République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU **G**ENIE **E**LECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Mathématiques et Informatique**
Filière : **Informatique**
Spécialité : **Conduite de projet informatique**

Présenté par
Sarah DAOUDI
Nassima CHERRAD

Thème

Comparaison des différentes approches de classification automatique des documents textuels.

Mémoire soutenu publiquement le 29/09/ 2016 devant le jury composé de :

Président : Mme Dalila TAOURI
Encadreur : M^elle^ Samia Iltache
Examinateur : M^elle^ Samia AIT ADDA
Examinateur : Mohamed Nabil AMIROUCHE

"Aucun de nous ne s'est élevé à la seule force de son poignet. Nous sommes arrivés parce que quelqu'un s'est baissé pour nous aider. "

Thurgood Marshall

Au terme de ce travail,

Nos remercions avant tout le bon Dieu tout puissant de nous avoir donné patience, courage et volonté pour réussir notre mémoire,

Nous tenons à exprimer notre profonde gratitude à Mlle **Iltache Samia**, enseignante à l'UMMTO, pour avoir encadré et dirigé notre travail. Nous la remercions pour ses conseils et ses remarques constructives qui nous ont permis d'améliorer la qualité de nos travaux,

Nos vifs remerciements vont aux membres du jury pour nous avoir fait l'honneur d'examiner et d'évaluer notre travail avec le poids de leurs compétences,

Un merci particulier à Mlle **Cherdioui Sabrina** et Mlle **Mestour Linda**, doctorantes à l'UMMTO, pour leur disponibilité, leur gentillesse, leur écoute et leurs conseils pertinents qui nous ont été d'une aide précieuse pour mener à bien ce travail,

A tous les enseignants qui ont assuré notre formation durant notre parcours universitaire, nous souhaitons qu'ils trouvent dans ce travail l'expression de notre infinie reconnaissance,

Nous souhaitons également exprimer notre gratitude à tous ceux qui de près ou de loin ont participé à l'élaboration du présent travail.

C'est avec un immense plaisir que je dédie ce travail,

A ma fierté, ma chère et mon adorable mère que j'adore, pour son sacrifice, son soutien tout au long de mes études et pour toutes les valeurs magnifiques qu'elle m'a donné durant toute ma vie,

*A la mémoire de mon cher père et ma chère sœur « **Karima** », que Dieu les garde dans son vaste paradis,*

*A mon cher fiancé **Rachid** et toute sa famille,*

A mes chers sœurs et frères ainsi que leurs familles,

*A ma très chère amie **Sarah** et toute sa famille,*

A tous mes amis sans exception.

Nassima

Je dédie ce travail,

A mes chers parents, qui ont toujours été à mes côtés et ont toujours cru en moi. Pour leur amour, et leur soutien permanent tout au long de mes années d'études. Que Dieu vous protège,

A mes chers beaux-parents, pour leur soutien précieux, leurs encouragements répétés et leur immense affection,

A mon magnifique fiancé, qui n'a jamais cessé de m'encourager, de me soutenir, qui a toujours été là, dans les bons comme dans les mauvais moments. Pour son aide précieuse et son amour immense,

A mes frères et sœurs, qui de par leur amour et leur soutien au quotidien ont contribués à la réalisation de ce travail,

A mon beau-frère et ma magnifique belle-sœur,

A tous mes amis, qui m'ont toujours soutenu,

Ainsi qu'à tous mes camarades du département informatique.

Sarah

Sommaire

Chapitre II: Classification

Chapitre III : présentation du logiciel Weka

Chapitre IV : implémentation et représentation des résultats

Liste des figures

Liste des tableaux

Liste des Annexes

INTRODUCTION GÉNÉRALE

Introduction générale

L'information joue inévitablement un rôle vital dans la société d'aujourd'hui. Effectivement, nous sommes privilégiés dans un monde riche en information, dans lequel la plupart, si ce n'est la totalité, de l'information dont nous avons besoin est au bout de nos doigts et est prête à être exploitée. Le texte a toujours été considéré comme le moyen le plus sûr pour stocker et pérenniser l'information ou la connaissance, mais vu sa prolifération incessante dans tous les domaines, se pose le nouveau problème de l'extraction de ces informations afin de les exploiter.

Historiquement, la croissance du volume de données textuelles comme les livres et les articles dans les bibliothèques durant des siècles a imposé de définir des mécanismes efficaces pour les localiser. Les premières techniques, comme l'indexation et l'utilisation des catégories de classification ont marqué le monde de l'information électronique. D'énormes efforts ont été déployés depuis, comme le montre la littérature, pour développer des approches et des techniques permettant de retrouver l'information voulue effectivement et efficacement à partir de vastes collections de données textuelles.

L'envie d'organiser pour simplifier a progressivement évolué vers l'ambition de classifier pour comprendre et, pourquoi pas pour prédire. Cette évolution a conduit à la création d'outils et de stratégies de classification automatique des données. La classification s'apparente toutefois, au problème de l'extraction de la sémantique d'un texte, puisque l'appartenance d'un document à une catégorie est étroitement liée à la signification de ce texte. De plus, les algorithmes de classification ne sont pas capables de traiter directement les textes dans leur forme brute, c'est pourquoi, une étape préliminaire dite *Indexation* est nécessaire. Cette étape consiste généralement en la représentation de chaque document par un vecteur, dont les composantes sont par exemple les mots contenus dans le texte, afin de le rendre exploitable par les algorithmes d'apprentissage. Une collection de textes peut être ainsi représentée par une matrice dont les lignes sont les termes pondérés les plus représentatifs, et les colonnes sont les documents de cette collection.

On distingue deux approches de classifications automatiques : supervisée et non-supervisée. Les deux approches partagent un but commun mais, à la différence de la classification non supervisée où l'ordinateur doit découvrir lui-même des groupes de documents, la classification supervisée suppose qu'il existe déjà une classification de documents, autrement dit, il existe au départ un échantillon dit d'apprentissage dont le classement est connu. Cet échantillon est utilisé pour l'apprentissage des règles de classement.

L'objectif de notre travail consiste à faire une comparaison entre les différentes méthodes de classification supervisée, à savoir, la méthode SVM, KNN, l'Arbre de décision et la méthode Naïve Bayes, et ceci, en les appliquant sur deux collections distinctes de documents textuels. A l'issus de ce travail, on va analyser les résultats afin d'évaluer les performances de chaque

algorithme dans chaque collection. Et pour la réalisation de cette expérimentation, on utilisera la boite à outil WEKA.

Le présent mémoire s'articule autour de quatre chapitres. Le premier est consacré à l'indexation, son processus, ses concepts et techniques de base et ses différentes étapes. Dans le deuxième, on trouve une introduction à la classification, ses différentes approches, ses méthodes et algorithmes en se basant essentiellement sur l'approche de classification supervisée. Ensuite, le troisième chapitre traitera du logiciel Weka, de ses différentes fonctionnalités et interfaces en se focalisant particulièrement sur son interface principale, son mode d'utilisation, spécialement dans le domaine de la classification supervisée de documents. Enfin, le dernier est consacré aux outils de développement utilisés, les étapes d'implémentation de notre travail et l'interprétation des résultats obtenus.

CHAPITRE I

Indexation de documents

I.1. Introduction

Les documents dans leur forme texte libre sont difficiles à exploiter par l'ordinateur. Un traitement préalable permettant leur représentation simplifiée est nécessaire : c'est l'*indexation.* L'indexation est utilisée pour faciliter la recherche, la classification et l'organisation des objets documentaire. Elle est une étape très importante dans le processus de classification de documents. Elle consiste à déterminer et extraire les termes représentatifs du contenu d'un document. La qualité de la classification dépend en grande partie de la qualité de l'indexation. Ainsi pour chaque document de notre collection il est nécessaire d'indexer son contenu.

Ce chapitre a pour but de présenter le processus d'indexation de manière générale, ses étapes, ses caractéristiques ainsi que ses concepts de base.

I.2. Définitions

- « L'*indexation* consiste à transformer des documents en substituts capables de représenter le contenu de ces documents ». [1]

- « L'*indexation* est un processus destiné à représenter, au moyen des termes ou indices d'un langage documentaire ou au moyen des éléments d'un langage libre, les notions caractéristiques du contenu d'un document (ressource, collection), en vue d'en faciliter la recherche, après les avoir identifiées par l'analyse». [2]

- « L'*indexation* est une opération intellectuelle par laquelle on va codifier le contenu d'un document. Cette opération consiste à analyser le contenu de ce document, et de le transcrire dans un langage documentaire ». [3]

- « Mathématiquement, un index est une fonction qui relie chaque document D_i à l'ensemble des mots clés du corpus T, décrivant le thème qu'il traite ». [4]

$$Index : D_i \xrightarrow{traite} \{t_j\} \text{ où } \{t_j\} \in 2^T$$

- De manière générale, l'*indexation* consiste à associer à chaque document une liste de mots-clés appelée aussi descripteur, susceptible de représenter au mieux le contenu des documents.

La finalité de l'indexation est donc de produire une représentation synthétique des documents, formés de termes ou groupe de termes significatifs pour l'unité textuelle correspondante, comme illustré dans la figure I.1:

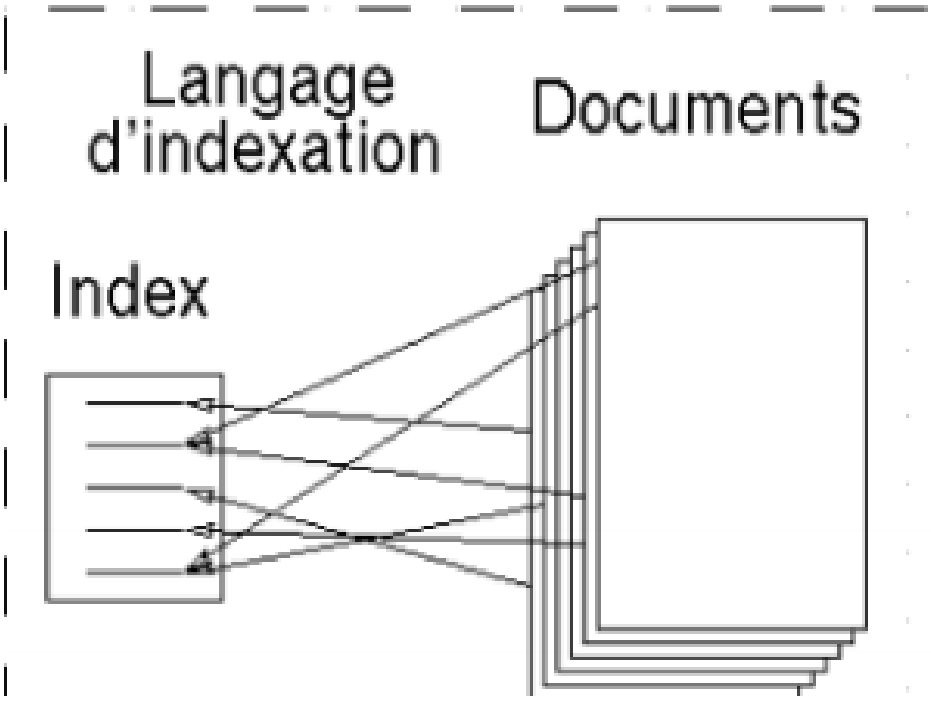

Figure I.1 : *Indexation*

I.3. Les modes d'indexation

L'indexation peut être manuelle, automatique ou semi-automatique :

I.3.1. Indexation manuelle

Dans ce cas, chaque document est analysé par un spécialiste du domaine ou par un documentaliste. Après la lecture des documents, ce spécialiste détermine, selon ses connaissances, les mots-clés qui lui semblent les plus adéquats pour représenter le contenu du document. Ce mode d'indexation est fondé sur le jugement humain. Il se caractérise par sa profondeur, sa cohérence et sa qualité. Cependant, il dépend de l'indexeur ce qui induit la subjectivité de ses résultats. De plus l'augmentation du nombre de documents à indexer rend la tâche d'indexation manuelle difficile et coûteuse en temps et en nombre de personnes impliquées, et est difficile à maintenir du fait de l'évolution de la terminologie.

