Chapitre 1

Identification du sens du résultat

Résumé. *L'extraction des demandes a été présentée dans le chapitre précédent comme l'identification du quantum demandé, du quantum résultat et du sens du résultat pour chaque demande d'une catégorie donnée. Le sens du résultat est l'information la plus importante pour le métier car elle donne une idée du taux d'acceptation des demandes dans les tribunaux pour une analyse descriptive du sens du résultat. Il a été proposé d'identifier le sens du résultat en interprétant simplement le verbe de décision de l'énoncé du résultat. Cette technique a l'inconvénient de dépendre de l'identification explicite du passage exprimant le résultat qui est donc une source supplémentaire d'erreurs. Le présent chapitre adresse cet inconvénient en reformulant l'identification du sens du résultat par la classification binaire (« accepte » / « rejette ») de la décision représentée en entrée sous forme vectorielle. Une décision pouvant comprendre plusieurs demandes de la catégorie traitée, nous ne traitons que le cas des décisions à une seule demande de la catégorie. Cependant, le défi de ce problème est le déséquilibre observé entre les 2 classes sur les données d'apprentissage. Nous observons en effet que la tendance est de rejeter une forte majorité des demandes, ce qui serait aussi le cas en général en justice pour la majorité des catégories. Sur la base de l'expertise d'une partie de l'encadrement de cette thèse, nous proposons une application de la méthode Gini-PLS de Mussard & Souissi-Benrejab [2018] qui a été conçue pour rendre robuste aux valeurs aberrantes, de vecteurs représentatifs, la régression par analyse des moindres carrés partiels (PLS). Nos expérimentations la compare à d'autres algorithmes de classification et sous différentes conditions de représentation dont divers modèles vectoriels et restriction du document à des sous-parties. Les meilleures résultats sont obtenus avec les arbres de classification avec une moyenne de F_1-mesure sur les catégories de ? ?%, maximale de ? ?%, minimale de ? ?%. Le Gini-PLS étant à ... ?*

1.1 Introduction

Comme le précédent, ce chapitre est relatif à l'extraction de données sur les demandes et résultats correspondants. Cependant, il est question ici d'extraire uniquement le sens du résultat d'une demande connaissant sa catégorie. Cette étude est intéressante parce que le problème devient

plus simple. En se passant de la localisation précise de l'énoncé du résultat, l'extraction du sens du résultat peut être formulée comme une tâche de classification de documents. Nous modélisons la tâche comme un problème de classification binaire consistant à entrainer un algorithme à reconnaitre si la demande a été rejetée (sens = rejette) ou acceptée (sens = accepte). Cette modélisation est proposée sur une restriction du problème définie par les postulats 1.1.1 et 1.1.2 basés sur nos observations des données annotées manuellement.

Postulat 1.1.1 *Pour toute catégorie de demande, les décisions ne contenant qu'une demande de cette catégorie sont très largement majoritaires.*

Ce postulat est légitime car les statistiques sur les données labellisées de la Figure **??** Page **??** montrent bien que dans chaque catégorie, les décisions contiennent en majorité une seule demande d'une même catégorie. En effet, dans les données annotées, les décisions à une demande sont au nombre de 23 sur 23 (100%) pour *acpa*, 19 sur 30 (63,33%) pour *concdel*, 189 sur 198 (95,45%) pour *dnais*, 73 sur 91 (80.22%) pour *dcppc*, et 73 sur 91 (76,21%) pour *doris*. On remarque néanmoins l'exception de la catégorie STYX (dommage-intérêt sur l'article 700 CPC), où dans la majorité des documents, on a plutôt 2 demandes. Cette exception peut se justifier par le fait que chaque partie fait généralement ce type de demande car elle porte sur le remboursement des frais de justice. Ce postulat présente cependant un inconvénient dû au fait que la majorité des demandes d'une catégorie peuvent se retrouver dans des décisions comprenant plus d'une demande de cette catégorie. Pour la catégorie *concdel* par exemple, 58 demandes ont été annotées manuellement (Figure 1.1 Page 3) mais 19 décisions sur 30 (63,33%) ont une seule demande de cette catégorie (Figure **??** Page **??**). Ces 19 décisions comprennent donc seulement 32,75% ($100 \times 19/58$) des demandes annotées pour *concdel*. Par conséquent, 67.24% de ces demandes sont dans les décisions annotées ayant plus d'une demande de cette catégorie. Il est donc possible de manquer un grand nombre de demandes.

Postulat 1.1.2 *Le sens du résultat est généralement binaire : accepte ou rejette.*

Ce postulat est justifié car les sens de résultat ont majoritairement l'une de ces deux valeurs (Figure 1.1 Page 3). Les autres valeurs sont très rares.

Cette étude porte sur l'analyse de l'impact de différents aspects techniques en général impliqués dans la classification de textes qui consistent en général en une combinaison de représentations des documents et d'algorithmes de classification. Cette analyse permettra de savoir s'il existe une certaine configuration permettant de déterminer le sens du résultat à

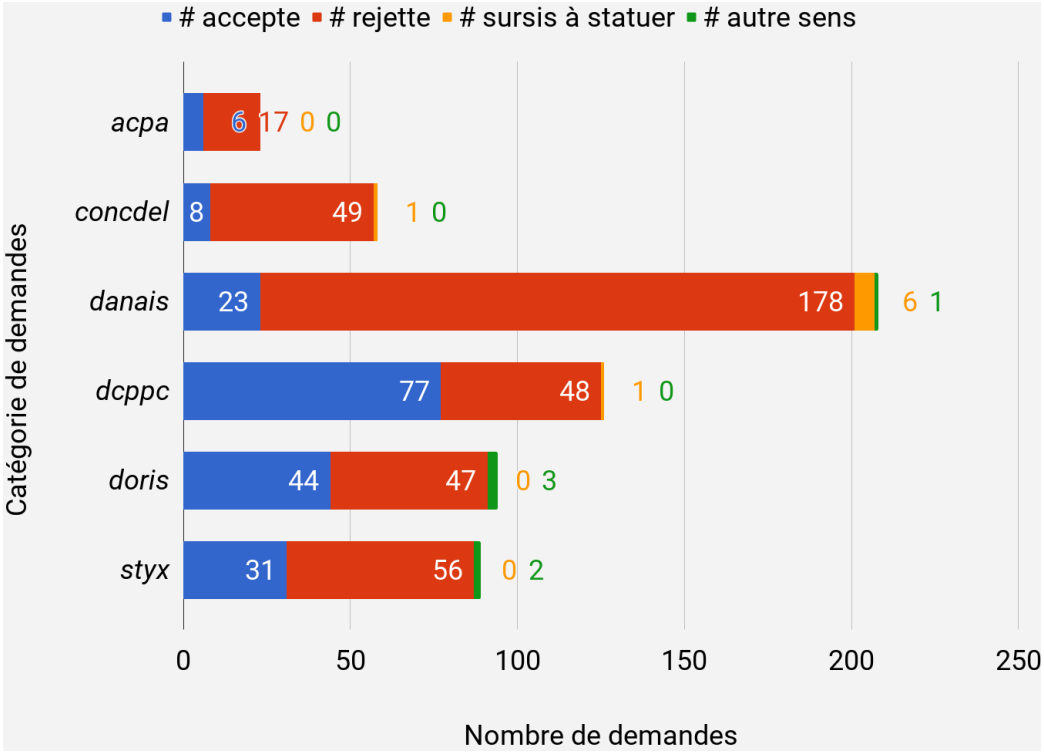


Figure 1.1 – Répartition des sens de résultat dans les données annotées.

une demande sans identifier précisément cette dernière dans le document. Nous proposons l'algorithme Gini-PLS généralisé qui est une extension du modèle Gini-PLS simple. Il s'agit d'un nouveau modèle dans lequel un paramètre de régularisation va permettre de mieux adapter la régression aux informations se situant en queues de distribution tout en atténuant, comme dans le Gini-PLS simple, l'influence exercée par les valeurs aberrantes. Nous proposons également une nouvelle régression (LOGIT-Gini-PLS) qui est mieux adaptée à l'explication d'une variable cible lorsque cette dernière est une variable binaire. Ces deux modèles n'ont par ailleurs jamais été appliqués à de la classification de textes.

1.2 Classification de documents

La classification de textes permet d'organiser des documents dans des groupes prédéfinis. Elle reçoit depuis longtemps beaucoup d'attention. Deux choix techniques influencent principalement les performances : la représentation des textes et l'algorithme de classification. Dans la suite,

la variable à prédire est notée y, et la base d'apprentissage comprend les observations de l'échantillon $D = \{(x_i, y_i)_{i=1..n}\}$. C représente l'ensemble des classes. Les notations du Tableau **??** sont utilisées dans cette section.

1.2.1 Représentation de textes

Chaque document d est représenté sous une forme vectorielle du type TF-IDF (*term frequency - inverse document frequency*) proposé par Salton & Buckley [1988] dont chaque dimension k est identifiée par un terme t_k. Tout document $d \in D$ est une séquence de mots $d = (d[1], \ldots, d[|d|])$, où d_i est le mot à la position i dans d. Sa représentation vectorielle est notée $\vec{d} = (\vec{d}[1], \vec{d}[2], \ldots, \vec{d}[m])$. Pour un modèle vectoriel de type TF-IDF de vocabulaire $V = \{t_1, t_2, \ldots, t_m\}$, $\vec{d}[k] = w(t_k, d)$ le poids du terme $t_k \in T$ dans le texte d (cf. **??**). Le poids $w(t_k, d)$ affecté à ce dernier est le produit normalisé d'un poids global $g(t_k)$ de t_k dans le corpus d'entraînement et d'un poids local $l(t_k, d)$ de t_k dans le document d : $w(t_k, d) = l(t, d) \times g(t) \times nf(d)$, où nf est un facteur de normalisation tel que la norme euclidienne $\sqrt{\sum_k (w(t_k, d))^2}$. Le poids global est calculé à partir d'une des méthodes de la section **??**. Le poids local est calculé à partir de la fréquence d'occurrence du terme dans le document à l'aide d'une des méthodes du Tableau 1.1 Page 4.

