Apprendre à prédire avec des données

Introduction et exemples

David Makowski

INRA

david.makowski@inra.fr

- Principes généraux et exemples
- Zoom 1: Random forest et gradient boosting
- Zoom 2: Réseaux de neurones multi-couches et deep learning

Question → Données → Entrainement → Test

Statistical Modeling: The Two Cultures (Breiman, 2001)

$$y = f(x) + e$$

Approche 1 : Trouver le vrai $f(x)$
Approche 2 : Prédire le plus précisément possible y à partir de x

Statistical Modeling: The Two Cultures (Breiman, 2001)

$$y = f(x) + e$$

Approche 1 : Trouver le vrai $f(x)$

Approche 2 : Prédire le plus précisément possible *y* à partir de *x*

Statistical Modeling: The Two Cultures (Breiman, 2001)

$$y = f(x) + e$$

Approche 1 : Trouver le vrai $f(x)$

Approche 2 : Prédire le plus précisément possible *y* à partir de *x*

- **On compare plusieurs méthodes pour prédire *y* avec *x***
- **La plus précise est choisie (pour un coût d'usage max. donné)**

Quoi de neuf ?

Des méthodes flexibles

+

Puissance de calcul

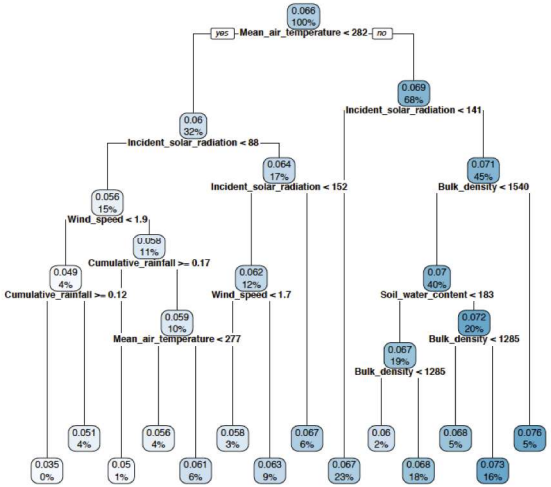
+

Des données nombreuses et diversifiées

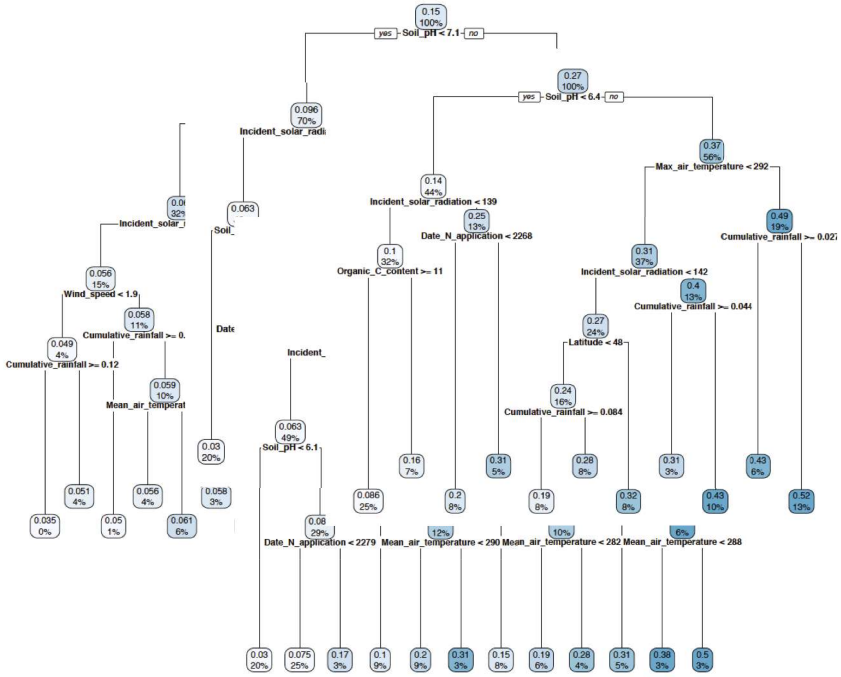
Plus de chance d'obtenir un algorithme prédictif précis

Des méthodes (très) flexibles

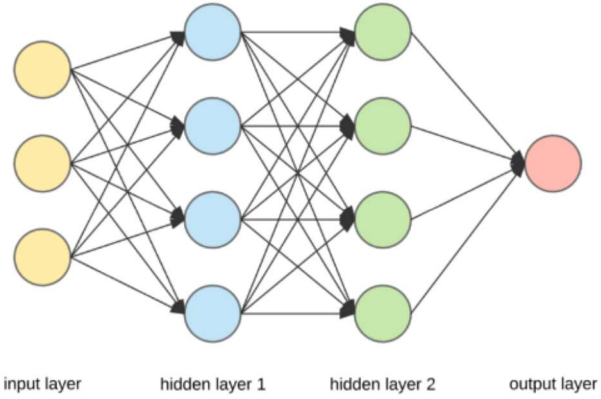
Arbre de régression



Forêt aléatoire



Réseau de neurones multicouches



Question → Données → Entrainement → Test

Place centrale des données et de l'évaluation

Question → **Données** → **Entrainement** → **Test**

Jeu de données indépendant

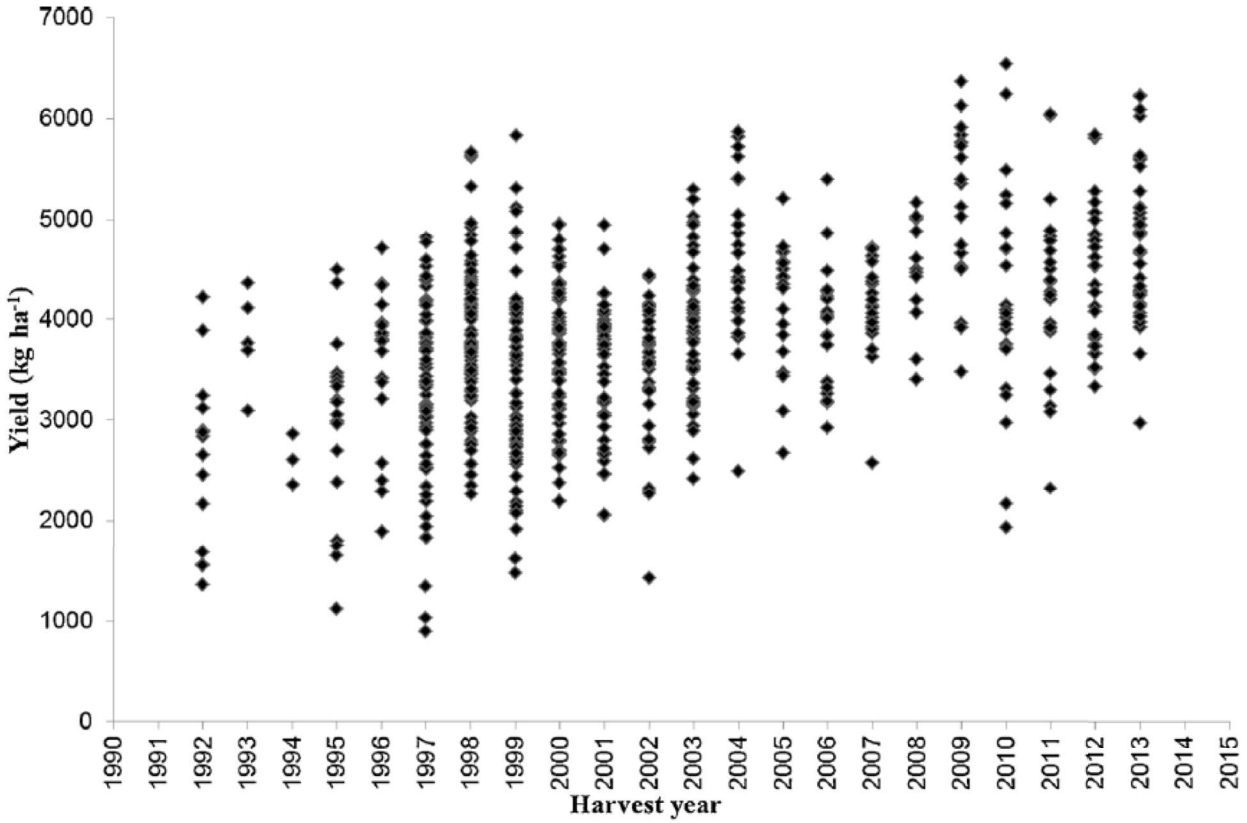
Validation croisée

Ex. 1. Prédiction des rendements

Comparison of regression techniques to predict response of oilseed rape yield to variation in climatic conditions in Denmark

Behzad Sharif[a,*], David Makowski[b], Finn Plauborg[a], Jørgen E. Olesen[a]

Prediction of oilseed rape yields in Denmark



Sharif et al. (2017)

Prediction of oilseed rape yields in Denmark

$$\log\left(Yield_j\right) = b_0 + b_1 \times YEAR_j + \sum_{i=1}^{n} b_{2i} \times TEMP_{ij} + \sum_{i=1}^{n} b_{3i}$$

$$\times RAD_{ij} + \sum_{i=1}^{n} b_{4i} \times PREC_{ij} + \sum_{i=1}^{n} b_{5i} \times TEMP_{ij}^{\,2} + \sum_{i=1}^{n} b_{6i}$$

$$\times RAD_{ij}^{\,2} + \sum_{i=1}^{n} b_{7i} \times PREC_{ij}^{\,2} + b_8 \times SOIL_j + b_9 \times Pre_{CROPj}$$

$$+ b_{10} \times Sowing_j + b_{11} \times Sowing_j^{\,2} + \varepsilon_j$$

Sharif et al. (2017)

Entrainement =
estimation des paramètres et sélection de variables

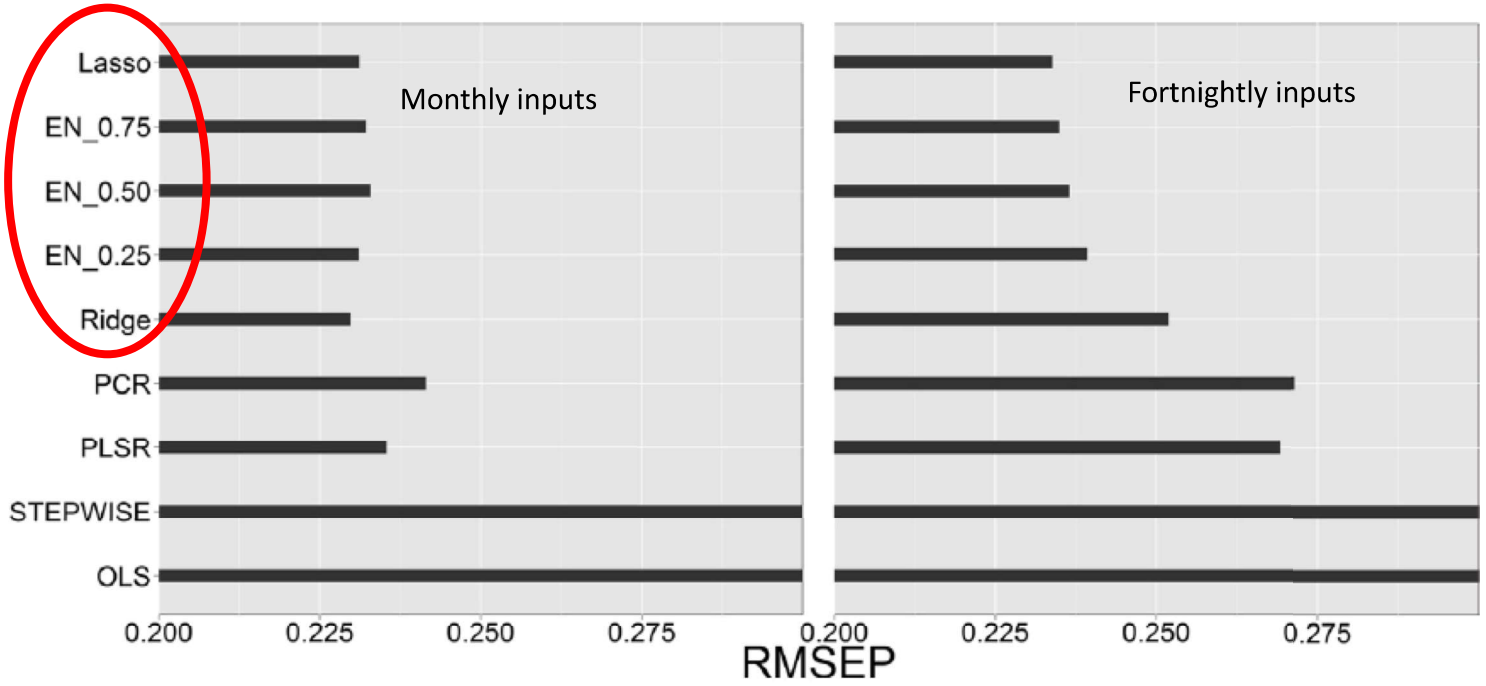
$$\sum_{i=1}^{p}\left(y_i - b_0 - \sum_{j=1}^{q} b_j \ .x_{ij}\right)^2 + \lambda \sum_{j=1}^{q} |b_j|$$

Moindres carrés ordinaires

Terme de pénalisation de la complexité

Prediction of oilseed rape yields in Denmark

Moindres carrés pénalisés



Sharif et al. (2017)

