Statistique et Science des Grosses Données

Philippe Besse

Université de Toulouse — INSA
Institut de Mathématiques — UMR CNRS 5219

Définition

- *Analyste, ça fait trop Wall Street ; statisticien, ça agace les économistes ; chercheur scientifique, ça fait trop académique. Pourquoi pas "data scientist" ?* (D. Patil *LinkedIn* et J. Hammerbacher, *Facebook*, 2008)
- Data scientist (n) : *Person who is better at statistics than any software engineer and better at software than any statistician* (J. Wills, *Cloudera*)

Un peu d'histoire

1930-70 h-Octets Statistique inférentielle

1970s kO Analyse des données et *exploratory data analysis*

1980s MO IA, Réseaux de neurones, Statistique fonctionnelle

1990s GO Data mining

2000s TO Bioinformatique et Apprentissage Statistique

2010s PO *Grosses Data*

Science des données et apprentissage

- Facteurs de risque épidémiologiques
- Facteur génétique ou biomarqueurs
- Reconnaissance de forme (caractères)
- Adaptation statistique en prévision météo (pic d'ozone)
- Score d'appétence ou d'attrition en GRC
- Méta modèle ou réduction de modèle physique
- Détection défaillance ou fraude (atypique)
- ...
- Estimer un modèle apprendre un algorihtme
- Minimiser une erreur de prévision ou risque

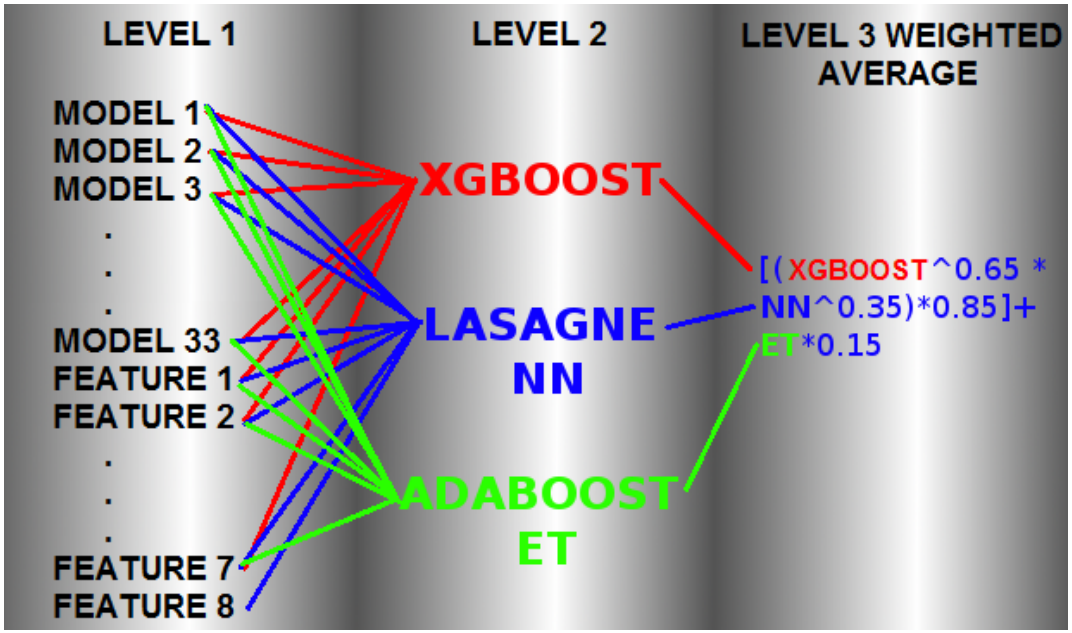
Quel Objectif ?

- Explorer ou vérifier, représenter, décrire
- Expliquer ou tester une influence
- Prévoir et sélectionner, interpréter
- Prévision "brute"

Statistique *vs.* Apprentissage Statistique *vs.* Machine

Quel But ?

- Publication académique (*Benchmarks — UCI repository*)
- Solution industrielle peu *glamour*
- Concours de type Kaggle



Concours Kaggle : Identify people who have a high degree of Psychopathy based on Twitter usage.

Les données

- p variables (*features*) explicatives ou prédictives $X = (X^1, \ldots, X^p)$
- n observations, individus, unités statistiques, *instances*
- Ensemble d'apprentissage $D_1^n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
- $\mathbf{x}_i \in \mathcal{X}$ $(= \mathbb{R}^p)$, $y_i \in \mathcal{Y}$ pour $i = 1 \ldots n$
- Choix d'un ensemble de modèles, méthodes, algorithmes

$$Y = f(\boldsymbol{X}), \quad f \in \mathcal{F}$$

Méthodes et algorithmes d'apprentissage

- Modèle linéaire avec sélection ou régularisation
- Régression PLS avec pénalisation
- Modèle linéaire binomial (logistique) avec sélection ou régularisation
- Analyse discriminante, k plus proches voisins
- Arbres binaires de décision (CART)
- Résaux de neurones, perceptron, apprentissage profond
- Agrégation de modèles : *random forest, boosting...*
- SVM ou séparateurs à vaste marge
- Imputation de données manquantes
- À venir : Détection d'anomalies ou d'atypiques
- ...

Quelles applications industrielles ?

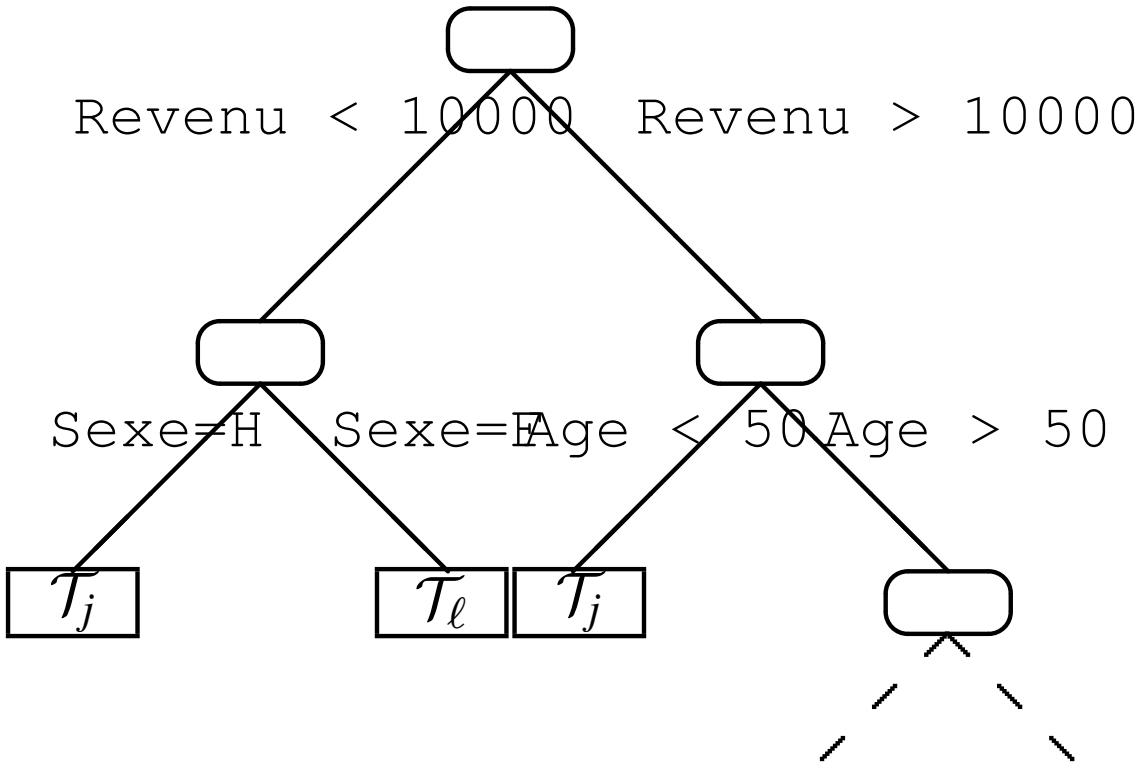
- Analyse d'image : réseaux de neurones convolutifs
- Traitement du signal : réseaux récurrents
- Méta modèles ou modèles réduits
  - Thèse de Matthias de Lozzo (ONERA, 2013) Patricia Klotz et Béatrice Laurent
  - Thèse d'Amandine Marrel (CEA, 2008) Bertrand Iooss et Béatrice Laurent
- Détection d'anomalie
  - *One Class Classification*, *Novelty detection*
  - SVM et beaucoup d'autres...
  - Plusieurs thèses en route : Airbus, Continental...