Description	Formule
Décompte brute du terme [Salton & Buckley, 1988]	$tf(t,d) =$ nombre d'occurrences de t dans d
Présence du terme [Salton & Buckley, 1988]	$tp(t,d) = \begin{cases} 1 & \text{, si } tf(t,d) > 0 \\ 0 & \text{, sinon} \end{cases}$
Normalisation logarithmique	$logtf(t,d) = 1 + \log(tf(t,d))$
Fréquence augmentée et normalisée du terme [Salton & Buckley, 1988]	$atf(t,d) = k + (1-k)\frac{tf(t,d)}{\max_{t \in T} tf(t,d)}$
Normalisation basée sur la fréquence moyenne du terme [Manning *et al.* , 2009] (avg représente la moyenne)	$logave(t,d) = \frac{1+\log tf(t,d)}{1+\log \operatorname{avg}_{t \in T} tf(t,d)}$

Tableau 1.1 – Métriques locales.

1.2.2 Algorithmes traditionnels de classification de données

Bien que la classification de documents voit se développer récemment des algorithmes propres aux textes, un grand nombre de méthodes ont été développées dans des contextes détachés des considérations applicatives. Ces méthodes sont généralement basée sur une représentation des textes dans un espace vectoriel $\mathcal{X}$ d'entrée et délimitent une frontière entre les classes dans un espace multidimensionnel.

1.2.2.1 Le classifieur bayésien naïf (NB)

Le classifieur naïf bayésien [Duda *et al.* , 1973] est un modèle probabiliste qui estime la probabilité qu'un texte appartienne à une classe à l'aide du théorème de Bayes [Raschka, 2014] :

$$\text{probabilité a posteriori} = \frac{\text{probabilité conditionnelle} \cdot \text{probabilité a priori}}{\text{évidence}}$$

La probabilité a posteriori peut être interprétée pour la classification de documents par la question "Quelle est la probabilité que le document d soit de la classe $y = c \in C$?". La réponse à cette question se formalise comme suit :

$$P(y = c|d) = \frac{P(d|c)P(c)}{P(d)}, \forall c \in C$$

ou plus simplement $P(y = c|d) \propto P(c)P(d|c)$ car $P(d)$ ne change pas en fonction de la classe et peut donc être ignorée [Rish, 2001]. d est catégorisé dans la classe c pour laquelle $P(c|d)$ est maximale :

$$y = \underset{c \in C}{\operatorname{argmax}} P(c|d).$$

La phase d'entrainement, appliquée à des exemples déjà labellisés, permet d'estimer les paramètres $P(c)$ et $P(d|c)$ qui servent à calculer $P(c|d)$.

$P(c)$ est estimée par la proportion de documents classés dans c parmi les exemples d'apprentissage : $P(c) = \frac{N_c}{N}, \forall c \in C$.

$P(c|d)$ est estimé grâce l'hypothèse 1.2.1 d'indépendance conditionnelle des descripteurs (termes). Une hypothèse naïve dont la violation, par les données réelles, n'empêche pas le NB de bien fonctionner [Rish, 2001].

Hypothèse 1.2.1 (indépendance conditionnelle des descripteurs) *[Un modèle naïf bayésien étant de type génératif], étant donnée la catégorie du texte, la position de chaque mot dans le texte est générée indépendamment de tout autre mot.*

Si l'ensemble des termes de d est $\{t_1, \ldots, t_m\} \subset V$ (vocabulaire), alors grâce à l'hypothèse 1.2.1, $P(d|c) = P(t_1, \ldots, t_m|c) = Prod_{k=1}^{m} P(t_k|c)$, et pour un terme t_k, la probabilité conditionnelle $P(t_k|c)$ est la proportion d'exemples de c qui contiennent t_k : $P(t_k|c) = \frac{N_{t_k,c}}{|D_c|}, \forall k \in \{1, \ldots, m\}$.

1.2.2.2 Machine à vecteurs de support (SVM)

La classification binaire par une machine à vecteurs de support (SVM) [Vapnik, 1995] affecte à tout objet en entrée x la classe y qui correspond au coté d'un hyperplan, séparant les exemples d'entraînement des classes candidates, où x se trouve. La phase d'apprentissage consiste à déterminer l'hyperplan optimal $w^T x + b = 0$ i.e. dont la marge [1] est maximale (Figure 1.2 [2]).

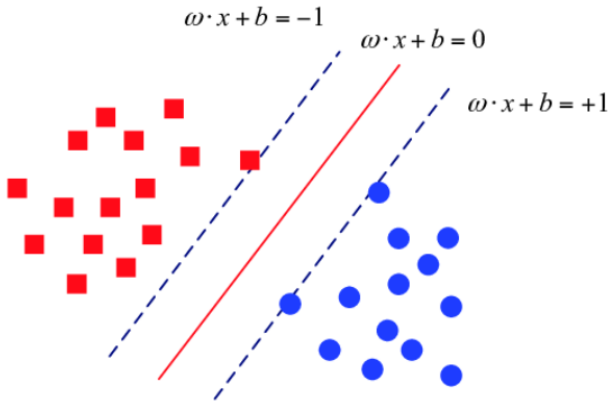


Figure 1.2 – Hyperplan optimal et marge maximale d'un SVM.

Le vecteur w des poids des caractéristiques et le biais b sont déterminés par le problème d'optimisation du « SVM à marges molles » de Cortes & Vapnik [1995] :

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.c.} \quad & y_i(w^T + b) \geq 1 - \xi_i, \xi_i \geq 0. \end{aligned}$$

où C est la constante de régularisation pour éviter un sur-apprentissage ou un sous-apprentissage, les ξ_i sont des variables ressort (*slack variables*) qui permettent à des points de se retrouver dans la marge.

1. La plus petite distance entre les exemples d'apprentissage et l'hyperplan séparateur.
2. `http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf`

Lorsque les exemples d'apprentissage des classes sont linéairement séparables, la classe d'un objet x correspond au signe de la fonction de décision $f(x) = w^T x + b$:

$$y = signe(f(x)) = \begin{cases} -1 & \text{si } w^T x + b < 0 \\ +1 & \text{si } w^T x + b > 0. \end{cases}$$

. Cependant, ils ne le sont pas toujours dans l'espace $\mathcal{X}$. Ainsi, une fonction « noyau » (*kernel*) $K : \mathcal{X} \longrightarrow \mathcal{F}$ doit être choisie pour transformer chaque donnée entrée x de l'espace original $\mathcal{X}$ vers un nouvel espace $\mathcal{F}$ dit de caractéristiques dans lequel les classes sont linéairement séparables. Par conséquent, la fonction de classification s'écrit

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) + b$$

où les α_i sont les coefficients de la combinaison linéaire des exemples d'apprentissage égale à w ($w = \sum_{i=1}^{n} \alpha_i x_i$) [Ben-Hur & Weston, 2010]. Parmi les multiples formes qu'il peut prendre, le noyau peut être, par exemple, soit linéaire($K(x, x_i) = x^T x_i + c$), soit polynomial ($K(x, x_i) = (\gamma x^T x_i + c)^d$), soit Gaussien ou RBF[3] ($K(x, x_i) = \exp -\gamma \|x - x_i\|$), soit une sigmoïde ($K(x, x_i) = \tanh(\gamma x^T x_i + c)$) [Amami *et al.* , 2013].

1.2.2.3 k-plus-proches-voisins (kNN)

L'algorithme des k-plus-proches-voisins [Cover & Hart, 1967] est un algorithme simple qui consiste à affecter à un nouvel objet x la classe majoritaire y' parmi ceux des k points d'exemples d'entrainement $\{(x_i, y_i)\}_{1 \leq i \leq k}$, les plus proches du point x selon la distance Dis choisie. Ainsi, trois éléments clés influencent l'efficacité de la classification :

1. Si les les données d'entrainement sont trop nombreuses, le processus de classification peut devenir coûteux en temps de calcul. En effet, la distance du nouvel objet à chaque point annoté est calculée.
2. Le nombre de voisins (c'est-à-dire la valeur de k) ne doit être trop petit pour limiter la sensibilité aux bruits / *outliers*. Il ne doit non plus être trop grand au risque d'avoir dans le voisinage trop de points d'une autre classe que celle attendue. La sensibilité au nombre de voisins peut être atténuée en pondérant les points par leur distance à l'objet à classer. Il a été proposé plusieurs variantes de cette stratégie de « vote pondéré par la distance », comme par exemple :

3. *radial base function*

- $y = \underset{c}{\text{argmax}} \sum_{(x_i,y_i)} \frac{1}{Dis(x,x_i)^2} \times I(c = y_i)$ [Dudani, 1976]

$$\text{où } I(c = y_i) = \begin{cases} 1 & \text{si } c = y_i \\ 0 & \text{sinon} \end{cases}$$

- $y = \underset{c}{\text{argmax}} \sum_{(x_i,y_i)} w_i \times I(c = y_i)$ [Gou *et al.* , 2011]

$$\text{avec } w_i = \begin{cases} \frac{Dis(x,d_k)-Dis(x,d_i)}{Dis(x,d_k)-Dis(x,d_1)} & \text{si } Dis(x,d_k) \neq Dis(x,d_1) \\ 1 & \text{sinon.} \end{cases}$$

3. La métrique de calcul de distance doit être adéquate pour le type de donnée et la tâche. Par exemple, la distance cosinus est souvent préférable à la distance euclidienne pour la classification de documents. En effet, la distance euclidienne se dégrade lorsque le nombre de dimensions augmente [Sohangir & Wang, 2017; Aggarwal *et al.* , 2001].