I.3.2. Indexation automatique

Ce mode d'indexation est entièrement informatisé. Il se base sur des procédures algorithmiques pour accomplir les tâches des indexeurs humains et met en œuvre un ensemble de techniques informatisées issues de *Traitements Automatiques de la Langue Naturelle* ***(TALN)***. Il détecte automatiquement les termes les plus représentatifs du contenu du document. Ici, le processus d'indexation tente d'atteindre une position optimale entre deux objectifs contradictoires, l'exhaustivité et la spécificité. Le premier a pour but de présenter la description la plus complète possible d'un document, alors que l'autre a pour objectif une meilleure différenciation entre les

descriptions des documents pour mieux les distinguer, ce type d'indexation est actuellement la méthode la plus répandue.

I.3.3. Indexation semi-automatique

C'est une combinaison des deux approches d'indexation précédente. Elle consiste à tirer profit des deux types d'indexation manuelle et automatique. L'indexation automatique donne les premiers éléments d'indexation, puis le spécialiste du domaine ou le documentaliste corrige, complète les informations obtenues. [5]

Le choix d'un mode par rapport à l'autre dépend d'un certain nombre de facteurs, dont le plus déterminant est le volume des collections.

I.4. Domaine d'application de l'indexation

L'indexation est la pierre angulaire de nombreux processus de gestion des connaissances, elle est utilisée :

- lors d'une recherche documentaire à travers un moteur qui indexe une base de documents : tous les documents indexés par les mêmes mots clés du langage documentaire sont trouvés, indépendamment de la langue ou de la présence ou non des mots dans le reste du document.

- Lors de l'analyse de corpus (*Text Mining*) : les mots clés sont utilisés pour analyser un corpus textuel avec des méthodes statistiques ou linguistiques et permettent d'obtenir des résultats précis.

I.5. Eléments d'indexation

I.5.1. Document

Un document est un élément essentiel en indexation. L'une des définitions possible du terme document est de le considérer comme un support physique de l'information, qui peut être du texte, une page web, une image, une séquence vidéo, etc.

Dans le cas d'un document texte on peut le représenter selon trois vues :

- **La vue sémantique (ou contenu) :** elle se concentre sur l'information véhiculée dans le document

.

- **La vue logique :** elle définit la structure logique du document (structuration en chapitres, sections)

- **La vue présentation :** elle consiste en la présentation sur un médium à deux dimensions (alignement de paragraphes, indentation, en-têtes et pieds de pages, etc.).

L'ensemble des documents manipulés se nomme collection de documents (ou base documentaire ou encore corpus).

L'objectif de l'indexation documentaire peut être de:

- Classer les documents dans une catégorie ;
- Explorer le contenu des documents ;
- Extraire des informations : identification d'éléments sémantiques dans un texte ;
- La construction de résumés (Summarisation) : abstraction et condensation d'un texte pour élaborer une version réduite conservant au maximum la sémantique ;
- La Recherche d'Information : c'est une étape primordiale en RI. Et la qualité de la recherche dépend en grande partie de la qualité de l'indexation.

I.5.2. Corpus

Un **corpus** est un ensemble de documents, artistiques (textes, images, vidéos, etc.), regroupés dans une optique précise. La notion de corpus n'est pas nouvelle, mais sa version moderne, celle d'un ensemble de textes authentiques assemblés sous forme électronique selon un certain nombre de critères, avait besoin de l'informatique pour que s'épanouissent toutes ses potentialités.

Un corpus peut prendre différents formats :

- **Texte brut :** Une suite de lettres et de chiffres, sans mise en forme, interprétable par tous les éditeurs de textes ;
- **PDF :** Langage de description de pages qui préserve la mise en forme ;
- **HTML :** langage interprété par les navigateurs internet, doté d'une mise en forme et d'une structuration du texte ;
- **Texte enrichi d'annotation**
  - **Extended Markup Language – XML** (délimitation de la zone du texte marqué par l'annotation).
  - **Format d'annotation de textes** : **TEI** (Text Encoding Initiative).

I.5.3. Thésaurus

Un thesaurus est une liste de termes contrôlés et normalisés, représentant chacun un concept unique d'un domaine de connaissance. Ces termes, appelé « descripteurs » (ou termes préférentiels), et leurs synonymes (ou « non-descripteurs ») sont organisés selon les relations qui structurent leur champs sémantique : relations hiérarchiques, d'équivalence (synonymie) ou d'association. Ainsi, un terme générique peut avoir un ou plusieurs synonymes, un ou plusieurs termes spécifiques – représentant des cous-concepts liés, et être associé à un ou plusieurs termes d'autres champs sémantiques.

Le thésaurus sert à réduire la variabilité des notions exprimées en langage naturel. Les relations aident à s'assurer du sens du concept et de son adéquation pour l'énoncé d'un sujet.

Un thésaurus s'organise très typiquement au tour des éléments et procédures suivants :

- **Des descripteurs (non-descripteurs) :** le descripteur est un mot ou un groupe de mots choisi (parmi, en général, un ensemble de mots, « candidats » pour représenter, exprimer une notion (un thème) dans le thésaurus. La classe des non-descripteurs couvrent par contre, les expressions qui ne sont pas employées comme descripteurs mais qui auraient pu l'être, autrement dit, qui servent également à exprimer un thème, une notion.

- **Un vocabulaire normalisé :** tout thésaurus repose sur un vocabulaire « Contrôlé », « normalisé ». Ceci est nécessaire pour des raisons à la fois techniques et de gestion.

- **Des relations principales qui organisent le vocabulaire :** comme connu, il existe un canon de relations « lexico-sémantique » pour structurer, organiser les descripteurs. Les plus souvent utilisés se résume dans le tableau suivant :

Tableau I.1 : *Relations lexico-sémantique*

Relation général/spécifique	
TG	terme dit générique, i. Général
TS	terme spécifique
Relation dite associative	
TA	terme associatif
Relation de restriction	
NA	note d'application

Un exemple de thésaurus :

- **Wordnet** : qui est une base de données lexicale pour la langue anglaise. Il est une combinaison entre dictionnaire et thésaurus.

I.6. Processus d'indexation

Il existe au moins trois grands niveaux dans le processus d'indexation, à savoir :

- **Niveau du découpage :** Tokénisation appelée aussi segmentation qui consiste à diviser un texte en unités lexicales (token) élémentaires ;
- **Niveau morphologique :** reconnaissance du mot. Chaque mot de la langue lui correspond une catégorie morpho syntaxique (lemme, racine, mot composé,…) ;
- **Niveau lexical :** réduction du mot à sa forme canonique ⟶ lemmatisation, racinisation.

I.6.1. Analyse lexicale (*tokenization*)

L'analyse lexicale est la première phase de l'indexation automatique. C'est une étape indispensable qui consiste en l'identification des unités élémentaires. Elle permet de convertir un texte de document en une liste de termes. Un terme est un groupe de caractères constituant un mot significatif [6]. L'analyse lexicale permet de reconnaître les espaces de séparation des mots, les chiffres, les ponctuations, etc. Ces unités (tokens ou termes) seront les candidats à l'indexation.

I.6.2. Elimination des ponctuations

Prétraitement qui consiste à éliminer tout symbole qui ne correspond pas à une lettre de l'alphabet (point, virgule, trait d'union, point d'interrogation,…). Cette opération est motivée par le fait que ces caractères ne sont pas liés au contenu des documents et ne change rien au sens s'ils sont omis et par conséquents ils peuvent être négligés.

I.6.3. Elimination des mots vides (*stopwords*)

Les mots vides (article, proposition, conjonction, etc.) sont des mots à haute fréquence mais peu porteur de sens dans un document, car ils ne traitent pas le sujet du document. C'est pourquoi on introduit des opérations de filtrage de ces mots pour une meilleure représentation des documents.

- **Exemples de mots vides**

  Anglais : *a, about, above, according, across, after, again, as, at, in, with, some, of, to*
  Français : *de, un, les, le, la*

On distingue deux techniques pour éliminer les mots vides :

- L'utilisation d'une liste préétablie de mots vides, appelée aussi *anti-dictionnaire* ou *stop-list* [7];
- L'utilisation de mesures statistiques sur la collection de documents pour déterminer les mots les plus fréquents ou les mots rares, qui sont « probablement » des mots vides.[7]

L'élimination des mots vides réduit la taille de l'index, ce qui améliore le temps de réponse du système et augmente sa performance, de ce fait, certains systèmes considèrent, aussi, comme des mots vides quelques verbes, adjectifs et adverbes.

Bien que ce traitement présente l'avantage d'améliorer la représentation des documents en éliminant les mots non significatifs, il peut cependant induire des effets de silence (comme par exemple en éliminant le mot **a** de *vitamine a).*

I.6.4. L'étiquetage (POS Tagging)

C'est le processus qui consiste à associer aux mots d'un texte leur catégorie morphosyntaxique (*nom, verbe, adjectif, etc.*) à l'aide d'un outil informatique. L'étiquetage peut servir à éliminer les mots inutiles.

La figure suivante décrit le processus d'étiquetage :

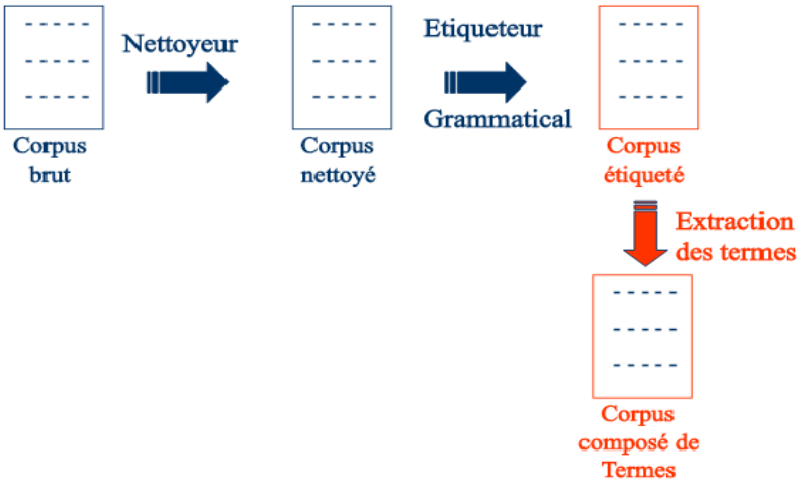

Figure I.2 : *processus d'étiquetage grammatical*

Les étiqueteurs grammaticaux sont très nombreux pour les langues saxonnes mais plus rares pour le français, on peut citer par exemple :

- **TreeTagger : (**IMS Stuttgart, Allemagne**)** [8], étiqueteur qui prend en entrée un texte brut ou HTML et en sortie on obtient un mot par ligne, suivi de l'étiquette. Accessible pour le français.

- **StandFord POSTagger :** Logiciel réalisé par l'Université Stanford, qui lit un texte et assigne les étiquettes à chaque mot (token) du texte.

- **LIA Tagg :** du Laboratoire Informatique d'Avignon.