Autres méthodes plébiscitées

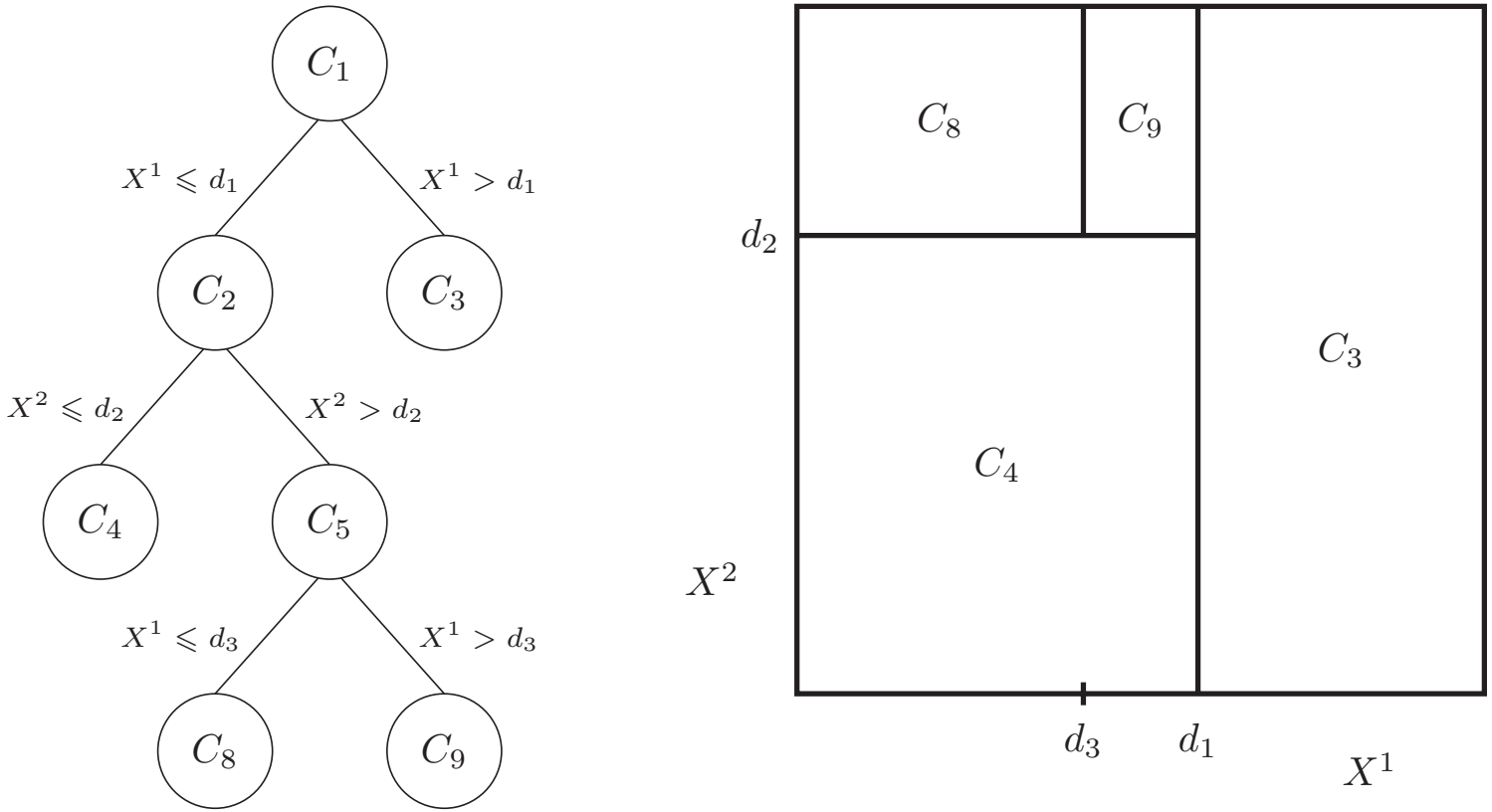
- *Random Forest* (Breiman 2001)
  Comparaisons systématiques
  (Fernandez-Delgado et al. 2014)
- *Extrem Gradient Boosting* (Chen et Guestrin, 2016)
  Concours *Kaggle*

Echelle grosses data

- Volume : *Hadoop, Spark, cloud computing*
- Variété : images, signaux, trajectoires, graphes...
- Vélocité : décision séquentielle, bandits, gradient stochastique



Exemple fictif d'arbre binaire de classification



Construction d'un arbre et pavage dyadique de l'espace

Y quantitative : hétérogénéité en régression

Hétérogénéité du nœud κ :

$$D_\kappa = \frac{1}{|\kappa|} \sum_{i \in \kappa} (y_i - \bar{y}_\kappa)^2$$

où $|\kappa|$ est l'effectif du nœud κ

Minimiser la variance intra-classe

Les nœud fils κ_G et κ_D minimisent :

$$\frac{|\kappa_G|}{n} \sum_{i \in \kappa_G} (y_i - \bar{y}_{\kappa_G})^2 + \frac{|\kappa_D|}{n} \sum_{i \in \kappa_D} (y_i - \bar{y}_{\kappa_D})^2$$

Hétérogénéité et *déviance* dans le cas gaussien (Breiman et al. 1984)

Y qualitative : hétérogénéité en discrimination

Hétérogénéité du nœud κ :

- Entropie avec la notation $0\log(0) = 0$

$$D_\kappa = -2\sum_{\ell=1}^{m} |\kappa| p_\kappa^\ell \log(p_\kappa^\ell)$$

  p_κ^ℓ : proportion de la classe $\mathcal{T}_\ell$ de Y dans κ.
- Concentration de Gini : $D_\kappa = \sum_{\ell=1}^{m} p_\kappa^\ell (1 - p_\kappa^\ell)$
- Statistique du test du χ^2 (CHAID)

Entropie et déviance d'un modèle multinomial
(Breiman et al. 1984)