1.2.2.4 Arbre de décision

Un arbre de décision est une structure arborescente qui associe un label prédéfini à des objets (classification), ou prédire la valeur d'une variable continue (régression). Il comprend des nœuds internes qui correspondent chacun à un test sur la valeur d'un attribut (test uni-varié), des arêtes correspondant à une sortie du test, et enfin des feuilles ou nœuds terminaux qui correspondent chacun à une prédiction. La classification d'un objet x consiste à faire passer successivement les tests en fonction des valeurs des attributs de x, de la racine jusqu'à une feuille dont le label est retourné comme classe de x (Algorithme 1).

Algorithme 1 : Classification par arbre de décision

Données : Objet x, Arbre A
Résultat : label

```
n := racine(A) ;
tant que n n'est pas une feuille faire
    Effectuer sur x le test associé à n;
    n := noeud fils de n correspondant au résultat du test ;
retourner le label associé à la feuille n;
```

La construction de l'arbre (phase d'apprentissage) consiste à générer une hiérarchie de tests, aussi courte que possible, qui divise successive-

ment l'ensemble D d'exemples d'apprentissage en sous-ensembles disjoints de plus en plus purs [4]. L'arbre est construit de la racine aux feuilles en divisant les données d'entraînement S_t à chaque nœud (t) de sorte à minimiser le degré d'impureté des sous-ensembles d'exemples S_{t_i} dans les nœuds fils (t_i). Le critère de coupe est généralement défini à partir d'une métrique d'impureté comme par exemple :

- l'entropie de la distribution des classes dans S_t :
  $h_C(S_t) = -\sum_{c \in C} [P(c|S_t) \log_2 P(c|S_t)]$;
- l'indice de Gini mesurant la divergence entre les distributions de probabilité des valeurs de la variable prédite : $g_C(S_t) = 1 - \sum_{c \in C} [P(c|S_t)]^2$;
- l'erreur de classification définie par : $e_C(S_t) = 1 - \max_{c \in C} [P(c|S_t)]$.

Pour ces métriques, $P(c|S_t)$ représente la proportion d'exemples du nœud t appartenant à c. Parmi les critères de séparation les plus populaires associés à ces métriques d'impureté, on retrouve :

- le gain d'information apporté par le test t portant sur l'attribut a (qui divise S_t en des sous-ensembles S_{t_i}) utilisant l'entropie comme métrique d'impureté, et est définie par la différence entre l'entropie de t et la moyenne des entropies des fils de t :
  $ig(S_t, a) = h_C(S_t) - i(S_t, t, a) = h_C(S_t) - \sum_i \frac{|S_{t_i}|}{|S_t|} \cdot h_C(S_{t_i})$;
- le rapport des gains, qui corrige le gain d'information, biaisé en faveur des tests ayant un grand nombre d'alternatives (sorties du nœud), en prenant en compte l'information intrinsèque $h_t(S_t)$ de la séparation de S_t suivant le test t en sous-ensembles S_{t_i} :
  $gr(S_t, t, a) = \frac{ig(S_t,t,a)}{h_t(S_t)}$ avec $h_t(S_t) = \sum_i \frac{|S_{t_i}|}{|S_t|} \log_2 \left(\frac{|S_{t_i}|}{|S_t|} \right)$
- le critère binaire de "doublage" (*twoing criteria*) qui ne s'emploie que pour les arbres binaires :
  $tc(t) = \frac{P(S_{t_R}|S_t)P(S_{t_L}|S_t)}{4} \left[\sum_{c \in C} |P(c|t_L) - P(c|t_R)| \right]^2$ où $P(S_{t_R}|S_t)$ et $P(S_{t_L}|S_t)$ sont les proportions de S_t qui vont respectivement dans les fils t_R et t_L après séparation suivant le test t.

Les variables nominales peuvent être divisées soit en utilisant autant de partitions que de valeurs distinctes, soit uniquement en des partitions binaires suivant des tests booléens nécessitant de rechercher la division

4. homogénéité des labels

optimale. Les variables numériques sont divisées quant à elles soit par discrétisation de leur domaine en les transformant en variables catégoriques ordinales, soit en recherchant la meilleure division binaire parmi toutes les séparations possibles.

La construction de l'arbre est une division récursive qui peut continuer tant qu'il est possible d'améliorer la pureté des nœuds, ce qui peut engendrer un arbre très grand résultant en un sur-apprentissage [5], et une forte complexité temporelle et spatiale lors de la prédiction. Pour s'arrêter plus tôt ("pré-élagage"), plusieurs conditions sont possibles comme par exemple, l'atteinte d'un seuil minimum par la taille des données ($|S_t|$), ou l'atteinte par l'arbre d'une profondeur maximale, ou l'amélioration du critère de division est très faible, etc. Le post-élagage [6] est appliqué après construction de l'arbre toujours dans le but de minimiser le sur-apprentissage et la complexité. Le post-élagage peut être basé, par exemple, soit sur la réduction du taux d'erreur (éliminer successivement les feuilles si cela ne fait pas croître le taux d'erreur sur les données d'entrainement), soit sur la stratégie coût-complexité de Breiman *et al.* [1984].

Les algorithmes de construction d'arbres diffèrent ainsi par leur critère de séparation, leur stratégie d'élagage, et leur capacité à gérer les types d'attributs, les valeurs manquantes et extrêmes. Singh & Gupta [2014] comparent les deux algorithmes CART [Breiman *et al.* , 1984] (critère de « doublage », élagage coût-complexité) et C4.5 [Quinlan, 1993] (rapport des gains, élagage à réduction d'erreur).

1.2.2.5 Analyses discriminantes linéaires et quadratiques

L'analyse discriminante comprend l'ensemble des méthodes déterminant les combinaisons linéaires de variables qui permettent de séparer le mieux possible $|C|$ catégories ou variables qualitatives. Les analyses linéaires et quadratiques sont des méthodes probabilistes basées sur la probabilité conditionnelle d'appartenance d'un objet $x \in \mathbb{R}^m$ à une classe $c_k \in C$:

$$P(y = c_k|x) = \frac{P(y = c_k)P(x|y = c_k)}{P(x)} = \frac{P(y = c_k)P(x|y = c_k)}{\sum_{j=1}^{K} P(y = c_j)P(x|y = c_j)}.$$

5. Un modèle trop précis a un très faible taux d'erreur sur les données d'entraînement (erreur d'apprentissage) mais un fort taux d'erreur sur les données de test (erreur de test).

6. Suppression de sous-arbres superflus après génération de l'arbre.

La classe de x est donc $y = \underset{k}{\text{argmax}}\, P(y = c_k|x)$ avec $P(y = c_k|x) \propto P(y = c_k)P(x|y = c_k)$ car le dénominateur est le même pour toutes les classes. Dans cette expression, $P(y = c_k)$ est la proportion d'exemples de classes c_k dans l'ensemble des données d'apprentissage. Il ne reste donc qu'à déterminer $P(x|y = c_k)$, pour trouver y. Deux hypothèses simplifient les calculs :

1. L'hypothèse de normalité statue que la probabilité conditionnelle $P(x|y)$ suit une loi normale multidimensionnelle :

   $$P(x|y = c_k) = \mathcal{N}(\mu_k, V_k) = \frac{1}{\sqrt{(2\pi)^m det(V_k)}} e^{-\frac{1}{2}(x-\mu_k)^\top V_k^{-1}(x-\mu_k)},$$

   μ_k étant le centre de gravité conditionnel (centre de gravité des données d'entraînement de la classe $c_k \in C$), et V_k la matrice de variance-covariance de la classe c_k. Le logarithme de $P(x|c_k)$ donne : $\ln(P(x|c_k)) \propto -\frac{m}{2}\ln(2\pi) - \frac{1}{2}\ln(det(V_k)) - \frac{1}{2}(x-\mu_k)^\top V_k^{-1}(x-\mu_k)$ [Ghojogh & Crowley, 2019]. Grâce à la proportionnalité de la probabilité conditionnelle à son logarithme, on déduit une fonction discriminante proportionnelle à $P(c_k|x)$:

   $$\delta_1(x, c_k) = -\frac{1}{2}\ln(det(V_k)) - \frac{1}{2}(x-\mu_k)^\top V_k^{-1}(x-\mu_k) + \ln(P(c_k))$$

   en éliminant le terme $-\frac{m}{2}\ln(2\pi)$ car il est le même pour toutes les classes.
2. l'hypothèse d'homoscédasticité statue que les matrices de variance co-variance conditionnelles sont identiques i.e. :

   $$\forall j, k \in \{1, ..., |C|\}, V_j = V_k = V.$$

   Cette hypothèse permet de simplifier $\delta_1(x, c_k)$ en :

   $$\begin{aligned}\delta_2(x, k) &= -\tfrac{1}{2}\ln(det(V_k)) - \tfrac{1}{2}x^\top V^{-1}x - \mu_k^\top V^{-1}\mu_k + \mu_k^\top V^{-1}x + \ln(P(c_k)) \\ &= \mu_k^\top V^{-1}x - \tfrac{1}{2}\mu_k^\top V^{-1}\mu_k + \ln(P(c_k))\end{aligned}$$

   car les termes $-\frac{1}{2}\ln(det(V_k))$ et $-\frac{1}{2}x^\top V^{-1}x$ sont indépendants des classes.

L'analyse discriminante linéaire (LDA) [Fisher, 1936] est définie à partir de la simplification de $P(x|c_k)$ sous ces deux hypothèses. Ainsi la classe de x est $y = \underset{k\in\{1,...,|C|\}}{\text{argmax}}\ \delta_2(x, c_k)$. L'analyse discriminante quadratique (QDA) [McLachlan, 1992] quand à elle ne considère pas l'hétéroscédasticité (i.e. $\exists k \neq j, V_k \neq V_j$), et ne s'appuie que sur l'hypothèse de normalité. La classe de x est par conséquent $y = \underset{k\in\{1,...,|C|\}}{\text{argmax}}\ \delta_1(x, c_k)$.