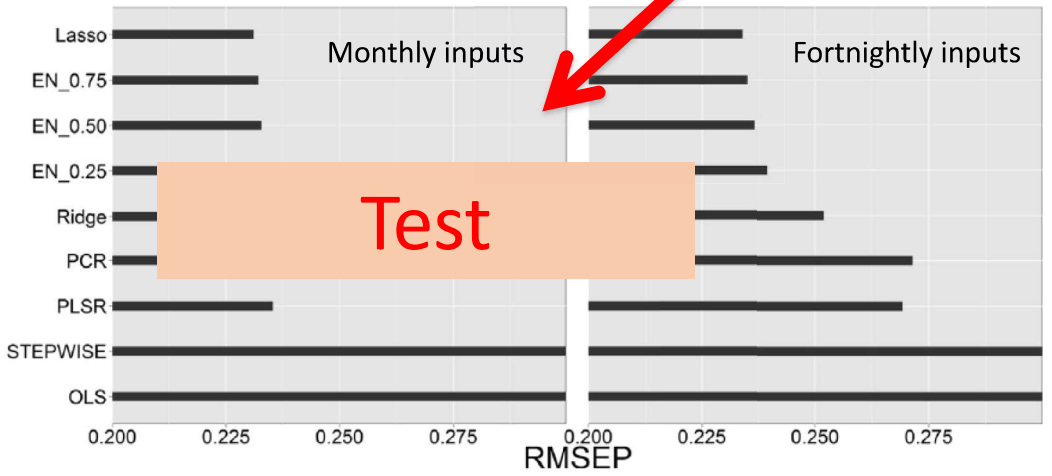
Prediction of oilseed rape yields in Denmark

$$\log\left(Yield_j\right) = b_0 + b_1 \times YEAR_j + \sum_{i=1}^{n} b_{2i} \times TEMP_{ij} + \sum_{i=1}^{n} b_{3i}$$

$$\times RAD_{ij} + \sum_{i=1}^{n} b_{4i} \times PREC_{ij} + \sum_{i=1}^{n} b_{5i} \times TEMP_{ij}^{\,2} + \sum_{i=1}^{n} b_{6i}$$

$$\times \ldots \quad \ldots re_{CROPj}$$

Entrainement

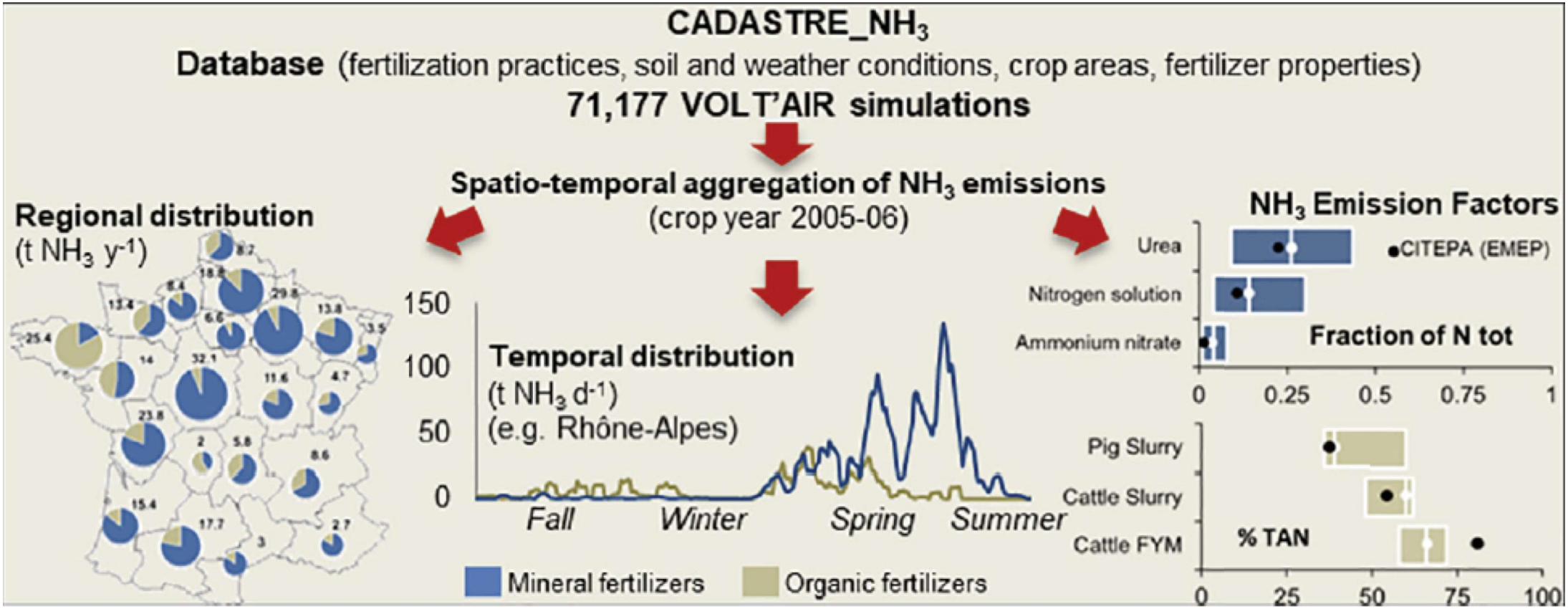


Sharif et al. (2017)

Ex. 2. Méta-modélisation: émissions de NH_3

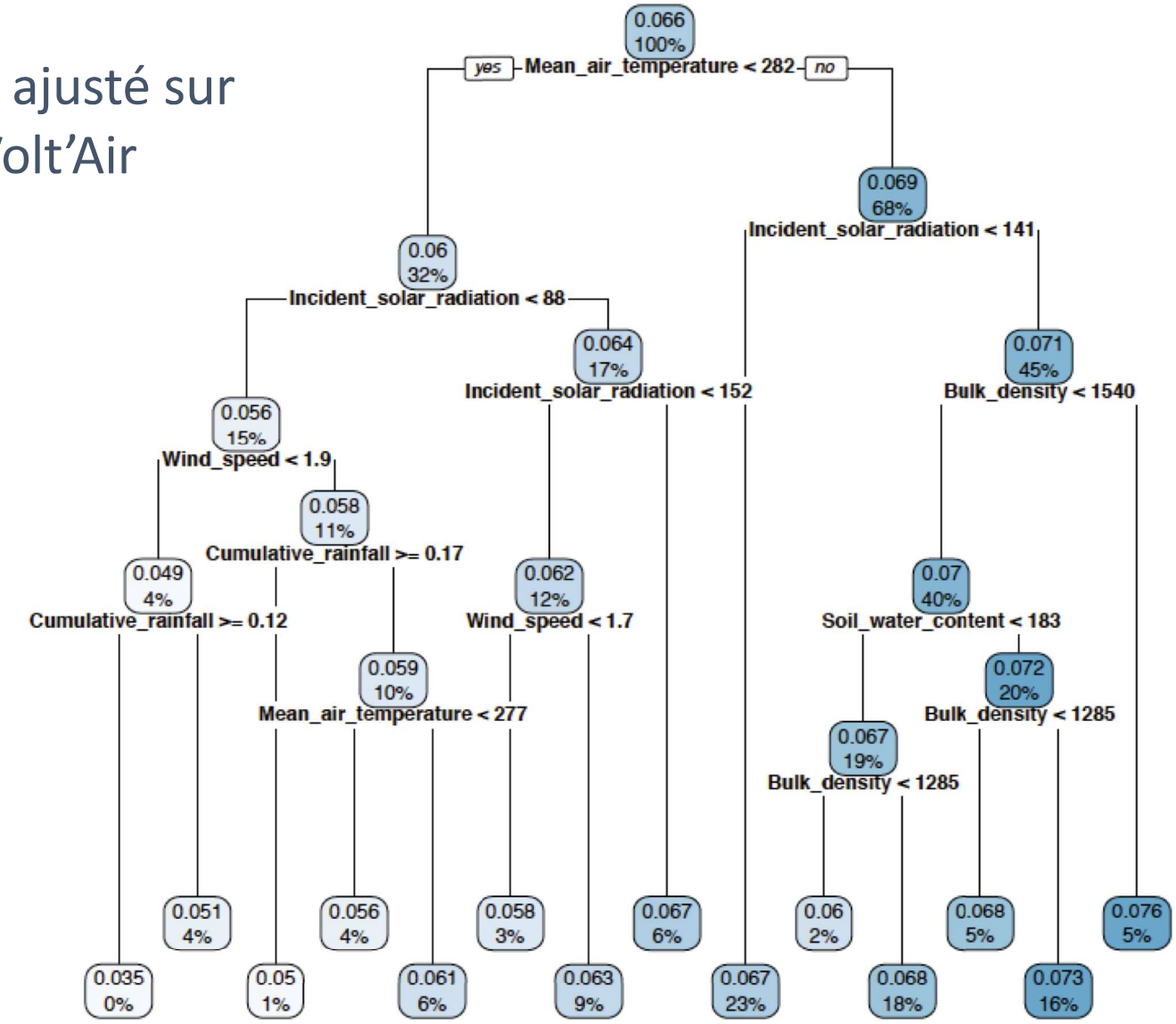
Meta-modeling methods for estimating ammonia volatilization from nitrogen fertilizer and manure applications

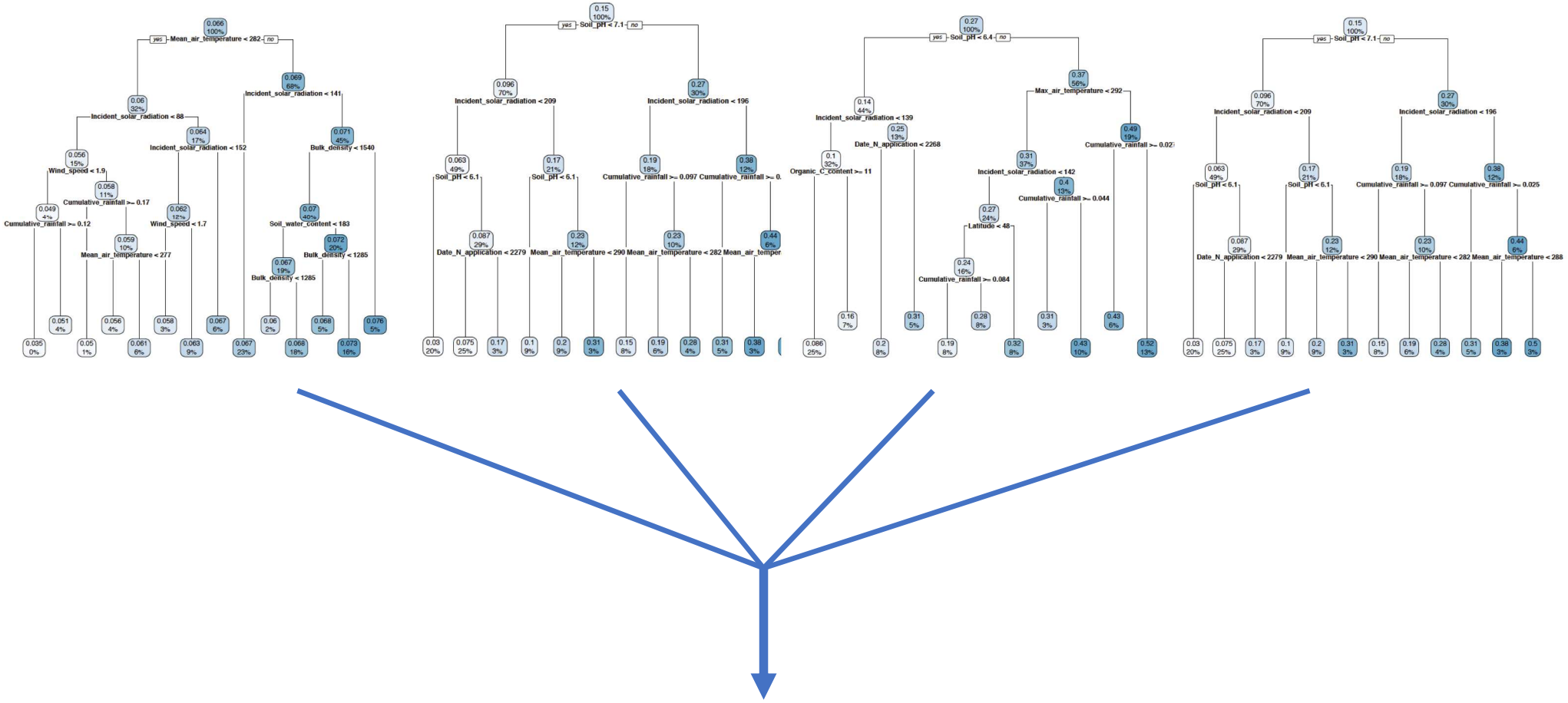
Maharavo Marie Julie Ramanantenasoa[a,b], Sophie Génermont[a,*], Jean-Marc Gilliot[a], Carole Bedos[a], David Makowski[c,d]



Ramanantenasoa et al., 2018

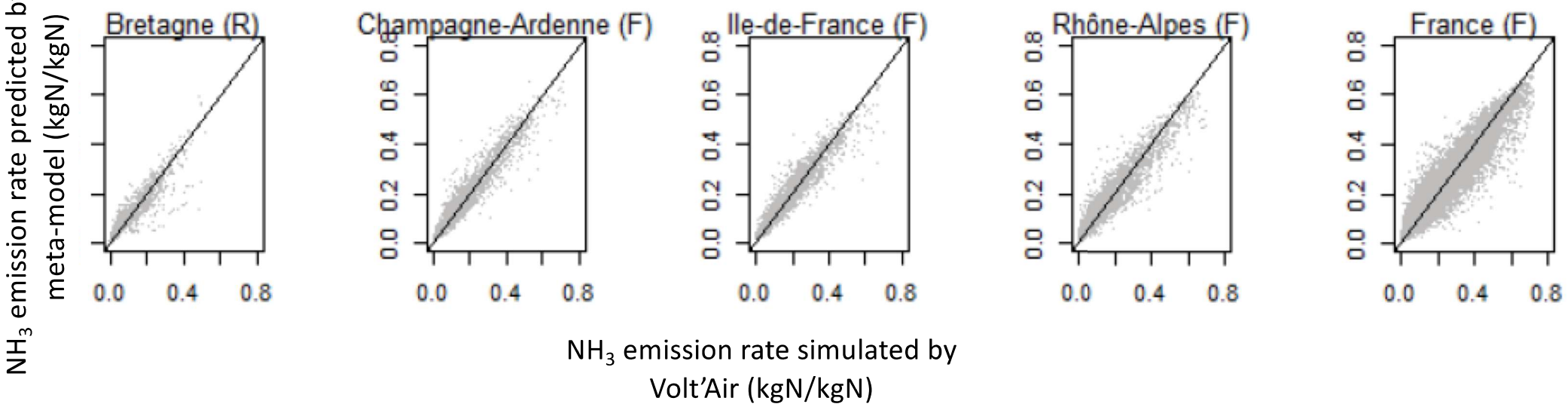
Arbre de régression ajusté sur les simulations de Volt'Air





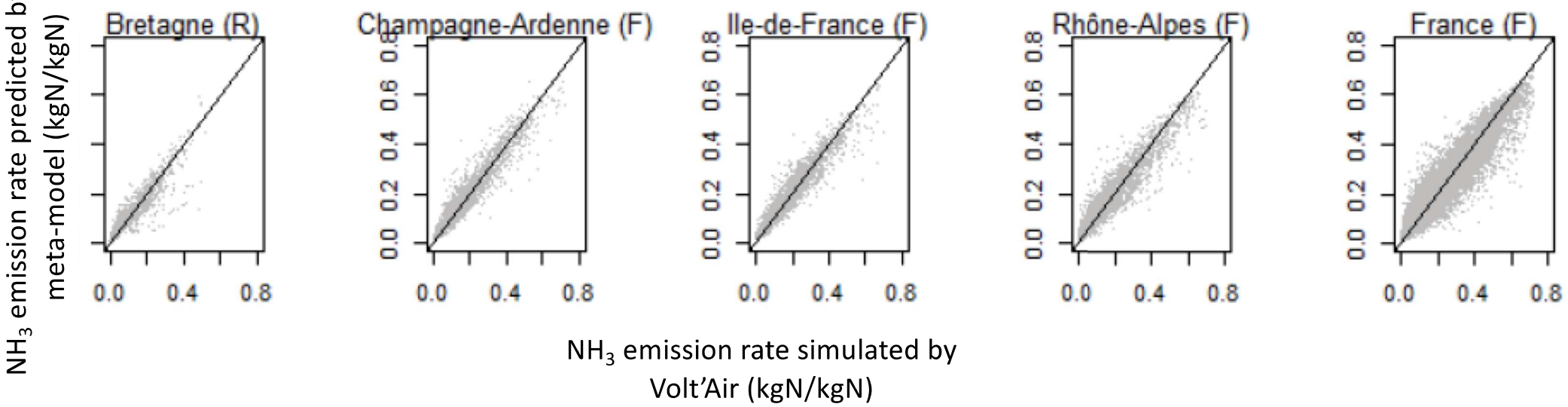
Prédiction de la « forêt »

Relation entre le méta-modèle et le modèle mécaniste Volt'Air



Avantage du méta-modèle

- Moins de variables d'entrée
- Temps de calcul beaucoup plus faible



De nombreuses sources d’informations

- Données expérimentales
- Simulations
- Enquêtes

De nombreuses sources d'informations

- Données expérimentales
- Simulations
- Enquêtes

Mais aussi

- Mesures chez les agriculteurs
- Données extraites d'articles et de rapports
- Données issues d'avis d'experts
- Données climatiques/satellitaires
- Photos/Images

Ex. 3. Vers du deep learning en agriculture?