Forêts aléatoires : algorithme

Soit $\mathbf{x}_0$ à prévoir et $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ un échantillon

for $b = 1$ à B **do**

Tirer un échantillon bootstrap $\mathbf{z}_b^*$

Estimer un arbre avec randomisation des variables :

Pour chaque nœud, tirage aléatoire de m prédicteurs

end for

Calculer l'estimation moyenne $\widehat{f}_B(\mathbf{x}_0) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)$

ou le vote

Forêts aléatoires : utilisation

- Élagage : Arbres de taille q, ou complet.
- La sélection aléatoire des m prédicteurs ($m = \sqrt{p}$ en classification, $\frac{p}{3}$ en régression) accroît la variabilité
- Chaque modèle de base est moins performant mais l'agrégation est performante
- Évaluation itérative de l'erreur out-of-bag

Aide à l'interprétation

- Indices d'importances
- Mean Decrease Accuracy
- Mean Decrease Gini
- Var used

Boosting : principe

- Améliorer les compétences d'un faible classifieur (Freund et Schapire, 1996)
- AdaBoost (Adaptative boosting) prévision d'une variable binaire
- Réduire la variance mais aussi le biais de prévision
- Agrégation d'une famille de modèles récurents
  Chaque modèle est une version adaptative du précédent en donnant plus de poids, lors de l'estimation suivante, aux observations mal ajustées
- Variantes : type de la variable à prédire (binaire, k classes, réelles), fonction perte (robustesse)

AdaBoost discret

Fonction δ de discrimination $\{-1, 1\}$
Soit $\mathbf{x}_0$ à prévoir et
$\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ un échantillon
Initialiser les poids $\boldsymbol{w} = \{w_i = 1/n \; ; \; i = 1, \ldots, n\}$
for do$m = 1$ à M

Estimer δ_m sur l'échantillon pondéré par $\boldsymbol{w}$

Calculer le taux d'erreur apparent : $\widehat{\mathcal{E}}_p = \frac{\sum_{i=1}^n w_i \mathbf{1}\{\delta_m(\boldsymbol{x}_i) \neq y_i\}}{\sum_{i=1}^n w_i}$

Calculer les logit : $c_m = \log((1 - \widehat{\mathcal{E}}_p)/\widehat{\mathcal{E}}_p)$

Nouvelles pondérations (normalisation) :

$w_i \leftarrow w_i . \exp\left[c_m \mathbf{1}\{\delta_m(\boldsymbol{x}_i) \neq y_i\}\right] ; i = 1, \ldots, n$

end for

Résultat du vote : $\widehat{f}_M(\boldsymbol{x}_0) = \text{signe}\left[\sum_{m=1}^M c_m \delta_m(\boldsymbol{x}_0)\right]$

Boosting : interprétation

- Approximation de f par un modèle additif pas à pas (Hastie et col., 2001)

$$\widehat{f}(\boldsymbol{x}) = \sum_{m=1}^{M} c_m \delta(\boldsymbol{x}; \gamma_m)$$

- c_m est un paramètre
- δ le classifieur de base fonction de $\boldsymbol{x}$ et dépendant d'un paramètre γ_m
- Q une fonction perte

Modèle additif : optimisation

- $(c_m, \gamma_m) = \arg\min_{(c,\gamma)} \sum_{i=1}^n Q(y_i, \widehat{f}_{m-1}(\boldsymbol{x}_i) + c\delta(\boldsymbol{x}_i;\gamma))$
- $\widehat{f}_m(\boldsymbol{x}) = \widehat{f}_{m-1}(\boldsymbol{x}) + c_m\delta(\boldsymbol{x};\gamma_m)$ améliore l'ajustement précédent
- f binaire, $Q(y, f(\boldsymbol{x})) = \exp[-yf(\boldsymbol{x})]$

$$(c_m, \gamma_m) = \arg\min_{(c,\gamma)} \sum_{i=1}^n \exp\left[-y_i(\widehat{f}_{m-1}(\boldsymbol{x}_i) + c\delta(\boldsymbol{x}_i;\gamma))\right]$$

$$= \arg\min_{(c,\gamma)} \sum_{i=1}^n w_i^m \exp\left[-cy_i\delta(\boldsymbol{x}_i;\gamma)\right] \text{avec} \quad w_i = \exp[-y_i\widehat{f}_{m-1}(\boldsymbol{x}_i)]$$

- w_i^m : poids fonction de la qualité de l'ajustement précédent

Modèle additif : solution

- Deux étapes : classifieur optimal puis optimisation de c_m

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} \mathbf{1}\{y_i \neq \delta(\boldsymbol{x}_i;\gamma)\} \quad \text{et} \quad c_m = \frac{1}{2}\log\frac{1-\widehat{\mathcal{E}}_p}{\mathcal{E}_p}$$

  avec $\widehat{\mathcal{E}}_p$ erreur apparente de prévision

- les w_i sont mis à jour avec : $w_i^{(m)} = w_i^{(m-1)} \exp[-c_m]$
- Adaboost approche f pas à pas par un modèle additif en utilisant une fonction perte exponentielle
- D'autres fonctions perte (autres pour robustesse)
  - adaBoost $Q(y,f(\boldsymbol{x})) = \exp[-yf(\boldsymbol{x})]$
  - LogitBoost $Q(y,f(\boldsymbol{x})) = \log_2(1+\exp[-2yf(\boldsymbol{x})]$
  - L^2Boost $Q(y,f(\boldsymbol{x})) = (y-f(\boldsymbol{x}))^2/2$

Gradient Boosting Machine (Friedman 2002)

- Fonction perte convexe différentiable
- Principe

$$min_\gamma \sum_{i=1}^{n} \left[Q\left(y_i, f_{m-1}(x_i) - \gamma \frac{\partial Q(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)} \right) \right]$$

  - Construire une séquence de modèles de sorte qu'à chaque étape, chaque modèle ajouté à la combinaison, apparaisse comme un pas vers une meilleure solution
  - Ce pas est franchi dans la direction du gradient de la fonction perte approché par un arbre de régression
- Paramètres à optimiser
  - Profondeur des arbres
  - Nombre d'itérations
  - Restriction du pas de descente (*schrinkage*)