1.2.3 Algorithmes dédiés aux textes

Les algorithmes dédiés aux textes intègrent leur propre représentation de document, contrairement aux algorithmes opérant sur des espaces vectoriels aux axes et poids paramétrables à volonté comme le SVM. Actuellement, les algorithmes NBSVM [Wang & Manning, 2012] et FastText [Grave *et al.* , 2017] sont les plus populaires pour la classification de documents avec une très bonne précision pour l'analyse de sentiments [citation] et exemples [Mohammad, 2018; Joulin *et al.* , 2016; Shirsath, 2017; Elsaadawy *et al.* , 2018].

1.2.3.1 NBSVM

Le NBSVM [Wang & Manning, 2012] est un classifieur binaire qui consiste à appliquer le SVM à une transformation par le classifieur bayésien multinomial des composantes du vecteur x.

consiste à transformer les poids $f^{(k)}$ caractéristiques V des textes $x^{(k)}$, réduits à leur simple présence $\widehat{f}^{(k)}$ en réalisant leur produit élément à élément ($\tilde{f}^{(k)} = r \circ \widehat{f}^{(k)}$) avec le vecteur de poids r du classifieurs bayésien multinomial (calculé avec le vecteur présence de caractéristique) : $r = \log\left(\frac{p/||p||_1}{q/||q||_1}\right)$ avec $p = \alpha + \sum_{k:y^{(k)}=1} f^{(k)}$, $q = \alpha + \sum_{k:y^{(k)}=-1} f^{(k)}$. L'ensemble des caractéristiques V est constitué de n-grammes de mots. Le nouveau vecteur issu de ce produit représente le texte ($x^{(k)} = \tilde{f}^{(k)}$) en entrée d'un SVM classique. La classe de $x^{(k)}$ est prédite par : $y^{(k)} = sign(\mathbf{w}^T x^{(k)} + b)$, $\mathbf{w}$ et b étant appris lors de l'entraînement du SVM. Une interpolation entre le bayésien multinomial et le SVM est nécessaire pour assurer la robustesse du NBSVM et des performances excellentes pour toute tâche de classification de documents ; les poids $\mathbf{w}$ sont réajustés par le modèle $\mathbf{w}' = (1-\beta)\overline{w} + \beta\mathbf{w}$, où $\overline{w} = ||\mathbf{w}||_1/|V|$ et $\beta \in [0;1]$.

1.2.3.2 FastText

FastText [Grave *et al.* , 2017], quant à lui, est un modèle de réseau de neurones dont l'architecture est semblable à celle de la variante CBOW de la méthode de plongement sémantique Word2Vec [Mikolov *et al.* , 2013] dans laquelle le mot du milieu a été remplacé par le label de la classe du texte et au-dessus de laquelle la fonction softmax $f(z) = \left[\frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}\right]_{\forall j \in \{1,\dots,K\}}$

est rajoutée pour réaliser la classification à partir de la représentation distribuée du texte. L'entraînement, sur un corpus manuellement annoté de n documents, consiste à minimiser $-\frac{1}{n} y_i \cdot \sum_{i=1}^{n} y_i \cdot \log f(B \cdot A \cdot x_i)$ qui estime la distribution de probabilité des classes ; A et B étant les matrices des poids à apprendre, et x_i la représentation vectorielle sous forme de « sac de N-grammes »[7] du i^{eme} document. Comment se déroule l'entrainement et l'apprentissage ?

1.2.4 Techniques d'amélioration de l'efficacité

La faible quantité [Ruparel *et al.* , 2013] et le déséquilibre des données sont susceptibles d'être des obstacles à l'entraînement des modèles de classification. De nombreuses techniques permettent néanmoins d'optimiser l'apprentissage en fonction des données. La sélection de modèle consiste à choisir les meilleures valeurs des hyper-paramètres (par exemple C et γ chez le SVM) en testant différentes combinaisons de valeurs candidates sur une fraction de la base d'entraînement (base de développement). La combinaison de classifieurs est aussi une méthode très étudiée [Kittler *et al.* , 1996; Kuncheva, 2004; Tulyakov *et al.* , 2008] notamment par l'exemple des forêts aléatoires [Breiman, 2001], ou de SVM ensembliste (*Ensemble SVM*) [Dong & Han, 2005]. Par ailleurs, la représentation vectorielle des textes résulte généralement en des vecteurs de haute dimension dont les coordonnées sont en majorité nulles. Par conséquent, les techniques de réduction ou transformation des dimensions, comme les analyses discriminantes, permettent de d'obtenir des vecteurs plus pertinents pour la classification.

1.3 Application du PLS à la classification des textes

Justification : Pourquoi le PLS ? à quel problème cette méthode répond spécifiquement ? Qu'est ce que moi j'ai fait/ajouté ? : le modèle a été déjà publié en temps que méthode de régression. nous l'adaptons pour de la classification de textes pourquoi ? : ça n'a jamais été appliqué en classif. de texte, orienté par l'expertise de l'encadrement sur cette technique déjà publiée.

La méthode ou l'analyse des moindres carrés partiels PLS [Wold, 1966] explique la dépendance entre une ou plusieurs variables y (dite dépen-

7. Modèle sac-de-mot calculé sur des occurrences de N-grammes (segments N mots consécutifs dans un texte.)

dantes) et des K variables $x_1, x_2, ..., x_K$ (dites explicatives ou indépendantes). Elle consiste principalement à transformer les variables explicatives en un nombre réduit de h composantes principales orthogonales $t_1, ..., t_h$. Il s'agit donc d'une méthode de réduction de dimension au même titre que l'analyse en composantes principales, l'analyse discriminante linéaire (LDA), et l'analyse discriminante quadratique (QDA). Les composantes t_h sont construites par étapes en appliquant l'algorithme du PLS de façon récurrente sur les données mal prédites (résidus). Plus précisément, à chaque itération h, la composante t_h est calculée par la formule $t_h = w_{h1}x_1 + \cdots + w_{hj}x_j + \cdots + w_{hp}x_K$ dont les coefficients w_{hj} sont à estimer. L'analyse PLS présente plusieurs avantages [Lacroux, 2011] dont la robustesse au problème de haute-dimension [8] comme on peut l'observer dans nos données (faible quantité de données annotées), et aussi prend en compte la multicolinéarité qui peut exister entre les variables explicatives, notamment quand celles-ci sont des mots/termes co-occurrents dans les documents. Cette méthode est étendue et appliquée avec succès pour divers problèmes de régression [Lacroux, 2011] ou de classification de données vectorielles en général [Liu & Rayens, 2007; Durif *et al.* , 2017; Bazzoli & Lambert-Lacroix, 2018], et de textes en particulier [Zeng *et al.* , 2007]. Nous nous sommes intéressés particulièrement à deux extensions : la régression Gini-PLS [Mussard & Souissi-Benrejab, 2018] dont l'intérêt est de réduire la sensibilité aux valeurs aberrantes des variables, et la méthode Logit-PLS [Tenenhaus, 2005] combinant la régression logistique et la PLS. Une combinaison de ces deux approches (Logit-Gini-PLS) est décrite dans la suite de cette section (Section § 1.3.3.2).

1.3.1 L'opérateur Gini covariance

Soit $\bar{x}_k$ la moyenne arithmétique de la variable $x_k, \forall k \in [1\,..\,m]$ sur les n observations d'apprentissage. L'opérateur de Gini covariance proposé par Schechtman & Yitzhaki [2003], encore appelé opérateur co-Gini est donné par :

$$\text{cog}(x_\ell, x_k) := \text{cov}(x_\ell, F(x_k)) = \frac{1}{n}\sum_{i=1}^{n}(x_{i\ell} - \bar{x_\ell})(F(x_{ik}) - \bar{F}_{x_k}), \qquad (1.1)$$

où $F(x_k)$ est la fonction de répartition de x_k, $\bar{F}_{x_k}$ sa moyenne, avec $\ell \neq k = 1, \ldots, K$. Lorsque $k = \ell$ le co-Gini mesure la variabilité entre une variable et elle-même (l'équivalent de la variance mesurée sur la norme ℓ_2). Le

8. Lorsque le nombre de variables explicatives est très grand devant le nombre d'observations ($n << K$).

co-Gini est une mesure basée sur la distance de Manhattan (distance de métrique ℓ_1), en effet :

$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}|x_{ik}-x_{jk}| = 4\text{cog}(x_k, x_k).$$

D'autre part, lorsque $k \neq \ell$, le co-Gini produit une mesure de la variabilité jointe entre deux variables. Puisque le co-Gini n'est pas symétrique :

$$\text{cog}(x_k, x_\ell) := \text{cov}(x_k, F(x_\ell)) = \frac{1}{n}\sum_{i=1}^{n}(x_{ik}-\bar{x_k})(F(x_{i\ell})-\bar{F}_{x_\ell}).$$

Définissons les rangs croissants d'une variable alétoire afin de fournir un estimateur de F,

$$R_\uparrow(x_{i\ell}) := nF(x_{i\ell)} = \begin{cases} \#\{x \leq x_{i\ell}\} & \text{si aucune observation similaire} \\ \frac{\sum_{i=1}^{p}\#\{x \leq x_{i\ell}\}}{p} & \text{s'il existe } p \text{ valeurs similaires } x_{i\ell}. \end{cases}$$

Alors, un estimateur du co-Gini est donné par,

$$\widehat{\text{cog}}(x_\ell, x_k) := \frac{1}{n}\sum_{i=1}^{n}(x_{i\ell}-\bar{x_\ell})(R_\uparrow(x_{ik})-\bar{R}_{\uparrow_{x_k}}),\ \forall k, \ell = 1, \ldots, K, \quad (1.2)$$

avec $\bar{R}_{\uparrow_{x_k}}$ la moyenne arithmétique du vecteur rang de la variable x_k.

1.3.2 Gini-PLS

Le premier algorithme Gini-PLS a été proposé par Mussard & Souissi-Benrejab [2018]. Nous le décrivons dans les lignes qui suivent. Il s'agit d'une méthode de compression avec débruitage qui consiste à réduire les dimensions de l'espace généré par X afin de trouver des composantes principales débruitées, dans le même esprit qu'une ACP débruitée, néanmoins l'approche est supervisée dans la mesure où une variable cible y est prise en compte dans le changement d'espace. Le sous-espace formé par les composantes principales $\{t_1, t_2, \cdots\}$ est construit de telle sorte que le lien entre les variables explicatives $X = [x_1, x_2, \cdots]$ et la cible y est maximisé.