Pré-traitement + réseau de neurones profond

MILDIOU

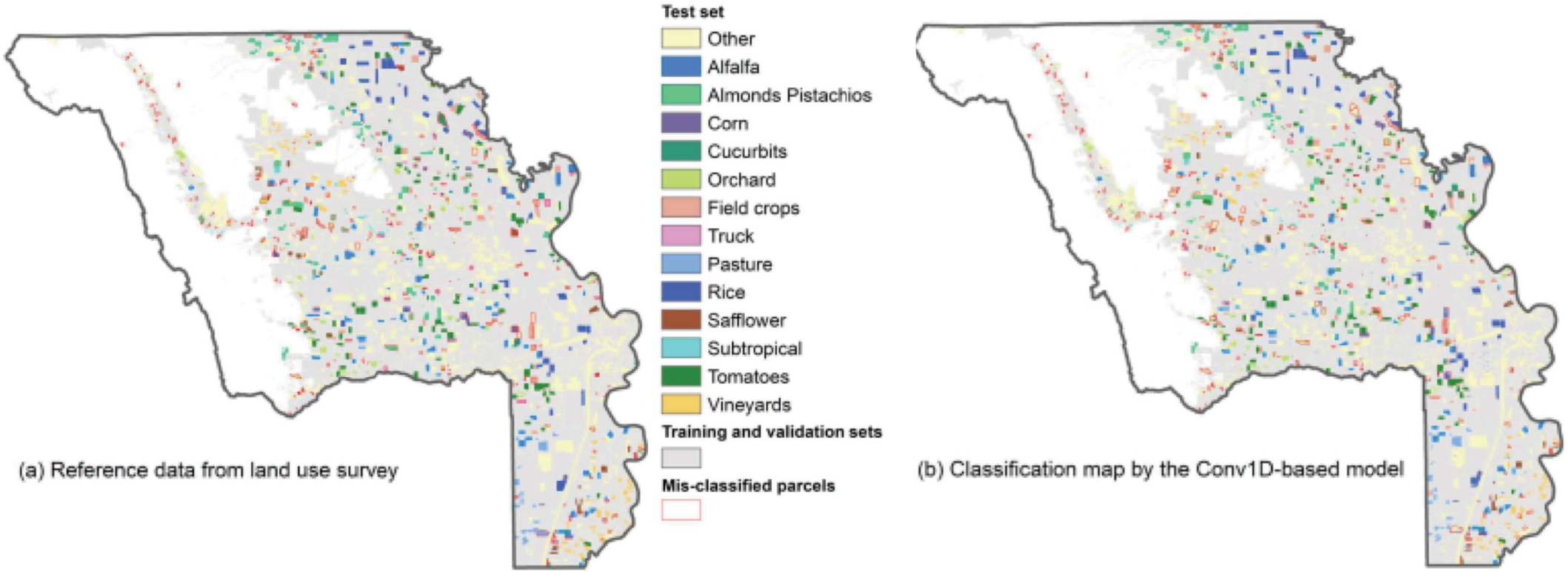
Oui/Non/Combien

Ex. 4. Vers du deep learning en agriculture (bis)?

Deep learning based multi-temporal crop classification☆

Liheng Zhong[a,*], Lina Hu[b], Hang Zhou[c]

Classification from Landsat surface reflectance in Yolo County, California

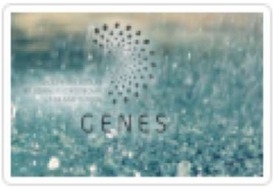


(a) Reference data from land use survey

(b) Classification map by the Conv1D-based model

Une autre façon de résoudre des problèmes : les « data challenges »

www.datascience.net



Prédiction d'intervalles de températures

L'objectif est de prévoir les meilleurs intervalles de températures de la ville de Candelia à partir des données météorologiques sur cette ville des cinq jours précédents. La qualité est mesurée par la longueur moyenne des intervalles. Le livrable est un ensemble d'intervalles.

1 000 €

Ouvrir

Challenge **terminé**

Prévision de consommation d'électricité d'un site tertiaire

Le challenge consiste à proposer un modèle de prévision à moyen terme de la consommation d'électricité d'un site tertiaire d'un industriel, par tranche de 10 minutes.

1 000 €

Ouvrir

Challenge **terminé**

http://cland.lsce.ipsl.fr/index.php/workshops/forecasting-crop-yields/33-data-challenge-in-french

Crop Data Challenge 2018 : Prédiction des rendements agricoles

Quels sont les objectifs ?

- Comparer les performances de méthodes statistiques et d'apprentissage automatique pour prédire les rendements agricoles,
- Promouvoir les échanges de connaissances autour des méthodes de prédiction pour l'enseignement et les applications agricoles.

Une procédure simple :

- Choisir un challenge (deux challenges sont proposés, un sur le blé et un sur le maïs),
- Développer votre algorithme de prédiction à partir du jeu de données d'entrainement pour le blé et/ou le maïs,
- Soumettre vos prédictions en ligne pour le jeu de données « test » pour le blé et/ou le maïs avant le 16 novembre 2018.

Règlements :

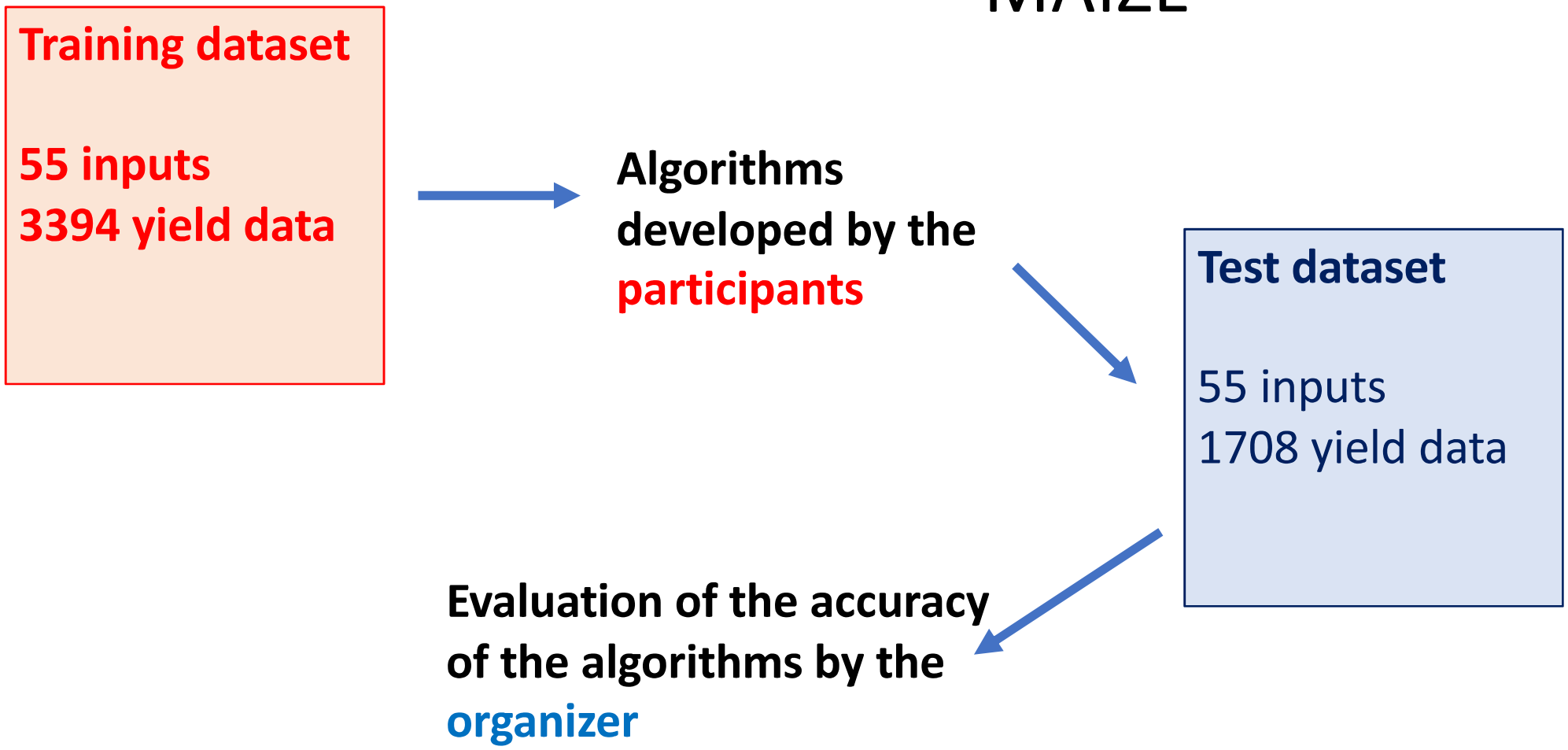
- « Crop Data Challenge 2018 - Prédiction des rendements du maïs en France ».
- « Crop Data Challenge 2018 - Prédiction des pertes de rendement du blé en France »

Fichiers "entrainement" et "test" ici

14 participants

- Des méthodes très diversifiées
  - Régression logistique
  - Régressions pénalisées
  - Forêts aléatoires
  - Réseaux de neurones simple et multi-couches
  - Clustering

MAIZE


Training dataset
55 inputs
3394 yield data
Algorithms developed by the participants
Test dataset
55 inputs
1708 yield data
Evaluation of the accuracy of the algorithms by the organizer

Benefits of data challenges

- Stimulate the development of shared datasets
- Allow rigorous comparison of methods
- Provide insights on pros and cons of different methods
- Useful to collect innovative ideas from outside
- Can be joined to a training session on machine learning

