**UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS**

Département d'informatique et d'ingénierie

# CLASSIFICATION SUPERVISÉE DE DOCUMENTS

# ÉTUDE COMPARATIVE

ESSAI PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN SCIENCES ET TECHNOLOGIES DE

L'INFORMATION

PAR

**LAHLOU OUCHIHA**

JANVIER 2016

## Jury d'évaluation

**Président du Jury :** **Dr. Mohand Said Allili**

**Directeur de recherche :** **Dr. Larbi Talbi**

# Dédicace

À la mémoire de ma petite sœur "Mina" qui est partie à la fleur de l'âge.

À tous ceux que j'ai de précieux dans ce monde, ma femme et mes deux enfants.

# Remerciements

Je remercie mon directeur de recherche le professeur Larbi Talbi pour avoir supervisé ce modeste travail et de m'avoir procuré un stage au sein de la Startup "Gnowit", je remercie également toute l'équipe de "Gnowit" pour leur accueil et soutient tout au long de mon stage, en particulier "Andrew" et "Shahzad".

Merci pour le professeur  Mohand Said  Allili qui m'a honoré en acceptant de présider le jury et ainsi évaluer mon travail.

Merci également pour les professeurs Rokia Missaoui, Stéphane Gagnon et Karim El Guemhioui.

# Table des matières

## Liste des figures

## Liste des tableaux

## Liste des abréviations, sigles et acronymes

MVS : Machine à Vecteur de support.

CNB : Classificateur Naïf Bayes.

AD : Arbre de Décision.

PDF : Portable Document Format.

XML : Extensible Markup Language.

HTML : Hypertext Markup Language.

TF : Term Frequancies.

NT : Nombre de Termes.

ST : Somme de Termes.

IDF : Inverse Documents Frequancies.

N_DOC : Nombre de Documents.

DOC_T : Nombre de Documents où le Terme est apparu.

VP : Vrais Positifs.

FP : Faux Positifs.

VN : Vrais Négatifs.

FN : Faux Négatifs.

KNIME : Konstanz Information Miner.

GPL : General Public License.

SDK : Software Development Kit.

RAM : Random Access Memory.

WEKA : Waikato Environment for Knowledge Analysis.

RBF : Radial Basis Function.

MDL : Minimum Description Length

ACP : Analyse à Composantes Principales.

ADL : Analyse Discriminante Linéaire.

**Résumé :**

La classification(catégorisation) de texte est plus qu'essentielle à l'ère d'internet et des Big data. Que ce soit dans la recherche documentaire, à l'image des moteurs de recherche qui s'émergent de toutes parts, ou bien de la catégorisation de documents texte, du classement de mails, du ranking des pages web, le principe reste le même mais les techniques différent. Et pour chaque besoin il y a une ou plusieurs solutions adéquates.

Qu'il s'agit de l'apprentissage supervisé(classification) ou non supervisé(clustering), le text minig ( Fouille de texte) est soumis à un ensemble de règles et à des algorithmes biens définis, parmi ces derniers, issus d'abord du domaine de l'intelligence artificielle et ensuite du data minig, qui sont souvent utilisés en text mining, nous pouvons citer : les arbres de décision, les réseaux de neurones, les k-moyens, les k les plus proches, le classificateur de bayes naïf CNB, la Machine à Vecteur de Support MVS et d'autres. Dans cet essai nous nous sommes intéressés plus exactement à la différence entre trois algorithmes à savoir : les arbres de décision, la machine à vecteur de support et le classificateur Naïf Bayes. nous aimerions d'abord définir les différentes étapes de la classification de texte ( apprentissage supervisé ), ensuite nous allons étudier la manière avec laquelle ces algorithmes réagissent, sachant que les données sont les mêmes, pour chacun d'entre d'eux, et elles ont subi le même processus de prétraitement. Des mesures de score, tel que la matrice de confusion ou bien la F-Mesure sont utilisés pour illustrer parfaitement, à la fois, le taux d'erreurs de chaque classificateur lors de la phase de test ainsi que sa généralisation en lui proposant de nouvelles données.

## Introduction :

Le text mining est une branche qui étudie la classification ou la catégorisation de textes ou de documents selon des catégories prédéfinies par les experts humains(apprentissage supervisé), ou prédites par le classificateur (apprentissage non supervisé). Dans le processus de classification de texte, les données sont souvent non structurées, c'est à dire, le texte en question peut s'agir d'un document Word, PDF etc., ou semi structurées comme par exemple les documents XML, HTML etc. C'est pour cette raison d'ailleurs que la partie prétraitement (preprocessing) dans ce processus est considérée comme étant l'étape la plus difficile et la plus délicate qui détermine souvent la fiabilité et surtout la généralisation du classificateur. On entend par généralisation, la capacité du classificateur à se comporter de la même manière lors de la phase d'apprentissage qu'avec de nouvelles données, ou presque (avec un taux d'erreurs acceptable).

Notre essai comporte deux parties, la première est un état de l'art de toutes les techniques et les algorithmes que nous avons utilisés. Quant à la seconde, c'est l'étude d'un cas pratique où nous construirons trois classificateurs distincts, ensuite nous mesurerons les taux d'erreurs de chaque classificateur ainsi que son degré de généralisation sur de nouvelles données, qui signifie aussi la robustesse d'un classificateur. Enfin, nous allons conclure sur la base des avantages et des inconvénients de chacun des classificateurs tout en indiquant le plus approprié pour nos données.

# Chapitre 1 - Le text mining

## 1.1 Définition :

«Le text mining regroupe l'ensemble des techniques issues du traitement automatique de la langue et de l'analyse de données permettant de retrouver des informations cachées dans de larges bases de données textuelles.» [1].

Le text mining est considéré comme étant une branche indissociable du data mining, du moment où ce dernier utilise les mêmes algorithmes et les mêmes mécanismes, à la différence prés que le data mining utilise des données structurées, contenues souvent dans des tables prêtes à l'emploi, tandis que le text mining manipule des données d'entrée non structurées (fichier plat), ce qui nécessite une étape de prétraitement cruciale. En somme, le text mining est le processus de découpage d'un texte en unités élémentaires (mots, N-grams, phrases), de calcul des fréquences de chaque unité. Des algorithmes spécifiques déterminent les associations et les similarités entres ces unités, et enfin former des groupes homogènes . «Il s'agit d'imiter la démarche d'une personne lisant un texte en diagonale et soulignant les passages qui l'intéressent.»[2].

## 1.2 Motivation :

Selon Lefébre et Venturi[2], le texte mining trouve ces origines dans trois domaines :

### 1.2.1 L'amélioration de l'interface homme-machine :

L'envie de voir un jour l'homme communiquer avec la machine par le biais de la voix, en se libérant des interfaces standards (clavier, souris ...).

### 1.2.2 La traduction automatique :

Il s'agit de la translation d'un texte d'une langue donnée (français) à une autre langue(anglais). C'est d'ailleurs l'activité qui a bénéficié du plus grand financement, à cause notamment de l'espionnage à l'époque de la guerre froide et.

### 1.2.3 La caractérisation des textes :

La recherche documentaire, avec l'avènement de l'internet et des moteurs de recherche, a propulsé considérablement le text mining .

### 1.2 Contraintes :

Comme tout autre domaine des sciences, le text mining est soumis lui aussi à plusieurs contraintes qui sont, d'ailleurs, essentielles à l'essor de cette discipline, du moment qu'elles favorisent la recherche. Il existe plusieurs contraintes dans la littérature, mais nous nous contentons de citer que quelques unes.

#### 1.2.1 La polysémie :

C'est le fait qu'un même mot peut avoir plusieurs sens. Ainsi le mot "livre" désigne à la fois une unité de mesure de poids, un document écrit, ou bien encore une unité monétaire. Utiliser un tel mot comme descripteur (terme qui discrimine un texte) est fortement déconseillé, du moment qu'il peut assigner à un document trois catégories distinctes.

#### 1.2.2 La redondance sémantique :

La redondance sémantique est le sens unique que peut y avoir plusieurs expressions distinctes, ou les différentes manières avec lesquelles on peut exprimer un sentiment, un fait, ou toute autre chose avec le langage naturel. Cette subtilité du langage naturel est illustrée dans l'exemple suivant: "les États Unis d'Amérique", "le pays de l'oncle Sam" et "le gendarme du monde", ces expressions désignent toutes la même chose, mais lors de la représentation vectorielle ces séquences de mots seront représentées différemment. C'est pourquoi dans certains cas, l'utilisation du mot comme descripteur peut s'avérer superflu.

#### 1.2.3 Le temps d'apprentissage :

Le document vecteur (données d'entrées pour l'algorithme de classification) qu'on génère à partir du corpus et d'une dimension inouïe. Imaginons le nombre de documents dans le corpus et le nombre de mots dans chaque document, au final nous nous retrouvons avec un tableau comportant tous les mots à traiter avec leurs poids respectifs, cela occupera beaucoup de ressources et prendra un temps d'exécution considérable.

#### 1.2.4 Le sur-apprentissage (overfitting) :

Les documents Vecteur de grande dimension engendrent généralement un sur-apprentissage, ça veut dire que les documents d'apprentissage sont bien classés, mais le classificateur réagit mal, et ne se généralise pas aux nouveaux documents. Il existe plusieurs méthodes de réduction de dimension pour parer à ce problème.

### 1.2.5 L'étiquetage :

Ce problème est spécifique à l'apprentissage supervisé, l'étiquetage de l'ensemble d'apprentissage est réalisé par des experts humains. Or, chaque expert a sa propre appréciation d'assigner une catégorie à un document. Ainsi, un document peut être classé dans deux ou plusieurs catégories différentes, selon l'appréciation des experts. En conséquence, le but d'avoir un classificateur idéal est compromis.

# Chapitre 2 - La classification de textes

## 2.1 Introduction:

Il s'agit de créer un classificateur à base d'un corpus de documents étiquetés (généralement à la main), ce corpus est partitionné en deux ensemble : l'ensemble d'apprentissage et celui de test. Cela consiste, en premier temps, à entrainer ce classificateur avec l'ensemble d'apprentissage, et une fois appris, nous testerons son efficacité avec l'ensemble test. Dans certain cas, nous terminerons ce processus par une étape de validation du classificateur avec un ensemble de nouveau documents. Le schéma ci-dessous illustre parfaitement ces différentes étapes.

