21, strides=3,
activation='relu'))
BatchNormalization())
Conv1D (filters=32, kernel_size=5, strides=3, activation='elu')
BatchNormalization()
Flatten()
Dense(16, activation='sigmoid',
Dense(1, activation='sigmoid')
```

[Droupout(), SpatialDropout(), BatchNormalization()]

Model

Architecture

Hyperparameters

Generic pipeline

PiNARD: a Pipeline for Nirs Analysis ReloadeD

A NIRS data processing pipeline based on scikit-learn pipelines

Input Data & split → Augmenter → Preprocessing → Estimator → Prediction → Explainer

Interoperable
(sklearn, tensorflow, pytorch, shap, etc.)

Parallel
(joblib)

Modular
(reuse scipy functions, sklearn transformers, etc.)

Reification

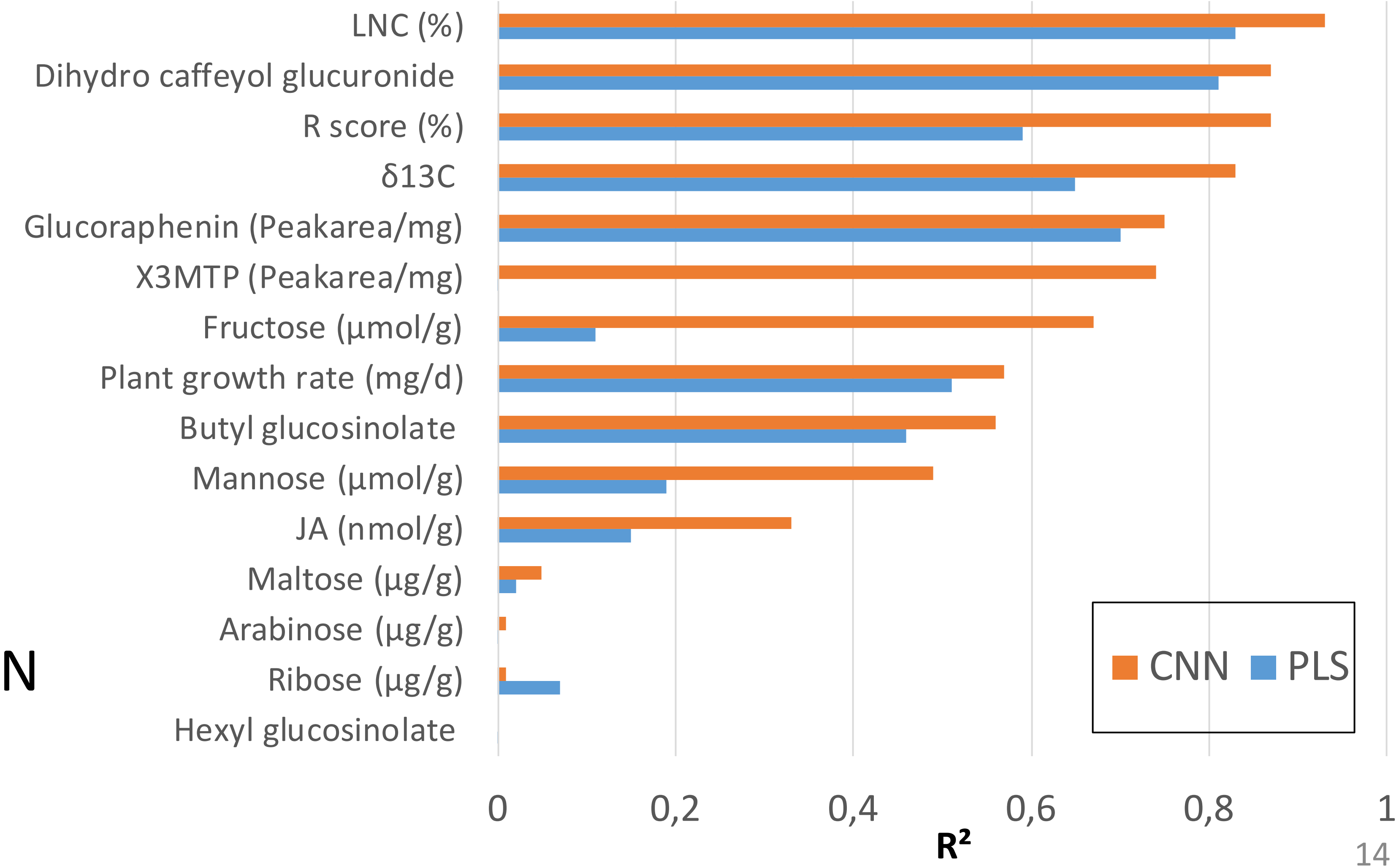
https://github.com/gbeurier/pinard

https://pypi.org/project/pinard/

Example - Regression

Vasseur et al. 2022 – Frontiers in Plant Science
Estarague et al. 2022 – Annals oy Botany

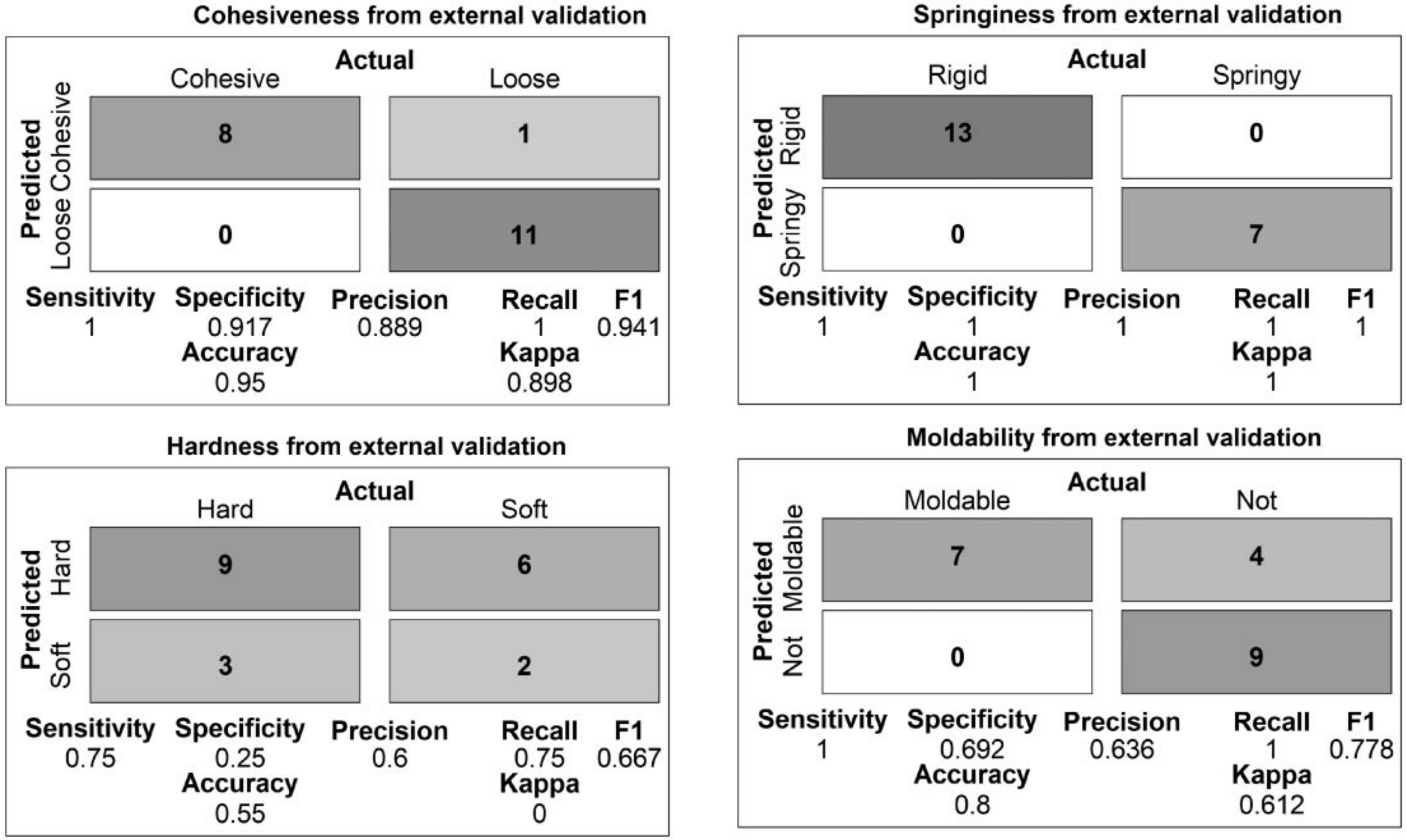
- *Arabidopsis thaliana*
- 21032 leaves spectra
- 108 traits
    - Physiology
    - Metabolic
    - Ecological strategy
- Optimized PLS vs BACON (CNN)



Example – Classification

Ehounou et al. 2021 – JNIRS

- *Dioscorea alata*
- Tuber flour