RMSE

2.949

1.26

0.928

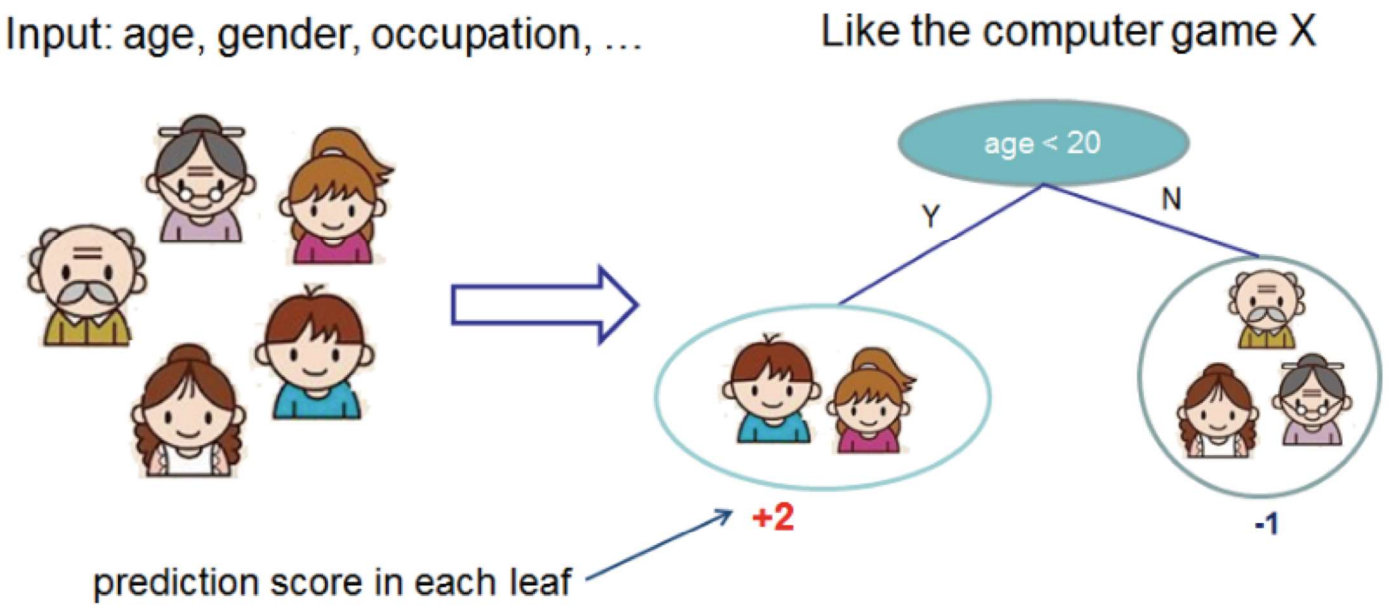
0.851

0.849

0.848

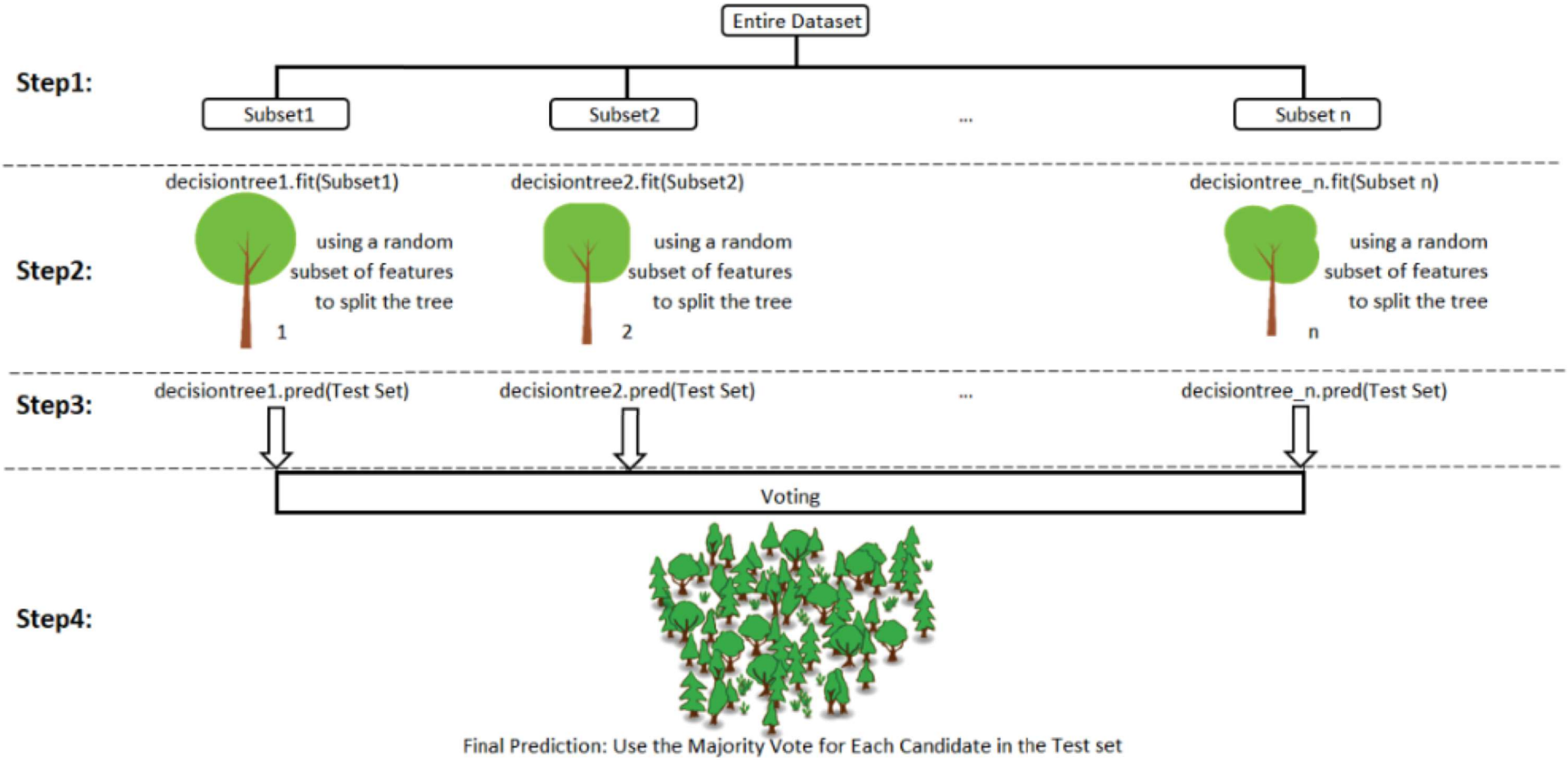
0.8399

Zoom 1: Random forest and gradient boosting



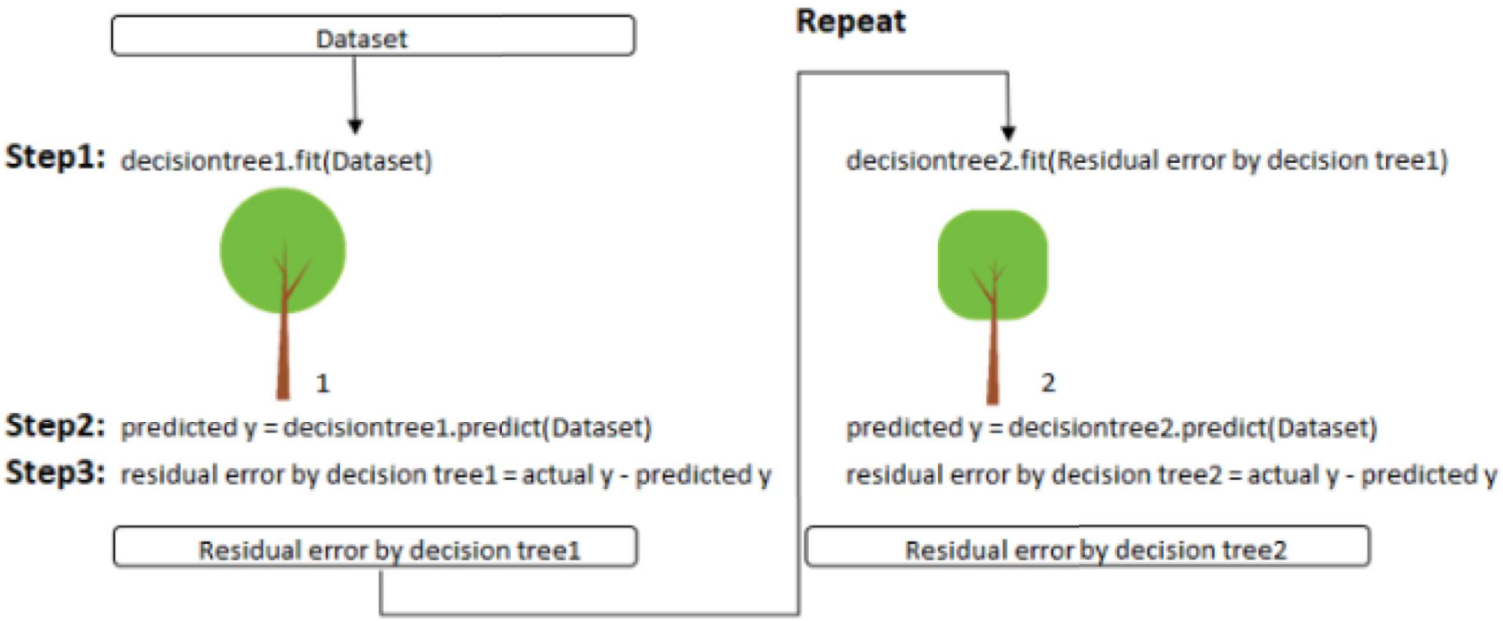
https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Random forest approach



https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725

Gradient boosting approach



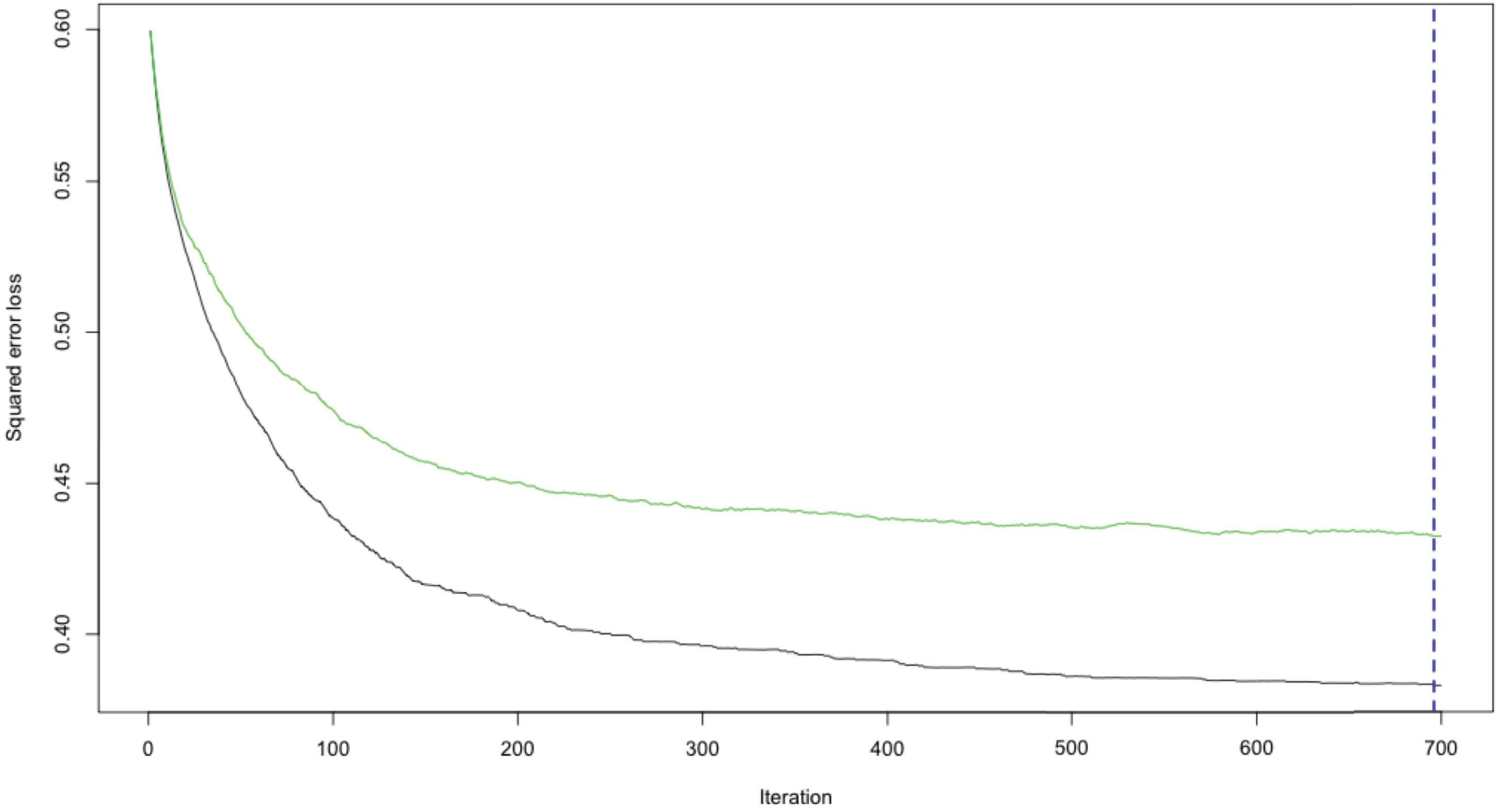
https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725

Application: prédire le rendement du maïs en France

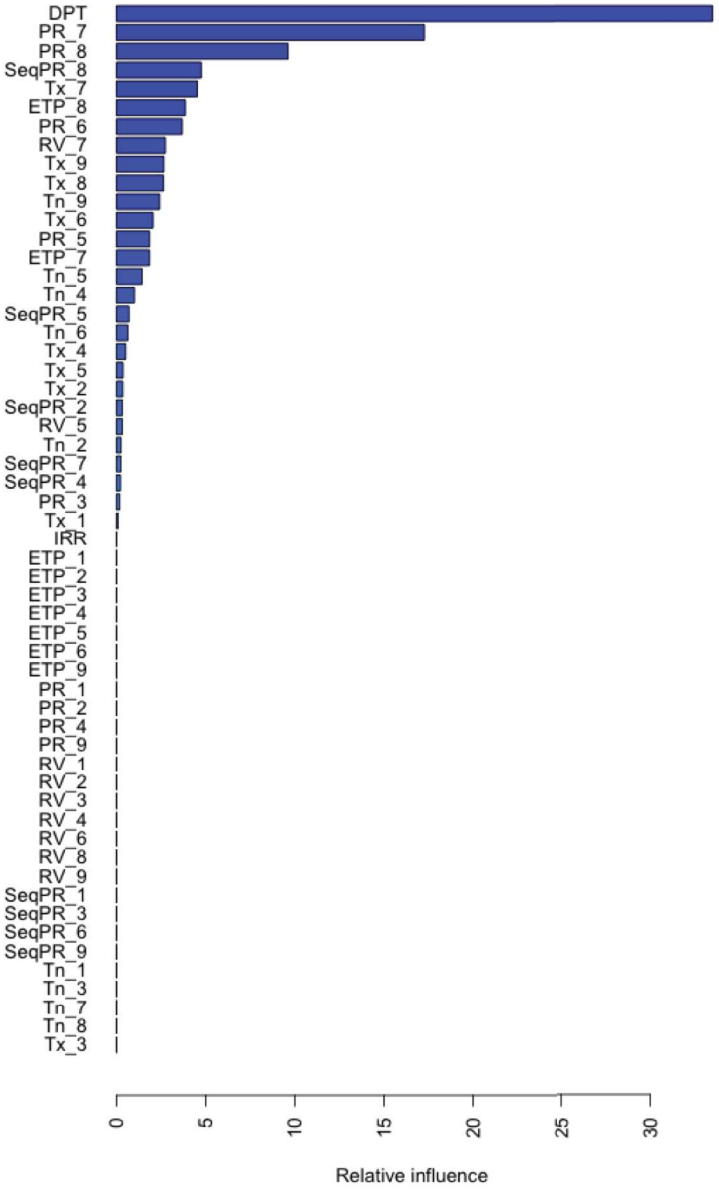
```
> head(DATAy)
       Yield Year IRR    ETP_1    ETP_2    ETP_3   ETP_4   ETP_5   ETP_6   ETP_7   ETP_8   ETP_9    PR_1      PR_2    PR_3    PR_4
1 -0.3862695 1959   3 0.480319 0.650041 0.952222 1.91736 2.72082 3.49425 4.28025 3.17082 1.75560 4.05646 0.0364394 4.80483 4.92613
2  0.1678717 1960   3 0.291583 0.577265 0.981122 2.12405 3.09640 3.57063 2.51640 2.39365 1.32290 3.49145 4.4775100 4.33698 0.96453
3 -0.2818916 1961   3 0.225639 0.573471 1.416320 1.55965 2.51640 3.24141 3.42725 2.86990 2.10262 3.48751 3.1553100 1.27740 3.88140
4 -0.8353380 1962   3 0.376123 0.402265 0.989877 1.85831 2.02216 3.92564 3.66216 3.68923 1.93872 3.72150 3.4689000 5.05512 3.17776
5  0.7080965 1963   3 0.201415 0.349833 0.920070 1.32144 2.47706 2.42592 3.32625 2.32699 1.29623 2.16936 3.6324400 4.63769 3.69405
6 -1.4500704 1964   3 0.235087 0.444469 0.510438 1.73645 2.41834 3.51096 4.41394 3.01169 1.94824 1.24272 1.5125900 5.31356 2.84759
     PR_5    PR_6     PR_7    PR_8     PR_9    RV_1     RV_2     RV_3    RV_4    RV_5    RV_6    RV_7    RV_8    RV_9  SeqPR_1
1 1.31709 2.59149 1.116000 2.96198 0.933639 51.0494 103.1240 100.3250 156.393 212.716 239.995 266.448 220.982 150.517 0.341515
2 1.82152 2.81858 4.092340 5.74437 7.676310 43.6871  72.6408 108.5880 180.077 222.952 234.407 186.702 176.072 123.731 0.416900
3 1.79057 2.33787 4.041520 2.38588 0.642520 32.9697  73.4175 166.0900 137.571 204.488 225.688 233.286 206.122 156.286 0.627279
4 3.04124 1.35275 0.676303 1.69645 3.138570 44.6328  73.7346 117.8080 168.202 168.662 277.445 240.403 247.275 158.200 0.510168
5 2.26305 6.25565 1.989530 7.40782 2.212850 52.4708  75.2086 102.6840 125.133 199.030 180.159 233.137 175.962 121.204 0.468794
6 2.15236 2.29069 1.000380 2.24094 2.327600 50.7146  72.1086  67.1326 148.749 187.654 231.976 276.754 203.253 153.330 0.196704
    SeqPR_2  SeqPR_3  SeqPR_4  SeqPR_5  SeqPR_6  SeqPR_7  SeqPR_8  SeqPR_9      Tn_1      Tn_2     Tn_3    Tn_4     Tn_5    Tn_6
1 0.0015528 0.317672 0.453623 0.292777 0.434420 0.255961 0.366760 0.269565 -1.334640 -1.567460  4.63960 5.51576  9.00653 12.9011
2 0.4760120 0.477560 0.312319 0.311010 0.363043 0.430575 0.514727 0.571377 -2.365930  0.841286  3.65670 3.87700  9.73954 13.1460
3 0.4274070 0.137097 0.567391 0.373773 0.510145 0.388850 0.280505 0.135870 -0.388616  2.692020  1.16167 7.85662  7.57757 12.7788
4 0.3905280 0.458275 0.474638 0.527700 0.167391 0.356942 0.152174 0.286594  0.619777 -2.754220 -1.29949 3.74210  7.24426 10.7943
5 0.4996120 0.601683 0.401449 0.360449 0.692391 0.422861 0.632188 0.358696 -7.751870 -5.703010  1.60362 5.56457  7.40275 12.5736
6 0.3092200 0.553647 0.466667 0.441445 0.428623 0.229313 0.470196 0.334783 -3.062120  0.103191  1.75535 5.66051 10.01580 12.9838
     Tn_7    Tn_8     Tn_9      Tx_1      Tx_2     Tx_3    Tx_4    Tx_5    Tx_6    Tx_7    Tx_8    Tx_9 DPT
1 15.8822 13.3508 12.60090  4.458120  6.997290 12.34220 14.0660 18.5257 22.3311 26.4056 23.4798 22.0578 AIN
2 12.1531 12.9508  9.08623  2.900720  8.175610 11.02550 13.2580 20.4857 22.5911 20.8484 21.1559 16.7304 AIN
3 12.8329 12.2605 13.76390  3.879340 10.042700 12.18090 16.3347 16.2052 21.4233 22.5490 22.4748 24.6548 AIN
4 12.9149 13.6179 10.29380  5.537650  3.681010  6.58445 12.8914 15.7882 20.8746 23.6048 25.8095 19.7318 AIN
5 14.8594 12.0416 10.68060 -2.122810  0.232842  9.23221 13.4903 16.2551 20.0708 23.5810 20.3687 18.5353 AIN
6 14.8936 12.3592 11.56320  0.187899  6.834210  7.26100 14.1535 18.9244 22.2869 26.1986 22.9112 20.6706 AIN
```