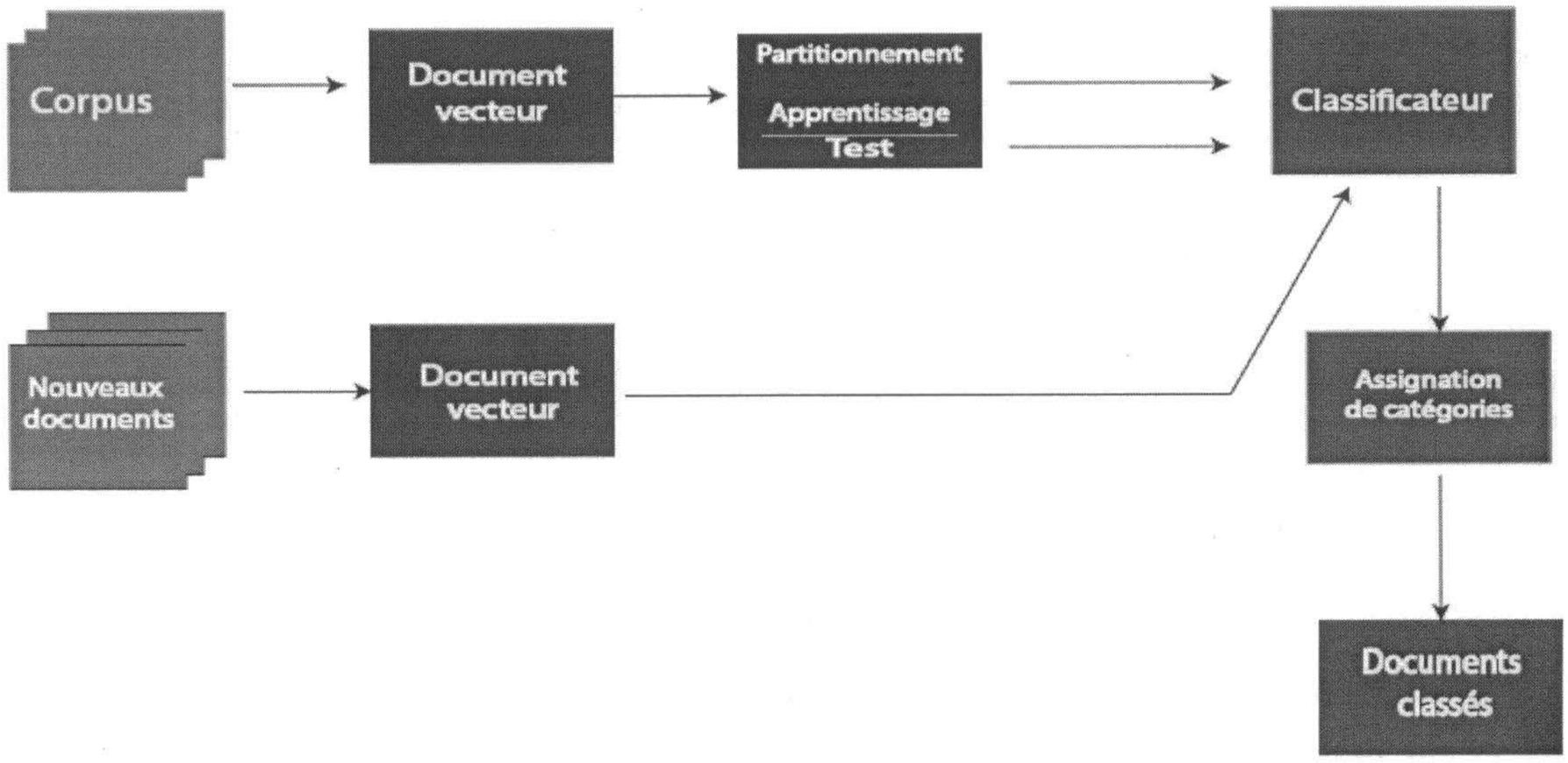


Figure 1: Processus de classification de documents[4].

Avant de commencer les étapes définies ci haut, nous devons effectuer certaines modifications pour transformer nos documents textes en documents vecteurs (entrées valides), pour ensuite les présenter aux algorithmes de classification. Ces transformations sont appelées le" prétraitement".

## 2.2 Le prétraitement :

Avant de pouvoir créer et entrainer un quelconque classificateur à partir d'un corpus de documents donné, nous devions impérativement transformer les documents textes en entrées valides compréhensibles par les différents algorithmes de classification ou de clustering. Ces entrées valides sont en fait des vecteurs ou des matrices qui définissent le poids de chaque

descripteur (mot ou groupe de mots, n-gram) dans chaque document texte où ils apparaissent. Si par contre un tel descripteur n'existe pas dans un tel document son poids sera nul (0). C'est cette transformation entre un document texte et un document vecteur qu'on appelle prétraitement, et qui comporte plusieurs sous étapes. Dans cette première partie, nous nous contentons juste de présenter les prétraitements classiques que nous avons d'ailleurs utilisés et qui figurent souvent dans les livres de data mining et dans les articles scientifiques. Cependant les traitements spécifiques, propre à notre cas, seront détaillés dans la deuxième partie.

### 2.2.1 Tokenisation :

Un token est une unité définie comme une séquence de caractères comprise entre deux séparateurs; les séparateurs étant les blancs, les signes de ponctuation et certains autres caractères comme les guillemets ou les parenthèses.

La tokenisation consiste à segmenter un document texte en tokens de mots, séquences de mots ou carrément de phrases, mais généralement elle s'opère souvent sur des mots.

### 2.2.2 Mots vides :

Une fois les documents textes découpés en tokens, nous apercevons que certains de ces tokens sont présents dans tous les textes du corpus, c'est ce que nous appelons les mots vides : les articles, les prépositions, les déterminants, les adverbes... comme "la ,le , dans, car," dans la langue française, et "the, and, after" dans la langue anglaise. Ils représentent 30% des mots dans un texte. La présence de ces mots n'apporte absolument aucune différence tant sur le plan sémantique que sur le plan lexical. Cela veut dire que leur présence dans tous les textes du corpus les rend non discriminants et du coup leurs utilisation pour une tâche de classification s'avère inutile. Par contre, leur suppression réduit la dimension de notre document vecteur, par conséquent, le temps de traitement et le temps d'apprentissage seront réduit considérablement.

### 2.2.3 La racinisation (Stemming) et la lemmatisation:

Lors de certaines représentations vectorielles, chaque mot présent dans le corpus est considéré comme un descripteur. Cette façon de faire contient certaines irrégularités, notamment pour les verbes à l'infinitif et les verbes conjugués (développer, développe, développé, développèrent...), nous remarquons que ces derniers ont le même sens mais chacun est considéré comme un descripteur à part entière. La racinisation vient parer à ce problème, en considérant uniquement la racine de ces mots plutôt que les mots en entier sans se soucier de l'analyse grammaticale.

La lemmatisation par contre est une forme compliquée de la racinisation puisqu'elle met en évidence l'analyse grammaticale, elle ramène les termes à leur forme canonique[4] en mettant tous les noms au singulier, les adjectifs au masculin singulier, et tous les verbes conjugués à l'infinitif. La substitution des mots par leurs racines permet une meilleure représentation, surtout dans le cas de la lemmatisation, et réduit considérablement la dimension des descripteurs. Toutefois, nous lui décelons certaines ambigüités dans le cas ou un descripteur représente un même mot qui a deux sens différents, notamment dans l'exemple suivant:
"actions" et "action" seront représentés par leur forme singulière par le descripteur "action", cependant, le mot peut avoir deux sens bien différents selon son utilisation; En économie le mot "actions" peut signifier les actions en bourse, qui n'a rien avoir avec "un film d'action"[3].

### 2.3 Les pondérations (fréquences) :

En text mining, la fréquence désigne le nombre de fois qu'un certain descripteur est apparu dans chacun des documents d'un corpus. À partir de ces fréquences, que nous nommons communément "poids", nous pouvons dire qu'un descripteur quelconque est discriminant ou pas, par rapport à un document donné, si son poids est élevé ou pas, respectivement. En d'autres termes si le descripteur apparait souvent ou pas du tout dans le document en question. Mais malheureusement cette manière très simpliste de mesurer les fréquences n'est pas toujours vraie, surtout dans des documents longs comportant plusieurs paragraphes.
Avant de parler des différentes manières avec lesquelles nous calculons le poids d'un descripteur dans un document, nous aimerions d'abord aborder les manières avec lesquelles nous définissons notre ensemble de descripteurs.

#### 2.3.1 Le sac à mots :

La façon la plus simple et la plus évidente pour la représentation d'un document texte par un document vecteur, est d'utiliser les mots comme descripteurs. Ainsi, nous construisons un sac à mots (bag of words) de tous les mots qui apparaissent au moins une fois dans le corpus. Cette méthode, qui conserve le sens naturel des descripteurs, est loin de répondre à toutes les attentes de la classification de texte, à cause, notamment, de certaines anomalies liées à la variation des fréquences par rapport à la longueur du document, au problème des mots composés, et principalement, au fait que l'ordre d'apparition des mots dans les phrases du document n'est pas pris en considération. Les mots sont regroupés en vrac et traités d'une manière indépendante, ce qui nuit considérablement à la sémantique du texte.

### 2.3.2 Les N-grams :

Cette méthode de représentation de documents texte consiste à partager ce dernier en séquences de n caractères. En effet, si nous considérons seulement les lettres de l'alphabet comme caractères et dans le cas ou "n" égal à 1, c'est à dire la séquence contient juste une seule lettre, est ce que c'est une bonne manière de représenter un document dans le but de le classifier? Certainement non, même si dans certains cas cela semble très efficace, notamment dans la reconnaissance de la langue. C'est pour cette raison d'ailleurs que le "n" est toujours supérieur à 1.

Prenons l'exemple de la phrase "La classification supervisée " et essayons de la représenter en n-grams caractères.

- si n=2 nous aurons {"La","a "," c","cl","la","as","ss","si","if",etc.}
- si n=3 nous aurons {"La ","a c"," cl","cla","las","ass","ssi","sif","ifi", etc.}
- si n=4 nous aurons {"La c","a cl"," cla","clas","lass","assi","ssif","sifi","ific", etc}

Dans la littérature, le consensus s'est porté sur n=3, car n<3 la représentation est très élémentaire, tandis que n>3 on génère beaucoup de colonnes. Les trigrammes de caractères semblent très efficaces et sont utilisés dans plusieurs applications .

### 2.3.3 Fréquence des termes (TF) :

On désigne par TF la fréquence d'un mot (descripteur) dans un texte donné. C'est un calcul de fréquence très simple, mais qui s'avère efficace et pratique. Nous l'utilisons souvent en association avec d'autres fréquences. Nous dénombrons plusieurs manières de calcul de la TF:

TF absolue: c'est le nombre de fois qu'un terme apparait dans un texte donné.

$$TF = \mathrm{NT} \qquad (1)$$

ou NT est le nombre de fois où le terme est apparu dans le texte.

TF relative : C'est le rapport entre le nombre de fois qu'un terme est apparu dans le texte sur le nombre de tous les termes du texte. Cette dernière est utilisée généralement pour limiter l'impacte de la longueur des textes. En effet, un terme qui apparait 6 fois dans un texte de 100 termes n'a pas le même degré de discrimination que celui qui apparait le même nombre de fois (6 fois) dans un texte de 20 termes.

$$TF = \frac{\mathrm{NT}}{\mathrm{ST}} \qquad (2)$$

où NT est le nombre de fois que le terme est apparu dans le document. et, ST est le nombre de tous les termes du document.

TF booléenne : se contente juste de la présence ou de l'absence du terme dans le texte.

$$TF = 1 \text{ ou } 0 \qquad (3)$$

Le principal inconvénient de la fréquence des termes est le fait qu'il est possible, et d'ailleurs c'est un cas très probable en pratique, qu'un terme apparait avec une fréquence assez grande dans tous les documents d'un corpus. Dans ce cas, le terme en question perd toute sa notion de discrimination relative au degré de présence. Une autre notion vient rectifier ce cas exceptionnel, nommée IDF(Inverse documents frequancies).

### 2.3.4 Fréquence documents inverses (IDF) :

Elle mesure en quelque sorte le degré de rareté d'un terme, non pas dans un document, mais dans tous les documents d'un corpus elle est définie par cette équation :

$$IDF = \log\left(\frac{\mathrm{N_{Doc}}}{\mathrm{Doc_T}}\right) \qquad (4)$$

Où $N_{Doc}$ est le nombre de documents dans le corpus, et $Doc_T$ est le nombre de documents dans lesquels le terme est apparu.