- Texture of pounded yam
  - Cohesiveness
  - Springiness
  - Hardness
  - Mouldability
- BACON (CNN)



Generic pipeline

PiNARD: a Pipeline for Nirs Analysis ReloadeD

A NIRS data processing pipeline based on scikit-learn pipelines

Input Data & split → **Augmenter** → **Preprocessing** → **Estimator** → **Prediction** → **Explainer**

Interoperable
(sklearn, tensorflow, pytorch, shap, etc.)

Parallel
(joblib)

Modular
(reuse scipy functions, sklearn transformers, etc.)

Reification

GitHub

PINARD

https://github.com/gbeurier/pinard

python™ Package Index

https://pypi.org/project/pinard/

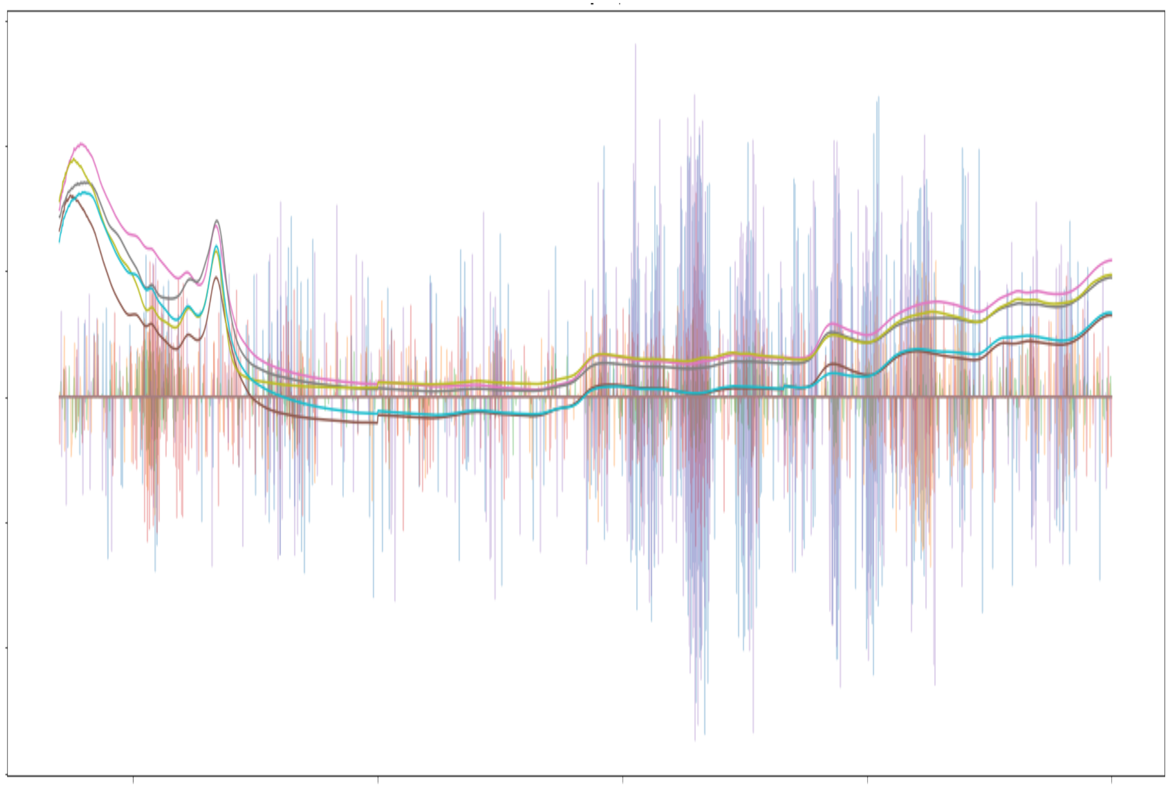
Generic pipeline

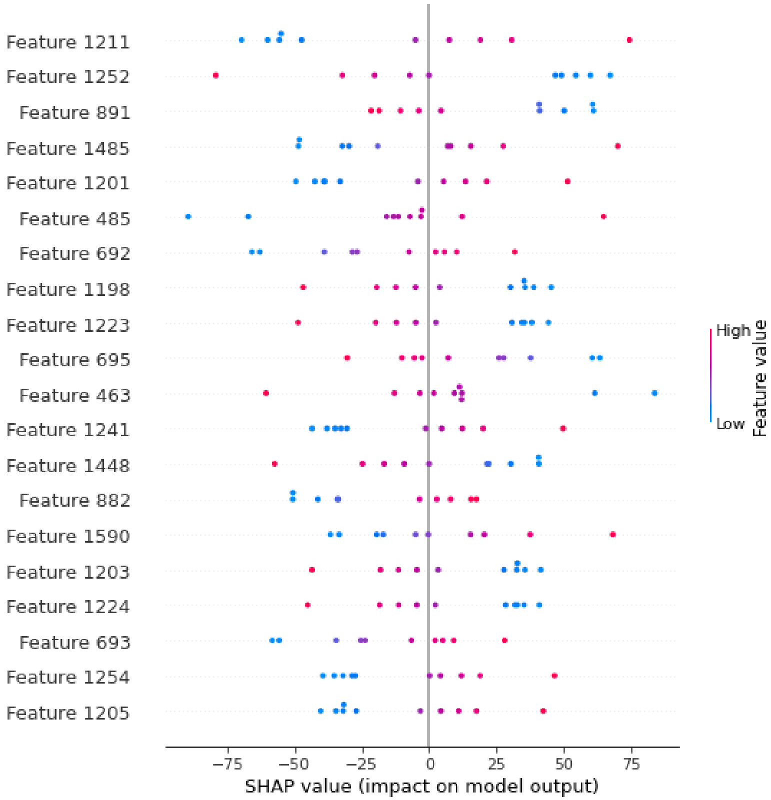
Intelligibility & Shapley values

SHAP

```
import shap

X_train_summary = shap.kmeans(X_train, 10)
explainer = shap.KernelExplainer(estimator.predict, X_train_summary)
shap_values = explainer.shap_values(X_test[0:5])
```





Generic pipeline

PiNARD: a Pipeline for Nirs Analysis ReloadeD

A NIRS data processing pipeline based on scikit-learn pipelines

Interoperable
(sklearn, tensorflow, pytorch, shap, etc.)

Parallel
(joblib)

Modular
(reuse scipy functions, sklearn transformers, etc.)