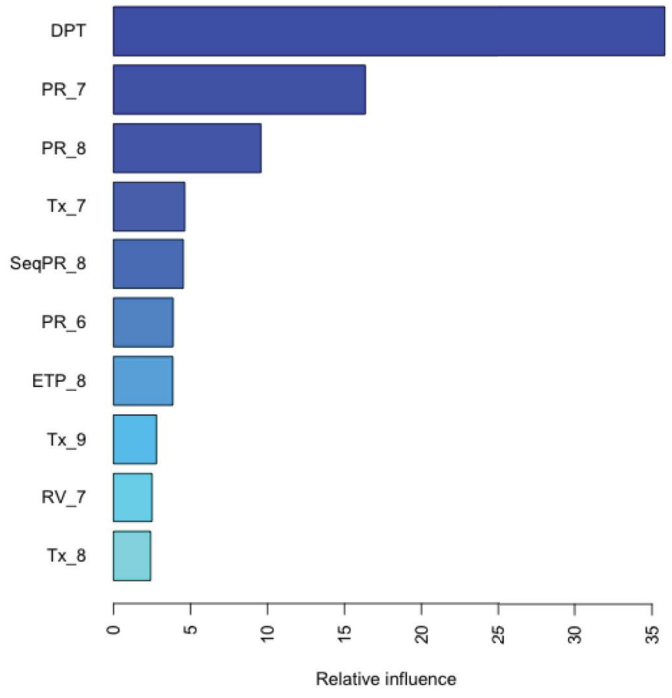
N=5102

Précision des prédictions en fonction du nombre d’arbres (gradient boosting)

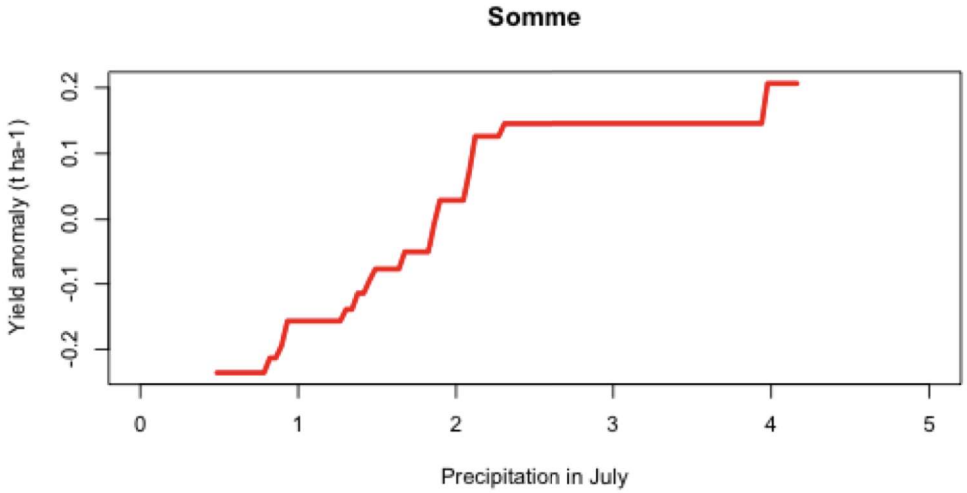


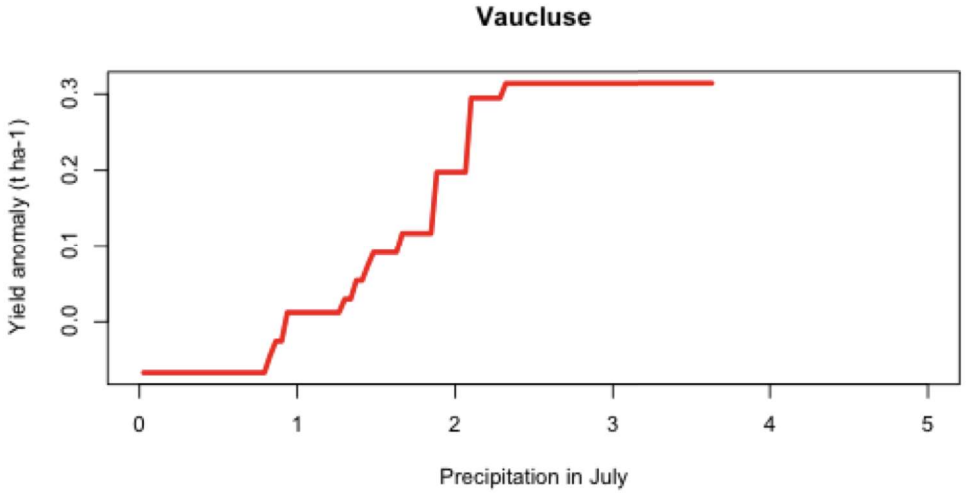
Importances des variables d'entrée (gradient boosting)

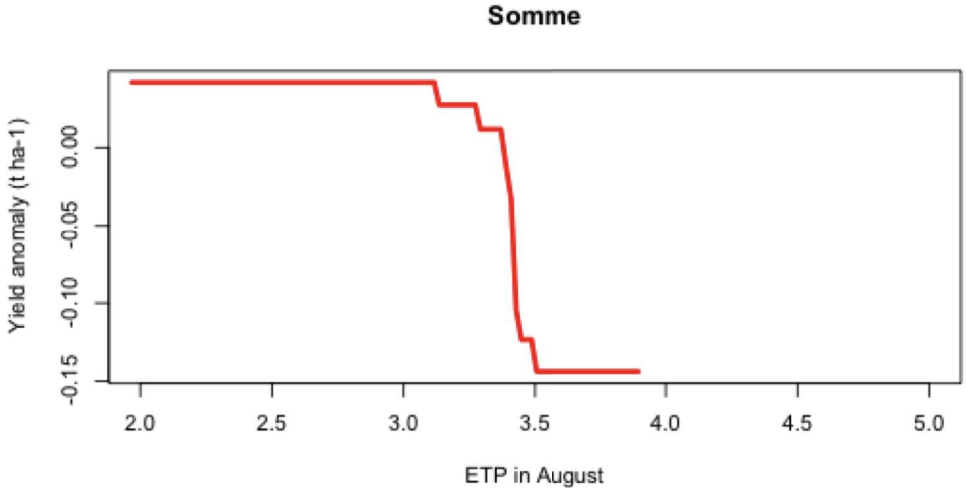


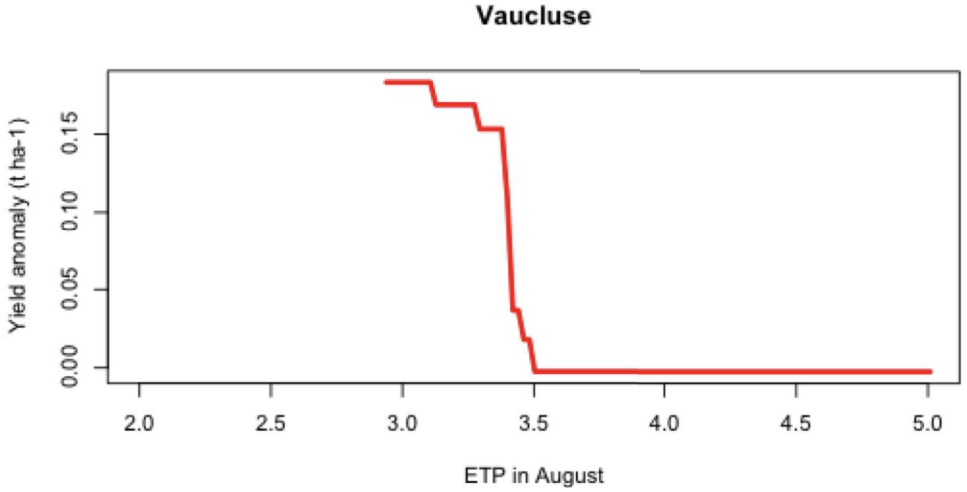


Réponses à des variables influentes (gradient boosting)