Si le terme est très présent dans tout le corpus alors le rapport sera égal à 1 et IDF = 0 donc le terme est neutralisé. Si par contre il apparait dans un seul document la valeur est maximale.

$$\mathrm{IDF} = \log(\mathrm{N_{Doc}}) \qquad (5)$$

Cette pondération à elle seule ne définie nullement le degré de discrimination d'un terme dans un document puisque elle est relative au corpus. Mais l'association de IDF avec TF donne des résultats intéressants.

### 2.3.5 TFIDF :

Nous avons vu que la fréquence d'un terme dans un document joue un rôle important dans le calcul de son degré de discrimination. En revanche, la représentation d'un texte, dans le but de le classifier, ne dépend pas seulement de son contenu, mais elle est liée étroitement au corpus auquel le texte appartient, la rareté de ce terme au sein des autres documents du corpus s'avère aussi importante que sa fréquence (abondance) dans le document en question.

Cette combinaison judicieuse de ces deux principes (abondance particulière et rareté générale) a engendré la pondération dite TFIDF.

Elle est calculée avec la formule suivante :

$$TFIDF = TF * \log\left(\frac{\mathrm{N_{Doc}}}{\mathrm{Doc_T}}\right) \qquad (6)$$

où TF est relative ou absolue.

**2.4 Les Algorithmes :**

Il existe plusieurs algorithmes pour la classification de texte[4]. Dans cet essai, nous nous contentons de définir les trois algorithmes que nous allons comparer dans le chapitre 3.

**2.4.1 Arbre de décision :**

L'arbre de décision est constitué d'un ensemble de nœuds liés par des branches, il s'étend du haut vers le bas [5]; c'est à dire, que le nœud racine est placé au sommet de l'arbre tandis que les nœuds feuilles sont placés au bas de l'arbre. L'arbre de décision est sensé classer les textes en plusieurs catégories. En commençant par le haut, nous devons placer dans le nœud racine le descripteur qui discrimine le mieux les textes de notre corpus, pour obtenir de nouveaux sous nœuds, ainsi de suite, on répète l'opération pour chaque nœud, jusqu'à ce que la séparation des textes n'est plus possible ou n'est plus souhaitable. Au final, les nœuds feuilles sont constitués d'ensemble de textes de même catégorie.

L'arbre de décision est considéré comme étant la technique la plus utilisée en data mining.

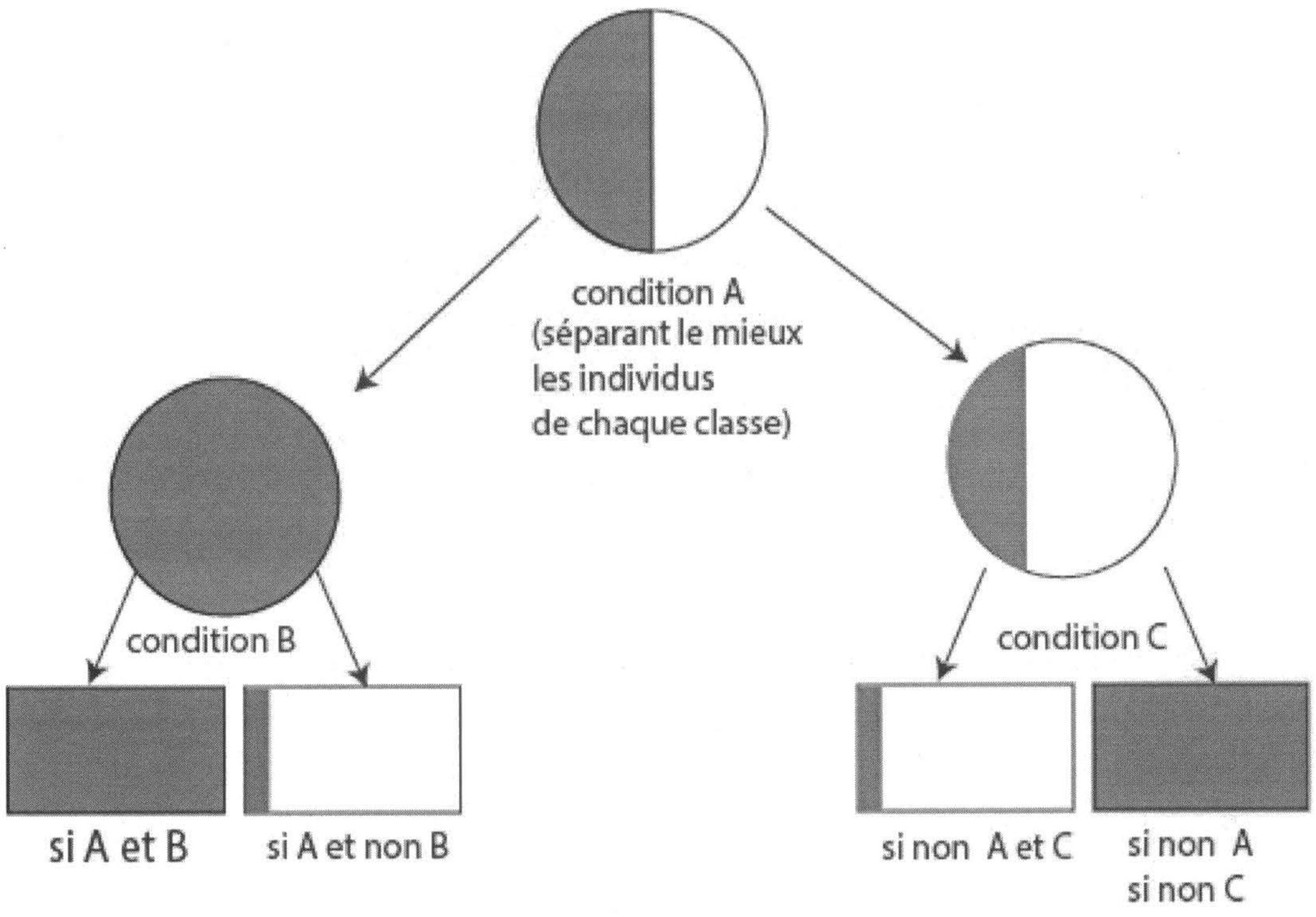


Figure 2 : Arbre de décision[4].

#### 2.4.1.1 Avantages :

Les arbres de décision sont bâtis selon un ensemble de règles très explicites ce qui rend les résultats bien compris par l'utilisateur. Ils demandent généralement peu de ressources, et leur temps d'apprentissage et de test sont relativement courts.

#### 2.4.1.2 Inconvénients :

Quand les arbres deviennent assez grands, ça veut dire, ils comportent beaucoup de feuilles, ils perdent leur pouvoir explicatif. De plus le fait que les sous-nœuds dépendent directement du nœud racine rend la présence ou l'absence d'un seul descripteur dans un texte déterminant sur le sort de son classement. D'autant plus que la modification d'un seul nœud, s'il est prés du sommet, modifie entièrement l'arbre. Notons enfin que les arbres de décision sont sujet au sur-apprentissage, mais au moins il est localisable contrairement au réseaux de neurones.

### 2.4.2 Machine à Vecteur de Support (MVS) :

Le MVS est un classificateur dit linéaire, ça veut dire que, dans le cas parfait, les données (document texte dans notre cas) doivent être linéairement séparables. Ainsi notre corpus est représenté comme étant un espace vectoriel, où chaque document texte est représenté par un point dans ce dernier. La problématique maintenant est de trouver le meilleur séparateur (ligne, plan ou hyperplan) qui partage notre corpus en deux catégories. L'espace entre ces deux catégories est appelé marge, qui est définie par les points (Vecteurs de support) les plus proches du séparateur, de part et d'autre. Le but étant essentiellement de maximiser cette marge, plus elle est grande meilleurs est le résultat. Le classificateur se généralise bien avec les nouvelles données.

Toutefois, si les données ne sont pas linéairement séparables, la MVS peut être modifiée pour tolérer un minimum d'erreurs. Désormais, le but est de maximiser la marge et de minimiser l'erreur de classification. Une autre alternative pour parer à la non séparabilité des données, est de passer à un espace de dimension supérieur.

La figure ci dessous illustre la façon dont MVS départage les données dans le cas où elles sont linéairement séparables, de plus, elle choisit la séparation la plus optimale où la marge est maximale.

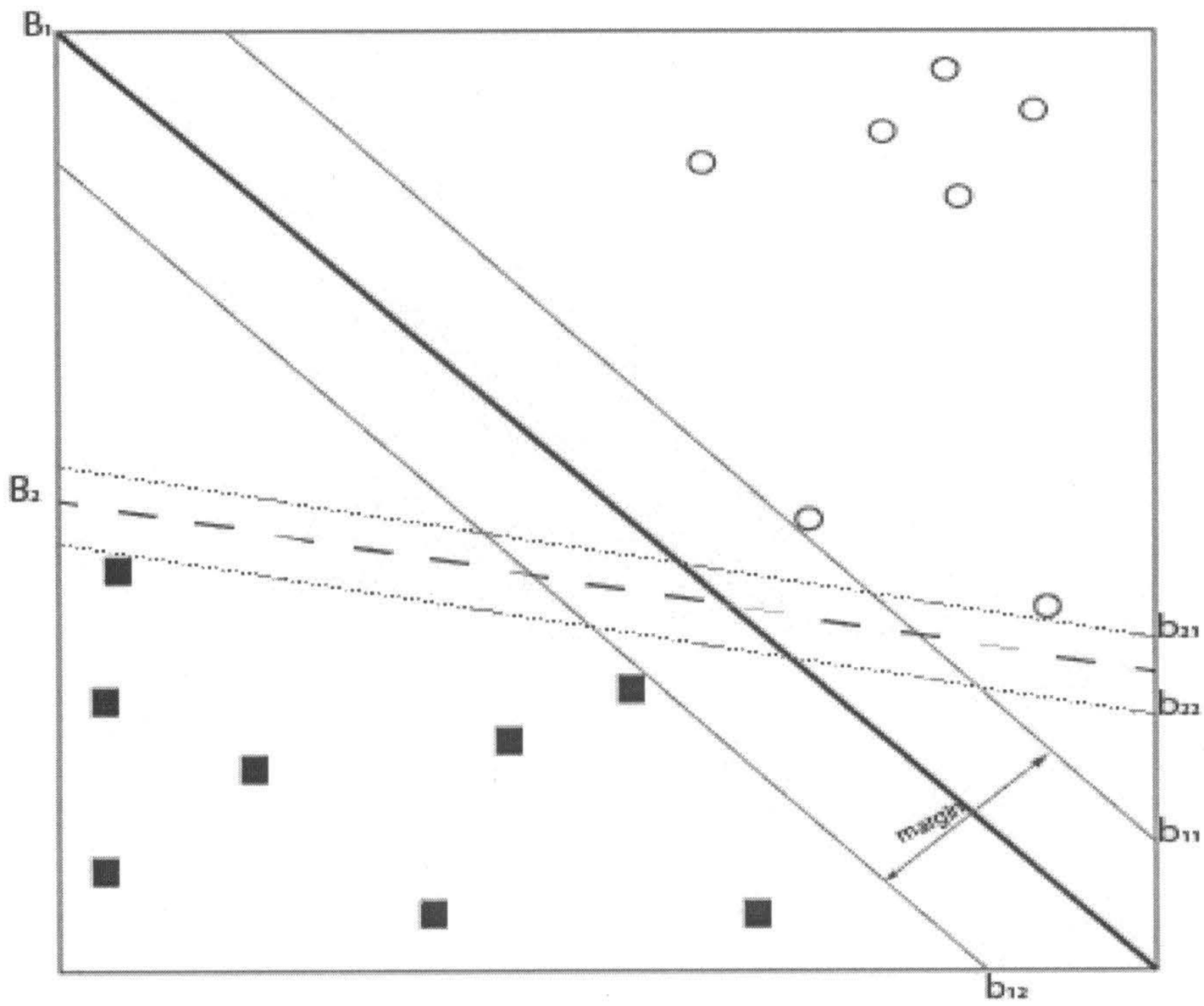


Figure 3 : Séparation correcte(B2) et séparation optimale(B1)[4].