• **Étape 1 :** La régression Gini permet de concevoir un nouveau type de lien entre la variable expliquée et les variables explicatives tout en évitant l'influence des valeurs aberrantes. Ceci est permis grâce notamment

à l'opérateur co-Gini dans lequel le rôle de la variable explicative est remplacé par celui de son vecteur rang dans un espace muni d'une métrique ℓ_1. Ainsi, il est possible de créer un nouveau vecteur de poids w_1 qui renforce le lien (co-Gini) entre la variable expliquée y et les régresseurs X dans le cadre d'une régression (linéaire ou non linéaire).
La solution du programme,

$$\max \operatorname{cog}(y, Xw_1) \text{ , s.c. } \|w_1\| = 1 \text{ , est}$$

$$w_{1j} = \frac{\operatorname{cog}(y, x_j)}{\sqrt{\sum_{k=1}^{K} cog^2(y, x_k)}} \text{ , } \forall j = 1 \ldots, p \text{ .}$$

La pondération est équivalente à :

$$w_{1k} = \frac{\operatorname{cov}(y, R(x_k))}{\sqrt{\sum_{k=1}^{K} \operatorname{cov}^2(y, R(x_k))}} \text{ , } \forall k = 1 \ldots, K \text{ .}$$

Comme dans la régression PLS, on régresse y sur la composante t_1 qui est construite de la manière suivante :

$$t_1 = \sum_{k=1}^{K} w_{1k} x_k \implies y = \hat{c}_1 t_1 + \hat{\varepsilon}_1 \text{ .}$$

• **Étape 2 :** On régresse le vecteur rang de chaque régreseur $R(x_k)$ sur la composante t_1 par moindres carrés ordinaires afin de récupérer les résidus $\hat{U}_{(1)k}$:

$$R(x_k) = \hat{\beta} t_1 + \hat{U}_{(1)k} \text{ , } \forall k = 1, \cdots, K \text{ .}$$

On construit le nouveau vecteur de pondération en utilisant les rangs des résidus des régressions partielles :

$$\max \operatorname{cog}(\hat{\varepsilon}_1, \hat{U}_{(1)} w_2) \text{ , s.c. } \|w_2\| = 1 \implies w_{2k} = \frac{\operatorname{cog}(\hat{\varepsilon}_1, \hat{U}_{(1)k})}{\sqrt{\sum_{k=1}^{K} \operatorname{cog}^2(\hat{\varepsilon}_1, \hat{U}_{(1)k})}}.$$

On utilise à présent les composantes t_1 et t_2 pour établir un lien entre y et les régresseurs x_k :

$$t_2 = \sum_{k=1}^{K} w_{2k} \hat{U}_{(1)k} \implies y = \hat{c}_1 t_1 + \hat{c}_2 t_2 + \hat{\varepsilon}_2 \text{ .}$$

La validation croisée permet de savoir si t_2 est significative.

• **Étape** h **:** Les régressions partielles sont réitérées en ajoutant l'influence de t_{h-1} :

$$R(x_k) = \beta t_1 + \cdots + \gamma t_{h-1} + \hat{U}_{(h-1)k}\ ,\ \forall k = 1, \cdots, K.$$

D'où, après maximisation :

$$w_{hk} = \frac{\mathrm{cog}(\hat{\varepsilon}_{h-1}, \hat{U}_{(h-1)k})}{\sqrt{\sum_{k=1}^{K} \mathrm{cog}^2(\hat{\varepsilon}_{h-1}, \hat{U}_{(h-1)k})}}\ ,$$

$$t_h = \sum_{k=1}^{K} w_{hk} \cdot \hat{U}_{(h-1)k} \implies y = \alpha_2 + c_1 t_1 + \cdots + c_h t_h + \varepsilon_h\ .$$

La procédure s'arrête lorsque la validation croisée indique que la composante t_h n'est pas significative. L'algorithme Gini-PLS1 est valable si toutes les composantes t_h et t_l sont orthogonales, $\forall h \neq l$.

La validation croisée permet de trouver le nombre optimal $h > 1$ de composantes à retenir. Pour tester une composante t_h, on calcule la prédiction du modèle avec h composantes comprenant l'observation i, $\hat{y}_{h_i}$, puis sans l'observation i, $\hat{y}_{h_{(-i)}}$. L'opération est répétée pour tout i variant de 1 à n : on enlève à chaque fois l'observation i et on ré-estime le modèle. [9] Pour mesurer la robustesse du modèle, on mesure l'écart entre la variable prédite et la variable observée :

$$PRESS_h = \sum_{i=1}^{n} \left(y_i - \hat{y}_{h_{(-i)}} \right)^2\ .$$

La somme des carrés résiduels obtenue avec le modèle à $(h-1)$ composantes est :

$$RSS_{h-1} = \sum_{i=1}^{n} \left(y_i - \hat{y}_{(h-1)_i} \right)^2\ .$$

Le critère RSS_h (Residual Sum of Squares) du modèle à h composantes et $PRESS_h$ (PRedicted Error Sum of Squares) sont comparés. Leur rapport permet de savoir si le modèle avec la composante t_h améliore la prédictibilité du modèle :

$$Q_h^2 = 1 - \frac{PRESS_h}{RSS_{h-1}}\ .$$

9. Les observations peuvent être éliminées par blocs *Cf.* Tenenhaus (1998), p. 77.

La composante t_h est retenue si : $\sqrt{PRESS_h} \leq 0,95\sqrt{RSS_h}$. Autrement dit, lorsque $Q_h^2 \geq 0,0975 = (1 - 0,95^2)$, la nouvelle composante t_h est significative, elle améliore la prévision de la variable y. Pour la significativité de la composante t_1, on utilise :

$$RSS_0 = \sum_{i=1}^{n} (y_i - \bar{y})^2 \ .$$

1.3.3 Régressions Gini-PLS généralisée

Schechtman & Yitzhaki [2003] ont récemment généralisé l'opérateur co-Gini afin d'imposer plus ou moins de poids en queue de distribution. Notons $r_k = (R_\downarrow(x_{1k}), \ldots, R_\downarrow(x_{nk}))$ le vecteur rang décroissant de la variable x_k, autrement dit, le vecteur qui assigne le rang le plus petit (1) à l'observation dont la valeur est la plus importante x_{ik} :

$$R_\downarrow(x_{ik}) := \begin{cases} n+1-\#\{x \leq x_{ik}\} & \text{pas d'observation similaire} \\ n+1-\frac{\sum_{i=1}^{p}\#\{x \leq x_{ik}\}}{p} & \text{si } p \text{ observations similaires } x_{ik}. \end{cases}$$

L'opérateur co-Gini est généralisé grâce au paramètre ν :

$$\text{cog}_\nu(x_\ell, x_k) := -\nu\text{cov}(x_\ell, r_k^{\nu-1});\ \nu > 1. \tag{1.3}$$

Afin de bien comprendre le rôle de l'opérateur co-Gini, revenons sur la mesure du coefficient de corrélation linéaire généralisé au sens de Gini :

$$GC_\nu(x_\ell, x_k) := \frac{-\nu\text{cov}(x_\ell, r_k^{\nu-1})}{-\nu\text{cov}(x_\ell, r_\ell^{\nu-1})}\ ;\ GC_\nu(x_k, x_\ell) := \frac{-\nu\text{cov}(x_k, r_\ell^{\nu-1})}{-\nu\text{cov}(x_k, r_k^{\nu-1})}.$$

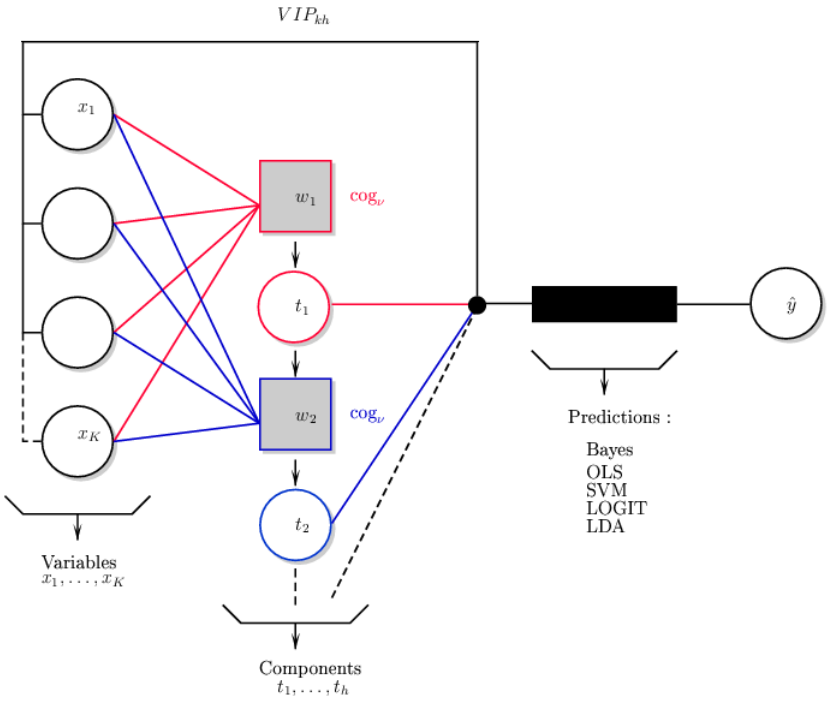
Property 1 – Schechtman & Yitzhaki [2003] :
(i) $GC_\nu(x_\ell, x_k) \leq 1$.
(ii) *Si les variables x_ℓ et x_k sont indépendantes, pour tout $k \neq \ell$, alors* $GC_\nu(x_\ell, x_k) = GC_\nu(x_k, x_\ell) = 0$.
(iii) *Une transformation monotone des données φ n'affecte pas le coefficient de corrélation,* $GC_\nu(x_\ell, \varphi(x_k)) = GC_\nu(x_\ell, x_k)$.
(iv) *Pour une transformation linéaire φ,* $GC_\nu(\varphi(x_\ell), x_k) = GC_\nu(x_\ell, x_k)$ [*comme le coefficient de corrélation de Pearson*].
(v) *Si x_k et x_ℓ sont deux variables échangeables à une transformation linéaire près, alors* $GC_\nu(x_\ell, x_k) = GC_\nu(x_k, x_\ell)$.