Exemple *:* Stanford POSTagger

Texte original : *"This is a sample sentence"*

Texte étiqueté : `This/DT is/VBZ a/DT sample/NN sentence/NN`

La figure suivante représente un exemple d'étiquettes que peut prendre un mot :

Tag	Description
JJ	Adjective
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
POS	Possessive ending
PRP	Personal pronoun
RB	Adverb
RP	Particle
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle

Figure I.3 : *Etiquettes grammaticales*

I.6.5. La normalisation

La normalisation consiste à représenter les différentes variantes d'un terme par un format unique appelé lemme ou racine. Il permet de regrouper les variantes morphologiques d'un mot sous une forme de base unique. Son objectif est de ne garder, dans le langage d'indexation, que les formes normalisées des mots représentatifs, ce qui a pour effet de réduire la taille de l'index et d'offrir un gain d'espace mémoire considérable.

Plusieurs stratégies de normalisation sont utilisées : la table de correspondance, l'élimination des affixes (*algorithme de Porter*), la troncature, l'utilisation des N-grammes.

La normalisation se base sur l'une des deux procédures suivantes :

I.6.5.1 Lemmatisation

La lemmatisation permet de regrouper les mots de la même catégorie grammaticale et les transformer en leur forme canonique appelée *lemme* (*exemple* : les différentes formes d'un verbe sont transformés à son infinitif). L'objectif étant d'éliminer les variations morphologiques des mots (nombre, genre,…). Cette technique est basée sur l'utilisation des patrons syntaxiques et des dictionnaires (*exemple* : TreeTagger).

Exemple : l'adjectif *petit* existe sous quatre formes : *petit*, *petite*, *petits* et *petites*. La forme canonique de tous ces mots est « *petit* ».

I.6.5.2 Racinisation [*stemming*]

C'est un procédé de transformation des flexions en leur radical ou racine (*stem*). La racine d'un mot correspond à la partie du mot restant une fois que l'on a supprimé son (ses) préfixe(s) ou/et suffixe(s), à savoir son radical. Contrairement au lemme qui correspond à un mot réel de la langue, la racine ou stemme ne correspond généralement pas à un mot réel. Par exemple, le mot **« chercher »** a pour radical ou stemme **« cherch »** qui ne correspond pas à un mot réel. Par contre dans l'exemple de **« frontal »**, le radical ou stemme est **« front »** qui lui l'est. Cette technique est réalisée par l'utilisation des règles de transformation de type condition action (*exemple* : *l'algorithme de Porter* **[**9], est l'un des plus connus pour *l'anglais*) ou par une troncature des mots à x caractères.

Ces divers algorithmes de racinisation procèdent en deux étapes : un pas de *désuffixation* qui consiste à ôter aux mots des terminaisons prédéfinies les plus longues possibles, et un pas de *recodage* qui ajoute aux racines obtenues des terminaisons prédéfinies. L'algorithme de Porter fait les deux étapes en simultané.

Racinisation et *Lemmatisation* sont deux notions très proches, mais il y a des différences fondamentales :

- Les méthodes utilisées pour la lemmatisation et désuffixation ne sont pas les mêmes.
- La lemmatisation a pour objectif de retrouver le lemme d'un mot, par exemple l'infinitif pour les verbes.
- La racinisation consiste à supprimer la fin des mots, ce qui peut résulter en un mot qui n'existe pas dans la langue.
- **Racinisation :** Obtention d'une forme tronquée du mot, commune à toutes les variantes morphologiques.
- **Lemmatisation :** obtention de la forme canonique (le lemme) à partir du mot.

Quel que soit l'outil retenu, la façon de procéder est toujours la même : le stemmatiseur recherche selon la forme du mot fléchi et la langue défini, le radical le plus probable pour ce mot. Contrairement à la lemmatisation qui repose donc sur une base de connaissance des formes fléchies de la langue auxquelles on associe les lemmes possibles, la stemmatisation fonctionne uniquement avec une base de connaissance des règles syntaxiques et grammaticales de la langue.

La figure I.4 résume le processus de normalisation

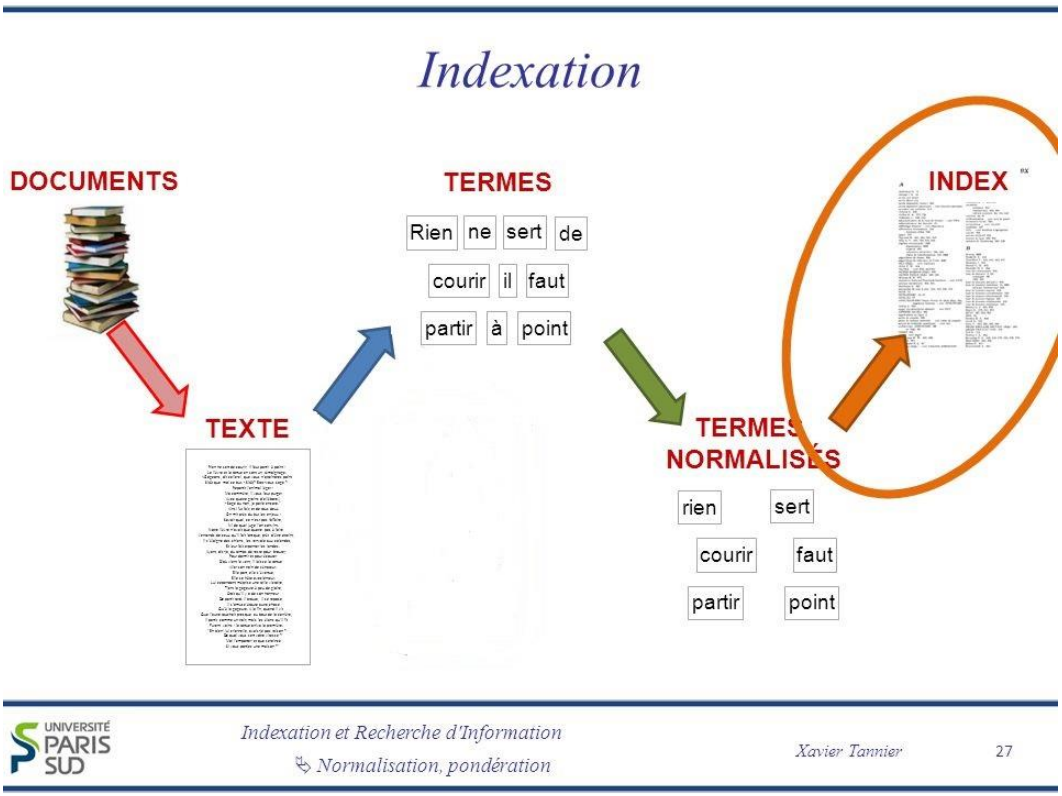

Figure I.4 : *normalisation*

I.7. Choix des descripteurs (mots clés)

Le but de l'indexation est l'extraction des descripteurs, et avec la représentation en sac de mots, chacun des mots d'un corpus est un descripteur potentiel. En général, il est admis que les mots les plus fréquents peuvent être supprimés : ils n'apportent pas d'information sur le sens d'un texte puisqu'ils sont présents sur l'ensemble des textes. Les mots très rares, sont également supprimés, car il n'est pas possible de construire des statistiques fiables à partir d'une ou deux occurrences. Cependant, même après la suppression de ces deux catégories de mots, le nombre de candidats reste encore très élevé, et il est nécessaire de sélectionner les descripteurs les plus représentatifs du contenu. On peut distinguer plusieurs types de descripteurs :

- **Des mots simples** du texte de document en éliminant les mots vides ;
- **Des lemmes** ou les racines des mots extraits ;

- **Des concepts**, qui sont des expressions contenant un ou plusieurs mots et pris généralement d'une structure conceptuelle, tels que les thésaurus ;

- **Des mots composés** : les groupes de mots ou expression sont souvent plus riches sémantiquement que les mots qui les composent pris séparément.

- **Des N-grammes** : un *n-gramme* est une succession de n lettres.
  Exemple : pour le mot « recherche », 1-gramme : r, e, c, h, e, r, c, h, e ; 2-gramme : re, ec, ch, he, er, rc, he.
  Les *n-grammes* permettent de reconnaître des mots de manière approximative et ainsi de corriger des flexions de mots ou même des fautes de frappe.

I.8. Pondération des termes

La pondération est une fonction fondamentale et un facteur prépondérant dans le processus d'indexation. Elle consiste à répondre à la question si tous les termes ont la même importance ? Et comment attribuer un poids aux termes extraits ? Autrement dit, elle nous permet de mesurer l'importance d'un terme dans un document en lui affectant un poids qui exprime son degré de représentativité.

I.8.1. Pondération en TF*IDF

Le schéma de pondération *tf***idf* [10] combine un facteur de pondération local *tf*, quantifiant la représentativité locale d'un terme dans le document, et un second facteur de pondération globale *idf*, mesurant la représentativité globale du terme vis-à-vis de la collection des documents.

La justification théorique de ce schéma de pondération repose sur l'observation empirique de la fréquence des mots dans un texte qui est donné par la *loi de Zipf* [11]. Si une requête contient le terme *T*, un document a d'autant plus de chances d'y répondre qu'il contient ce terme : la fréquence du terme au sein du document **(TF)** est grande. Néanmoins, si le terme *T* est lui-même très fréquent au sein du corpus, c'est-à-dire qu'il est présent dans de nombreux documents, il est en fait peu discriminant. C'est pourquoi le schéma propose d'augmenter la pertinence d'un terme en fonction de sa rareté au sein du corpus **(IDF)**. Ainsi, la présence d'un terme rare dans le contenu d'un document fait croître le « score » de ce dernier.

I.8.1.1. Pondération locale TF [Term Frequency]

Cette mesure est proportionnelle à la fréquence du terme dans le document. L'idée sous-jacente est que plus un terme est fréquent dans un document, plus il est important dans la description de ce document. Le TF est souvent exprimé selon la déclinaison suivante :

- **Fonction brute de tf_{ij} :** elle correspond au nombre d'occurrence du terme $\mathbf{t_i}$ dans le document $\mathbf{D_j}$

I.8.1.2. Pondération globale IDF [Inverse of Document Frequency]

Mesure l'importance d'un terme dans toute la collection. L'idée sous-jacente est que les termes qui apparaissent dans peu de documents de la collection sont plus représentatifs du contenu de ces documents que ceux qui apparaissent dans tous les documents de la collection. Un poids plus important est donné aux termes qui apparaissent moins fréquemment dans la collection. Cette mesure est exprimée selon l'une des déclinaisons suivantes :

$$idf = log\left(\frac{N}{n_i}\right)$$

$$idf = log\left(\frac{N - n_i}{N}\right)$$

Où : $\mathbf{n_i}$ est le nombre de documents contenant le terme $\mathbf{t_i}$ et **N** le nombre total de documents de la collection.

Finalement, le poids s'obtient en multipliant les deux mesures : $\mathbf{tfidf_{i,j} = tf_{i,j} * idf_i}$

Les termes importants dans un document doivent avoir un poids fort.

Dans le corpus où les documents sont de tailles homogènes, la mesure ***tf*idf*** donne une bonne approximation de l'importance d'un terme dans un document. Cependant, dans les corpus de documents de tailles variables cette mesure n'est pas un bon indice de l'importance des termes. En effet, lors des compagnes d'évaluation internationales, la mesure a eu des performances très limitées dans des corpus de taille très variable. Le problème posé est que les termes appartenant aux documents longs apparaissent très fréquemment et emportent le poids sur les termes appartenant à des documents moins longs. Les documents longs auront alors plus de chance d'être sélectionnés.

I.9. Création de l'index

Au terme du processus d'indexation, un ensemble de structure de données sont créés. Ces dernières permettent un accès efficace à la représentation des documents. Le fichier inverse est la structure de données la plus utilisée, il enregistre pour chaque descripteur les identificateurs des documents qui le contiennent et sa fréquence dans chacun de ces documents.