**2.4.2.1 Avantages :**

C'est un classificateur qui se généralise très bien avec les nouvelles données, puisque les vecteurs de support sont les points les plus proches du séparateur est non les plus loins. Du coup, les valeurs aberrantes n'affectent en aucun cas le classificateur.

L'autre avantage est sa flexibilité quant au traitement des cas non-linéairement séparables.

**2.4.2.2 Inconvénients :**

Son temps d'apprentissage est relativement supérieur, comparativement aux arbres de décision.

La MVS est considérée comme un classificateur bi-classes, mais des solutions ont été proposées dans ce sens, notamment, la transformation du problème de classification multi-classes en sous problème bi-classes.

**2.4.3 Classificateur Naïf Bayes (CNB) :**

C'est un classificateur probabiliste, basé sur le théorème de Bayes.

$$P(A/B) = P(B/A)\frac{P(A)}{P(B)} \qquad (7)$$

Naïf parce que nous posons, au préalable, l'hypothèse que les variables sont indépendantes. Dans le text mining on désigne par variables les descripteurs (termes).

### 2.4.3.1 Avantages :

l'hypothèse d'indépendance des descripteurs du classificateur Naïf Bayes le rend simple et efficace. Son entrainement ne nécessite pas beaucoup de documents, il a fait ses preuves dans la classification de documents courts, notamment les courriels (Ham/Spam)

### 2.4.3.2 Inconvénients :

Contrairement aux documents courts, les documents longs posent un grand problème pour le classificateur Naïf Bayes, un riche vocabulaire favorise les dépendances entre les descripteurs(termes).

## 2.5 Les Scores:

Les scores sont aussi importants au même titre que les autres phases du processus de construction d'un classificateur. Sinon, comment savoir si un tel classificateur est fiable, et comment savoir s'il se comporte bien avec de nouvelles données? Plusieurs mesures ou scores existent pour justement vérifier et estimer le degré de généralisation d'un classificateur. Toutefois, il faut signaler que ces mesures sont à l'origine utilisées dans la recherche documentaire, puis les spécialistes du data mining les ont adaptées aux autres branches de cette science. Il existe plusieurs indicateurs qui reflètent la réussite ou l'échec d'un classificateur.

### 2.5.1 la matrice de confusion :

| | | **Classes Prédites** | |
|---|---|---|---|
| | | Classe01 | Classe02 |
| **Classes Réelles** | Classe01 | VP | FN |
| | Classe02 | FP | VN |

Tableau 1 : Matrice de confusion.

La matrice de confusion en text mining sert à vérifier le bon classement des documents préalablement étiquetés. En d'autres termes, elle indique si un classificateur fonctionne bien et à quel degré de fiabilité. Chaque classe du classificateur est représentée par une colonne et une

ligne. La ligne indique le nombre de documents réels appartenant à la classe (C) et la colonne indique à quel nombre de documents cette classe (C) est assignée.

On entend par :

**VP (Vrais positifs) :** Les documents appartenant à la classe 01que le classificateur a classés à la classe 01.

**FP (Faux positifs) :** Les documents appartenant à la classe 02 que le classificateur a classés à la classe 01.

**VN (Vrais négatifs):** Les documents appartenant à la classe 02 que le classificateur a classés à la classe 02.

**FN (Faux négatifs):** Les documents appartenant à la classe 01 que le classificateur a classés à la classe 02.

Notons bien que c'est avec ces quatre paramètres que toutes les autres mesures sont calculées.

### 2.5.2 La précision :

C'est le rapport entre le nombre de documents correctement classés dans la classe (C) sur le nombre de document auxquels la classe (C) est assignée.

$$\text{Précision(C)} = \frac{\text{VP}}{\text{VP+FP}} \qquad (8)$$

### 2.5.3 Le rappel :

C'est le rapport entre le nombre de documents correctement classés dans la classe (C) sur le nombre de documents appartenant à la classe (C).

$$\text{Rappel (C)} = \frac{\text{VP}}{\text{VP+FN}} \qquad (9)$$

### 2.5.4 Le taux d'erreur et de succès:

Le taux de succès est le rapport entre les documents bien classés sur le nombre total des documents du corpus.

$$\text{Taux de succès} = \frac{\text{VP+VN}}{\text{VP+FP+VN+FN}} \qquad (10)$$

D'autre part le taux d'erreur est le rapport entre les documents mal classés sur le nombre total des documents du corpus.

$$\text{Taux d'erreur} = \frac{\text{FP+FN}}{\text{VP+FP+VN+FN}} \qquad (11)$$

### 2.5.5 La F-Mesure :

La F-Mesure est un indicateur qui combine le rappel et la précision elle est donnée par la formule suivante :

$$\text{F-Mesure} = \frac{2*(\text{Précision}*\text{Rappel})}{\text{Précision}+\text{Rappel}} \qquad (12)$$

# Chapitre 3 - Le cas pratique

## 3.1 L'outil KNIME :

Konstanz Information Miner, connu sous l'acronyme KNIME, est un logiciel de data mining développé en 2004 à l'université de Constance( Konstanz) en Allemagne. C'est un logiciel open source sous la licence GPL(General Public License ), écrit en langage JAVA sous la plate forme Éclipse SDK.

KNIME est une plate forme modulaire pour la conception et l'exécution de flux de travail(workflows) en utilisant des composants prédéfinis appelés nœuds. Contrairement à beaucoup de logiciels du domaine, qui offrent une version allégée gratuite et une autre version complète(professionnelle) payante, la plateforme de KNIME est complète, il n'ya pas de restrictions par rapport au volume de données, ainsi les projets qui manipulent de grands volumes de données dépendent entièrement des ressources de l'utilisateur(taille disque, RAM ...) et non pas de la plate forme KNIME[6].

KNIME est doté d'une grande flexibilité qui se traduit par sa facilité à interagir avec d'autres logiciels comme par exemple WEKA, du coup nous pouvons facilement utiliser les composants WEKA que nous ajoutons comme une librairie externe. Son point fort est le fait qu'il nous offre la possibilité d'écrire notre propre code Java dans certains nœuds, notamment avec le nœud "Java Snippet". Ainsi, nous pourrons par exemple créer nos propres pondérations en se passant de celles déjà existantes.

KNIME est un logiciel facile à utiliser, à condition qu'il faut bien choisir le nœud correspondant à la tache que nous aimerions effectuer. La logique du processus et l'ordre des opérations sont très importants pour la réussite d'un projet KNIME.

### 3.1.1 Le Nœud :

Le nœud est une entité qui exécute une tâche bien définie. L'algorithme de racinisation "Porter" est un bel exemple de nœud KNIME.

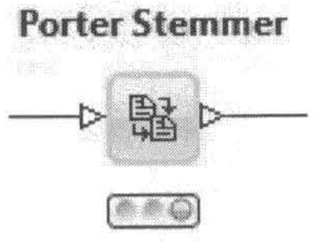


Figure 4 : Exemple de nœud, Porter Stemmer.

### 3.1.2 Le flux de travail (Workflows) :

Le Workflows est une succession logique de nœuds .

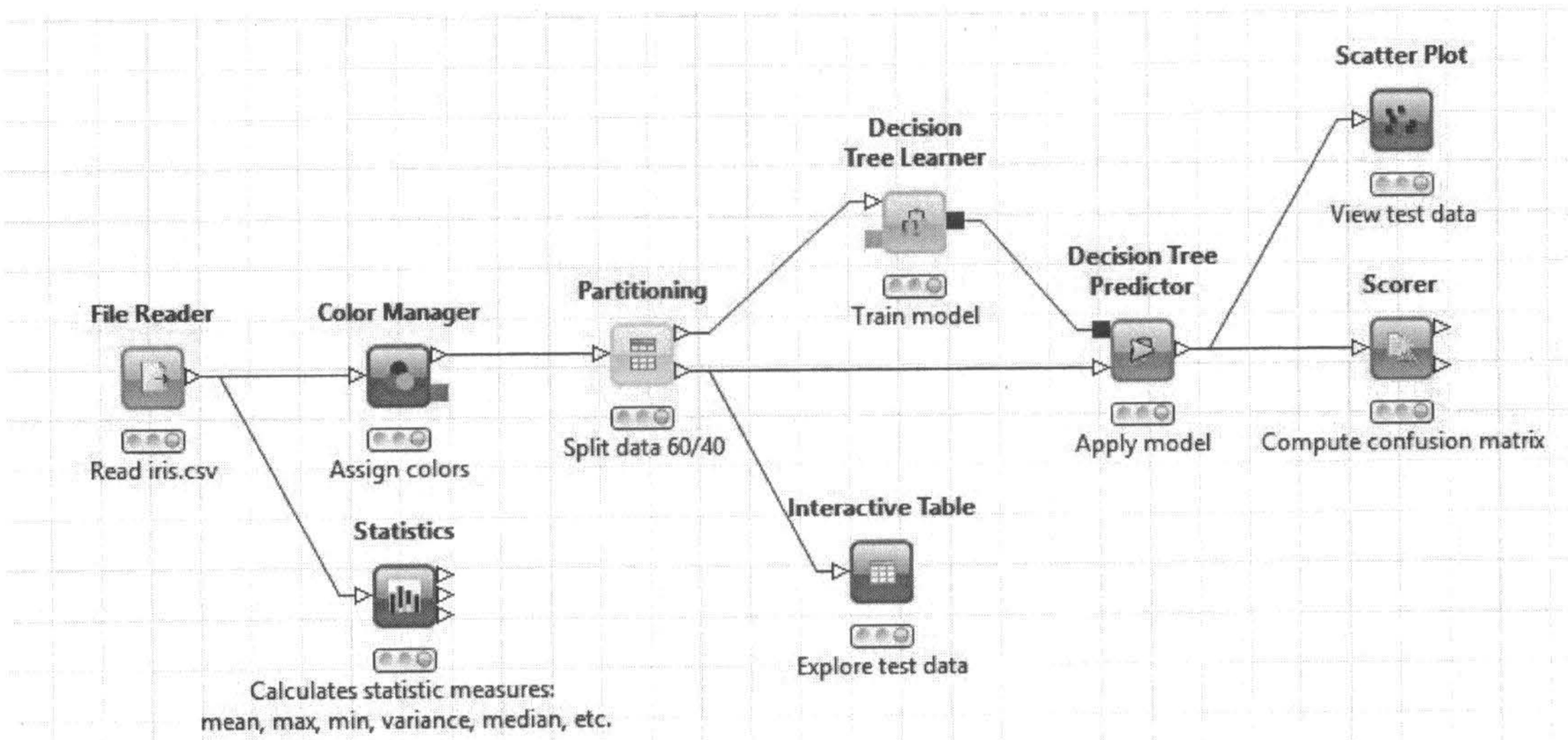


Figure 5 : Workflows de KNIME.