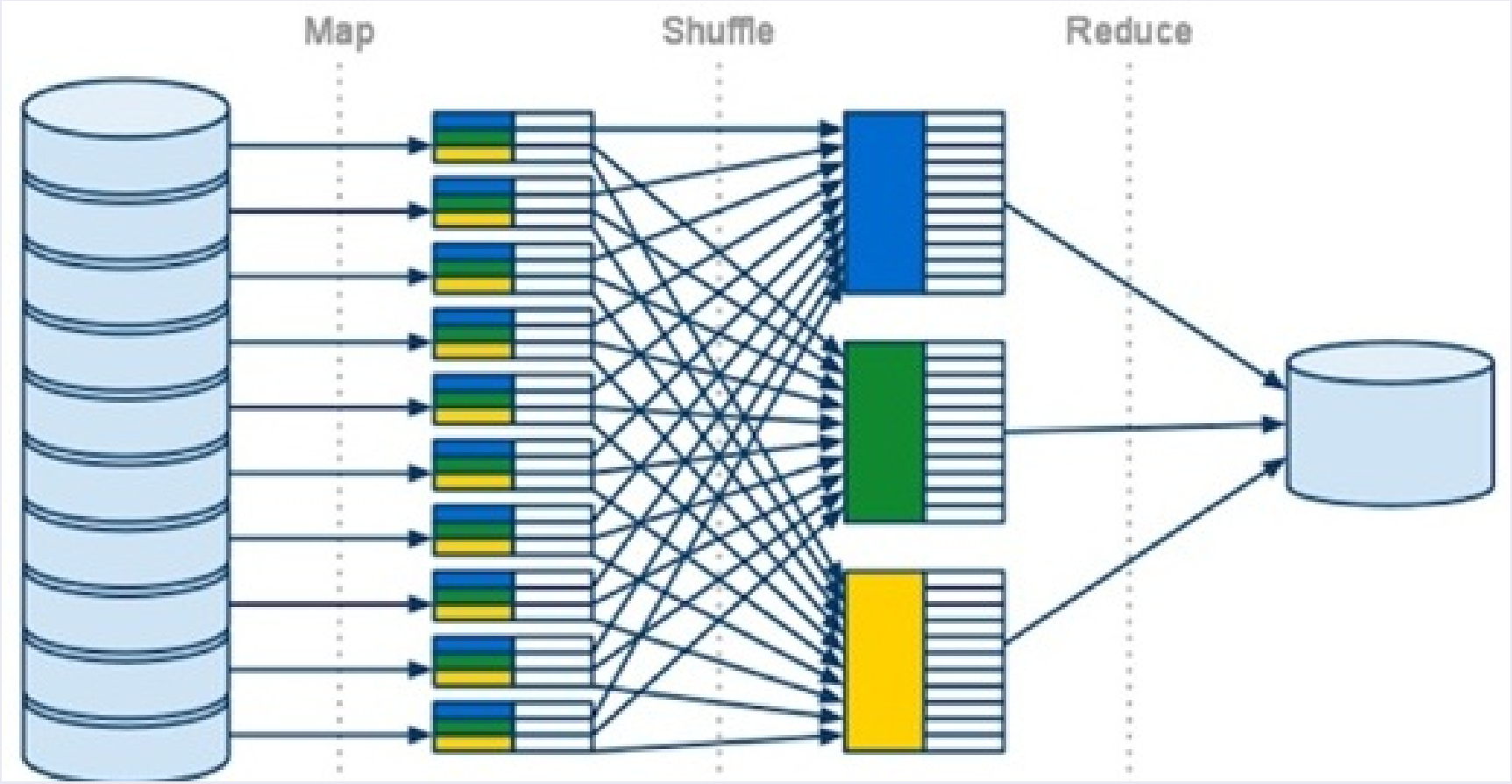
eXtrem Gradient Boosting (Chen et Gustrin 2016)

- Librairies `XGBoost`
  pour tout environnement (R, Pyhton, Scala, AWS...)
- Code optimisé pour parallélisation (GPU)
- Autres "astuces" et autres paramètres
  - Paramètres de pénalisation de la fonction perte

$$\mathcal{L}(f) = \sum_{i=1}^{n} Q(\widehat{[y)}_i, y_i) + \sum_{m=1}^{M} \Omega(\delta_m)$$

  avec $\Omega(\delta) = \alpha|\delta| + \frac{1}{2}\beta||\boldsymbol{w}||^2$
  - Développement de Taylor du gradient
  - Réduction du nombre de divisions possibles
  - Nombreuses astuces d'implémentation et de parallélisation

Pourquoi *Hadoop*?

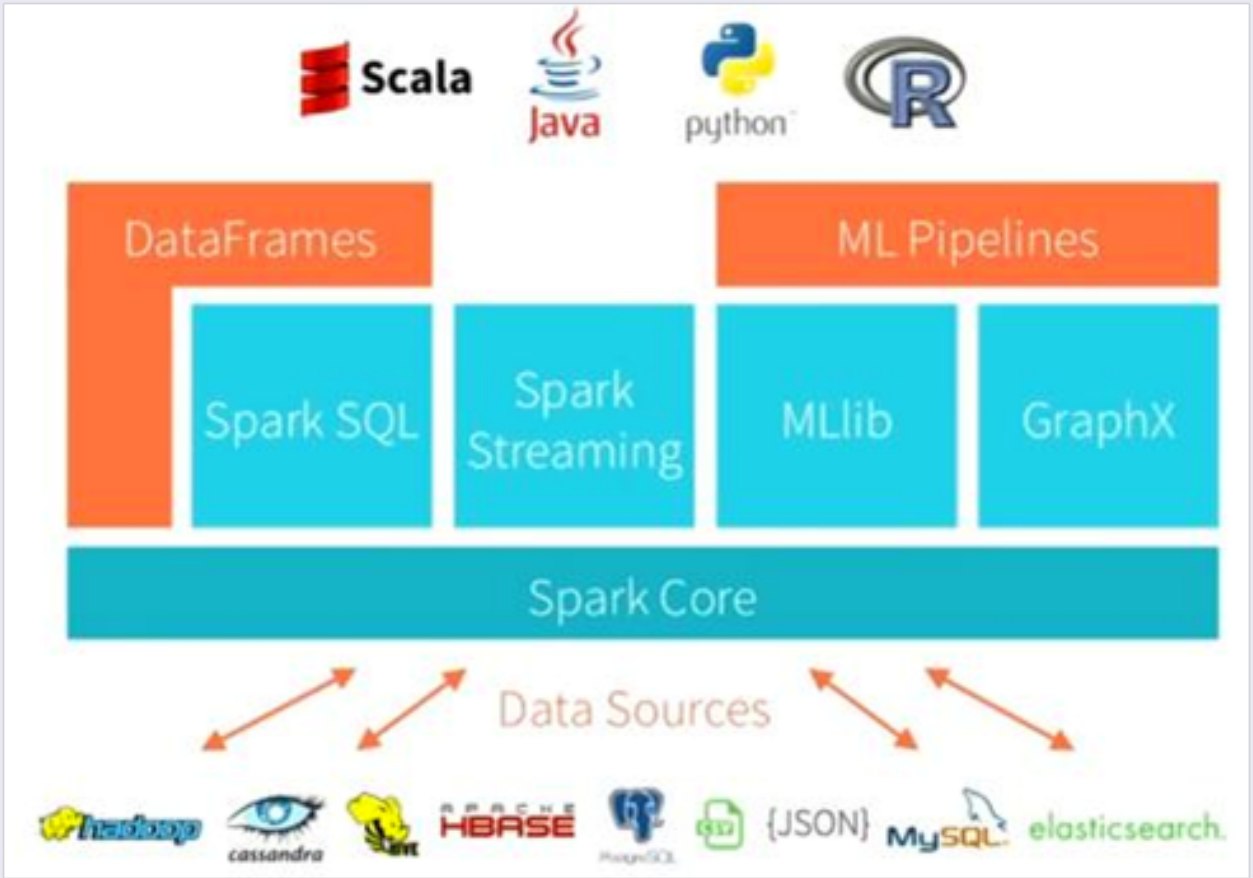


Hadoop Distributed File System (HDFS) & MapReduce

Classification par centres mobiles ($\approx$ *k-means*)

- Définition d'une distance euclidienne
- Algorithme de Forgy (1965)
  - Initialisation des k centres
  - Itération des étapes *MapReduce*
    - Map : Affectation de chaque individu `(valeur)` au centre `(clef)` le plus proche
    - Reduce : Calcul des centres des individus de même clef
    - Mise à jour des centres
- Problème : accès disques à chaque itération
- Solution actuelle de Spark : *Resilient Distributed Dataset*, (Zaharia *et al.*, 2012)

Pourquoi Spark ?