I.10. Conclusion

Face au volume important des unités documentaires à traiter, l'indexation est une nécessité. Elle nous permet de reformuler le contenu d'un document sous une forme plus adaptée à son exploitation.

On admet généralement, qu'un mot qui apparaît souvent dans un texte représente un concept important. Ainsi, la première approche d'indexation consiste à déterminer les mots représentatifs par leur fréquence après avoir éliminé les mots vides. Une autre approche est ensuite appliquée, c'est la normalisation afin de retourner les racines des mots représentatifs des documents.

On peut rendre compte de l'importance d'un terme dans un document relativement à un corpus, en utilisant des techniques de pondération, dont les plus connues sont TF-IDF qui calcul le poids d'un terme dans un document. Ce poids peut augmenter avec la fréquence du terme dans un document (TF) et peut aussi augmenter avec la rareté du terme dans les documents du corpus (IDF).

Chaque unité documentaire peut alors faire l'objet d'une représentation vectorielle : les coordonnées représentent les fréquences des mots non vides. Lorsque l'on effectue cette opération pour un corpus de documents on obtient une matrice dont les colonnes représentent un document et les coordonnées la fréquence des termes.

CHAPITRE II

Classification

II.1. Introduction

La classification de documents est l'activité du Traitement Automatique des Langues naturelles qui consiste à classer de façon automatique des ressources documentaires, généralement en provenance d'un corpus.

Cette classification peut prendre une infinité de formes, dont, la classification par genre, par thème, par opinion, ou encore par rapport au contenu du document. Elle connaît ces derniers temps un fort regain d'intérêt. Cela est dû essentiellement à la forte croissance des documents numériques disponibles et à la nécessité de les organiser de façon rapide. Et puis, regrouper des documents par catégories améliore la cohérence et la lisibilité d'une collection.

La tâche de classification est réalisée avec des algorithmes spécifiques, mis en œuvre par des systèmes de traitement de l'information. C'est une tâche d'automatisation d'un processus de classement, qui fait le plus souvent appel à des méthodes numériques (c'est-à-dire des algorithmes de classification de type mathématique). Bien que les premières bases de l'approche algorithmique de la classification automatique soient relativement anciennes, ce n'est qu'avec le développement de l'informatique qu'il est devenu possible de les mettre en œuvre.

Dans ce chapitre, nous allons définir la classification, ses méthodes , ses techniques et ses différents algorithmes en se basant essentiellement sur l'approche supervisée.

II.2. Définitions

- En son sens générique, la classification est une répartition systématique en classes, en catégories, d'êtres, d'objets ou de notions ayant des caractères communs notamment afin d'en faciliter l'étude ; résultat de cette opération ». [12] Elle répond à un besoin constant de l'ésprit humain d'organiser les savoirs pour les articuler entre eux, et de catégoriser les savoirs au sein d'une hiérarchie ou d'une chronologie. [13]

- La classification est une discipline reliée à plusieurs domaines, elle est connue sous divers appellations (classification, catégorisation, clustering, segmentation,…) selon les objets qu'elle traite et les objectifs qu'elle vise à atteindre. Pour attribuer une définition au terme « classification », il faudrait d'abord définir ses racines, ça vient du verbe « classer » qui désigne plus une action qu'un domaine, ou plutôt une série de méthodes qu'une théorie unifiée [14]. Le terme « classification » en anglais fait référence à l'affectation d'un individu à une classe (existant à priori) dans le cadre de l'analyse discriminante. [15]

- En mathématique, on appelle classification, la catégorisation algorithmique d'objets. Elle consiste à attribuer une classe ou catégorie à chaque objet (ou individu) à classer, en se basant sur des données statistiques. [16]

- La classification est une méthode mathématique d'analyse de données, pour faciliter l'étude d'une population d'affectif important, on les regroupe en plusieurs classes de telle sorte que les individus d'une même classe soient le plus semblables possible et que les classes soient les plus distinctes possibles. [17]

- En règle générale, La classification automatique consiste à regrouper divers objets (les individus) en sous-ensembles d'objets (les classes) dans le but d'identifier les classes auxquelles appartiennent des objets à partir de traits descriptifs (attributs, caractéristique, features). Elle peut être :

  - **Supervisée :** les classes sont connues à priori, elles ont en général une sémantique associée.

  - **Non-supervisée (clustering) :** les classes sont fondées sur la structure des objets, la sémantique associée aux classes est plus difficile à déterminer.

II.3. Domaines d'application :

- Affectation de documents à des thèmes prédéfinis
  - Traitement des e-mails (redirection, filtrage) ;
  - Organisation des documents par catégories ;
  - Classement de contacts au centre d'appel.

- Recherche d'information (Information retrieval)
  - Interrogation de textes par concepts, mots-clés, sujets, phrases visant à obtenir des résultats triés par ordre de pertinence.

II.4. Etapes de la classification automatique

Le processus de classification passe par plusieurs étapes qui sont :

- **Extraction des données :** extraire les données et en distinguer des objets spécifiques (individus). C'est ici que les données sont préparées, nettoyées.
- **Représentation des données :** obtenir une représentation des données adaptée à l'algorithme, choisir une fonction de similarité suivant la nature de cette représentation, appréhender les différents critères sur ces données.
- **Stratégie de classement :** choisir l'algorithme suivant les exigences (performances, représentation des résultats, types de données) et fixer les éventuels paramètres.
- **Validation :** estimer la qualité de la construction produite.
- **Interprétation et utilisation :** juger si la classification répond bien aux besoins exprimés en utilisant la classification avec les données réelles.

La figure qui suit illustre le processus de classification et ses étapes

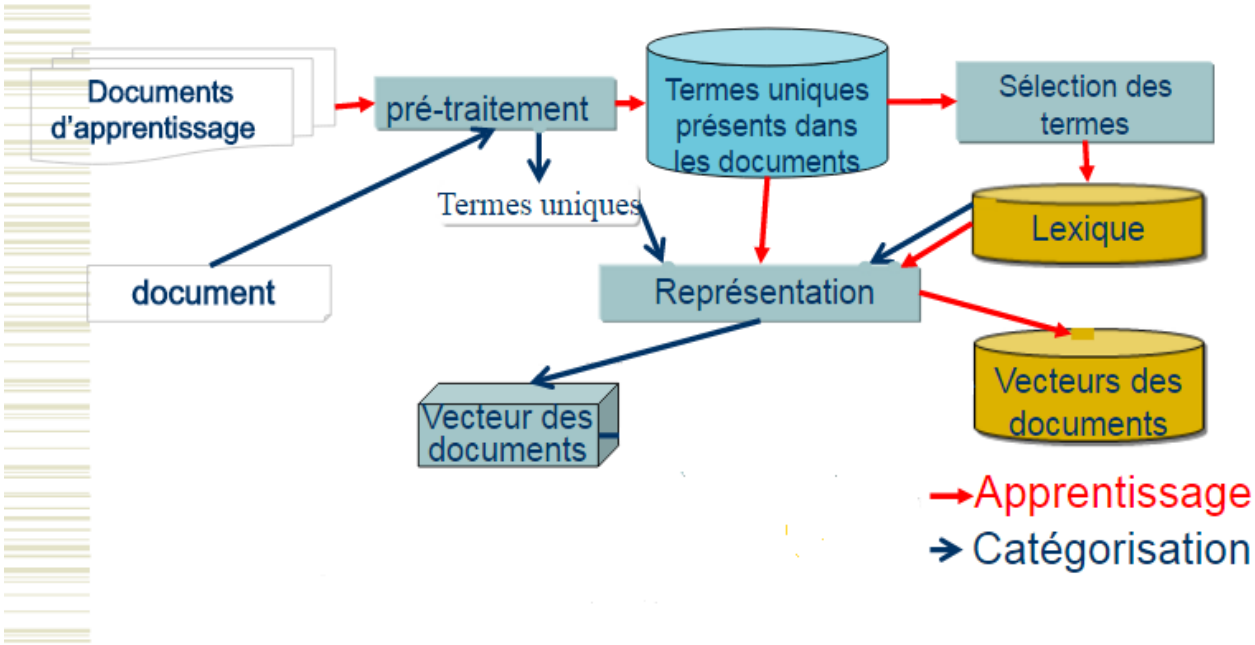

Figure II.1 : *Processus de classification*

II.5. Approches de classification

Les méthodes de classification ont pour objet d'identifier la classe d'appartenance d'objets définis par leur description. Il existe deux types d'approches de classification, une approche non-supervisée et une approche supervisée. La différence entre les deux situations est la connaissance des classes. Dans le cas non-supervisée, on fait une recherche à l'aveugle alors qu'en supervisée, on a un échantillon d'apprentissage. En supervisé on connaît un indicateur de performance : le taux de mal classé, mais en non-supervisé, difficile de valider les résultats.

II.5.1. Classification Non-supervisée (Clustering)

La classification non-supervisée, ou *clustering* est l'une des techniques fondamentales de l'extraction de données structurées ou non structurées. Elle est utilisée lorsque l'on possède des documents qui ne sont pas classés et dont on ne connaît pas de classification, autrement dit, dont les classes et leur nombre sont inconnus. Dans ce cas, l'apprentissage se ramène alors à cibler les groupes homogènes d'exemples existants dans les données, c'est-à-dire à identifier des groupes appelés cluster, et que les exemples les plus différents soient séparés dans des groupes distincts. A la fin du processus de classification non-supervisée, les documents doivent appartenir à l'une des classes générées par la classification.

On distingue deux catégories de classifications non-supervisées : hiérarchiques et non-hiérarchique.

- **Classification hiérarchique (CH)**

Dans la classification hiérarchique (CH), les sous-ensembles créés sont emboîtés de manière hiérarchique les uns dans les autres. On distingue la CH *descendante* qui part de l'ensemble de tous les individus et les fractionne en un certain nombre de sous-ensembles, chaque sous-ensemble étant alors fractionné en un certain nombre de sous-ensembles, et ainsi de suite. Et la CH *ascendante* qui part des individus seuls que l'on regroupe en sous-ensembles, qui sont à leur tour regroupés, et ainsi de suite. Pour déterminer quelles classes on va fusionner, on utilise le critère d'agrégation.

- **Classification non-hiérarchique**

Dans la classification non-hiérarchique, les individus ne sont pas structurés de manière hiérarchique. Si chaque individu ne fait pas partie que d'un sous-ensemble, on parle de *partition*.

Si chaque individu peut appartenir à plusieurs groupes, avec la probabilité $\boldsymbol{P_i}$ d'appartenir au groupe i, alors on parle de *recouvrement*.

II.5.1.1. Algorithmes de classification non-supervisée

A. Méthode *k-means*

L'algorithme *k-means*, créé par **MacQueen** en **1976** est l'algorithme de Clustering le plus connu et le plus utilisé car il s'avère être très simple à mettre en œuvre et efficace. Il suit une procédure simple de classification d'un ensemble d'objets en un certain nombre k de clusters, k fixé à priori. Dans cette algorithme, chaque cluster est caractérisé par son centre qui se trouve être la moyenne des éléments composant le cluster.