Reification

GitHub

PINARD

https://github.com/gbeurier/pinard

python™ Package Index

https://pypi.org/project/pinard/

Datasets

- De 150 à 8700 spectres
- De 100 à 2500 features
- Gamme large (350-2500) à restreinte (850-1050)
- 7 modèles de spectromètres différents
- 16 modèles publiés sur 20
- Échantillons frais et secs, broyés ou intacts

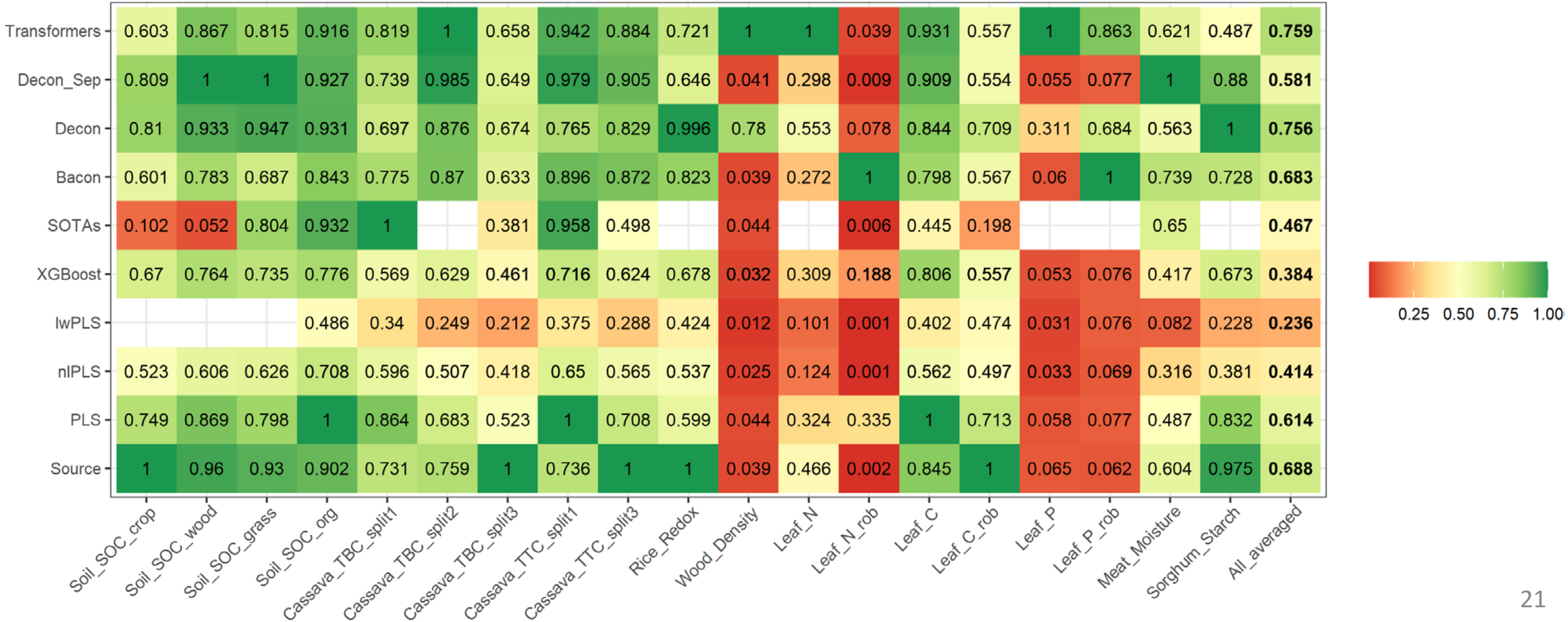
Dataset	State	Product	Measure	Instrument	λ_{min}	λ_{max}	N
Soil	dry	European soil	SOC	FOSS XDS	400	2500	3800 à 8700
Rice	fresh	Rice leaf	REDOX	"MEMS"	900	1700	3700
Cassava	fresh	Blended cassava root	TBC, TTC	FOSS 6500	400	2498	3500
Wood	dry	Eucalyptus wood	Density	Bruker MPA	1100	2500	1650