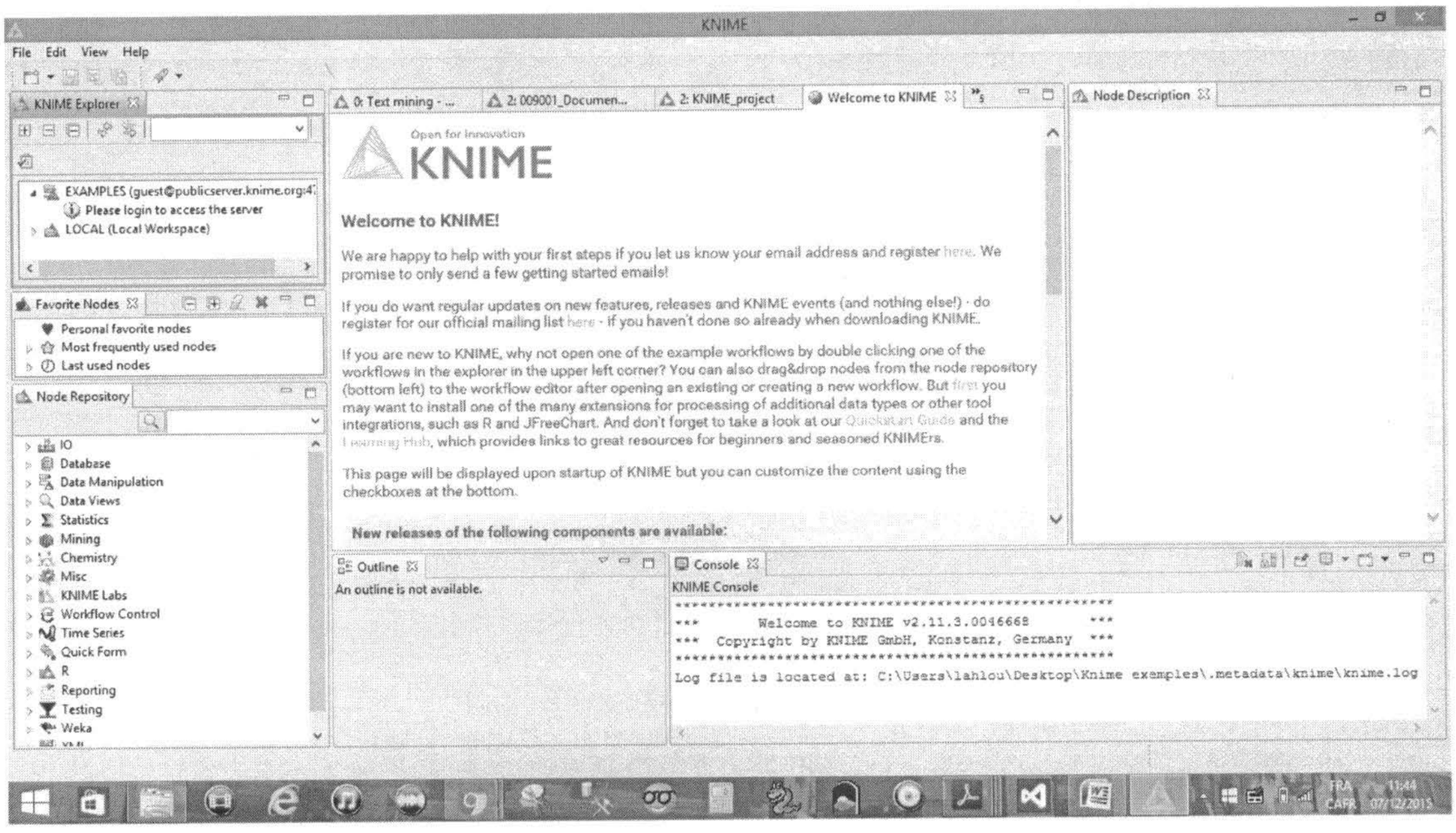


Figure 6 : Vue globale du le logiciel KNIME.

### 3.2 La méthodologie:

#### 3.2.1 Le corpus :

Plusieurs corpus de documents texte sont disponibles sur internet, à l'image de Reuters-21578, 20Newsgroups, Ohsumed[7], DEFT'05 et bien d'autres. Certains sont entièrement gratuits et disponibles à 100%, d'autres nécessitent plus au moins des étapes d'authentifications ainsi que des renseignements personnels et professionnels (tel que l'organisme de recherche par exemple), tandis que d'autres exigent un coût pour certaines fonctionnalités.

Pour notre étude, nous avons opté pour le corpus Ohsumed. C'est un ensemble de 23 176 documents, extrait de la base MEDLINE (base d'informations médicales en ligne), il est constitué principalement d'extraits de journaux médicaux rédigés en langue Anglaise. Les documents sont divisé en 23 catégories distinctes, et chaque catégorie est représentée par un répertoire contenant tous les documents à qui cette dernière est assignée. Le nombre de documents diffère d'une catégorie à une autre. Enfin l'ensemble des documents ( 23176) sont répartis comme suit: 10 443 pour l'ensemble d'apprentissage et 12773 pour l'ensemble de test selon la représentation déjà décrite . Les catégories sont désignées par l'acronyme C01 jusqu'à C23, mais un fichier décrivant le nom de chaque catégorie est mis en annexe.

Pour notre cas nous avons utilisé uniquement quatre (4) catégories parmi les vingt trois (23), à savoir la catégorie C01, la catégorie C02, la catégorie C04 et la catégorie C05.

Ce choix est motivé par le but même de notre étude de cas. L'approche était de construire en premier temps des classificateurs binaires (deux classes), ensuite d'ajouter les catégories une par une en analysant les performances de chaque classificateur.

#### 3.2.2 Le prétraitement :

Dans cette section nous aborderons les différentes manipulations que nous avons effectuées sur nos documents d'entrée, nous détaillerons le fonctionnement de chaque nœud ainsi que son paramétrage et surtout les motivations de son choix. Notons que cette première étape sert seulement à la préparation de nos documents, afin qu'il soit possible d'extraire les descripteurs et ainsi calculer leur pondérations.

**- Analyseur des fichiers plats :**

Ce nœud sert à lire les fichiers plats. Nous insérons notre ensemble de documents au travers d'un champ de chargement. Chaque catégorie est insérée avec un seul nœud. Une fois le paramétrage effectué, nous avons qu'à cliquer sur le bouton exécute pour lancer l'analyse et lire

l'ensemble de nos documents. Le résultat fourni est sous forme d'un tableau où chaque enregistrement représente un seul document.

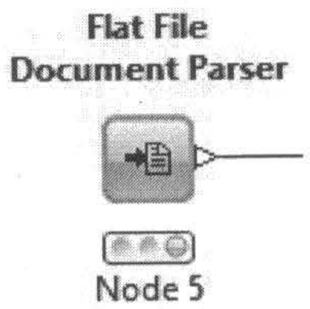


Figure 7 : Analyseur des fichiers plats.

**- Filtre d'enregistrements :** Ce nœud sert à filtrer les enregistrements, nous pouvons les sélectionner selon leurs numéros ou la valeur de l'un de leurs attributs. Dans notre cas nous l'avons choisi afin de réduire le nombre de documents dans chaque nœud et veiller à ce que ce dernier soit égal dans chaque catégorie.

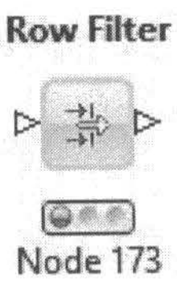


Figure 8 : Filtre d'enregistrements.

**- Concaténation :**

Ce nœud sert à regrouper toute les entrées en un seul document.

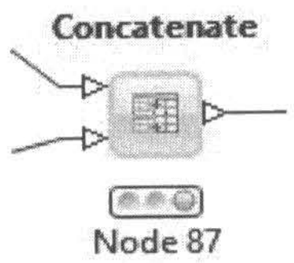


Figure 9 : Concaténation.

**- Suppression des signes de ponctuation:**

Ce nœud sert à éliminer tous les signes de ponctuation, en l'exécutant, nous aurons une table dans laquelle nous pouvons comparer le document original et celui traité.

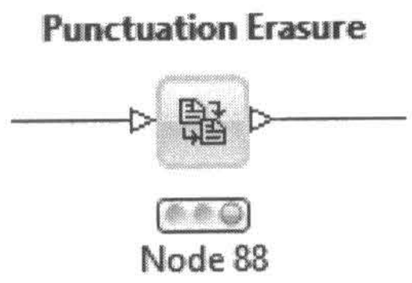


Figure 10 : Suppression des signes de ponctuation.

**- Filtre de nombres :**

Sert à éliminer tous les nombres.

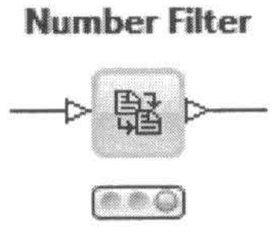


Figure 11 : Filtre de nombres.

**- Convertisseur :**

Sert à ramener toutes les lettres minuscules vers la majuscule ou vis versa.

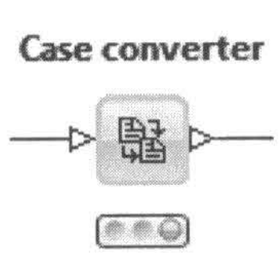


Figure 12 : Convertisseur.

**- Filtre de mots vides :**

Sert à éliminer les mots vides que nous avons défini au deuxième chapitre.

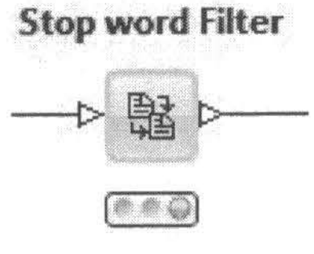


Figure 13 : Filtre de mots vides.

**- Nœud Porter :**

C'est un nœud qui utilise le plus célèbre des algorithmes de racinisation(vu au chapitre 2) en langue anglaise.

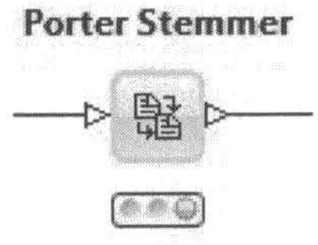


Figure 14 : Nœud Porter.

### 3.2.3 Les pondérations :

La deuxième étape de la préparation de nos documents est le calcul des pondérations. Il ya, certes, différentes pondérations dans le domaine du texte mining, nous rencontrons souvent de nouvelles manières de les calculer à travers notamment les articles scientifiques et les livres de data mining, mais dans notre cas nous allons aborder seulement les plus célèbres d'entre elles que nous avons évoqué au deuxième chapitre.

**- Sac à mots :**

Sert à grouper tous les descripteurs qui apparaissent au moins une fois dans tout le corpus.

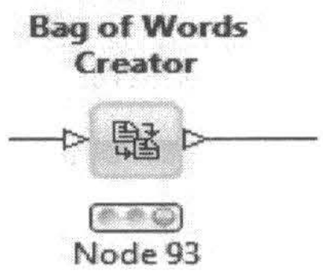


Figure 15 : Sac à mots.

**- Fréquences des termes (TF) :**

Dans notre cas nous avons utilisé la fréquence absolue.

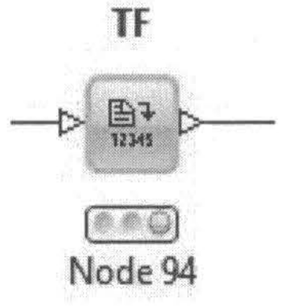


Figure 16 : Fréquences des termes.