Les figures suivantes représentent un exemple de classification selon l'algorithme ***k-means***

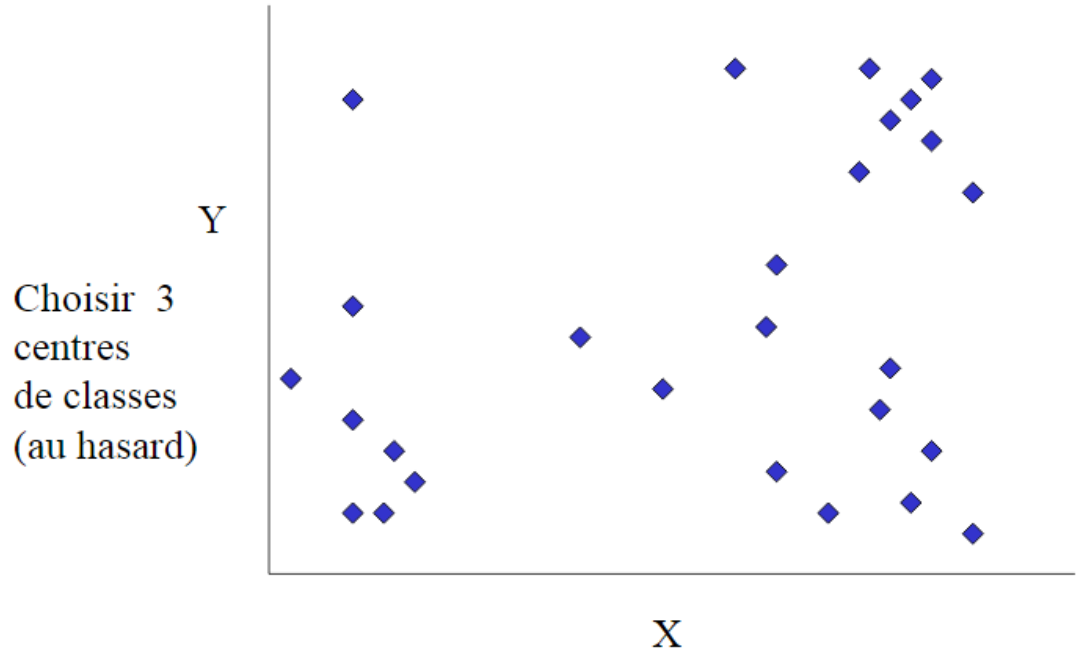

Figure II.2 : *Exemple de classification k-means, étape 1*

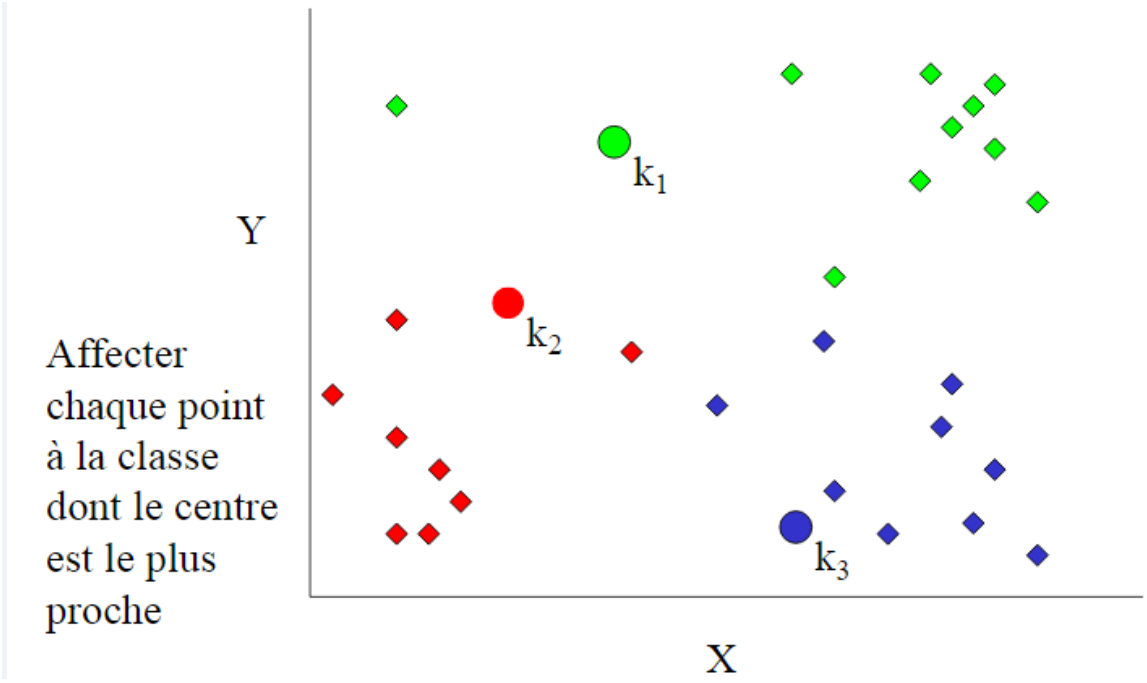

Figure II.3 : *Exemple de classification k-means, étape 2*

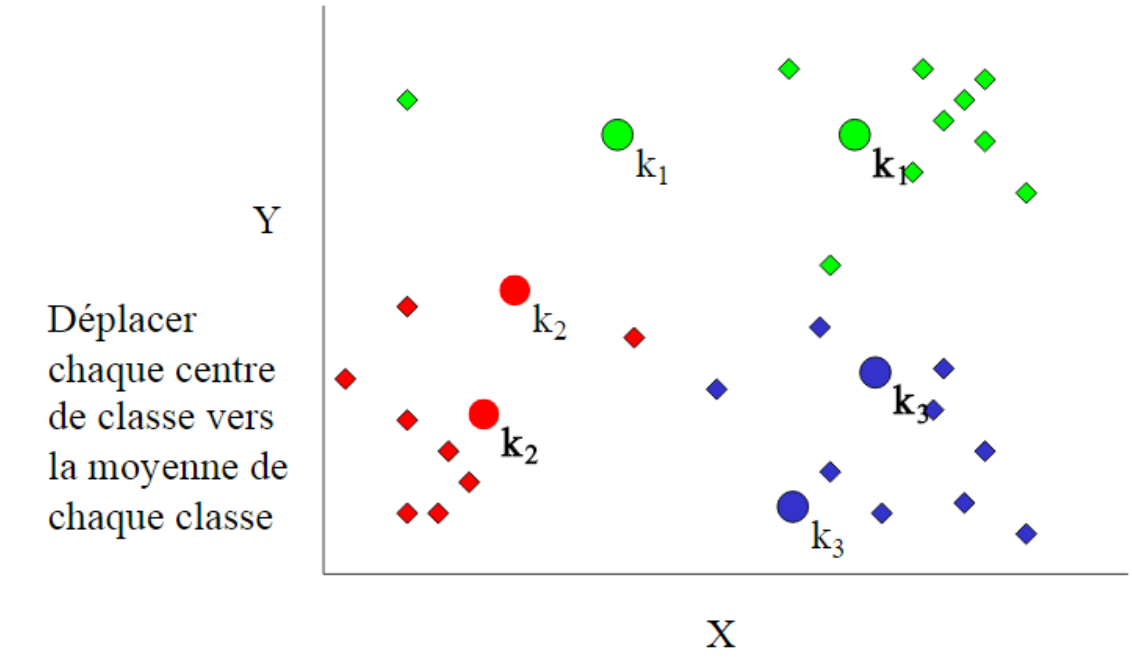

Figure II.4 : Exemple *de classification k-means, étape 3*

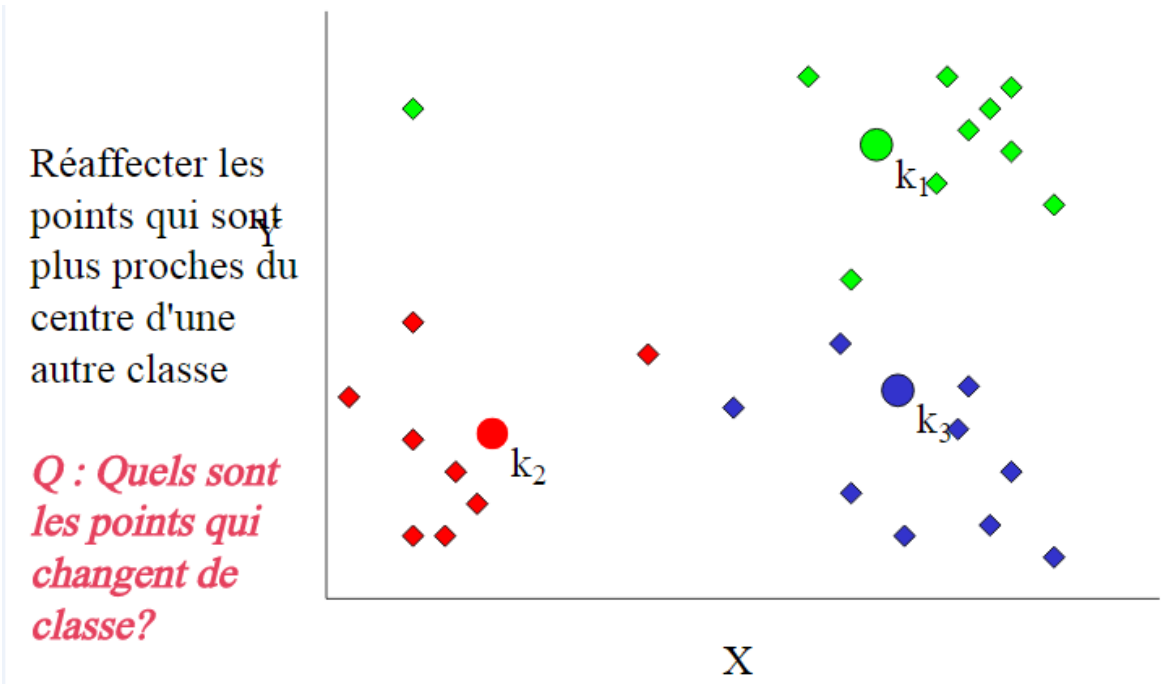

Figure II.5 : *Exemple de classification k-means, étape 4*

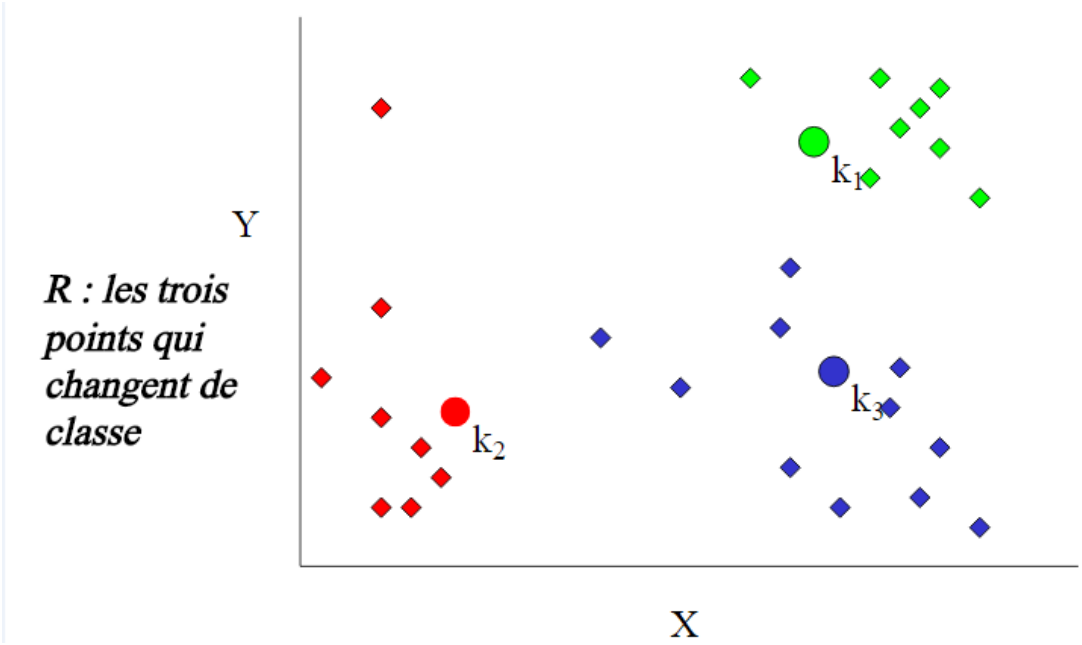

Figure II.6 : *Exemple de classification k-means, étape 5*

- **Principe**

On suppose qu'il existe ***K*** classes distinctes. On commence par désigner ***K*** centres de classes $u_1,\ldots,u_k$ parmi les individus. Ces centres peuvent être soit choisis par l'utilisateur pour leur « représentativité », soit désignés aléatoirement. On réalise ensuite itérativement les deux étapes suivantes :

- Pour chaque individu qui n'est pas un centre de classe, on regarde quel est le centre de classe le plus proche. On définit ainsi ***K*** classes $C_1,\ldots,C_k$,

Où : ***Ci***= {ensemble des points les plus proches du centre **ui**}

- Dans chaque nouvelle classe C_i.