Le rôle de l'opérateur co-Gini peut être expliqué de la manière suivante. Lorsque $\nu \to 1$, la variabilité des variables est atténuée de telle sorte que $\text{cog}_\nu(x_k, x_\ell)$ tend vers zéro (même si les variables x_k et x_ℓ sont fortement corrélées). Au contraire, si $\nu \to \infty$ alors $\text{cog}_\nu(x_k, x_\ell)$ permet de se focaliser sur les queues de distribution x_ℓ. Comme le montrent Olkin & Yitzhaki [1992], l'emploi de l'opérateur co-Gini atténue la présence d'outliers, du fait que le vecteur rang agit comme un instrument dans la régression de y sur X (régression par variables instrumentales).

Ainsi, en proposant une régression Gini-PLS basée sur le paramètre ν, nous pouvons calibrer la puissance du débruitage grâce à l'opérateur co-Gini qui va localiser le bruit dans la distribution. Cette régression Gini-PLS généralisée devient une régression Gini-PLS régularisée où le paramètre ν joue le rôle de paramètre de régularisation.

1.3.3.1 L'algorithme Gini-PLS généralisé

Dans ce qui suit nous généralisons la régression Gini-PLS de Mussard & Souissi-Benrejab [2018] avec renforcement du pouvoir de débruitage par l'intermédiaire du paramètre ν.



La première étape consiste à trouver des poids de débruitage associés à chaque variable x_k afin d'en déduire la première composante t_1 (ou première variable latente). Cette opération est bouclée jusqu'à la composante t_{h^*}, où h^* est le nombre optimal de variable latentes. Ainsi, le modèle est estimé :

$$y = \sum_{h=1}^{h^*} c_h t_h + \varepsilon_h. \tag{1.4}$$

La statistique VIP_{hj} est mesurée afin de sélectionner la variable x_j qui a l'impact significatif le plus important sur y estimé. Les variables les plus significatives sont celles dont $VIP_{hj} > 1$ avec :

$$VIP_{hj} := \sqrt{\frac{K \sum_{\ell=1}^{h} Rd(y; t_\ell) w_{\ell j}^2}{Rd(y; t_1, \ldots, t_h)}}$$

et

$$Rd(y; t_1, \ldots, t_h) := \frac{1}{K} \sum_{\ell=1}^{h} \text{cor}^2(y, t_\ell) =: \sum_{\ell=1}^{h} Rd(y; t_\ell).$$

où $\text{cor}^2(y, t_\ell)$ est le coefficient de corrélation de Pearson entre y et la composante t_ℓ. Cette information est rétro-propagée dans le modèle (une seule fois) afin d'obtenir les variables latentes t_{h^*} et leurs coefficients estimés $\hat{c}_{h^*}$ sur les données d'entraînement. La variable cible y est ensuite prédite grâce à (1.4). Cette prévision est comparée aux modèles standards SVM, LOGIT, Bayes et LDA lorsque les données tests sont projetées dans le sous-espace $\{t_1, \ldots, t_{h^*}\}$.

1.3.3.2 L'algorithme LOGIT-Gini-PLS généralisé

Comme nous le constatons dans l'algorithme Gini-PLS généralisé que nous avons proposé dans le section précédente, les poids w_j proviennent de l'opérateur co-Gini appliqué à une variable booléenne $y \in \{0; 1\}$. Afin de trouver les poids w_j qui maximisent le lien entre les variables x_j et la variable cible y, nous proposons d'utiliser la régression LOGIT, autrement dit, une sigmoïde qui est mieux adaptée aux variables booléennes. Ainsi, dans chaque étape de la régression Gini-PLS nous remplaçons la maximisation du co-Gini par la mesure de la probabilité conditionnelle suivante :

$$\mathbb{P}(y_i = 1 / X = X_i) = \frac{\exp\{X_i \beta\}}{1 + \exp\{X_i \beta\}} \tag{LOGIT}$$

Algorithme 2 : Gini-PLS Généralisé

Résultat : Prédiction du juge $y = 0; 1$

répéter
  répéter
    $\max \text{cog}_v(y, w_h X)$ s.t. $\|w_h\| = 1 \Longrightarrow$ poids w_h de X ;
    MCO équation : $y = \sum_h c_h t_h + \varepsilon_h$;
    MCO équation : $R(x_j) = \sum_h \beta_h t_h + \epsilon_k \; \forall k = 1, \ldots, K$;
    $X := (\hat{\epsilon}_1, \ldots, \hat{\epsilon}_K)$ $y := \hat{\varepsilon}_h$;
  jusqu'à $h = 10$ $[h = h + 1]$;
  Mesurer VIP_{kh}, Q_h^2 ;
  Sélectionner le nombre optimal de composantes h^* ;
jusqu'à $\nu = 14$ $[\nu = \nu + 2]$;
Déduire le paramètre optimal ν^* qui minimise l'erreur ;
retourner Prédiction $\hat{y}$ avec Gini-PLS (h^*, ν^*) ;
retourner Prédiction $\hat{y}$ avec SVM, LOGIT, Bayes, LDA sur les composantes $(t_1, \ldots, t_{h^*})$;

où X_i est la i-ème ligne de la matrice X (observation des caractéristiques/-dimensions de la décision juridique i). L'estimation du vecteur β se fait par maximum de vraisemblance. On en déduit alors les pondérations w_j :

$$w_j = \frac{\beta_j}{\|\beta\|}$$

L'algorithme LOGIT-Gini-PLS généralisé est donc le suivant :

1.4 Expérimentations et résultats

Nous discutons ici les performances de divers algorithmes populaires et l'impact de la quantité et du déséquilibre des données, de l'heuristique, et de la restriction explicite des documents aux passages relatifs à la catégorie de demandes, ainsi que leur capacité à faire abstraction des autres demandes du document. Ces expériences visent aussi à comparer l'efficacité du Gini-Logit-PLS par rapport à d'autres analyses discriminantes. Comme Im *et al.* [2017], différentes combinaisons d'algorithmes de classification et méthodes de pondération sont comparées ; ce qui représente un total de 8 algorithmes * 11 métriques globales * 5 métriques locales = 440 configurations (+ 2 algorithmes i.e. FastText et NBSVM). La méthode Gini-Logit-PLS réalisant une projection dans un espace de petite dimension, elle a été comparée aussi à d'autres méthode de réduction de dimension.

Algorithme 3 : LOGIT-Gini-PLS Généralisé

Résultat : Prédiction du juge $y = 0; 1$

répéter
  répéter
    LOGIT équation : $\Longrightarrow$ poids w_j de X ;
    MCO équation : $y = \sum_h c_h t_h + \varepsilon_h$;
    $X := (\hat{\epsilon}_1, \ldots, \hat{\epsilon}_K)$ $y := \hat{\varepsilon}_h$;
  jusqu'à $h = 10$ $[h = h + 1]$;
  Mesurer VIP_{kh}, Q_h^2 ;
  Sélectionner le nombre optimal de composantes h^* ;
jusqu'à $\nu = 14$ $[\nu = \nu + 2]$;
Déduire le paramètre optimal ν^* qui minimise l'erreur ;
retourner Prédiction $\hat{y}$ avec Gini-PLS (h^*, ν^*) ;
retourner Prédiction $\hat{y}$ avec SVM, LOGIT, Bayes, LDA sur les composantes $(t_1, \ldots, t_{h^*})$;

1.4.1 Protocole d'évaluation

Deux métriques d'évaluation sont utilisées : la précision et la F_1-mesure. Pour tenir compte du déséquilibre entre les classes, la moyenne macro est préférée. Il s'agit de l'agrégation de la contribution individuelle de chaque classe : $F_{1macro} = \frac{2 \times P_{macro} \times R_{macro}}{P_{macro} + R_{macro}}$, où les macro-moyennes de la précision (P_{macro}) et du rappel (R_{macro}) sont calculées en fonction des nombres moyens de vrais positifs ($\overline{TP}$), faux positifs ($\overline{FP}$), et faux négatifs ($\overline{FN}$) comme suit [Van Asch, 2013] : $P_{macro} = \frac{\overline{TP}}{\overline{TP} + \overline{FP}}$, $R_{macro} = \frac{\overline{TP}}{\overline{TP} + \overline{FN}}$.

Les données utilisées sont une restriction, des données du chapitre précédent, aux documents n'ayant qu'une seule demande annotée pour chacune des catégories de demande. Le déséquilibre entre les classes est illustrée par la figure 1.3. En effet, les demandes sont en majorité rejetées pour les catégories ACPA, CONCDEL, DANAIS et STYX. Le contraire est observé pour DCPPC, et le rapport est légèrement équilibré pour DORIS.

L'efficacité des algorithmes dépend souvent des méta-paramètres dont il faut déterminer des valeurs optimales. Scikit-learn implémente deux stratégies de recherche des ces valeurs : RandomSearch et GridSearch. Malgré la rapidité de la méthode RandomSearch, elle est non déterministe et les valeurs qu'elle trouve donnent une prédiction moins précise que les valeurs par défaut. Idem pour la méthode GridSearch, qui est très lente, et donc peu pratique face au grand nombre de configurations à évaluer. Par conséquent, les valeurs utilisées pour les expérimentations sont les valeurs par défaut définies par Scikit-learn (Tableau 1.2).