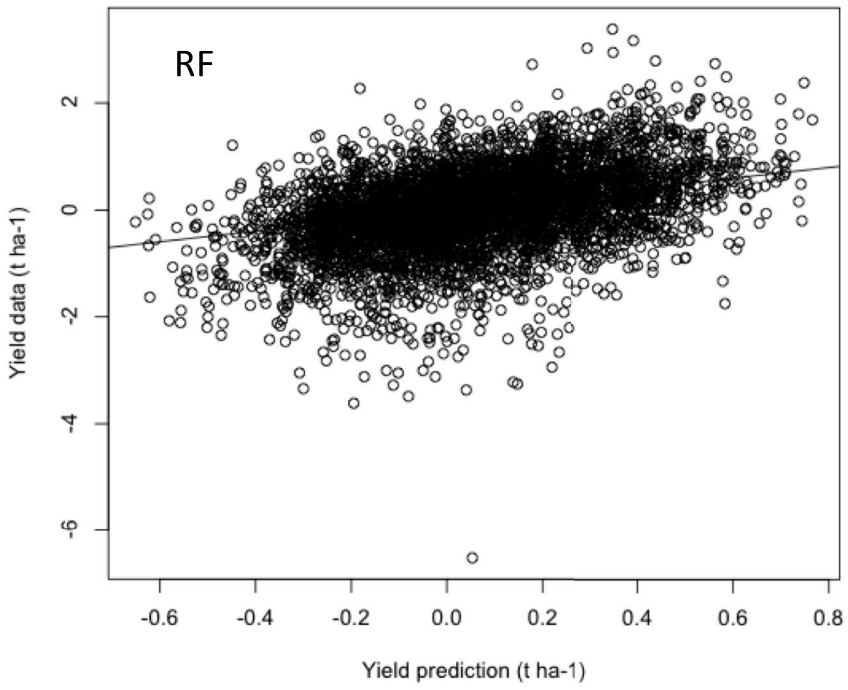


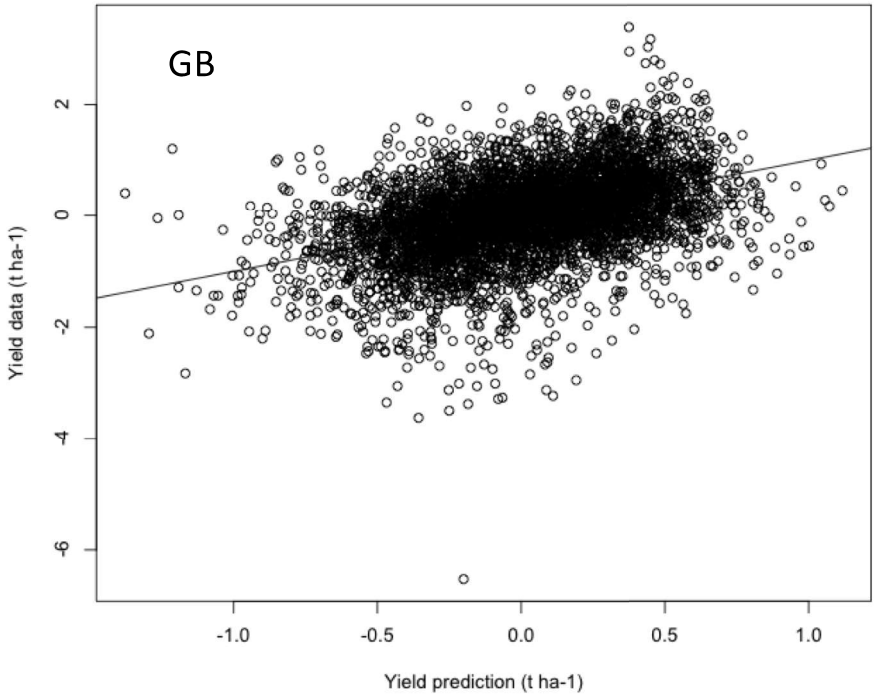




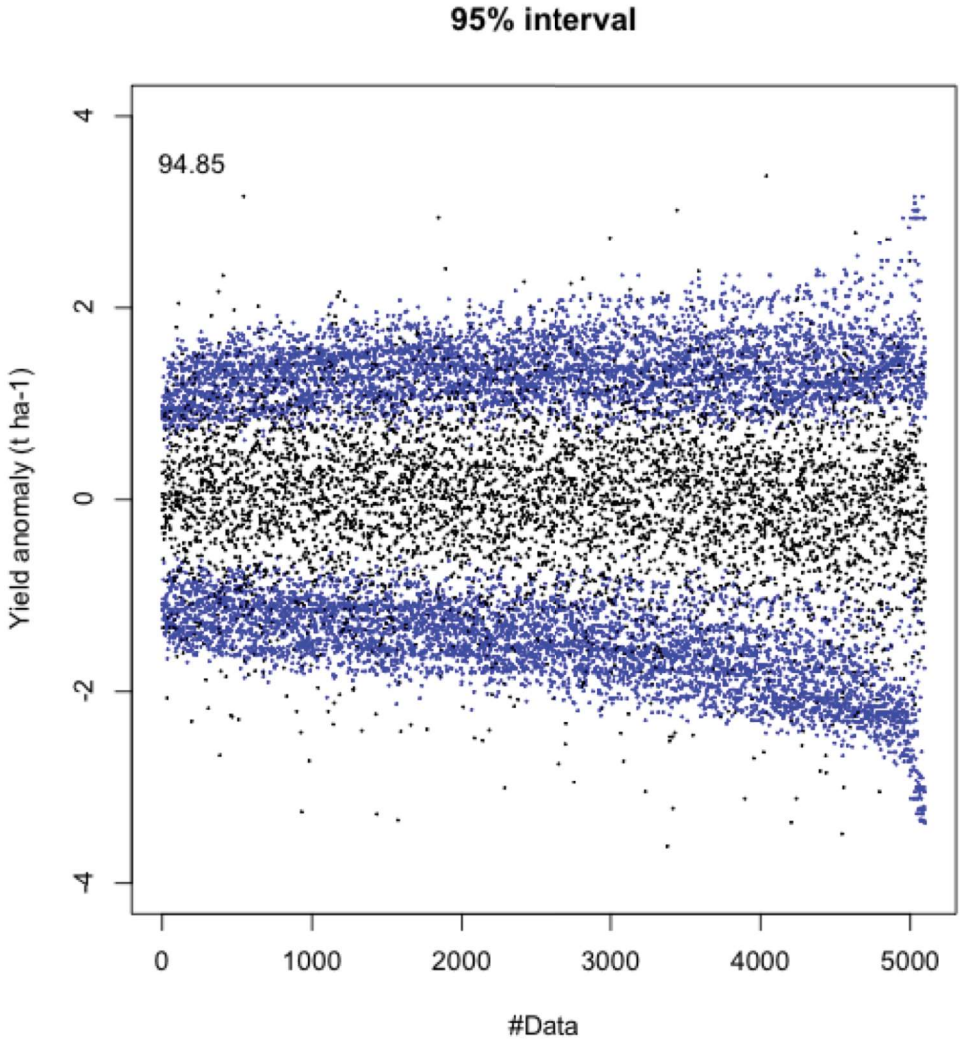


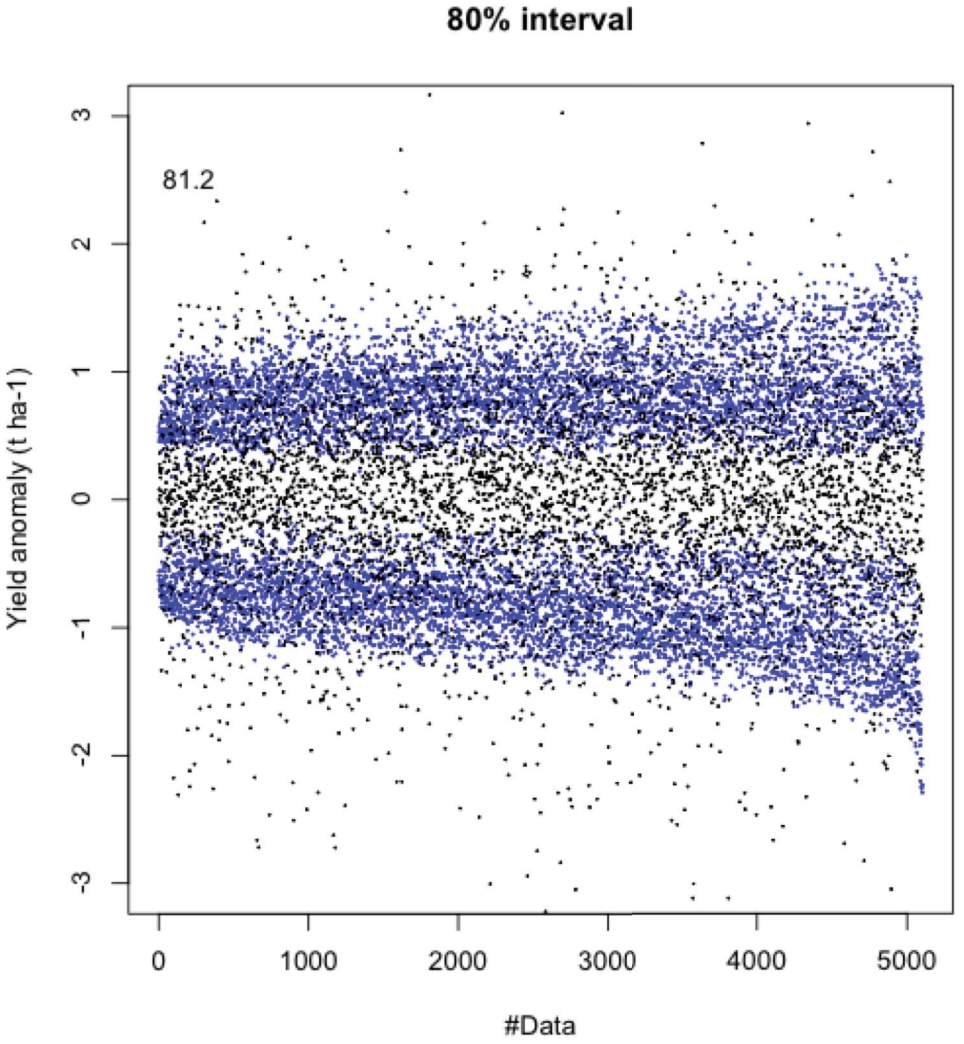
Méthode	RMSE (CV année par année)
Random Forest	0.71
Gradient boosting	0.70





Intervalles de prédiction (quantile random forest)





Zoom 2: Réseau de neurones multi-couches

Application : reconnaissance de chiffres

- 60000 images de chiffres disponibles pour l'entrainement
- Entrainement d'un algorithme prédictif
- 10000 images de chiffres disponibles pour le test de l'algorithme entrainé

Lecture des images

```
#Images
x_train <- mnist$train$x
#Etiquettes
y_train <- mnist$train$y
```

Entrainement

```
#Images
x_test <- mnist$test$x
x_test_image <- mnist$test$x
#Etiquettes
y_test <- mnist$test$y
```

Test

Une image de x_train et son étiquette y_train

```
#Exemple chiffre
plot(as.raster(x_train[26,,]), max=255)
y_train[26]
```

```
> y_train[26]
[1] 2
```

Une autre image de x_train et son étiquette y_train

```
#Exemple chiffre
plot(as.raster(x_train[62,,]), max=255)
image(1:28,1:28,NUM)
y_train[62]
```

```
> y_train[62]
[1] 4
```

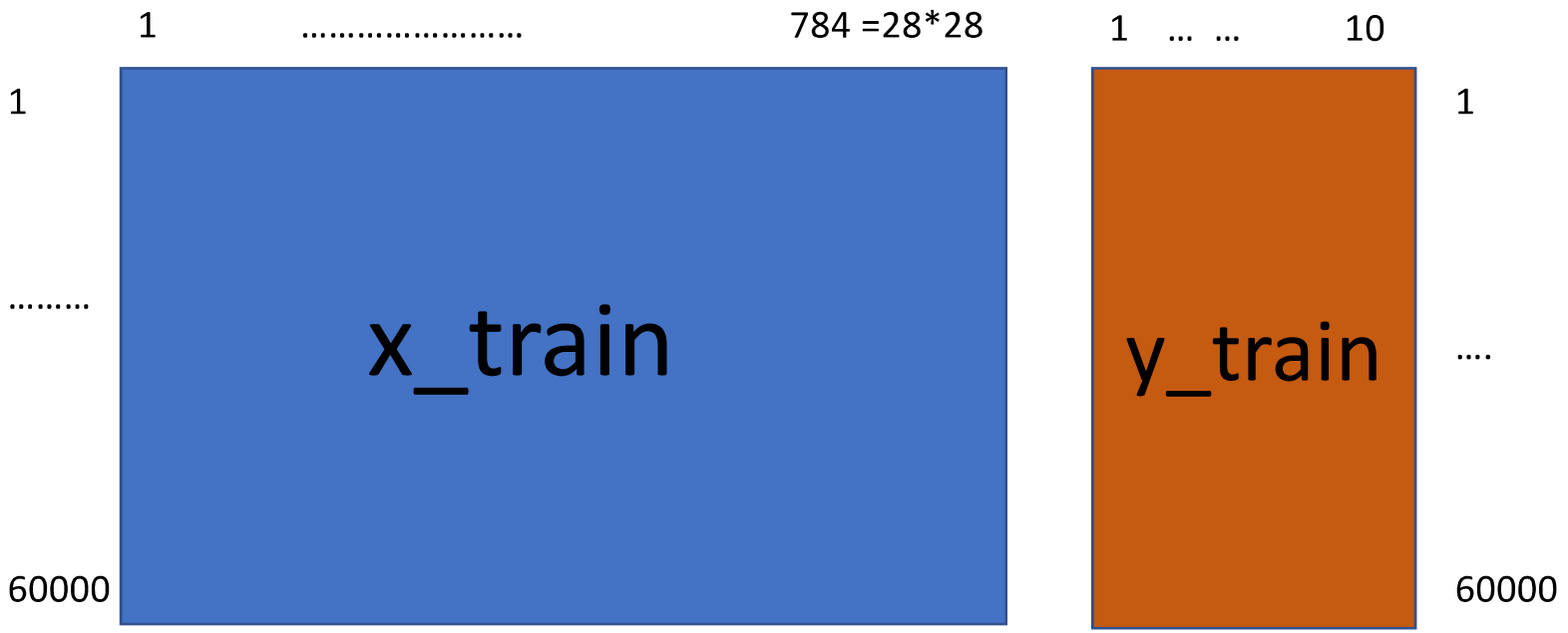
```
> dim(x_train)
[1] 60000    28    28
> x_train[1,,]
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]
 [1,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
 [2,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
 [3,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
 [4,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
 [5,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
 [6,]    0    0    0    0    0    0    0    0    0     0     0     0     3    18    18    18   126
 [7,]    0    0    0    0    0    0    0    0   30    36    94   154   170   253   253   253   253
 [8,]    0    0    0    0    0    0    0   49  238   253   253   253   253   253   253   253   253
 [9,]    0    0    0    0    0    0    0   18  219   253   253   253   253   253   198   182   247
[10,]    0    0    0    0    0    0    0    0   80   156   107   253   253   205    11     0    43
[11,]    0    0    0    0    0    0    0    0    0    14     1   154   253    90     0     0     0
[12,]    0    0    0    0    0    0    0    0    0     0     0   139   253   190     2     0     0
[13,]    0    0    0    0    0    0    0    0    0     0     0    11   190   253    70     0     0
[14,]    0    0    0    0    0    0    0    0    0     0     0     0    35   241   225   160   108
[15,]    0    0    0    0    0    0    0    0    0     0     0     0     0    81   240   253   253
[16,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0    45   186   253
[17,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0    16    93
```

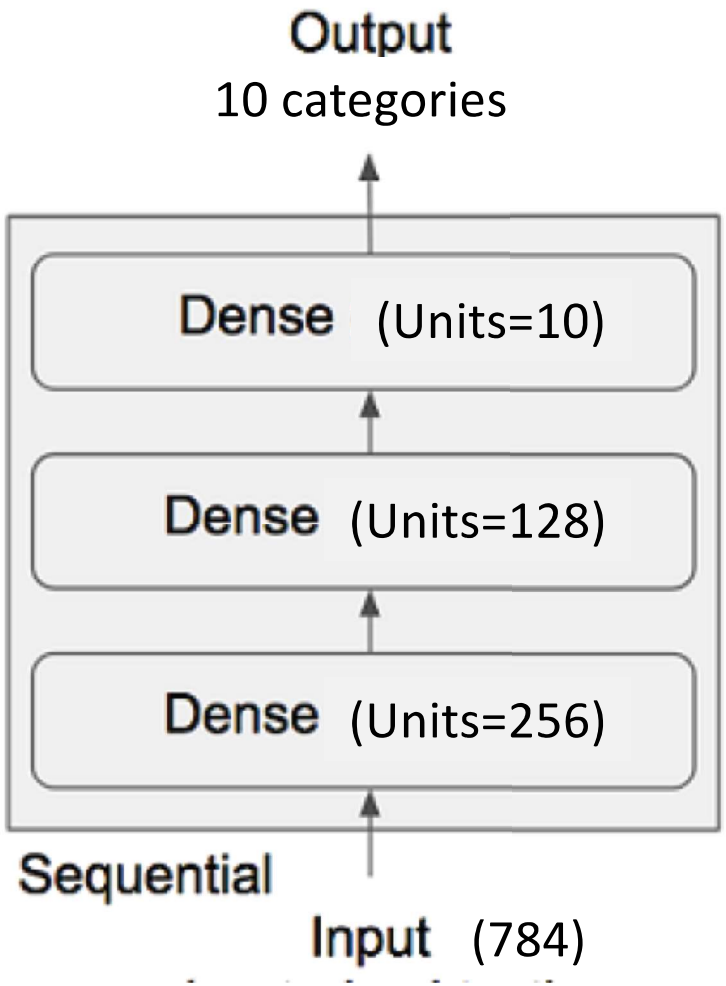
```
# reshape
x_train <- array_reshape(x_train, c(nrow(x_train), 784))
x_test <- array_reshape(x_test, c(nrow(x_test), 784))
# rescale
x_train <- x_train / 255
x_test <- x_test / 255

y_train <- to_categorical(y_train, 10)
y_test <- to_categorical(y_test, 10)
```