L'algorithme s'arrête suivant un critère d'arrêt fixé par l'utilisateur qui peut être choisi parmi les suivants : soit le nombre limite d'itérations est atteint, soit l'algorithme a convergé, c'est-à-dire qu'entre deux itérations les classes formées restent les mêmes, soit l'algorithme a « presque » convergé, c'est-à-dire que l'inertie intra-classe ne s'améliore quasiment plus entre deux itérations.

- **Avantages et Inconvénients de la méthode k-means**

  - **Avantages**

- Simple et compréhensible ;
- Eléments affectés automatiquement aux classes.

  - **Inconvénients**

- Nombre de classes fixé à l'avance ;
- Chaque élément doit être dans une classe ;
- Très sensible aux éléments marginaux (intérêt d'une normalisation).

B. Fuzzy C-means

Fuzzy C-means (FCM) est une méthode de clustering qui permet à un objet de données d'appartenir à deux ou plusieurs clusters. Cette méthode dérivée de l'algorithme *c-means* identique à l'algorithme *k-means* décrit précédemment, elle a été développée par Dunn en 1973 et améliorée par Bezdek en 1981. [14]

C. Hierarchical clustering

Le processus basique des méthodes hiérarchiques a été donné par Johnson, Ce type de clustering consiste à effectuer une suite de regroupements en Clusters de moins en moins fines en agrégeant à chaque étape les objets (simple élément) ou les groupes d'objets (un Cluster-partition-) les plus proches. Ce qui nous donne une arborescence de clusters. Cette approche utilise la mesure de similarité pour refléter l'homogénéité ou l'hétérogénéité des classes. [14]
Dans ce chapitre, nous nous intéressons plus particulièrement à l'étude de la classification supervisée.

II.5.2. Classification supervisée

La classification supervisée consiste à inférer, à partir d'un échantillon de données classées, une procédure de classification. Selon **Gérard Govaert** : « la classification supervisé a pour objectif « *d'apprendre* » par l'exemple. Elle cherche à expliquer et à prédire l'appartenance de documents à des classes connues à priori ».

Autrement dit l'objectif de la classification supervisée est principalement de définir des règles permettant de classer des objets dans des classes à partir de variables qualitatives ou quantitatives caractérisant ces objets. Ainsi, on dispose au départ d'un échantillon dit d'apprentissage dont le classement est connu. Cet échantillon est utilisé pour l'apprentissage de règles de classement.

II.5.2.1. Algorithmes de classification supervisée

A. Méthode de k plus proches voisins

La méthode des plus proches voisins (noté parfois **k-PPV** ou **k-NN** pour **Nearest-Neighbor**) consiste à déterminer pour chaque nouvel individu que l'on veut classer, la liste des plus proches voisins parmi les individus déjà classés. Autrement dit, l'objectif de l'algorithme est de classer les exemples non étiquetés sur la base de leurs similarités avec les exemples de la base d'apprentissage. Cette méthode nécessite de choisir une distance, la plus classique est la distance euclidienne, et le nombre de voisins à prendre en compte.

Le principe de cet algorithme est très simple. On lui fournit :

- Un ensemble de données d'apprentissage **D** ;
- Une fonction de distance **d** ;
- Et un entier **k**.

Pour tout nouveau point de test **x**, pour lequel il doit prendre une décision, l'algorithme recherche dans **D** les **k** points les plus proches de **x** au sens de la distance **d**, et attribue **x** à la classe qui est la plus fréquente parmi ces **k** voisins.

- **Comment choisir la valeur de k ?**

  - **K=1** : frontières des classes très complexes.
    - ✓ Très sensible aux fluctuations (variation, défaut de fixité, de permanence) des données (variance élevée) ;
    - ✓ Risque de sur-ajustement (sur apprentissage) ;
    - ✓ Résiste mal aux données bruitées.
  - **K=n** : frontière **g** rigide.
    - ✓ Moins sensible au bruit ;
    - ✓ Plus la valeur de **k** est grande plus le résultat d'affectation est bien réalisé

➢ **Distance euclidienne**

Soient deux données représentées par deux vecteurs x_i et x_j, la distance entre ces deux données est donnée par :

$$d(\mathrm{x}_i, \mathrm{x}_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$$

➢ **Algorithme**

Soit **D** = *{**(x',c)**, **c** ∈**C}* l'ensemble d'apprentissage ;

Soit ***x*** l'exemple dont on souhaite déterminer la classe.

Début

Pour chaque (**(x', c)** appartient à **D**) **faire**

Calculer la distance **dist(x, x')**

Fin

Pour chaque {**x'** appartient à **kppv(x)**} **faire**

Compter le nombre d'occurrence de chaque classe

Fin

Fin

➢ **Avantages et Inconvénients de la méthode k-PPV**

- **Avantages de la méthode k-PPV**

- Apprentissage rapide ;
- Méthode facile à comprendre ;
- Adapté aux domaines où chaque classe est représentée par plusieurs prototypes et où les frontières sont irrégulières (*Exemple :* reconnaissance de chiffre manuscrit ou d'images satellites).

- **Inconvénients de la méthode k-PPV**

- Prédiction lente car il faut revoir tous les exemples à chaque fois ;
- Méthode gourmande en place mémoire ;
- Sensible aux attributs non pertinents et corrélés.

B. Machine à vecteurs de support (SVM)

La machine à vecteurs de support (*Support Vector Machine, SVM*) appelée aussi *séparateur à vaste marge* est une technique de classification binaire par apprentissage supervisé. Elle fut introduite par Vladimir Vapnik en 1995. Elle repose sur l'existence d'un classificateur linéaire appelé hyperplan dans un espace approprié. Il est évident qu'il existe une multitude d'hyperplan valide mais la propriété remarquable des SVM est que cet hyperplan doit être optimal. Nous allons donc en plus chercher parmi les hyperplans valides, celui qui passe « au milieu » des points des deux classes d'exemples. L'hyperplan calculé permet ainsi de séparer l'espace en deux zones. Pour classer les nouveaux documents, on calcule dans quelle région de l'espace ils se situent et on leur attribue la classe correspondante.

La **SVM est** basée sur l'utilisation de fonctions dites noyau (*kernel*) qui permet une séparation optimale des données. Pour deux classes de données, le but de SVM est de retrouver un classificateur qui va séparer les données et maximiser la distance entre ces deux classes

- **Notions de base**

  - **Hyperplan :** sépare les deux ensembles de points (de documents).
  - **Vecteurs de support :** points les plus proches qui déterminent l'hyperplan.
  - **Marge :** hyperplan dont la distance minimale aux exemples est maximale.

La figure ci-dessous illustre ces notions de base

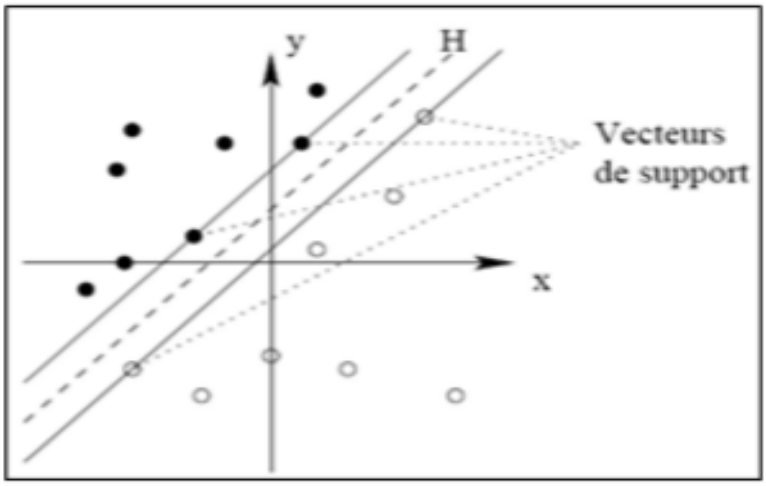

Figure II.7 : *Machine à Vecteurs de Support*

Les deux figures suivantes illustrent l'hyperplan sans séparation linéaire puis avec séparation linéaire

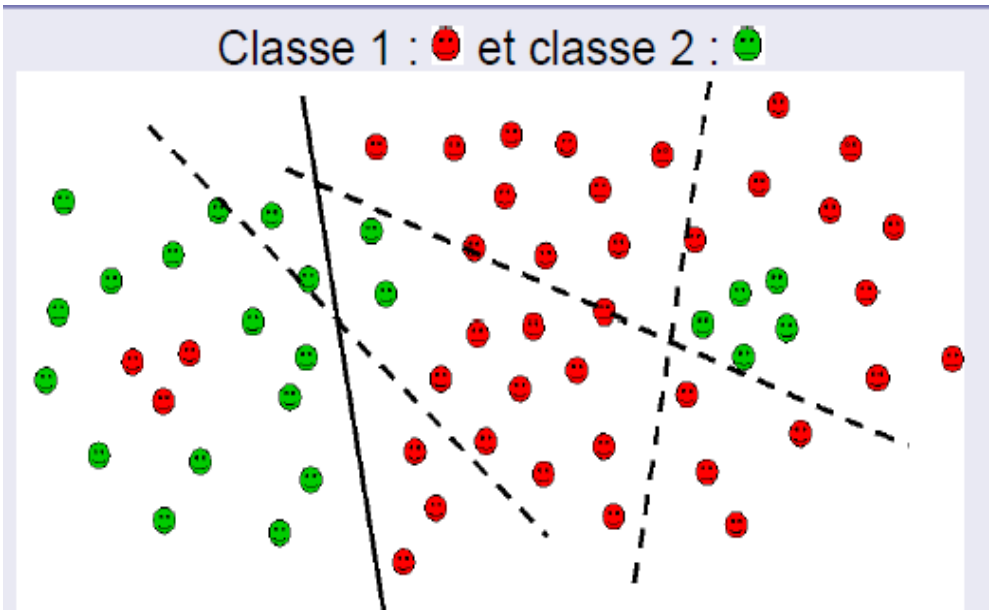

Figure II.8: *pas de séparation linéaire*

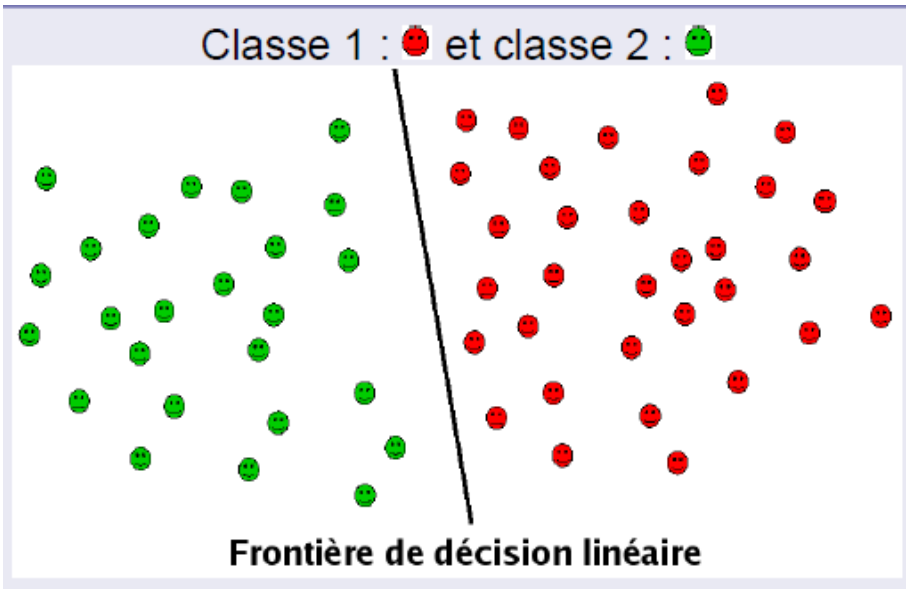

Figure II.9: *séparation linéaire*

- **Principe général**

Cette technique est une méthode de classification à deux classes qui tente de séparer les exemples positifs des exemples négatifs dans l'ensemble des exemples. La méthode cherche alors l'hyperplan qui sépare les exemples positifs des exemples négatifs, en garantissant que la marge entre le plus proche des positifs et des négatifs soit maximale. Cela garantit une généralisation du principe car de nouveaux exemples pourront ne pas être trop similaires à ceux utilisés pour trouver l'hyperplan mais être situés d'un côté ou l'autre de la frontière.