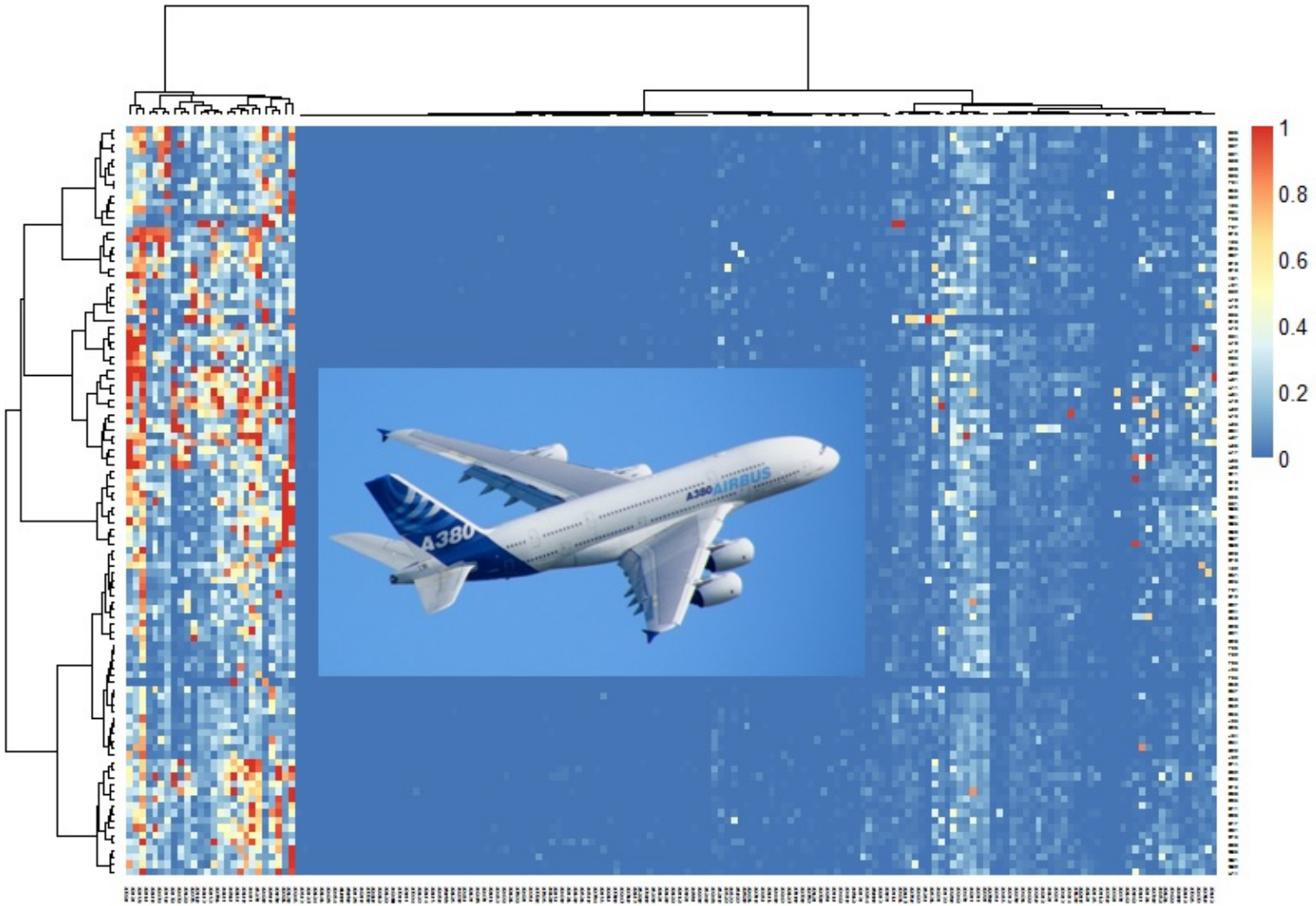
La technologie Spark et son écosystème

Librairie MLlib

- Spark 2.1
- MLlib (Resilient Distributed Dataset) : k-*means*, SVD, NMF (ALS), Régression linéaire et logistique (l_1 et l_2), perceptron, Classifieur Bayésien Naïf, Arbre, Forêt Aléatoire, *Gradient Boosting*
- Évolution de MLlib : *DataFrame*, *pipeline*
- Peu de méthodes mais passage à l'échelle "Volume"

Spark MLlib *vs.* Python Scikit-learn *vs.* R

- Thèse de Brendan Guillouet (2016), Besse et al. (2007)
- Trois cas d'usage
  - MNIST : Reconnaissance de caractères avec *random forest*
  - MovieLens : recommandation par complétion de matrice 22M d'évaluations de 138 000 utilisateurs $\times$ 27 000 films
  - Cdiscount : catégorisation de produits
- Comparaison des implémentations de
  - Random Forest
  - NMF Non negative Matrix Factorisation
  - Régression logistique
- Dans les environnements
  - Spark MLlib
  - Python Scikit-learn
  - R `randomForest ranger softimpute`



Airbus : Analyse des messages d'incidents en vol (700 000 en 6 mois)

Catégorisation de produits (Cdiscount)

- Préparation ou *data munging*
  - Nettoyages (ponctuation, erreurs de code, casse...)
  - Suppression des mots "vides" (*stopwords*)
  - Racinisation (*stemming*) : card(Dictionnaire)$= N$
- Vectorisation de (grosses ?) données
  - Hashage (*hashing trick*)) : $\texttt{n\_hash} < N$

$$i = h(j) \quad h : \{1, \ldots, N\} \longmapsto \{1, \ldots, \texttt{n\_hash}\}$$

  - Xgram : h appliquée à un mot ou couple (bigram) ou...
  - TF-IDF
- Apprentissage