Leaf	dry	Plant leaf	N, P, C content	ASD FieldSpec	350	2500	290 à 550
Meat	fresh	Pork meat	Moisture, fat	Tecator	850	1050	215
Sorghum	dry	Sorghum grain	Starch content	Bruker Tango	867	2535	152

Classes de modèles

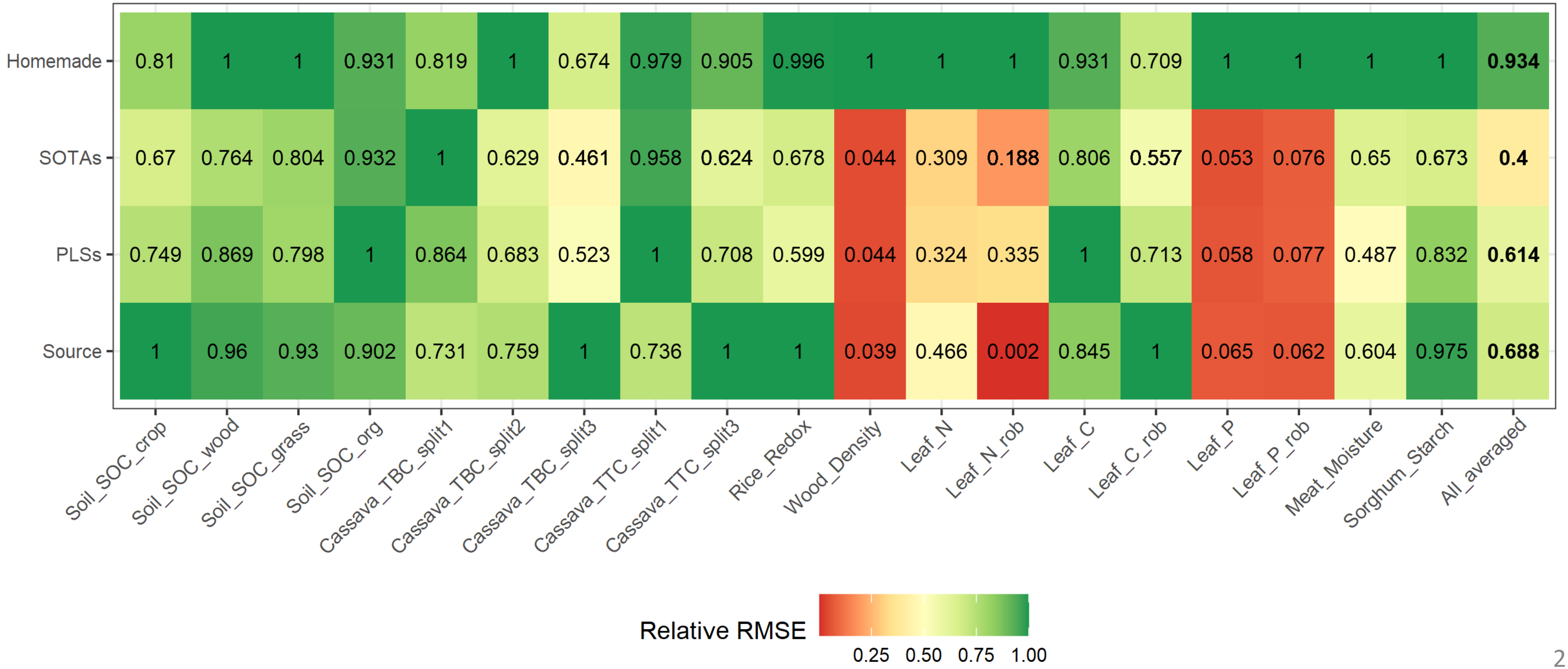
- PLS optimisées pour chaque jeu de données
  - Prétraitements (16 combinaisons)
  - Nombre de composantes (1 à 120)
  - Typess (PLS, lwPL, nlPLS)
  - 5760 modèles par jeu de données
- SOTAs (State Of The Art)
  - Adapté 2D -> 1D si nécessaire
  - Pas hyperparamétré
  - 5 classes : ResNet2, VGG1D, Xception1D, XGBoost, FFT_Conv
  - 5 combinaisons de prétraitements différentes
- Homemade