L'intérêt de cette méthode est la sélection de vecteurs supports qui représentent les vecteurs discriminant grâce auxquels est déterminé l'hyperplan. Les exemples utilisés lors de la recherche de l'hyperplan ne sont alors plus utiles et seuls ces vecteurs supports sont utilisés pour classer un nouveau cas, ce qui peut être considéré comme un avantage pour cette méthode. [14]

- **Exemples d'applications**
  - Classification de données biologiques/ physique
  - Classification de documents numériques
  - Classification d'expressions faciales
- **Avantage et Inconvénients des SVM**
  - **Avantage des SVM**

Les avantages théoriques et pratiques des SVM en ont fait un outil très prisé dans de nombreux problèmes pratiques de classification. On cite parmi ses avantages :

- Minimisation de l'erreur empirique et structurelle
- Algorithmes optimisés
- Simple, peu de paramètres à régler.
- Les SVM possèdent des fondements mathématiques solides
- Les exemples de test sont comparés juste avec les supports vecteurs et non pas avec tous les exemples d'apprentissage.
- Décision rapide. La classification d'un nouvel exemple consiste à voir le signe de la fonction de décision f(x).

  - **Inconvénients des SVM**

- Les données dans des espaces de grande dimension sont souvent non linéairement séparables ;
- Classification binaire, d'où la nécessité d'utiliser l'approche un-contre-un ;
- Grande quantité d'exemples en entrées implique un calcul matriciel important ;
- Temps de calcul élevé lors d'une régularisation des paramètres de la fonction noyau.

C. Réseau de neurones

Un réseau de neurones artificiel est un ensemble d'algorithmes dont la conception est à l'origine très schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques. Les approches neuronales furent les premières à être utilisées afin de réaliser un apprentissage de type statistique grâce à leur capacité de classification et de généralisation. Un réseau de neurones artificiel est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. A chaque synapse est associé un poids synaptique, de sorte que les Ni-1 sont multipliés par ce poids, puis additionnés par les neurones de niveau i, ce qui est équivalent à multiplier le vecteur d'entrée par une matrice de transformation. Nous pouvons ainsi voir un réseau de neurones artificiels comme un réseau ou graphe orienté dont les nœuds sont les neurones artificiels. Le but va être d'attribuer des poids synaptiques à chaque neurone afin d'obtenir le résultat voulu en sortie.

La Figure II.10 représente la Structure d'un neurone artificiel

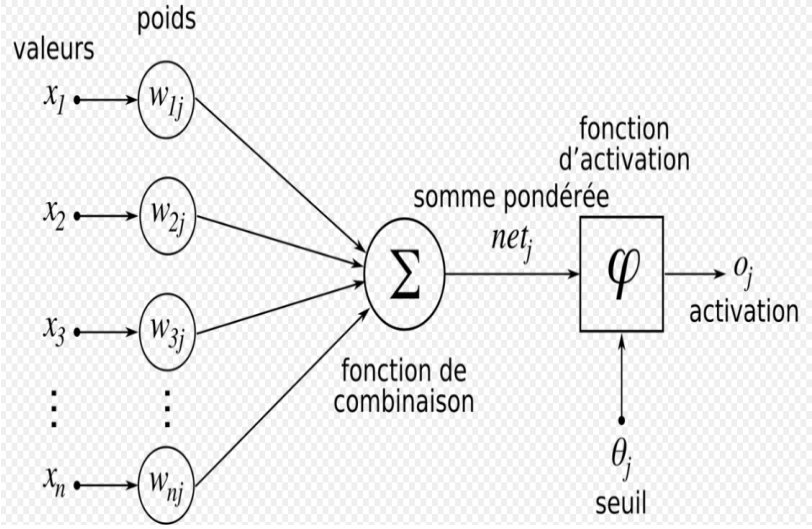

Figure II.10 : *Structure d'un neurone artificiel*

Le neurone calcule la somme de ses entrées puis cette valeur passe à travers la fonction d'activation pour produire sa sortie.

- **Avantages et inconvénients**

  - **Avantages**

- Classification très précise (si bien paramétré).
- Résistance aux pannes (si un neurone ne fonctionne plus, le réseau ne se perturbe pas).

  - **Inconvénients**

- La détermination de l'architecture du réseau est complexe.
- Paramètres difficiles à interpréter (boite noire).
- Difficulté de paramétrage surtout pour le nombre de neurone dans la couche cachée.

D. Arbre de décision

Un arbre de décision est un outil d'aide à la décision représentant un ensemble de choix sous la forme graphique d'un arbre. C'est un graphe orienté, sans cycles, dont les nœuds portent une question, les arcs des réponses, et les feuilles des conclusions .Un classificateur de texte basé sur la méthode d'arbre de décision est un arbre de nœuds internes qui sont marqués par des termes, les branches qui sortent des nœuds sont des tests sur les termes, et les feuilles sont marquées par catégories**.** Une méthode pour effectuer l'apprentissage d'un arbre de décision pour une catégorie Ci consiste à vérifier si tous les exemples d'apprentissage ont la même étiquette. Dans le cas contraire, nous sélectionnons un terme **Tk**, et nous partitionnons l'ensemble d'apprentissage en classes de documents qui ont la même valeur pour **Tk**, et à la fin nous créons les sous-arbres pour chacune de ces classes .Ce processus est répété récursivement sur les sous arbres jusqu'à ce que chaque feuille de l'arbre généré de cette façon contienne des exemples d'apprentissage attribués à la même catégorie Ci, qui est alors choisie comme l'étiquette de la feuille. [14]

La Figure II.11 montre un exemple d'arbre de décision

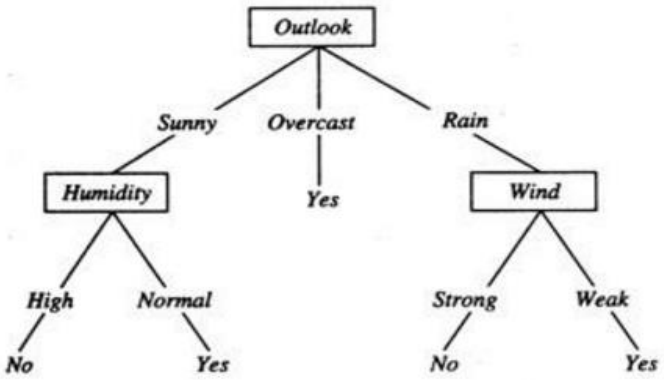

Figure II.11 : *exemple d'arbre de décision*

- **Avantages et inconvénients**

  - **Avantages**

- Leur capacité à travailler sur des données symboliques.
- Leur grande capacité et efficacité à faire de la classification.
- Leur facilité d'apprentissage et d'utilisation.

  - **Inconvénients**

- Leur sensibilité au changement des données.
- Une détection difficile des interactions entre les variables.

E. Naïve Bayes

Comme son nom l'indique, ce classificateur se base sur le théorème de Bayes permettant de calculer les probabilités conditionnelles. Dans le cas de la classification des textes, on cherche la classification qui maximise la probabilité d'observer les mots du document. Lors de la phase

d'entraînement, le classificateur calcule les probabilités qu'un nouveau document appartienne à telle catégorie à partir de la proportion des documents d'entraînement appartenant à cette catégorie. Il calcule aussi la probabilité qu'un mot donné soit présent dans un texte, sachant que ce texte appartient à telle catégorie.

Quand un nouveau document doit être classé, on calcule les probabilités qu'il appartienne à chacune des catégories à l'aide de la règle de Bayes [14].

- **Avantages et inconvénients**

  - **Avantages**

- La facilité et la simplicité de leur implémentation.
- Leur rapidité.
- Les méthodes Naïve Bayes donnent de bons résultats.

  - **Inconvénients**

- Performances limitées quand il s'agit d'une grande quantité de lexiques à traiter.
- Modèle qualifié de naïve ou simple à cause de l'hypothèse d'indépendance des mots.

F. Rocchio

L'algorithme Rocchio, [18] est un des plus vieux algorithmes de classification et l'un des plus simples. Utilisant une méthode probabiliste, il permet de construire un vecteur représentatif d'une catégorie à partir des documents pertinents pour cette catégorie et classe un document dans la catégorie dont il est le plus proche du barycentre.

Cette méthode est basée sur une phase d'apprentissage, où certains documents servent à établir une base contre laquelle les autres documents seront classés. Un ensemble de documents bien choisi est donc fourni au départ comme représentatif des catégories voulus.

- **Description de l'algorithme**
  - On commence par construire des vecteurs « de base » des catégories considérées. En clair, il s'agit pour chaque catégorie d'avoir un vecteur pondéré associant à chaque mot clé son poids dans la catégorie (**ex :** si on a comme catégorie **« animaux »** et **« lieu »**, alors **« chien »** aura un poids dort dans **« animaux »** et faible dans **« lieux »**).
    Le vecteur de base de ma catégorie k se calcule comme suit :

    a. Pour chaque mot clé on tire de la matrice terme-document pondérée l'ensemble de ces poids dans les documents de la catégorie k.

    b. On calcule le poids du mot dans la catégorie k e faisant la différence entre la moyenne des poids du mot dans les documents appartenant à la catégorie k et la moyenne du poids du mot dans les documents n'appartenant pas à la catégorie k.
    On obtient ainsi un ensemble de vecteurs de poids pour chaque catégorie (vecteurs et barycentres).

- Pour chaque document à classer, il suffit de calculer la distance avec chacun des vecteurs « de base » et d'associer le document à la catégorie dont il est le plus proche.