TF-IDF

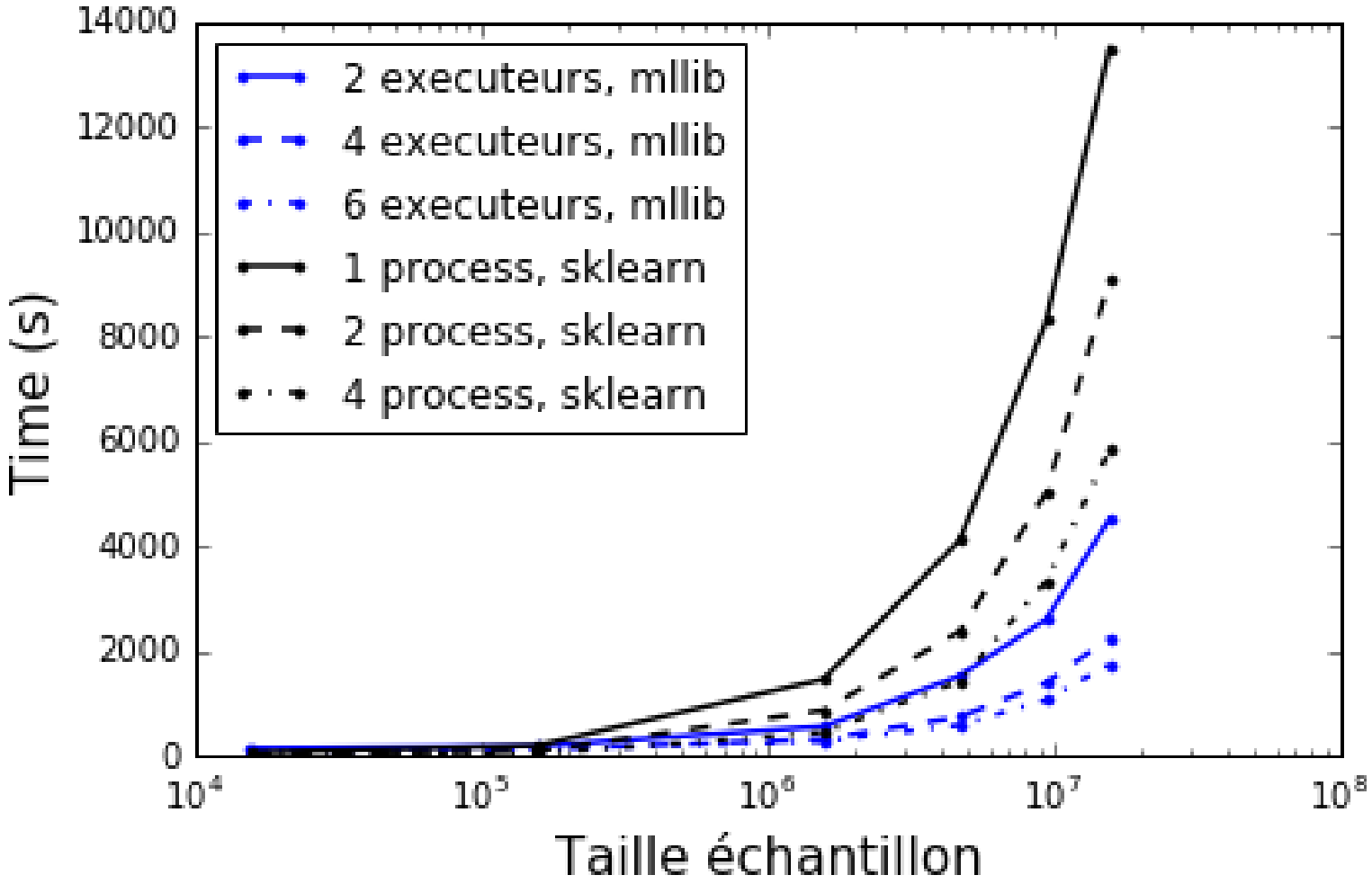
- Importance relative de chaque mot (ou xgram) dans un document par rapport à l'ensemble des documents.
- D : nombre de documents
- $TF(m,d)$: nombre d'occurrences du mot m dans le document d
- $f(m)$: nombre de documents contenant le mot m
- $IDF(m) = \log \frac{D+1}{f(m)+1}$ (version *smooth*)
- TF-IDF : nouvelles variables ou *features* par pondération des effectifs conjoints :

$$V_m(d) = TF(m,d) \times IDF(m)$$

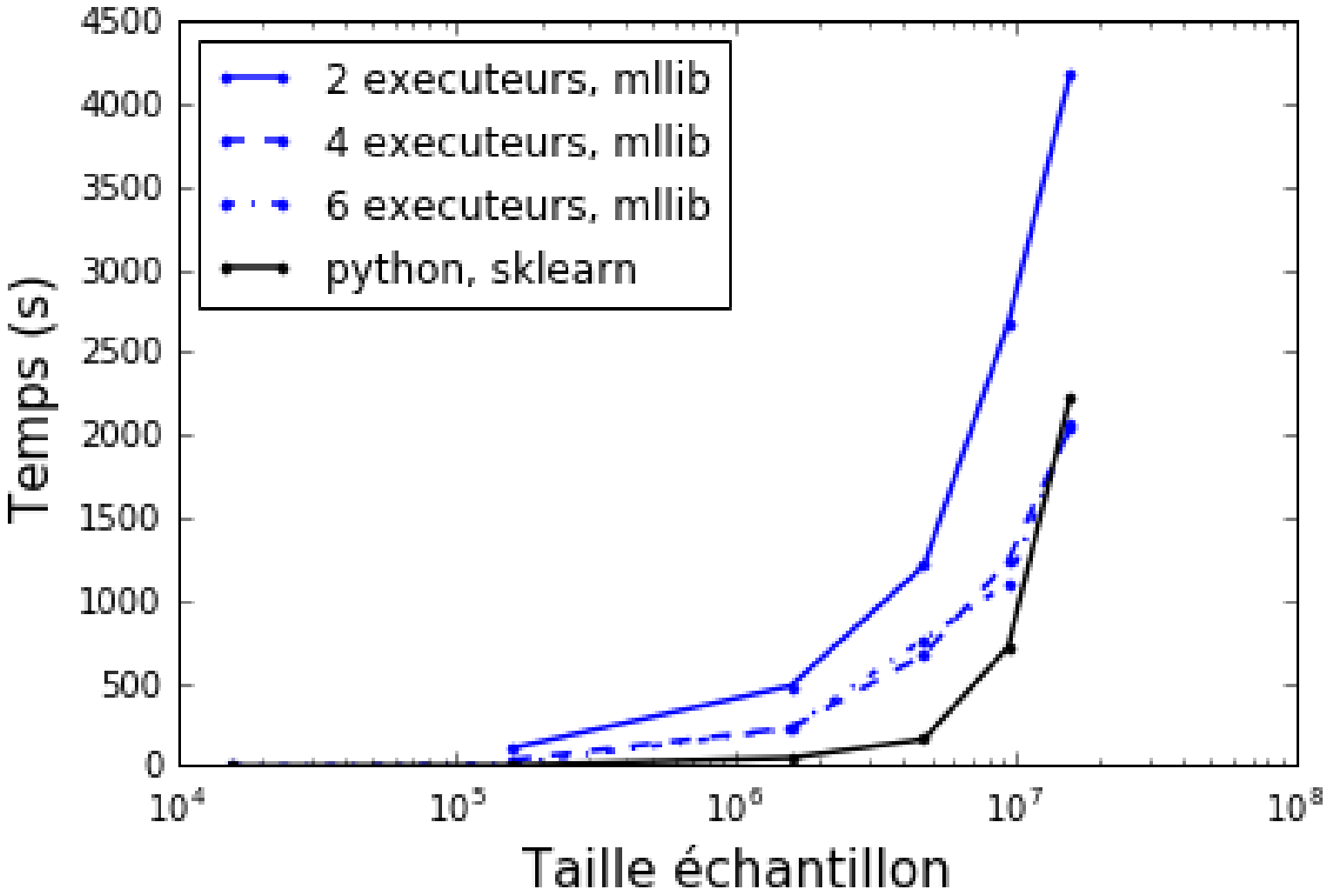
- Même vectorisation (hashage, TF-IDF) sur le test