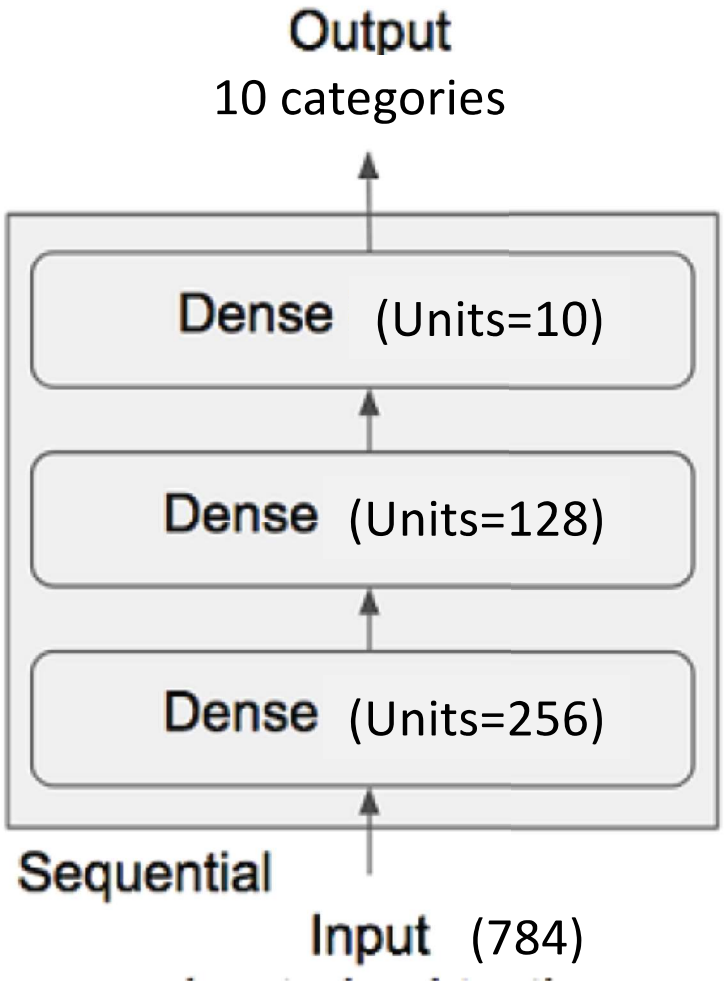
```
> dim(x_train)
[1] 60000   784
```

```
       [,129]    [,130]    [,131]    [,132] [,133] [,134] [,135] [,136] [,137] [,138] [,139]
[1,] 0.0000000 0.0000000 0.0000000 0.0000000      0      0      0      0      0      0      0
[2,] 0.6235294 0.9921569 0.6235294 0.1960784      0      0      0      0      0      0      0
[3,] 0.0000000 0.0000000 0.0000000 0.0000000      0      0      0      0      0      0      0
[4,] 0.0000000 0.0000000 0.0000000 0.0000000      0      0      0      0      0      0      0
[5,] 0.0000000 0.0000000 0.0000000 0.0000000      0      0      0      0      0      0      0
[6,] 0.0000000 0.0000000 0.0000000 0.0000000      0      0      0      0      0      0      0
```

```
> dim(y_train)
[1] 60000    10
> head(y_train)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    0    0    0    0    0    1    0    0    0     0
[2,]    1    0    0    0    0    0    0    0    0     0
[3,]    0    0    0    0    1    0    0    0    0     0
[4,]    0    1    0    0    0    0    0    0    0     0
[5,]    0    0    0    0    0    0    0    0    0     1
[6,]    0    0    1    0    0    0    0    0    0     0
```


1
.....
784 =28*28
1
60000
x_train
1 10
1
....
60000
y_train


Output
10 categories
Dense (Units=10)
Dense (Units=128)
Dense (Units=256)
Sequential
Input (784)

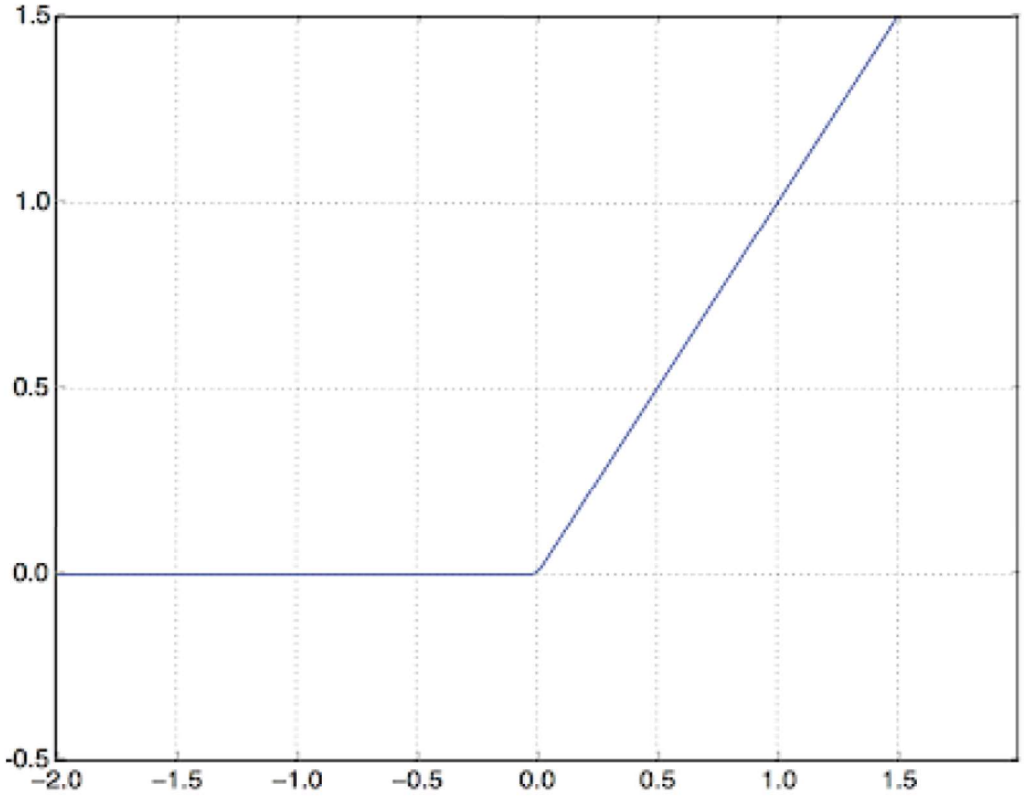


$$f\left(b3 + \sum_{k=1}^{10} w_{k,3} * X_{k,3}\right) \quad *10$$

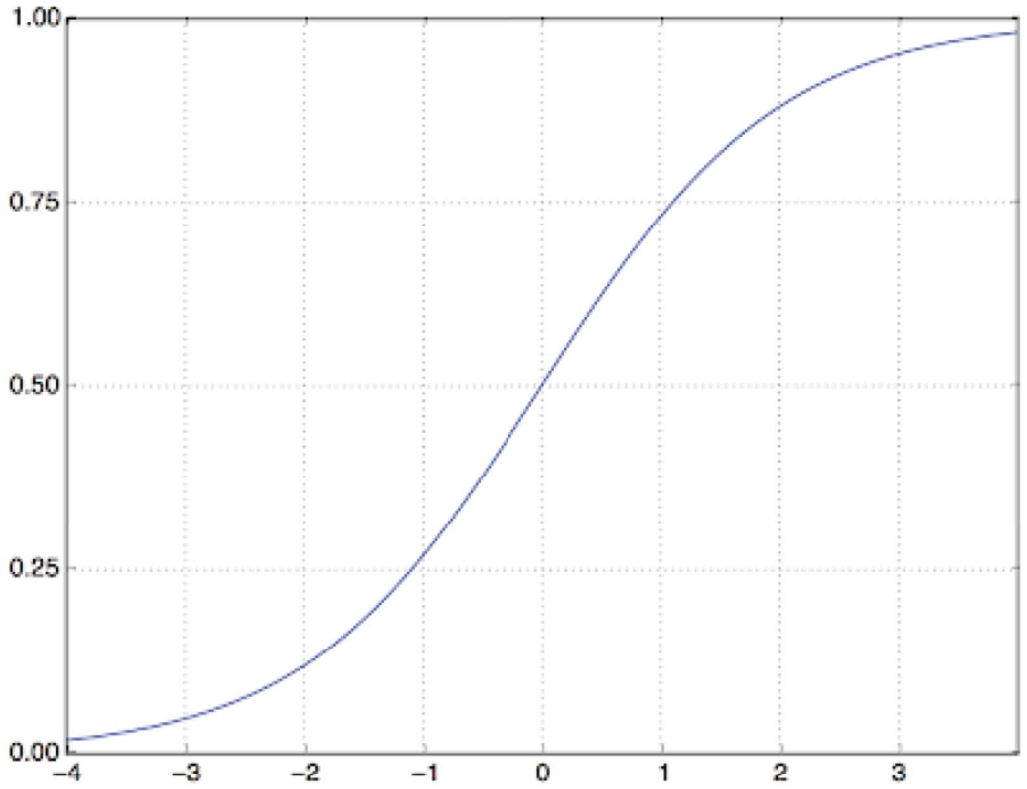
$$f\left(b2 + \sum_{j=1}^{256} w_{j,2} * X_{j,2}\right) \quad *128$$

$$f\left(b1 + \sum_{i=1}^{784} w_{i,1} * X_{i,1}\right) \quad *256$$

f=RELU



f=softmax

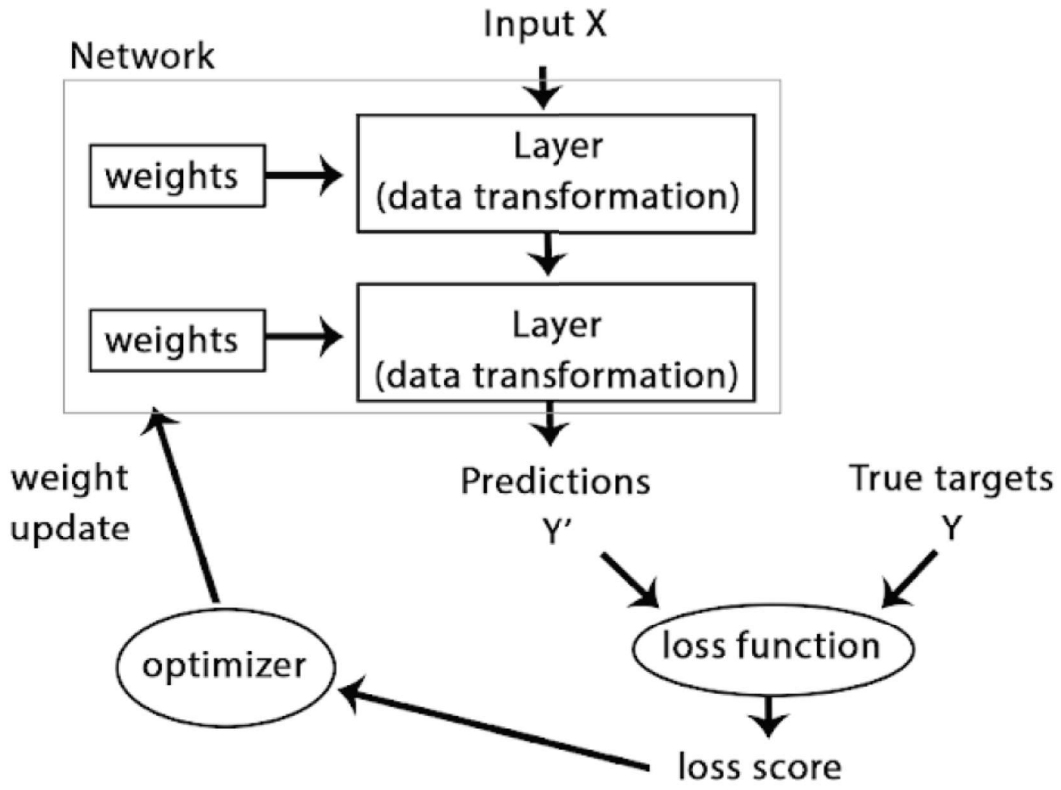


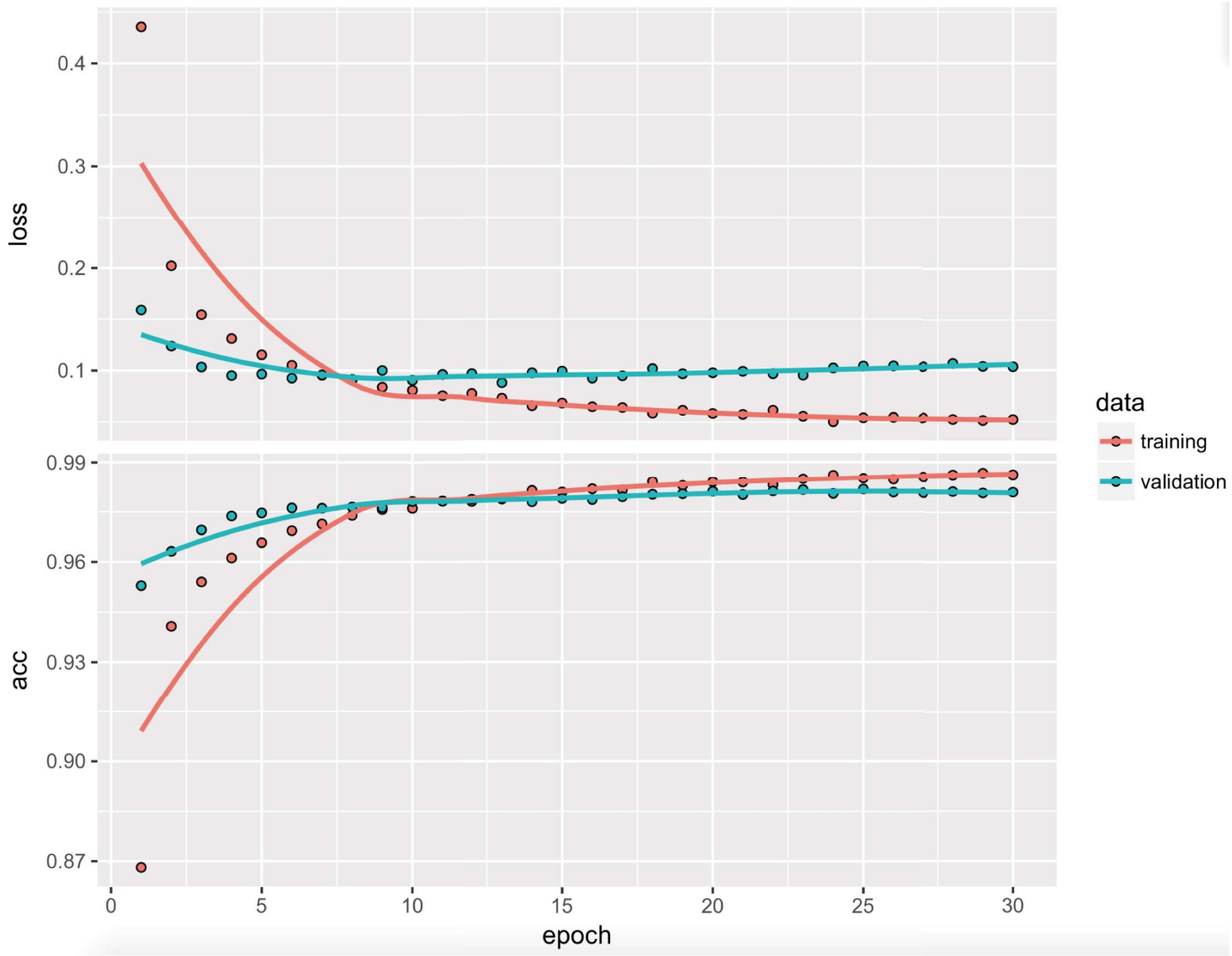
Réseau de neurones multi-couches

```
model <- keras_model_sequential()
model %>%

  layer_dense(units = 256, activation = 'relu',
       input_shape = c(784)) %>%

  layer_dropout(rate = 0.4) %>%

  layer_dense(units = 128, activation = 'relu') %>%

  layer_dropout(rate = 0.3) %>%

  layer_dense(units = 10, activation = 'softmax')

summary(model)
```

```
> summary(model)
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 256)               200960
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
=================================================================
Total params: 235,146
Trainable params: 235,146
Non-trainable params: 0
```

```
model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)


history <- model %>% fit(
  x_train, y_train,
  epochs = 30, batch_size = 128,
  validation_split = 0.2
)

plot(history)
```