  - Hyperparamétrisés sur 2 jeux de données indépendants (architecture, hyperparams et prétraitements)
  - CNN, Depthwise CNN, Separable Depthwise CNN, Conv LSTM, Transformer, hybrids

Performances par modèle

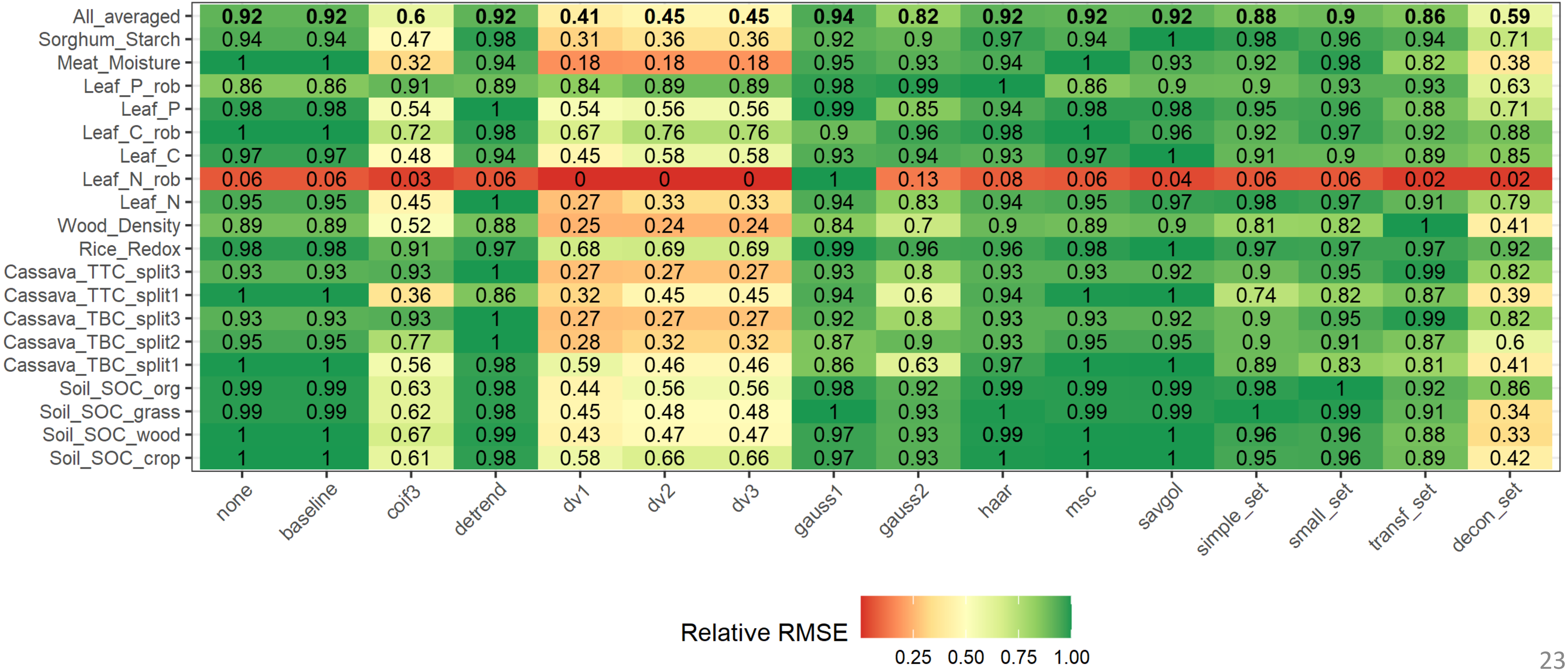
$$RRMSE_{dataset} = \frac{\min(RMSE)}{RMSE}$$



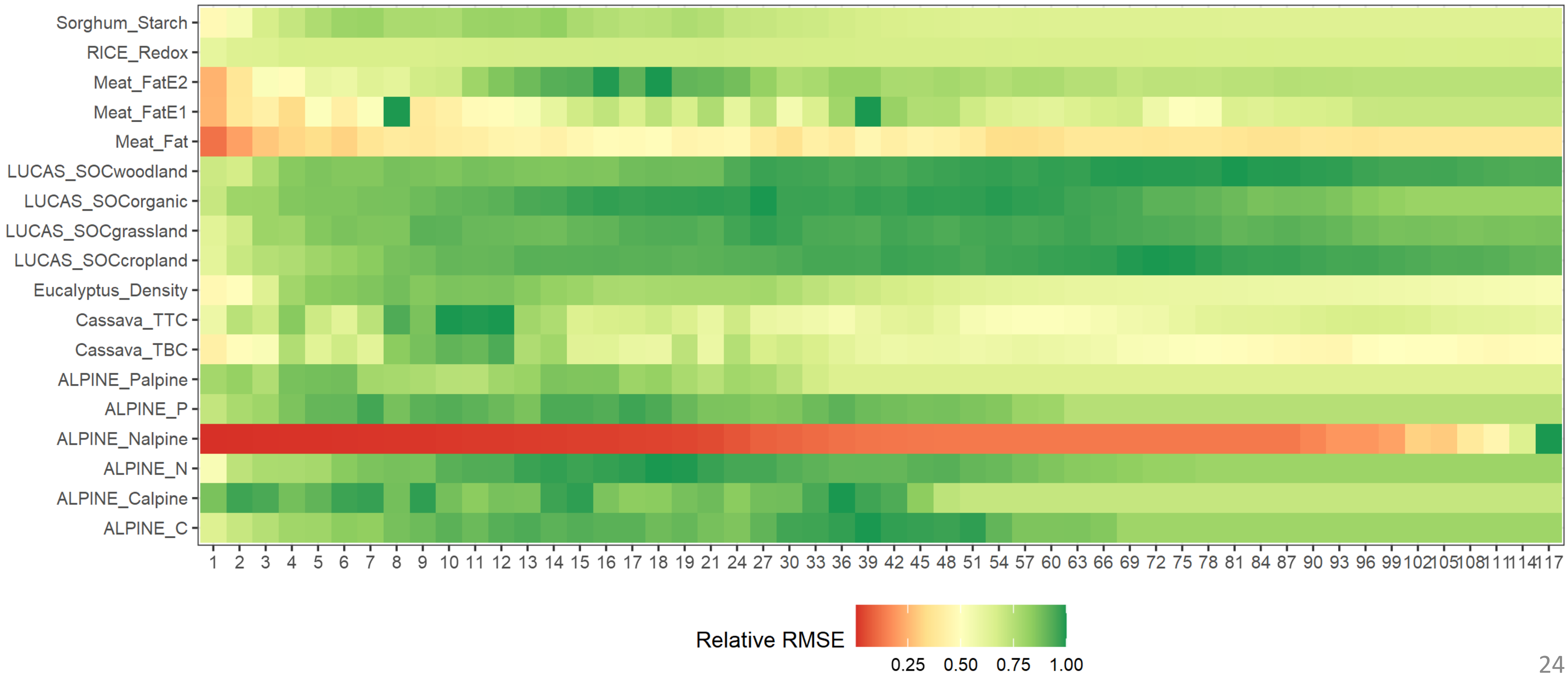
Performances par classe de modèles



PLS: preprocessing performances



PLS: optimal number of components?



Perspectives

- Environner PINAR pour faciliter le prototypage (French PINARD: Fast-track Robust EvaluatioN and Calibration Helper for PINARD)
- Assemblage de modèles pour plus de généricité
- Inclusions de données hétérogènes
  - Phenomic : données environnementales & NIRS
  - Modèles prédictifs : variables explicatives supplémentaires
  - Multimodale (NIRS, MIRS, raman OU gestion individuelle des capteurs VNIR, SWIR1, SWIR2)
- Choix des metrics de distance
  - Impacte : outlier detection, average repetition, identify reference spectra, dimension reduction, k-means et lwPLS…
  - Lock-step measures (e.g. Euclidien, mahalanobis, T^2…) -> elastic measures (e.g. DTW…)
- Standardisation (GAN…)
- Self supervised denoising
  - Database (BDD de 1 à 2 millions de spectres sans mesure de reference)
  - Applications : resampling, in/out painting, denoise, data augmentation…

Thanks