Cdiscount

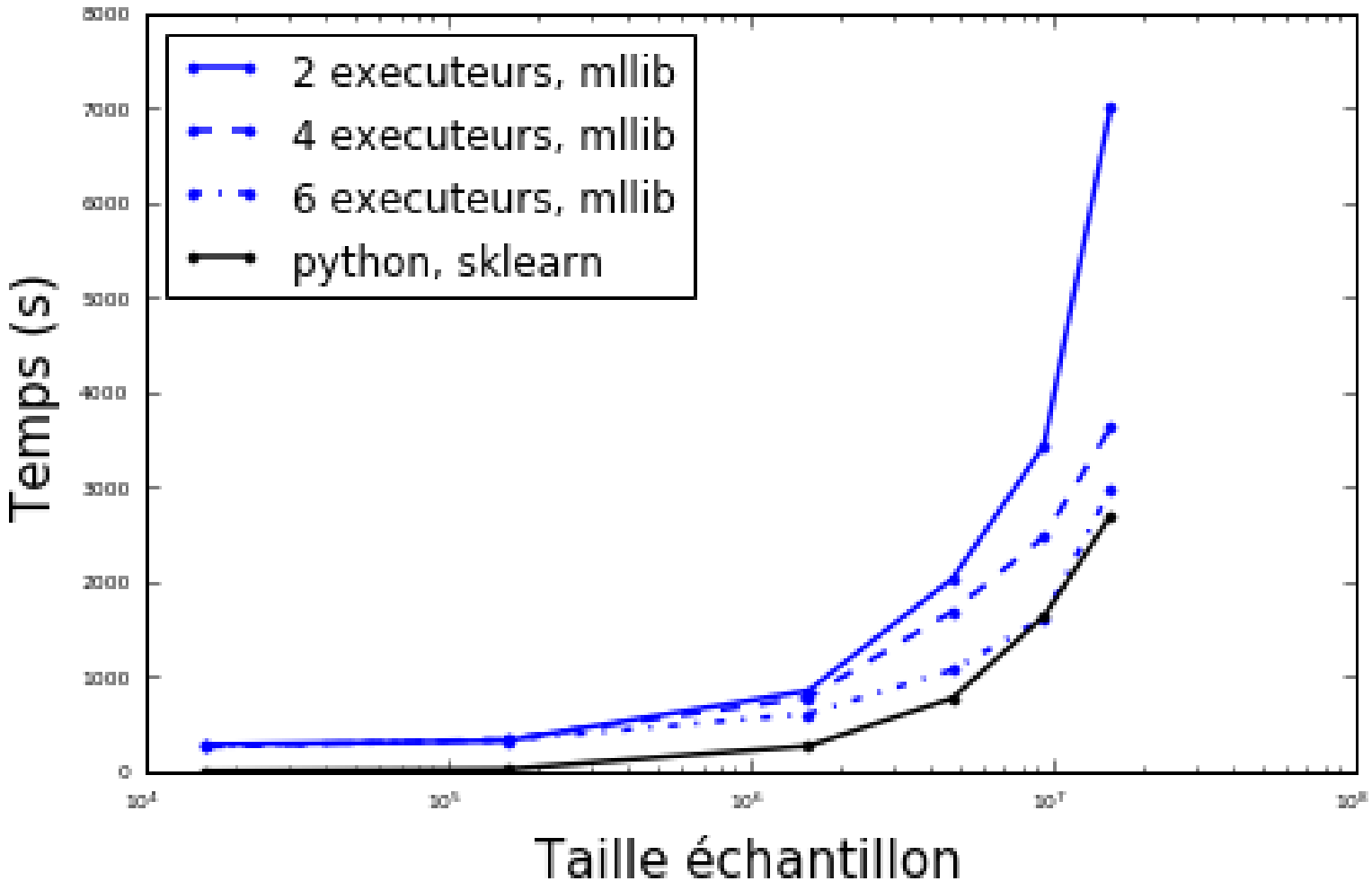
- Données publiques du concours `datascience.net`
- 15M de produits (3.5 Go), 3 niveaux : 5789 classes
- Classes déséquilibrées
- Solution gagnante (Goutorbe et al. 2016)
- Pyramide (Python) de régressions logistiques
- Simplifier : prévoir le 1er niveau : 47 classes
- Comparaison de Python Scikit-learn *vs.* Spark MLlib
- Trois phases : Nettoyage, Vectorisation, Apprentissage