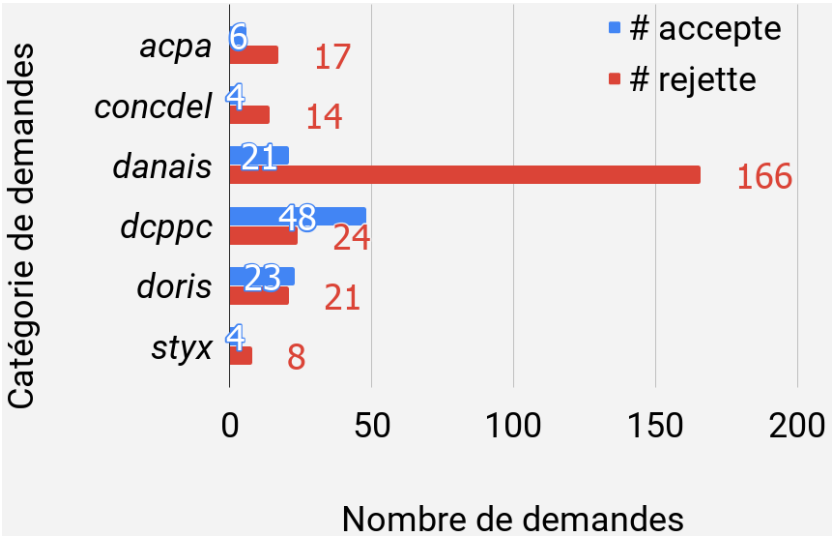


Figure 1.3 – Répartition des documents entre *accepte* et *rejette*.

algorithmes	hyper-paramètres
SVM	$C = 1.0; \gamma = \frac{1}{n\_features * var(X)}; noyau = RBF$ (fonction de base radiale)
KNN	$k = 5, weights =' uniform', algorithm =' auto'$
LDA	$solver =' svd', n\_components = 10$ [10]
QDA	
Arbre	critère de séparation='gini'
NBSVM	$--ngram = 123, linearSVM,$
FastText	-minCount=5, -wordNgrams=1, -lr=0.05,
Gini-PLS	$h = n\_components = 10$
Logit-PLS	$h = n\_components = 10$
Gini-Logit-PLS	$h = n\_components = 10; \nu = 14$

Tableau 1.2 – Valeurs utilisées pour les méta-paramètres des algorithmes de classification.

1.4.2 Classification de l'ensemble du document

En représentant l'ensemble du document à l'aide de diverses représentations vectorielles, les algorithmes sont comparés avec les représentations qui leurs sont optimales. On remarque d'après les résultats du Tableau 1.3 que les arbres sont en moyenne meilleurs sur l'ensemble des catégories même si en moyenne la F_1-mesure moyenne est limité à 0.668. Les résultats des extensions du PLS ne sont pas très éloignées de ceux des arbres avec des différences de F_1 à moins de 0.1 (si on choisit le bon schéma de "vectorisation").

Les scores F_1 moyens des algorithmes NBSVM et FastText n'excédent en général pas 0.5 malgré qu'ils soient spécialement conçus pour les textes. On peut estimer qu'ils sont très sensibles au déséquilibre des données

ajouter les F_1 ou erreur de rejette et de accepte
mettre aussi en avant les analyses discriminantes comme réducteur de dimension et comme classifieurs

Vecteur	**algorithme**	F_1	**min**	**Cat. min**	**max**	**Cat. max**	F_1 **- 1er**F_1	**max - min**	**rang**
GSS*TF	Arbre	0.668	0.5	doris	0.92	dcppc	0	0.42	1
AVG-G*TF	LogitPLS	0.648	0.518	danais	0.781	dcppc	0.02	0.263	13
AVG-G*TF	StandardPLS	0.636	0.49	danais	0.836	dcppc	0.032	0.346	24
DELTADF*TF	GiniPLS	0.586	0.411	danais	0.837	dcppc	0.082	0.426	169
DELTADF*TF	GiniLogitPLS	0.578	0.225	styx	0.772	dcppc	0.09	0.547	220
-	NBSVM	0.494	0.4	styx	0.834	dcppc	0.174	0.434	
-	FastText	0.412	0.343	doris	0.47	danais	0.256	0.127	

Tableau 1.3 – Comparaison des algorithmes sur une représentation globale des documents pour la détection du sens du résultat.

entre les catégories (plus de rejets que d'acceptations), soit il est plus difficile de détecter l'acceptation des demandes. En effet, ces algorithmes classent toutes les données test avec le label (sens) majoritaire i.e. le rejet, et par conséquent, ils ne détectent quasiment pas d'acceptation de demande. Le cas des catégories DORIS et DCPPC pour le NBSVM ($F_{1macro} = 0.834$) tend à démontrer la forte sensibilité aux cas négatifs de ces algorithmes puisque même avec presque autant de labels "accepte" que "rejette", la F_1-mesure de "rejette" est toujours supérieure à celle de "accepte" (Tableau 1.4).

Cat. Dmd.	**Algo.**	**Préc.**	**Préc. équi.**	**err-0**	**err-1**	$F_1(0)$	$F_1(1)$	F_{1macro}
dcppc	nbsvm	0.875	0.812	0.375	0	0.752	0.916	**0.834**
danais	fasttext	0.888	0.5	0	1	0.941	0	0.47
danais	nbsvm	0.888	0.5	0	1	0.941	0	0.47
concdel	fasttext	0.775	0.5	0	1	0.853	0	0.437
concdel	nbsvm	0.775	0.5	0	1	0.873	0	0.437
acpa	fasttext	0.745	0.5	0	1	0.853	0	0.426
acpa	nbsvm	0.745	0.5	0	1	0.853	0	0.426
doris	nbsvm	0.5	0.492	0.85	0.167	0.174	0.63	0.402
dcppc	fasttext	0.667	0.5	1	0	0.8	0	0.4
styx	fasttext	0.667	0.5	1	0	0.8	0	0.4
styx	nbsvm	0.667	0.5	0	1	0.8	0	0.4
doris	fasttext	0.523	0.5	1	0	0	0.686	0.343

$0 ==' accepte'; 1 ==' rejette'$

Tableau 1.4 – Détails des résultats de FastText et NBSVM.

1.4.3 Réduction du document aux régions comprenant le vocabulaire de la catégorie

Etant donné que les décisions portent sur plusieurs catégories de demande, nous avons expérimenté la restriction du document aux passages

comprenant du vocabulaire de la catégorie d'intérêt : demande, résultat, résultat antérieur (resultat_a), paragraphes dans les motifs (motifs). Les combinaisons passages-représentation vectorielle-algorithme sont comparées dans le Tableau 1.5. Les résultats s'améliorent énormément avec les réductions, sauf pour la catégorie DORIS. La meilleure restriction combine les passages comprenant le vocabulaire de la catégorie dans la section Litige (demande et résultat antérieur), dans la section Motifs (contexte), et dans la section Dispositif (résultat).

Catégorie	zone	Vecteur (pondération)	classifieur	F_1
acpa	**demande_resultat_a_resultat_context**	**DBIDF*TF**	**Tree**	**0.846**
	litige_motifs_dispositif	DELTADF*TF	StandardPLS	0.697
	litige_motifs_dispositif	AVERAGEGlobals*TF	LogitPLS	0.683
concdel	**litige_motifs_dispositif**	**GSS*TF**	**Tree**	**0.798**
	motifs	IDF*TF	GiniLogitPLS	0.703
	context	DBIDF*LOGAVE	StandardPLS	0.657
danais	**demande_resultat_a_resultat_context**	**CHI2*AVERAGELocals**	**Tree**	**0.813**
	demande_resultat_a_resultat_context	AVERAGEGlobals*ATF	LogitPLS	0.721
	demande_resultat_a_resultat_context	AVERAGEGlobals*ATF	StandardPLS	0.695
dcppc	**demande_resultat_a_resultat_context**	**CHI2*TF**	**Tree**	**0.985**
	demande_resultat_a_resultat_context	CHI2*TF	LogitPLS	0.94
	litige_motifs_dispositif	MARASCUILO*TP	StandardPLS	0.934
doris	**litige_motifs_dispositif**	**DSIDF*TP**	**GiniPLS**	**0.806**
	litige_motifs_dispositif	DSIDF*TP	GiniLogitPLS	0.806
	litige_motifs_dispositif	IG*ATF	StandardPLS	0.772
styx	**motifs**	**DSIDF*TF**	**Tree**	**1**
	demande_resultat_a_resultat_context	DSIDF*LOGAVE	GiniLogitPLS	0.917
	litige_motifs_dispositif	RF*TF	GiniPLS	0.833

Tableau 1.5 – Détection du sens du résultat : Comparaison des réductions du document.

1.5 Conclusion

L'étude de ce chapitre tente de simplifier l'extraction du sens du résultat rendu par les juges sur une demande de catégorie donnée. Elle a consisté à formuler le problème comme une tâche de classification de documents. On évite ainsi de passer par la détection ad-hoc [11] des passages et données à l'aide de termes-clés qui est un inconvénient de la méthode à règles du chapitre précédent car elle n'est peut-être pas généralisable à tous types de décisions (i.e. il pourrait être nécessaire d'établir de nouvelles listes de mots-clés pour d'autres domaines). Au total dix algorithmes de classification on été expérimentés sur 55 méthodes de représentations vectorielles de texte. Nous avons remarqué que les résultats de classifi-

11. i.e. spécialement conçu pour nos données.

cation sont principalement influencés par 3 caractéristiques de nos données. Tout d'abord, le très faible nombre d'exemples d'entraînement défavorise certains algorithmes (sensibilité aux valeurs aberrantes ou *outliers*), comme par exemple FastText qui nécessite plusieurs milliers d'exemples pour mettre à jour le pas du gradient (*learning rate*). Ensuite, le fort déséquilibre entre les classes ("accepte" vs. "rejette") rend difficile la reconnaissance de la classe minoritaire qui est généralement la classe "accepte". Le fort gap entre les erreurs sur "rejette" et celles sur "accepte", ainsi que les bons résultats obtenus sur DCPPC en sont la preuve. Enfin, la présence d'autres catégories de demande dans le document dégrade l'efficacité de la classification parce que les algorithmes ne parviennent pas seuls à retrouver les éléments en rapport direct avec la catégorie choisie. Ceci est démontré par l'impact positif de la restriction du contenu à classer à certains passages particuliers, même si la restriction adéquate est fonction de la catégorie.