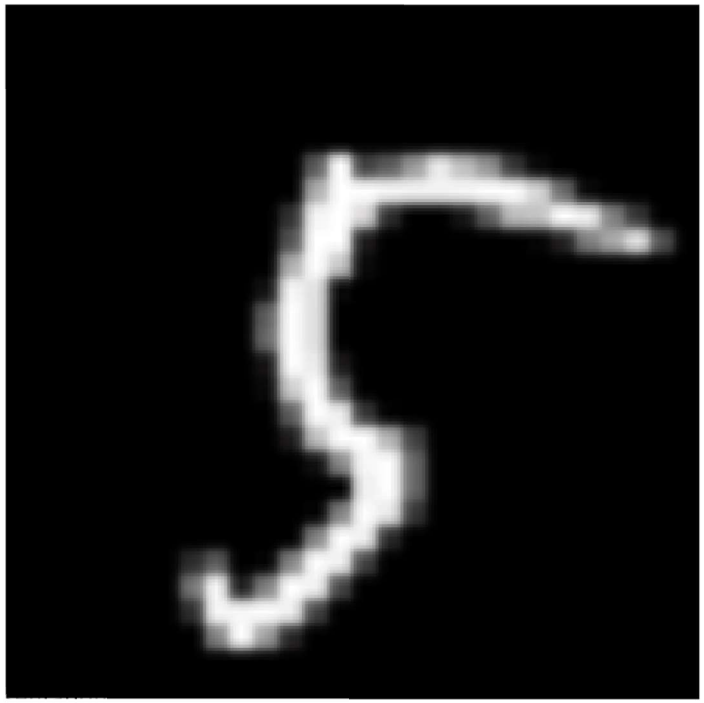

Input X
Network
weights
Layer
(data transformation)
weights
Layer
(data transformation)
weight
update
Predictions
Y'
True targets
Y
optimizer
loss function
loss score


loss
0.4
0.3
0.2
0.1
acc
0.99
0.96
0.93
0.90
0.87
0
5
10
15
20
25
30
epoch
data
training
validation

```
##Evaluation
results<- model%>%evaluate(x_test,y_test)
results
```

```
> results
$loss
[1] 0.1017851

$acc
[1] 0.981
```

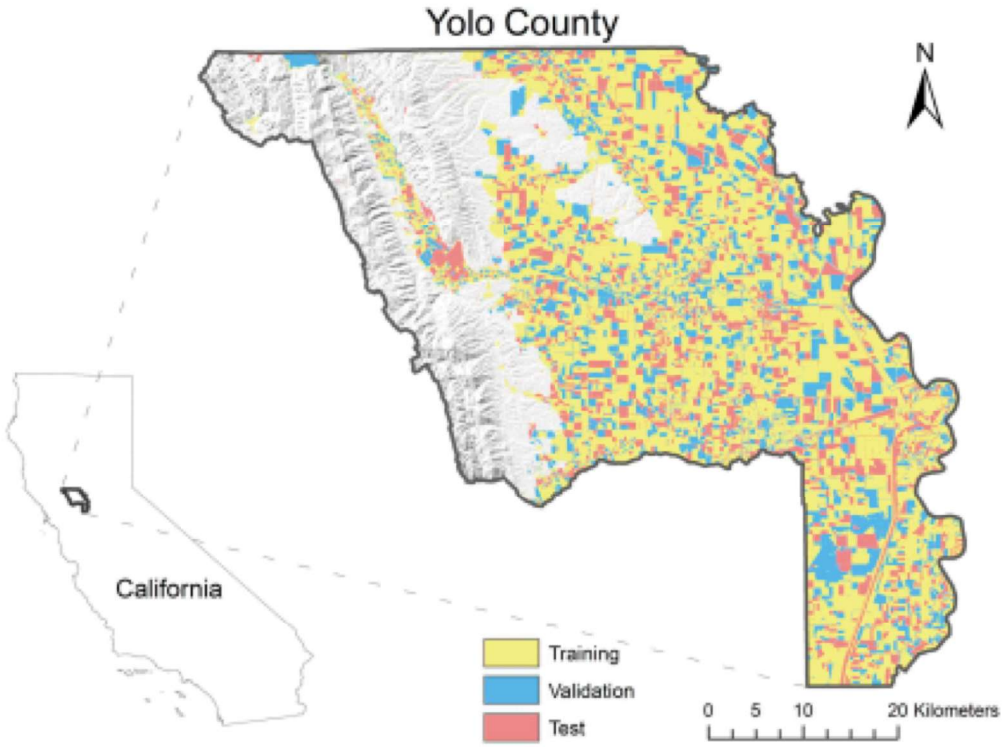
```
> Prediction<-model %>% predict_classes(x_test)
> #Exemple
> k<-780
> plot(as.raster(x_test_image[k,,], max=255))
> y_test[k,]
 [1] 0 0 0 0 0 1 0 0 0 0
> Prediction[k]
[1] 5
```



Des applications de plus en plus nombreuses en agriculture

Deep learning based multi-temporal crop classification☆

Liheng Zhong[a,*], Lina Hu[b], Hang Zhou[c]



Classification from Landsat surface reflectance

Comparison of several techniques:

- Multi-layer NN
- SVM
- Random forest
- Gradient boosting
- Deep learning

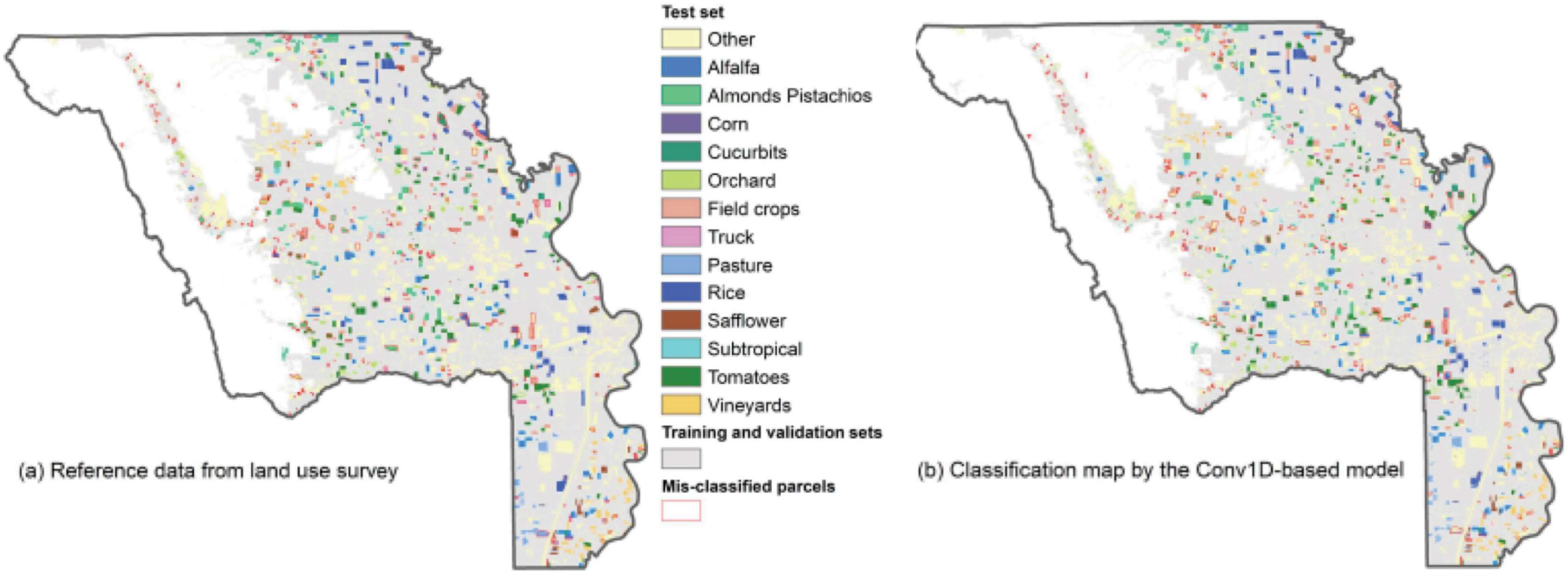
Datasets

- 1 221 396 pixels in the training dataset
- 431 920 pixels in the validation dataset (for the optimization of the hyper parameters
- 415 251 pixels in the test dataset

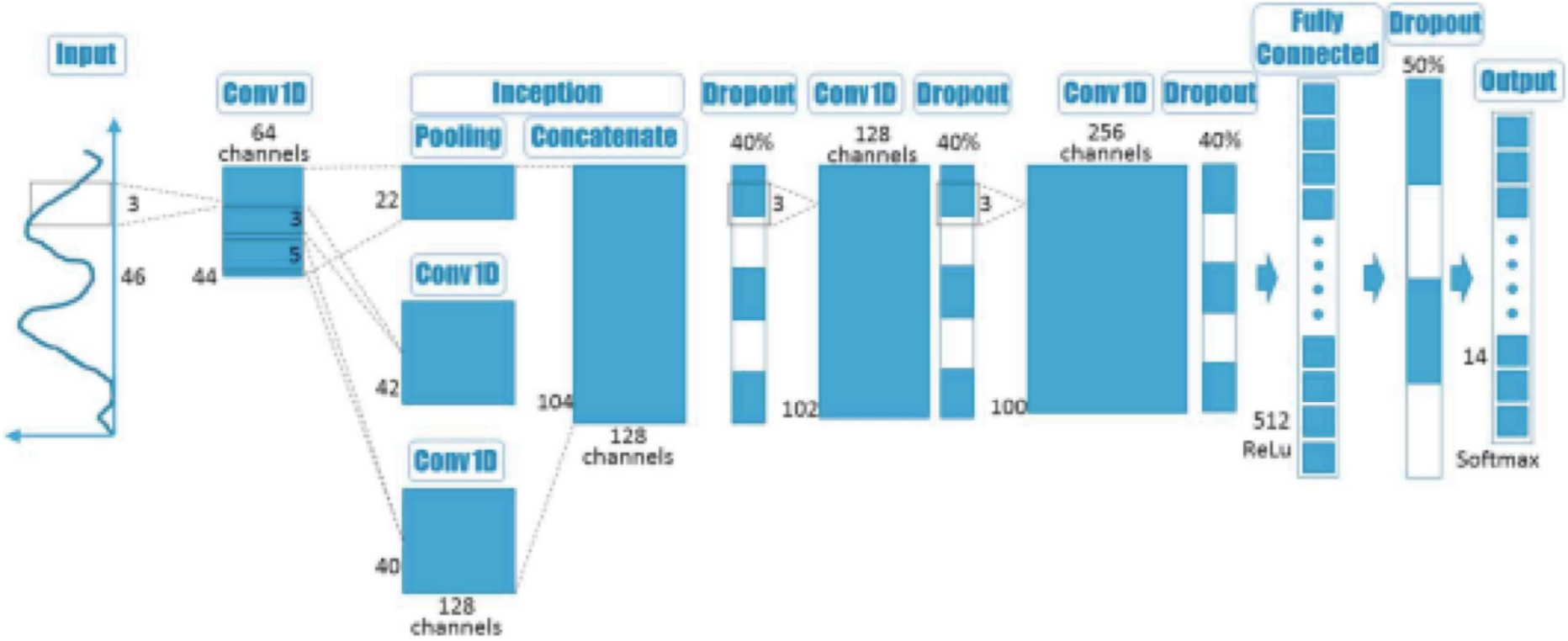
- Inputs: Time series of Enhanced Vegetation Index of length 46 in each pixel (computed from Landsat surface reflectance)
- Output: 14 categories of crops

Deep learning based multi-temporal crop classification☆

Liheng Zhong[a,*], Lina Hu[b], Hang Zhou[c]



Classification from Landsat surface reflectance


Input
Conv1D
64
channels
3
46
44
3
5
Inception
Pooling
Concatenate
22
Conv1D
42
Conv1D
40
128
channels
104
128
channels
Dropout
40%
3
Conv1D
128
channels
102
Dropout
40%
3
Conv1D
256
channels
100
Dropout
40%
Fully
Connected
512
ReLu
Dropout
50%
Output
14
Softmax

Classifier type	Input	Overall accuracy
MLP	Series only	83.81%
Conv1D-based	Series only	85.54%
LSTM-based	Series only	82.41%
XGBoost	Series only	84.12%
	Series + HANTS	84.17%
	Series + TIMESAT	84.09%
RF	Series only	83.38%
	Series + HANTS	83.25%
	Series + TIMESAT	83.26%
SVM	Series only	82.45%
	Series + HANTS	83.09%
	Series + TIMESAT	82.95%

Une démarche collective … en petits groupes

Question → Données → Entrainement → Test

Porteurs d'enjeux

Experts en analyse de données

Scientifiques biotechniques

Responsables d'expérimentation

Gestionnaires de bases de données

Des défis

- Définir des questions prioritaires
- Capitaliser rapidement des données
- Développer des groupes projets sur des périodes courtes
- Veille méthodologique pour bénéficier des innovations