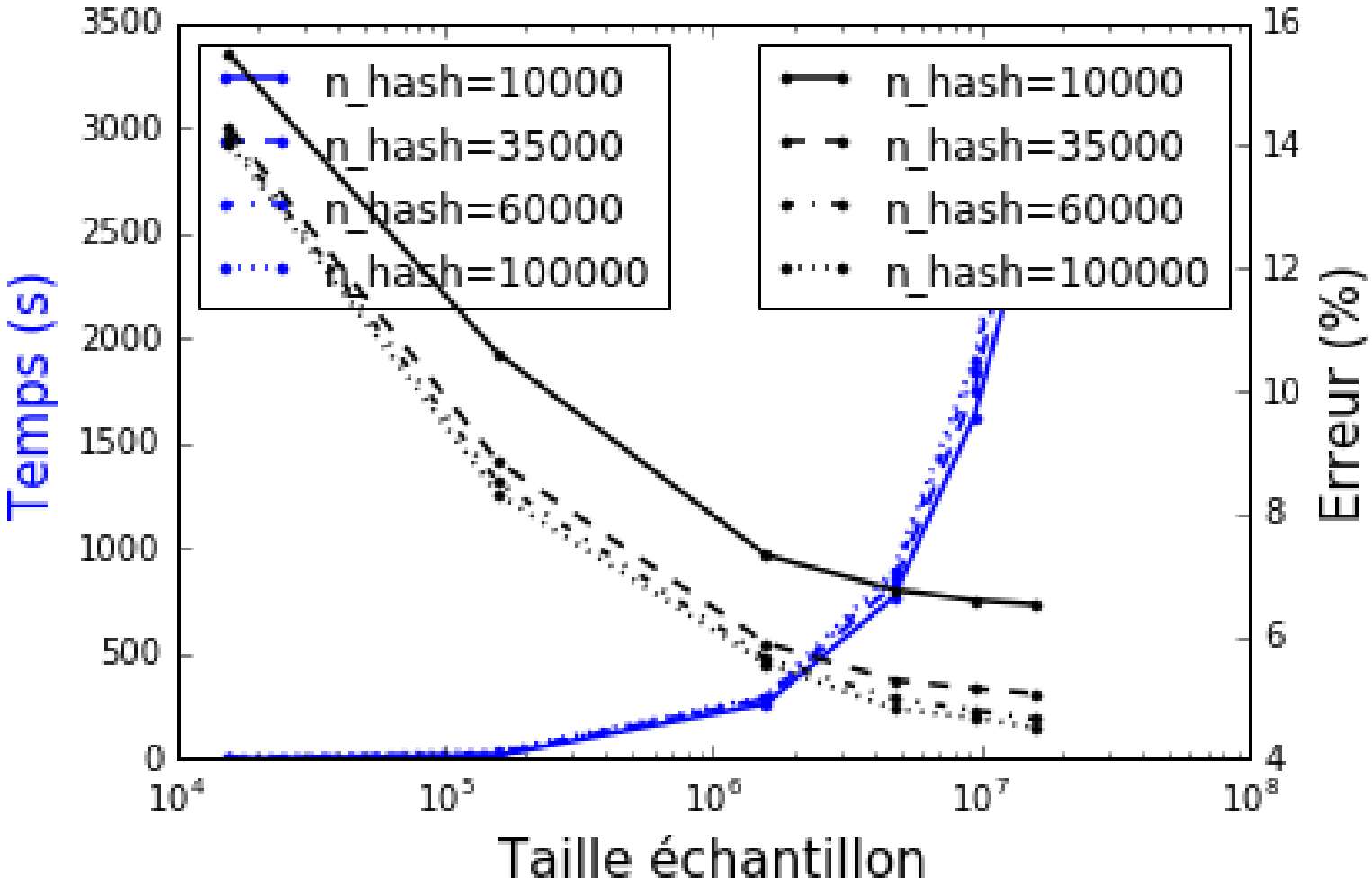
Cdiscount : nettoyage des données



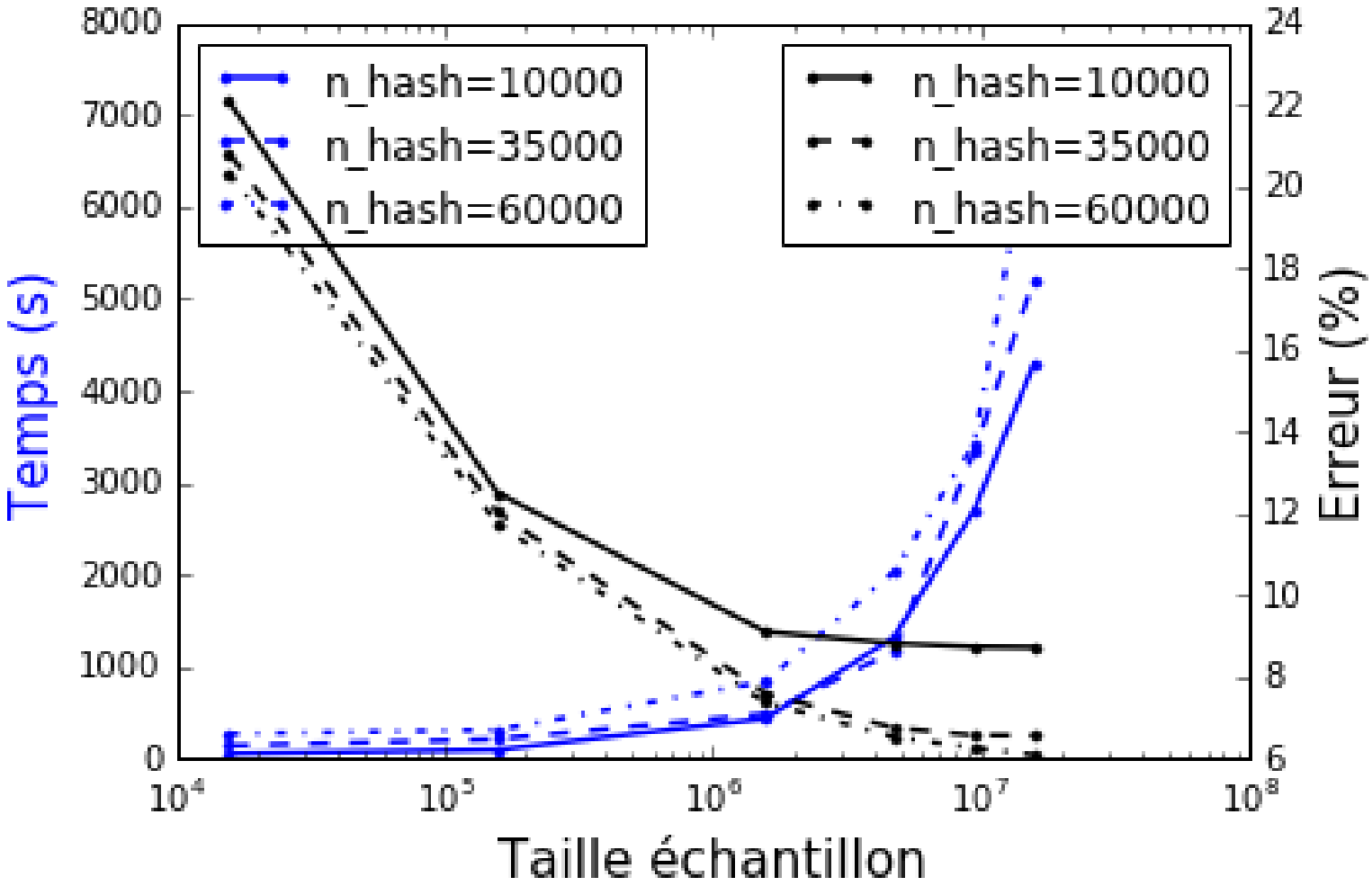
Cdiscount : vectorisation avec `n_hash` *= 60 000*



Cdiscount : apprentissage avec Spark, Python (Scikit-learn) et `n_hash` *= 60 000*



Cdiscount : Apprentissage avec Python Scikit-learn



Cdiscount : Apprentissage avec Spark MLlib

Conclusion

- Technologies en pleine effervescence (volatiles)
- Problème de maturation et sélection naturelle
- Comparaison entre architectures distribuée *vs.* intégrée
- Trois étapes :
  - *Data munging*, *streaming* : Spark, SparkSQL
  - Vectorisation : Python, Scikit-learn, Lucene...
  - Apprentissage : grosses data et gros modèles
- R *vs.* Python Scikit-learn *vs.* SparkSQL, MLlib
- Nettoyage et Apprentissage en ligne ?
- Cdiscount, Critéo, Deepky, Tinyclues, Hupi (ppml)...
- Ne pas oublier : fiabilité, représentativité des données
- Important : veille technologique
  XGBoost, Spark, TensorFlow, Keras ... ?

Références

- Besse P., Guillouet B., Loubes J.-M. (2016). Apprentissage sur données massives, trois cas d'usage avec R, Pyhton, Spark , *Apprentissage Statistique et Données Massives*, Maumy-Bertrand M., Saporta G. et Thomas Agnan C. (eds), Technip.
- Breiman L. (2001). Random forests, *Machine Learning*, 45, 5-32.
- Breiman L., Friedman J., Stone C. J., Olshen R. A. (1984). *Classification and Regression Trees*, Taylor & Francis.
- Chen T., Guestrin C. (2016). XGBoost : A Scalable Tree Boosting System, *KDD Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.
- Fernandez-Delgado M. , Cernadas E., Barro S., Amorim D. (2014). Do we Need Hundreds of Classiers to Solve Real World Classication Problems ?, *IEEE Trans. Pattern Anal. Mach. Intell*, 15.
- Freund Y., Schapire R.E. (1996). Experiments with a new boosting algorithm, *Machine Learning : proceedings of the Thirteenth International Conference*, Morgan Kaufman, San Francisco, 148-156.
- Goutorbe B., Jiao Y., Cornec M., Grauer C., Jakubowicz J. (2016). A large e-commerce data set released to benchmark categorization methods, in *Journées de Statistique de Montpellier*, SFdS.
- Zaharia M., Chowdhury M., Das T., Dave A., Ma J., McCauley M., Franklin M. J., Shenker S., Stoica I.

  (2012). Resilient Distributed Datasets : A Fault-Tolerant Abstraction for In-Memory Cluster Computing, *9th*

  USENIX Symposium on Networked Systems Design and Implementation, 15-28.