Au final, les arbres de décision sont adaptés pour la tâche, mais l'usage du Gini-PLS et du Gini-Logit-PLS permet d'obtenir des performances assez proches de celles des arbres. Il serait intéressant de combiner ces variantes de l'analyse PLS, à d'autres comme le Sparse-PLS qui pourrait peut-être aider à résoudre le problème de vecteurs/matrices creuses dont sont victimes les représentations vectorielles de texte. Il existe aussi un grand nombre d'architectures neuronales pour la classification de document et de très grands nombres de métriques de pondération de termes pour la représentation des textes, mais aucune ne semble s'adapter à toutes les catégories. Par conséquent, une étude sur l'usage des représentations par plongement sémantique comme Word2Vec [Mikolov *et al.* , 2013], Sent2Vec [Pagliardini *et al.* , 2018] ou Doc2Vec [Le & Mikolov, 2014] serait intéressante.

Bibliographie

Aggarwal, Charu C., Hinneburg, Alexander, & Keim, Daniel A. 2001. On the surprising behavior of distance metrics in high dimensional space. *Pages 420–434 of : International Conference on Database Theory*. Springer.

Amami, Rimah, Ayed, Dorra Ben, & Ellouze, Noureddine. 2013. Practical Selection of SVM Supervised Parameters with Different Feature Representations for Vowel Recognition. *International Journal of Digital Content Technology and its Applications (JDCTA)*, **7**(9).

Bazzoli, Caroline, & Lambert-Lacroix, Sophie. 2018. Classification based on extensions of LS-PLS using logistic regression : application to clinical and multiple genomic data. *BMC bioinformatics*, **19**(1), 314.

Ben-Hur, Asa, & Weston, Jason. 2010. A User's Guide to Support Vector Machines. *Chap. 13, pages 223–239 of : Data Mining Techniques for the Life Sciences*. Totowa, NJ : Humana Press.

Breiman, Leo. 2001. Random Forests. *Machine Learning*, **45**(1), 5–32.

Breiman, Leo, Friedman, J. H., Olshen, R. A., & Stone, C. J. 1984. *Classification and Regression Trees*. Statistics/Probability Series. Belmont, California, U.S.A. : Wadsworth Publishing Company.

Cortes, Corinna, & Vapnik, Vladimir. 1995. Support-vector networks. *Machine Learning*, **20**(3), 273–297.

Cover, Thomas, & Hart, Peter. 1967. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, **13**(1), 21–27.

Dong, Yan-Shi, & Han, Ke-Song. 2005. Boosting SVM classifiers by ensemble. *Pages 1072–1073 of : Special Interest Tracks And Posters Of The 14th International Conference On World Wide Web*. ACM.

Duda, Richard O., Hart, Peter E., *et al.* . 1973. *Pattern Classification And Scene Analysis*. Vol. 3. New York : John Wiley & Sons.

Dudani, Sahibsingh A. 1976. The Distance-weighted k-Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-6**(4), 325–327.

Durif, Ghislain, Modolo, Laurent, Michaelsson, Jakob, Mold, Jeff E., Lambert-Lacroix, Sophie, & Picard, Franck. 2017. High dimensional classification with combined adaptive sparse PLS and logistic regression. *Bioinformatics*, **34**(3), 485–493.

Elsaadawy, AbdAllah, Torki, Marwan, & Ei-Makky, Nagwa. 2018. A text classifier using weighted average word embedding. *Pages 151–154 of : 2018 international japan-africa conference on electronics, communications and computations (jac-ecc)*. IEEE.

Fisher, Ronald A. 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**(2), 179–188.

Ghojogh, Benyamin, & Crowley, Mark. 2019. *Linear and quadratic discriminant analysis : Tutorial*. arXiv preprint arXiv :1906.02590 [stat.ML].

Gou, Jianping, Xiong, Taisong, & Kuang, Yin. 2011. A Novel Weighted Voting for K-Nearest Neighbor Rule. *JCP*, **6**(5), 833–840.

Grave, E., Mikolov, T., Joulin, A., & Bojanowski, P. 2017. Bag of tricks for efficient text classification. *Pages 427–431 of : Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Im, Chan Jong, Mandl, Thomas, *et al.* . 2017. Text Classification for Patents : Experiments with Unigrams, Bigrams and Different Weighting Methods. *International Journal of Contents*, **13**(2).

Joulin, Armand, Grave, Edouard, Bojanowski, Piotr, Douze, Matthijs, Jégou, Hérve, & Mikolov, Tomas. 2016. Fasttext. zip : Compressing text classification models. *arxiv preprint arxiv :1612.03651*.

Kittler, Josef, Hater, Mohamad, & Duin, Robert P.W. 1996. Combining classifiers. *Pages 897–901 of : Proceedings of 13th international conference on pattern recognition*, vol. 2. IEEE.

Kuncheva, Ludmila I. 2004. *Combining pattern classifiers : methods and algorithms*. John Wiley & Sons.

Lacroux, Alain. 2011. Les avantages et les limites de la méthode «Partial Least Square »(PLS) : une illustration empirique dans le domaine de la GRH. *Revue de gestion des ressources humaines*, **80**(2), 45–64.

Le, Quoc, & Mikolov, Tomas. 2014. Distributed representations of sentences and documents. *Pages 1188–1196 of : International conference on machine learning*.

Liu, Yushu, & Rayens, William. 2007. PLS and dimension reduction for classification. *Computational Statistics*, **22**(2), 189–208.

Manning, Christopher D, Raghavan, Prabhakar, & Schütze, Hinrich. 2009. Scoring, term weighting and the vector space model. *Chap. 6, pages 109–133 of : Introduction to information retrieval*. Cambridge : Cambridge university press.

McLachlan, Geoffrey J. 1992. *Discriminant analysis and statistical pattern recognition*. John Wiley & Sons.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, & Dean, Jeffrey. 2013. Efficient estimation of word representations in vector space. *In : Proceedings of the International Conference on Learning Representations (ICLR)*.

Mohammad, Fahim. 2018. Is preprocessing of text really worth your time for toxic comment classification ? *Pages 447–453 of : Proceedings on the international conference on artificial intelligence (icai)*. The Steering Committee of The World Congress in Computer Science, Computer ....

Mussard, Stéphane, & Souissi-Benrejab, Fattouma. 2018. Gini-PLS Regressions. *Journal of Quantitative Economics*, April, 1–36.

Olkin, Ingram, & Yitzhaki, Shlomo. 1992. Gini regression analysis. *International Statistical Review/Revue Internationale de Statistique*, 185–196.

Pagliardini, Matteo, Gupta, Prakhar, & Jaggi, Martin. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. *In : NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Quinlan, J. Ross. 1993. C4.5 : Programming for machine learning. *Morgan Kauffmann*, **38**, 48.

Raschka, Sebastian. 2014. *Naive Bayes and Text Classification I : Introduction and Theory*. preprint arXiv :1410.5329 [cs.LG].

Rish, Irina. 2001. An Empirical Study Of The Naive Bayes Classifier. *Pages 41–46 of : IJCAI 2001 Workshop On Empirical Methods In Artificial Intelligence*, vol. 3. IBM New York.

Ruparel, Nidhi H, Shahane, Nitin M, & Bhamare, Devyani P. 2013. Learning from small data set to build classification model : A survey. *Internationla Journal of Computer Applications*, **975**(8887), 23–26.

Salton, Gerard, & Buckley, Christopher. 1988. Term-weighting Approaches In Automatic Text Retrieval. *Information Processing & Management*, **24**(5), 513–523.

Schechtman, Edna, & Yitzhaki, Shlomo. 2003. A family of correlation coefficients based on the extended Gini index. *The Journal of Economic Inequality*, **1**(2), 129–146.

Shirsath, Ashlesha. 2017. Two stage smart crawler with nbsvm classifier. *International research journal of engineering and technology*, **4**(07), 2051–2054.

Singh, Sonia, & Gupta, Priyanka. 2014. Comparative study ID3, CART and C4.5 decision tree algorithm : a survey. *International Journal of Advanced Information Science and Technology (IJAIST)*, **27**, 97–103.

Sohangir, Sahar, & Wang, Dingding. 2017. Improved sqrt-cosine similarity measurement. *Journal of Big Data*, **4**(1), 25.

Tenenhaus, Michel. 2005. La regression logistique PLS. *Chap. 12, pages 263–276 of :* Droesbeke, Jean-Jacques and Lejeune, Michel and Saporta, Gilbert (ed), *Modèles statistiques pour données qualitatives*. Editions Technip.

Tulyakov, Sergey, Jaeger, Stefan, Govindaraju, Venu, & Doermann, David. 2008. Review of classifier combination methods. *Pages 361–386 of : Machine learning in document analysis and recognition*. Springer.

Van Asch, Vincent. 2013. *Macro- and micro-averaged evaluation measures*. Tech. rept. Computational Linguistics & Psycholinguistics (CLiPS), Belgium. `https://pdfs.semanticscholar.org/1d10/6a2730801b6210a67f7622e4d192bb309303.pdf`.

Vapnik, Vladimir N. 1995. *The Nature of Statistical Learning Theory*. Springer.

Wang, Sida, & Manning, Christopher D. 2012. Baselines and bigrams : Simple, good sentiment and topic classification. *Pages 90–94 of : Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics : Short Papers-Volume 2*. Association for Computational Linguistics.

Wold, Herman. 1966. Estimation of principal components and related models by iterative least squares. *Multivariate Analysis*, 391–420.

Zeng, Xue-Qiang, Wang, Ming-Wen, & Nie, Jian-Yun. 2007. Text classification based on partial least square analysis. *Pages 834–838 of : Proceedings of the 2007 ACM symposium on Applied computing*. ACM.