**- Fréquences inverses documents (IDF) :**

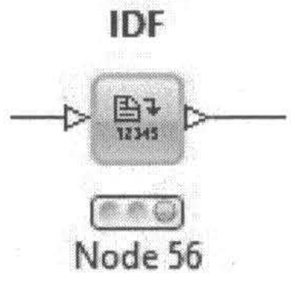


Figure 17 : Fréquences inverses documents.

**- Java Snippet :**

C'est l'un des nœuds les plus importants de Knime. Nous avons évoqué un peu plus haut que l'une des plus intéressante caractéristique que nous offre Knime était la possibilité d'écrire du code JAVA et de combiner des équations mathématiques. C'est avec le nœud Java Snippet que nous pouvons le faire. Dans notre cas nous l'avions utilisé pour combiner la pondération TFIDF.

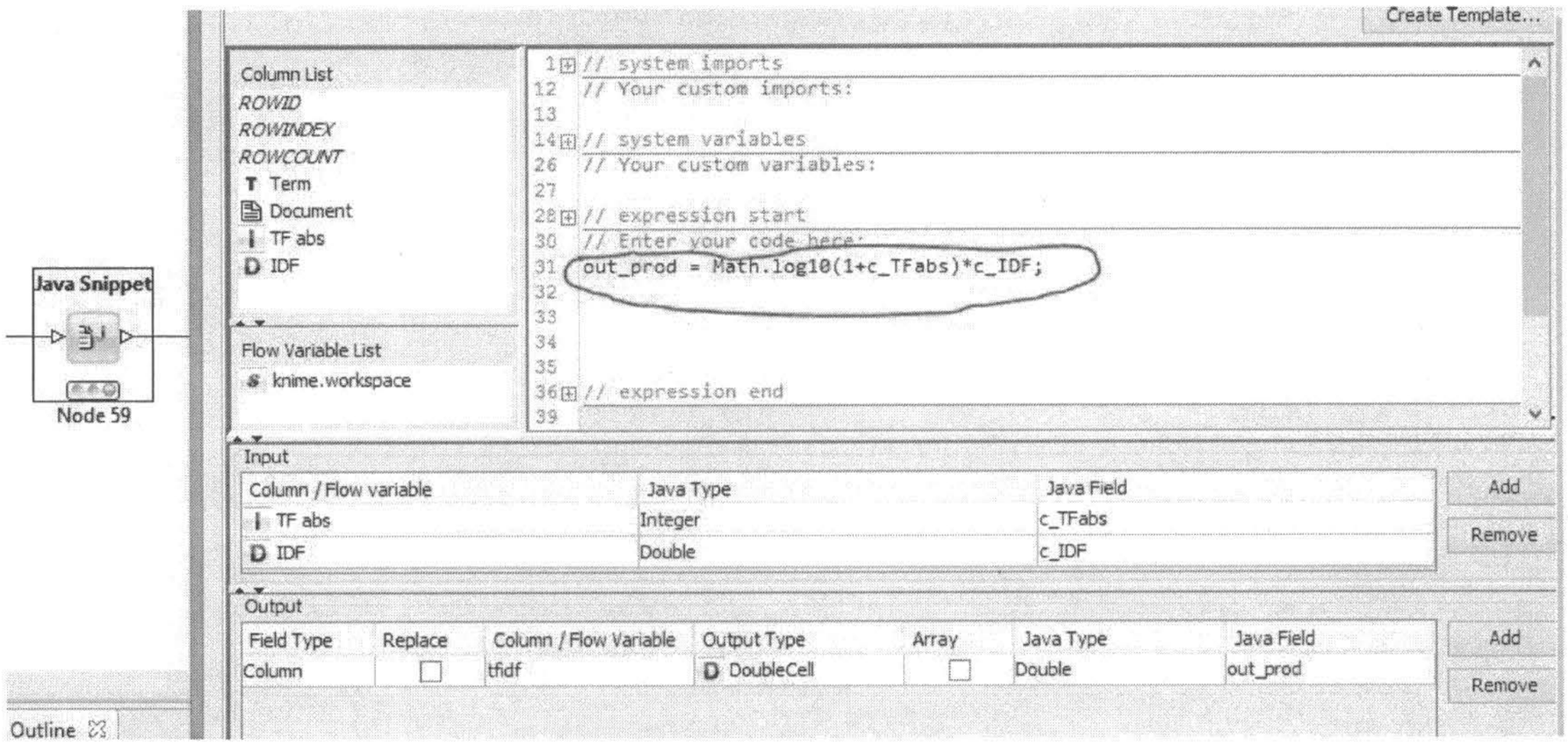


Figure 18 : Java Snippet.

**- Document vector :**

Notre vecteur d'entrée, comportant tous les poids des descripteurs, est désormais prêt à l'emploi, même si, il nous reste encore quelques nœuds qui vont servir à l'ajout et à la suppression de certaines colonnes et notamment pour les besoin de la visualisation.

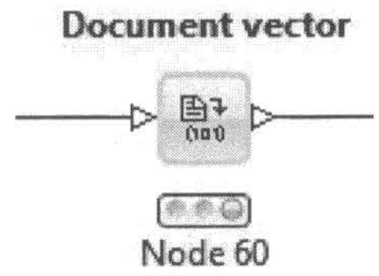


Figure 19 : Document Vecteur.

**- Ajout de la classe :**

Maintenant que tout le prétraitement est fait et notre document vecteur est fin prêt, il nous reste qu'introduire la colonne "Documents class" qui correspond à la catégorie de chaque document.

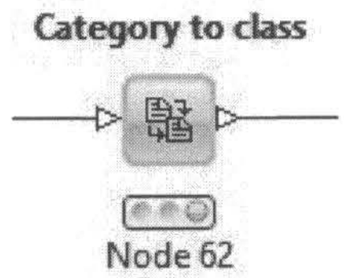


Figure 20 : Ajout de la classe.

**- Nœud couleur :**

Nous utilisons ce nœud pour des fin de visualisation, ainsi, nous attribuons à chaque catégorie une couleur bien distincte. nous allons voir son utilité lorsque nous visualiserons, un peu plus bas, les arbre de décision.

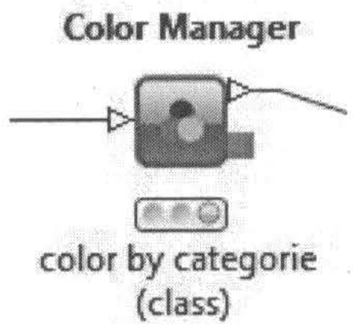


Figure 21 : Nœud couleur.

**- Filtre de colonne :**

Dans notre cas nous l'avons utilisé pour éliminer la colonne qui comporte notre texte sur lequel nous avons effectué tout le traitement, il ne reste que les descripteurs ainsi que leurs poids et la catégorie correspondante.

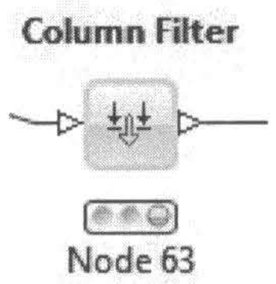


Figure 22 : Filtre de colonne.

**- Partitionnement :**

C'est la dernière étapes avant l'entrainement des algorithmes. Pour notre cas nous avons préféré partitionner notre ensemble en 90/10, qui signifie que 90% des documents seront dédiés pour l'entrainement et 10% pour le test..

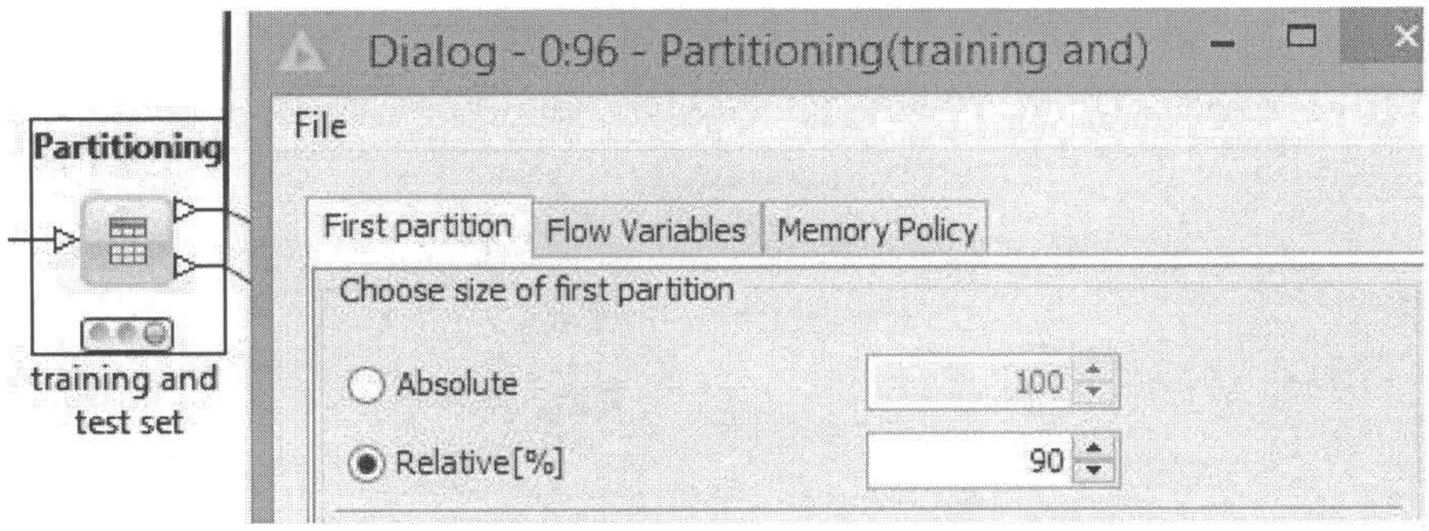


Figure 23 : Partitionnement.

### 3.2.4 Les classificateurs :

Une fois le prétraitement, le calcul des pondérations et le partitionnement du document vecteur effectués, nous procédons maintenant à l'entrainement des classificateurs avec l'ensembles d'entrainement, ensuite nous prédisons la classe(catégorie) de chaque document de l'ensemble test. Pour ce faire nous allons utiliser trois classificateurs très répandus dans la littérature à savoir naïf bayes, Machine à vecteurs de support et l'arbre de décision.

#### 3.2.4.1 Classificateurs Naïf Bayes :

Nous insérons le nœud "Naive Bayes Learner" pour entrainer d'abord le classificateur, mais avant d'exécuter quoi que ce soit nous devons effectuer certains paramétrage.

Default probability : C'est la probabilité apriori, que nous ajoutons à chaque descripteur pour éviter le cas où un descripteur n'apparait pas dans un document. le mieux et que cette dernière soit la plus petite valeur possible. Désormais nous pouvons exécuter notre "nœud Naive Bayes Learner". Une fois l'apprentissage du classificateur est effectué avec les données d'apprentissage, nous insérons le nœud "Naive Bayes Predictor" pour tester la fiabilité de ce dernier mais cette fois avec les données test.

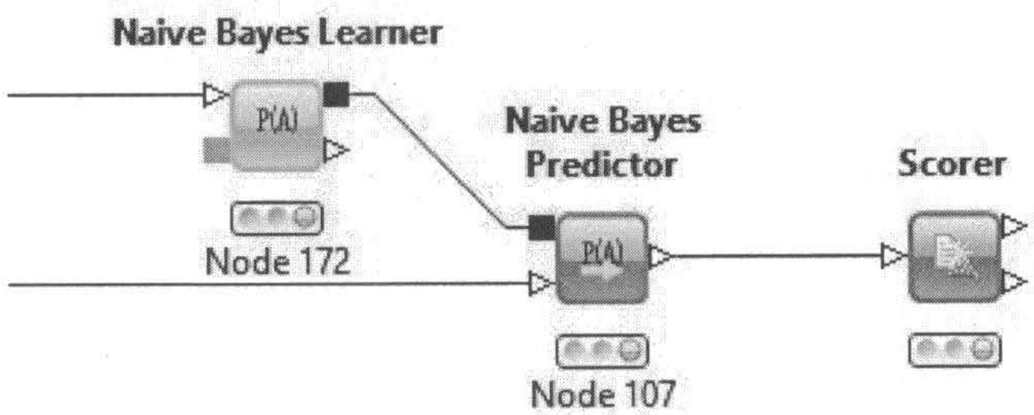


Figure 24 : Classificateur Naïf Bayes.

### 3.2.4.2 Machine à vecteurs de support (MVS) :

Nous procédons de la même manière, décrite précédemment (classificateur de bayes), pour la machine à vecteurs de support. Seulement cette dernière nécessite un peu plus de paramétrage.

Overlapping penality : 2 : C'est la pénalité de chevauchement, elle est utilisée particulièrement lorsque les données ne sont pas linéairement séparables.

Nous devons choisir un noyau parmi les trois noyau offert par Knime (Polynomial, HyperTangent, RBF) et leurs paramètres respectifs.

Dans notre cas nous choisissons le Polynomial avec : Power : 0,1 ; Bias : 0,1; Gamma :0,1.

Nous exécutons et nous passons à "SVM Predictor".

La non disponibilité d'une MVS linéaire sur KNIME nous a reconduit à une autre implémentation de cette dernière disponible sur WEKA, puisque KNIME nous permet cette interaction.

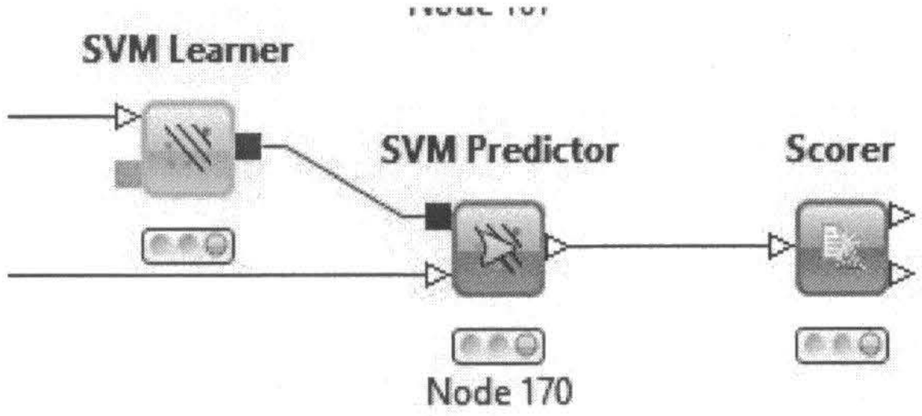


Figure 25 : Machine à vecteurs de support.

### 3.2.4.3 Arbre de décision (AD) :

De même pour l'arbre de décision nous effectuons les paramétrages suivants:

Quality measure : Gini index ( Knime utilise dans ce cas l'algorithme CART).

Pruning method : MDL

Min number records per node : 12

Une fois le paramétrage effectué nous passons à l'exécution, et voilà que notre classificateur est entrainé, maintenant nous allons juste insérer le nœud "Decision Tree Predictor" pour vérifier l'efficacité de notre classificateur avec les données test. Ce dernier ne nécessite pas de paramétrage particulier.

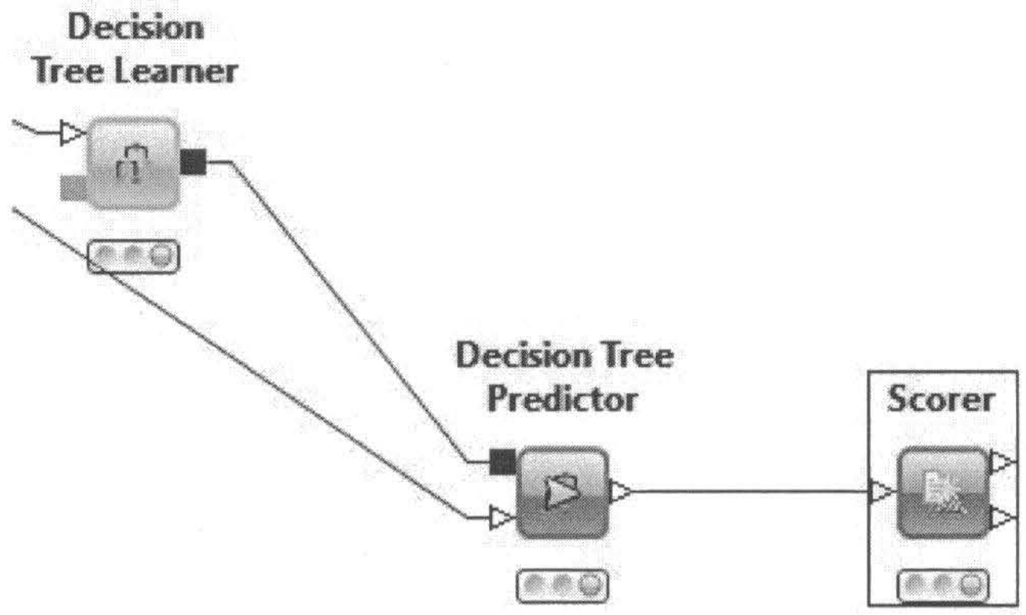


Figure 26 : Arbre de décision.

### 3.2.5 Les Résultats :

#### 3.2.5.1 Cas de deux classes :

**A- Classificateur de Naïf Bayes (CNB):**

| | | **Classes prédites** | |
|---|---|---|---|
| | | C02 | C01 |
| **Classes Réelles** | C02 | 10 | 0 |
| | C01 | 2 | 8 |

Tableau 2 : Matrice de confusion CNB Bi-Classes

D'après la matrice de confusion du classificateur de Bayes Bi-classes, nous voyons bien que parmi les dix(10) documents appartenant à la classe C01, huit(08) documents étaient bien classés tandis que deux (02) documents sont mal classés étant donné qu'ils sont attribués à classe C02.
Par contre tous les documents (10) de la classe C02 sont très bien classés.
Le tableau ci-dessous récapitule toutes les mesures que nous avons calculés à partir de la matrice de confusion.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 10 | 2 | 8 | 0 | 1 | 0.833 | 0.909 | 18 | 2 | 90% | 10% |
| C01 | 8 | 0 | 10 | 2 | 0.8 | 1 | 0.889 | | | | |

Tableau 3 : Tableau récapitulatif CNB Bi-classes.

**B- Machine à Vecteurs de Support polynomiale :**

| | | **Classes prédites** | |
|---|---|---|---|
| | | C02 | C01 |
| **Classes Réelles** | C02 | 9 | 1 |
| | C01 | 1 | 9 |

Tableau 4 : Matrice de confusion MVS polynomiale Bi-Classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 9 | 1 | 9 | 1 | 0.9 | 0.9 | 0.9 | 18 | 2 | 90% | 10% |
| C01 | 9 | 1 | 9 | 1 | 0.9 | 0.9 | 0.9 | | | | |

Tableau 5 : Tableau récapitulatif MVS polynomiale Bi-Classes.

**C- Machine à Vecteurs de Support linéaire :**

| | | **Classes prédites** | |
|---|---|---|---|
| | | C02 | C01 |
| **Classes Réelles** | C02 | 23 | 25 |
| | C01 | 9 | 118 |

Tableau 6 : Matrice de confusion MVS linéaire Bi-Classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 23 | 9 | 118 | 25 | 0.479 | 0.719 | 0.575 | 141 | 34 | 80.571% | 19.429% |
| C01 | 118 | 25 | 23 | 9 | 0.929 | 0.825 | 0.874 | | | | |

Tableau 7 : Tableau récapitulatif MVS linéaire Bi-Classes.

**D- Arbre de Décision (AD):**

| | | **Classes prédites** | |
|---|---|---|---|
| | | C02 | C01 |
| **Classes Réelles** | C02 | 5 | 5 |
| | C01 | 0 | 10 |

Tableau 8 : Matrice de confusion AD Bi-Classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 5 | 0 | 10 | 5 | 0.5 | 1 | 0.667 | 15 | 5 | 75% | 25% |
| C01 | 10 | 5 | 5 | 0 | 1 | 0.667 | 0.8 | | | | |

Tableau 9 : Tableau récapitulatif AD Bi-Classes.

### 3.2.5.2 Cas de trois classes :

**A- Classificateur de Naïf Bayes (CNB):**

| | | **Classes prédites** | | |
|---|---|---|---|---|
| | | C02 | C03 | C01 |
| **Classes Réelles** | C02 | 13 | 0 | 3 |
| | C03 | 2 | 12 | 2 |
| | C01 | 2 | 3 | 11 |

Tableau 10 : Matrice de confusion CBN trois classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 13 | 4 | 28 | 3 | 0.812 | 0.765 | 0.788 | 36 | 12 | 75% | 25% |
| C03 | 12 | 3 | 29 | 4 | 0.75 | 0.8 | 0.774 | | | | |
| C01 | 11 | 5 | 27 | 5 | 0.688 | .0688 | 0.688 | | | | |

Tableau 11 : Tableau récapitulatif CNB trois classes.

**B- Machine à Vecteurs de Support polynomiale :**

| | | **Classes prédites** | | |
|---|---|---|---|---|
| | | C02 | C03 | C01 |
| **Classes Réelles** | C02 | 14 | 0 | 2 |
| | C03 | 1 | 14 | 1 |
| | C01 | 3 | 3 | 10 |

Tableau 12 : Matrice de confusion MVS polynomiale trois classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 14 | 4 | 28 | 2 | 0.875 | 0.778 | 0.824 | 38 | 10 | 79.167% | 20.833% |
| C03 | 14 | 3 | 29 | 2 | 0.875 | 0.824 | 0.848 | | | | |
| C01 | 10 | 3 | 29 | 6 | 0.625 | 0.769 | 0.69 | | | | |

Tableau 13 : Tableau récapitulatif MVS polynomiale trois classes.

**C- Machine à Vecteurs de Support linéaire :**

| | | **Classes prédites** | | |
|---|---|---|---|---|
| | | C02 | C03 | C01 |
| **Classes Réelles** | C02 | 22 | 0 | 26 |
| | C03 | 1 | 71 | 13 |
| | C01 | 6 | 15 | 106 |

Tableau 14 : Matrice de confusion MVS linéaire trois classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 22 | 7 | 205 | 26 | 0.458 | 0.759 | 0.571 | 199 | 61 | 76.538% | 23.462% |
| C03 | 71 | 15 | 160 | 14 | 0.835 | 0.826 | 0.83 | | | | |
| C01 | 106 | 39 | 94 | 21 | 0.835 | 0.731 | 0.779 | | | | |

Tableau 15 : Tableau récapitulatif MVS linéaire trois classes.

**D- Arbre de Décision (AD):**

| | | Classes prédites | | |
|---|---|---|---|---|
| | | C02 | C03 | C01 |
| **Classes Réelles** | C02 | 7 | 3 | 6 |
| | C03 | 2 | 14 | 0 |
| | C01 | 1 | 7 | 8 |

Tableau 16 : Matrice de confusion AD trois classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 7 | 3 | 29 | 9 | 0.438 | 0.7 | 0.538 | 29 | 19 | 60.417% | 39.583% |
| C03 | 14 | 10 | 22 | 2 | 0.875 | 0.583 | 0.7 | | | | |
| C01 | 8 | 6 | 26 | 8 | 0.5 | 0.571 | 0.533 | | | | |

Tableau 17 : Tableau récapitulatif AD trois classes.

### 3.2.5.3 Cas de Quatre classes :

**A- Classificateur de Naïf Bayes (CNB):**

| | | Classes prédites | | | |
|---|---|---|---|---|---|
| | | C02 | C04 | C01 | C05 |
| **Classes Réelles** | C02 | 3 | 2 | 1 | 0 |
| | C04 | 0 | 5 | 1 | 0 |
| | C01 | 0 | 1 | 5 | 0 |
| | C05 | 1 | 2 | 0 | 3 |

Tableau 18 : Matrice de confusion CNB quatre classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 3 | 1 | 17 | 3 | 0.5 | 0.75 | 0.6 | 16 | 8 | 66.667% | 33.333% |
| C04 | 5 | 5 | 13 | 1 | 0.833 | 0.5 | 0.625 | | | | |
| C01 | 5 | 2 | 16 | 1 | 0.833 | 0.714 | 0.769 | | | | |
| C05 | 3 | 0 | 18 | 3 | 0.5 | 1 | 0.667 | | | | |

Tableau 19 : Tableau récapitulatif CNB quatre classes.

**B- Machine à Vecteurs de Support polynomiale :**

| | | **Classes prédites** | | | |
|---|---|---|---|---|---|
| | | C02 | C04 | C01 | C05 |
| **Classes Réelles** | C02 | 4 | 2 | 0 | 0 |
| | C04 | 0 | 5 | 0 | 1 |
| | C01 | 0 | 2 | 4 | 0 |
| | C05 | 1 | 0 | 0 | 5 |

Tableau 20 : Matrice de confusion MVS polynomiale quatre classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 4 | 1 | 17 | 2 | 0.667 | 0.8 | 0.727 | 18 | 6 | 75% | 25% |
| C04 | 5 | 4 | 14 | 1 | 0.833 | 0.556 | 0.667 | | | | |
| C01 | 4 | 0 | 18 | 2 | 0.667 | 0.1 | 0.8 | | | | |
| C05 | 5 | 1 | 17 | 1 | 0.833 | 0.833 | 0.883 | | | | |

Tableau 21 : Tableau récapitulatif MVS polynomiale quatre classes.

### C- Machine à Vecteurs de Support linéaire :

| | | Classes prédites | | | |
|---|---|---|---|---|---|
| | | C02 | C04 | C01 | C05 |
| **Classes Réelles** | C02 | 19 | 5 | 23 | 1 |
| | C04 | 3 | 112 | 15 | 8 |
| | C01 | 6 | 11 | 93 | 10 |
| | C05 | 1 | 12 | 17 | 55 |

Tableau 22 : Matrice de confusion MVS linéaire quatre classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 19 | 10 | 333 | 29 | 0.396 | 0.655 | 0.494 | 279 | 112 | 71.355% | 28.645% |
| C04 | 112 | 28 | 225 | 26 | 0.812 | 0.8 | 0.806 | | | | |
| C01 | 93 | 55 | 216 | 27 | 0.775 | 0.628 | 0.694 | | | | |
| C05 | 55 | 19 | 287 | 30 | 0.647 | 0.743 | 0.692 | | | | |

Tableau 23 : Tableau récapitulatif MVS linéaire quatre classes.

### D- Arbre de Décision (AD):

| | | Classes prédites | | | |
|---|---|---|---|---|---|
| | | C02 | C04 | C01 | C05 |
| **Classes Réelles** | C02 | 4 | 0 | 0 | 2 |
| | C04 | 0 | 2 | 0 | 4 |
| | C01 | 0 | 0 | 3 | 3 |
| | C05 | 0 | 0 | 1 | 5 |

Tableau 24 : Matrice de confusion AD quatre classes.

| Mesures / Classes | VP | FP | VN | FN | Rappel | Précision | F-Mesure | Bien Classés | Mal classés | Taux de succès | Taux d'erreur |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C02 | 4 | 0 | 18 | 2 | 0.667 | 1 | 0.8 | 14 | 10 | 58.333% | 41.667% |
| C04 | 2 | 0 | 18 | 4 | 0.333 | 1 | 0.5 | | | | |
| C01 | 3 | 1 | 17 | 3 | 0.5 | 0.75 | 0.6 | | | | |
| C05 | 5 | 9 | 9 | 1 | 0.833 | 0.357 | 0.5 | | | | |

Tableau 25 : Tableau récapitulatif AD quatre classes.

### 3.2.5.4 Discussion:

À première vue, nous remarquons que la Machine à vecteurs de support(MVS) se distingue et occupe la première place de part sa performance dans les trois cas de figures. Ainsi, elle confirme sa bonne réputation dans la littérature comme étant l'un des meilleurs classificateurs.

Nous avons défini un peu plus haut que MVS est un classificateur dit linéaire que nous utilisons dans la classification Bi-classes, mais nous avons évoqué également sa flexibilité par rapport aux données non linéairement séparables et surtout son adaptation au cas de la classification multi-classes avec une variante (autre implémentation) de son algorithme. Cette variante de MVS consiste à simplifier le problème de multi-classes en un problème de Bi-classes, du coup, différentes approches ont vu le jour, entre autre, l'approche "un contre tous" et l'approche "un contre un"[8].

Knime utilise la variante MVS multi-classes avec noyau pour parer à la fois au problème des données non linéairement séparables et celui des multi-classes, malheureusement il nous offre juste le choix entre trois noyaux, à savoir, Polynomial, Hyper Tangent et RBF(Radial Basis Function). L'absence de MVS linéaire nous pénalise en quelque sorte, car avec le corpus Oshumed que nous avons utilisés, les documents sont linéairement séparables [9]. Toutefois, la flexibilité de KNIME et son interaction avec d'autres logiciels nous offre la possibilité d'explorer d'autres alternatives, notamment le logiciel WEKA.

Malgré que les performances de MVS polynomiale dépassent de loin celles de l'arbre de décision (AD), on note que son temps d'exécution est nettement plus grand que celui de l'AD. Cet état de fait nous a conduit à utiliser le MVS linéaire disponible sur WEKA et qui a donné de très bonnes performances tant sur le plan du taux d'erreurs de classification que sur celui du temps d'exécution.

Deux choses qui nous ont surpris dans cette étude, la première est la performance inattendue du CNB, qui était un sérieux concurrent pour la MVS. En effet, ce que nous croyons connaitre à propos du CNB, c'est qu'il est très performant avec la classification des documents dits courts, qui comportent un petit nombre de mots, notamment dans le cas des Spams et Hams, ce qui semble tout à fais logique, puisque avec un volume très réduit de descripteurs, l'hypothèse de l'indépendance de ces derniers est facilement vérifiable. Notre étude a démontré que CNB est performant également avec les documents longs, cela dû à son implémentation particulière sous KNIME, comme dans le cas de MVS.

La deuxième étant notre déception par rapport à AD, pourtant réputé comme étant une solution alternative pour les problèmes de classification multi-classes. En effet, au fur et à mesure que nous ajoutons de catégories, les performances de AD se détériorent davantage. Notre première

interprétation qui semble logique, est le fait que notre AD est soumis à une très grande dimension de descripteurs qui a conduit à son sur-apprentissage. Le recours à une technique de réduction de dimension comme l'ACP (Analyse à Composantes Principales) ou l'ADL (Analyse Discriminante Linéaire), aurait amélioré ses performances. La manière avec laquelle nous avons représenté les documents (bag of word) ne s'adhère pas forcément avec AD. La méthode N-gram ou Keygraph Keyword aurait amélioré considérablement les résultats.

# Conclusion

La prolifération des outils du data mining et leurs interfaces très conviviales, à l'exemple de KNIME, semblent simplifier davantage des tâches, comme la classification de documents et la rendent plus facilement compréhensibles. Cependant il ne faut pas tomber dans cette simplicité, car la maitrise des mécanismes et de la logique du data mining est plus primordiale que la maitrise de ces logiciels. En réalité, lorsque nous traitons les problèmes de classification de textes avec une manière très simpliste, les résultats peuvent être incohérents et moins significatifs.

Tout au long de cette étude, nous avons remarqué que la manipulation des petits paramètres peut avoir un grand impacte sur le résultat final. Cette délicatesse nous oblige à essayer de résoudre les problèmes de la classification de documents avec une approche particulière; ceci veut dire au cas par cas, ains, certaines pondérations marchent très bien avec certains cas mais donnent des résultats peu luisant avec d'autres. Même constatation pour les algorithmes ils sont souvent spécifiques à des tâches bien définies.

Évoquer la simplicité dans le traitement de la tâche de classification, dans notre cas, n'est pas un hasard ni un subterfuge, c'est un fait. Notre modèle est vraiment simple.

Notre omission volontaire d'utiliser des mécanismes de réduction de dimensions tel que l'ACP ou l'ADL, le fait de ne pas utiliser d'autres méthodes de définition des descripteurs, à l'image des N-grams et du Graphe de mots clé (Keygraph Keywords), semblent confirmer nos résultats (taux de succès et d'erreurs). Or ce qui nous intéresse dans cette étude c'est la comparaison de ces trois classificateurs et non pas la construction d'un classificateur aussi performant qui soit.

Finalement, du moment que les trois classificateurs sont testés dans les mêmes conditions, avec les mêmes données, que ce soit au niveau du volume que dans le processus de leur prétraitement, nous pouvons avancer que l'objectif est pleinement atteint et nos résultats semblent être en concordance avec ceux de la littérature.

Notons aussi que travailler avec KNIME était très passionnant et très enrichissant, car une fois notre modèle bâtit nous pourrons réaliser une multitude de modifications et de paramétrages, et l'impacte de ces derniers est presque instantanés.

# Annexe A

## Workflow de Knime

Il s'agit des trois modèles de classificateurs selon le nombre de classes que nous avons utilisés. Notons tout de même qu'il faut recharger les données d'une source local ensuite exécuter le Workflow en totalité.

Mais avant d'importer les Workflows, nous devons installer le logiciel Knime, disponible à l'adresse suivante: https://www.knime.org/downloads/overview.

# Annexe B

## Le corpus Ohsumed

C'est le corpus de documents que nous avons utilisé pour entrainer et tester nos classificateurs. Il comporte 23 classes, mais dans notre cas juste les classes C01, C02, C04, C05 ont été utilisées. Un fichier décrivant l'appellation de chacune des catégories est joint avec le corpus.

# Bibliographie

[1] Bara M. et Nanceri C., «Le texte mining, un outil pour le marketing bancaire», Banque et informatique, 1999.

[2] René Lefébre et Gilles Venturi., «Data mining, gestion de la relation client, personnalisation de sites Web», Eyrolles, 200, pp. 629-638.

[3] https://www.neurones.espci.fr.

[4] Stéphane TUFFÉRY. «Data Mining et statistique décisionnelle, L'intelligence des données», TECHNIP 2010, pp. 297-515.

[5] Daniel T.Larose. Traduction et adaptation de Thierry Vallaud. «Des données à la connaissance, Une introduction au data mining », Vuibert, Paris, 2005, pp. 105-126.

[6] https://www.knime.org.

[7] http://disi.unitn.it.

[8] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04,Royal Holloway, University of London, Department of Computer Science, 1998.

[9] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. Technical Report 23, University of Dortmund, LS VIII, 1997.