La figure ci-dessous illustre un exemple de classification avec la méthode **Rocchio** :

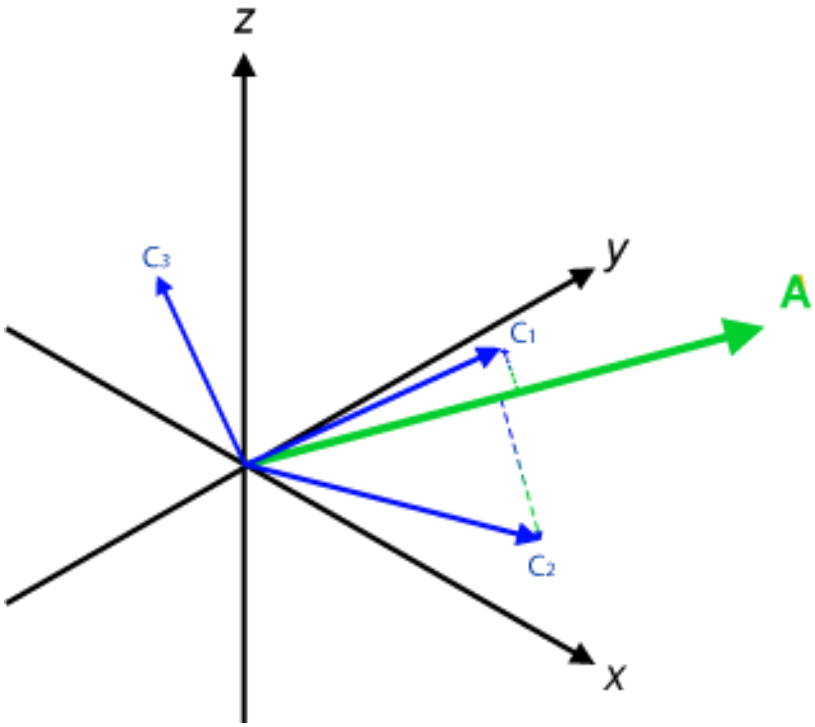

Figure II.12: *exemple de classification avec la méthode Rocchio*

Où, les catégories sont représentées en bleu et le document à classer en vert.

On remarque que le document **A** en vert va être classé avec la catégorie C_1 car c'est le plus proche.

II.5.3. Comparaison entre la méthode supervisée et non-supervisée

L'apprentissage non supervisé consiste à inférer les connaissances des classes sur la seule base des échantillons d'apprentissage, et sans savoir a priori à quelles classes ils appartiennent. Contrairement à l'apprentissage supervisé, on ne dispose que d'une base d'entrées et c'est le système qui doit déterminer ses sorties en fonction des similarités détectées entres les différentes entrées (règle d'auto organisation). On pourrait imaginer que l'algorithme d'apprentissage décide lui-même des classes qui existent et de la classification de chaque exemple.

Contrairement à l'apprentissage supervisé, dans l'apprentissage non-supervisé il n'y a pas d'oracle qui explicite les étiquettes. L'utilisation de ce type d'algorithme permet de trouver des structures, des dépendances entre descripteurs qui nous sont inconnues.

II.6. Conclusion

La classification de documents s'est avérée au cours des dernières années comme un domaine majeur de recherche. Elle devient de plus en plus indispensable dans de nombreuses situations où la quantité de documents électroniques rend impossible tout traitement manuel. Cette dernière a énormément progressée grâce à l'introduction des techniques héritées de l'apprentissage automatique.

Différentes méthodes de classificateurs ont été mis au point, et cela dans le but d'atteindre un degré maximal de précision et d'efficacité, chacun ayant ses avantages et ses inconvénients, mais, ils partagent toutefois des caractéristiques communes.

Dans ce chapitre, nous avons présenté les différentes approches de classifications automatiques, leurs méthodes et leur principe de fonctionnement, en se basant essentiellement sur l'approche de classification supervisée.

CHAPITRE III

Présentation du Logiciel Weka

III.1. Introduction

De nos jours, le domaine de la fouille de textes s'est développé pour répondre à volonté à la gestion par contenu des sources volumineuses de textes. A l'heure actuelle, de nombreux logiciels de classification de textes sont disponibles, ils ont fait l'objet de publications et leur champ d'application s'élargit de jour en jour. En général, ces systèmes sont basés sur des algorithmes d'apprentissage automatique (approche statistique, approche syntaxique). Parmi ces logiciels, on cite « Weka », un package populaire, une suite de logiciels d'apprentissage automatique et d'exploration de données implémentant plusieurs algorithmes de classification, et qui offre une interface conviviale, en plus des outils de prétraitement de données et d'analyse.

Ce chapitre a pour but de présenter le package « Weka », ses outils, ses différentes interfaces, ses concepts de base, son mode d'utilisation en se basant particulièrement sur la partie classification.

III.2. Présentation

Weka, autrement dit, environnement Waikato pour l'analyse de connaissances [19], est un package Open Source très populaire, en d'autres termes, un ensemble de classes et d'algorithmes développé sous Java, à *l'Université de Waikato* en *Nouvelle-Zélande*. Il propose différents algorithmes d'apprentissage automatique (supervisée ou non), à savoir, *Naïve Bayes*, *Arbre de décision*, *SVM*, *réseau de neurones*, etc. avec les fonctionnalités de prétraitement des données, analyse et évaluation des résultats. Il offre une interface GUI conviviale pour manipuler et inspecter les données et visualiser les résultats. Ce package peut fonctionner sur les plateformes Linux, Windows et Mac.

La figure ci-dessous présente la fenêtre d'invite au lancement de Weka

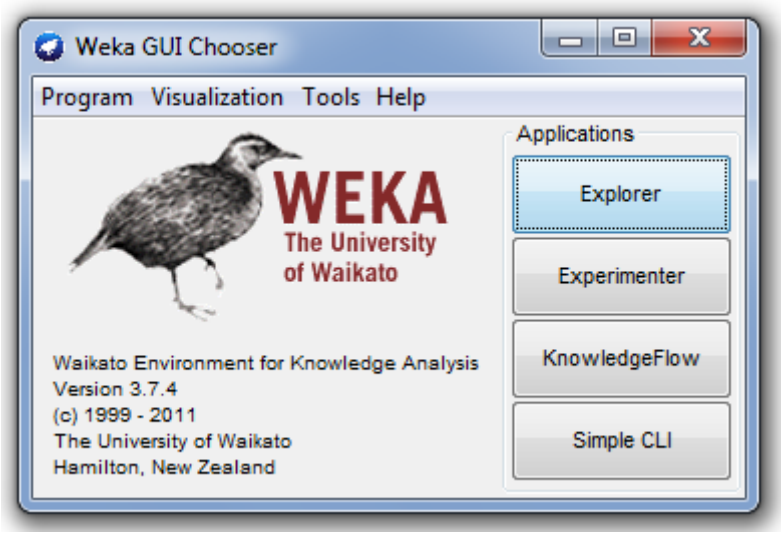

Figure III.1 : *Fenêtre d'invite Weka*

Weka se compose principalement :

- De classes Java permettant de charger et de manipuler les données ;
- De classes pour les principaux algorithmes de classification supervisée ou non supervisée ;
- D'outils de sélection d'attributs, de statistiques sur ces attributs ;
- De classes permettant de visualiser les résultats.

On peut l'utiliser à trois niveaux :

- Via l'interface graphique, pour charger un fichier de données, lui appliquer un algorithme, vérifier son efficacité ;
- Invoquer un algorithme sur la ligne de commande ;
- Utiliser les classes définies dans ses propres programmes pour créer d'autres méthodes, implémenter d'autres algorithmes, comparer ou combiner plusieurs méthodes.

III.3. Historique

- En **1992**, **l'Université de Waikato en Nouvelle-Zélande** commença le développement de la version originale de *Weka*.
- En **1997**, la décision fut prise de développer une nouvelle fois *Weka* à partir de zéro en Java, y compris l'implémentation des algorithmes de modélisation.
- En **2005**, *Weka* reçoit le **SIGKDD** (*Data Mining and Knowledge Discovery Service Award*).
- En **2006**, **Pentaho** acquiert une licence exclusive pour utiliser *Weka* pour de l'informatique décisionnelle. Il forme le composant d'exploration de données analytique et prédictif de la suite de logiciels décisionnels Pentaho.

III.4. Concepts de base de Weka

III.4.1. Format de données

Weka comporte son propre format de fichier appelé ARFF (Attribute Relation File Format), mais peut aussi traiter des données issues de bases relationnelles (BDD SQL), binaire, des fichiers type CSV (Open File), ou encore, charger des fichiers sur le Web (Open URL).

III.4.4.1. Caractéristiques du format de fichier ARFF

Le format ARFF précise essentiellement trois champs :

- Définition du nom de l'ensemble de données avec `@relation ;` Le nom doit être aussi compréhensible que possible
- Définition des *features* avec `@attribute`

- Attributs nominaux suivis des valeurs entre accolades
- Attributs numériques avec `real`
- Attributs chaines avec `string`, les valeurs doivent être entre doubles guillemets
- Attributs dates avec `date (yyyy-MM-dd-THH :mm :ss)`

➢ `@data` signale le début des instances

Note : Les commentaires sont précédés de % (`% Ceci est un commentaire`)
La figure suivante illustre un exemple de fichier ARFF

```
% Ensemble de donnees sur la meteo
@relation weather

% Definition des features
@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

% Debut des instances
@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
...
```

Figure III.2 : *Fichier ARFF*

III.4.2. Interfaces Weka

Weka dispose de plusieurs interfaces graphiques qui sont :

III.4.2.1. Explorer

C'est l'interface principale de Weka. Elle possède plusieurs onglets qui donnent accès aux principaux composants de l'espace de travail.

La figure III.3 Présente l'interface principale *Explorer* de Weka

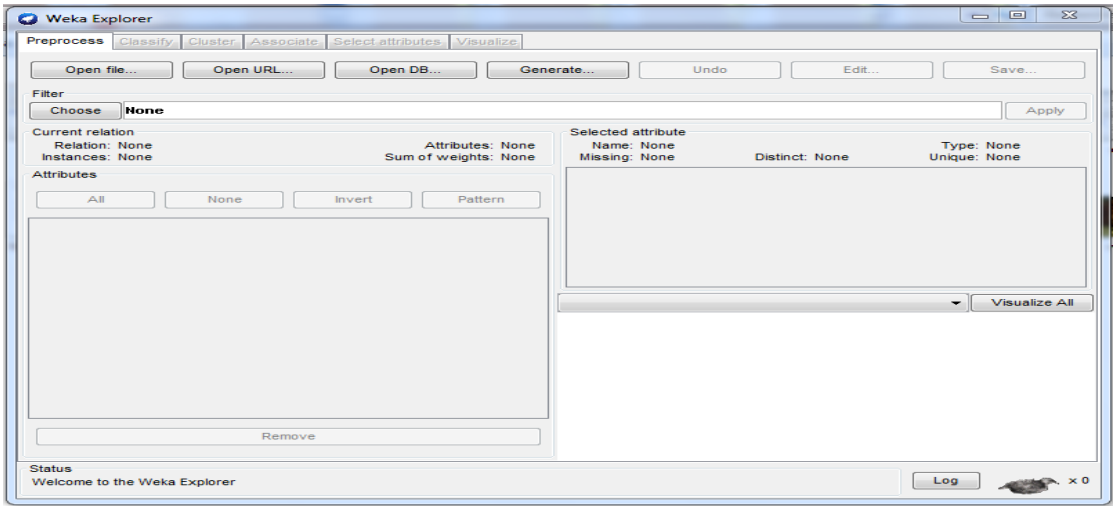

Figure III.3 : Interface *Explorer*

Les différents onglets de l'interface principale Explorer sont :

- **L'onglet Preprocess**

L'onglet **Preprocess**, ou « préprocesseur » a plusieurs fonctionnalités d'import de données (saisie des données, examen et sélection des attributs, transformations d'attributs), il nous permet de charger un fichier au format spécifique de Weka, le format ARFF, ou, depuis des bases de données ou un fichier CSV et pour prétraiter ces données avec un algorithme appelé *filtering*. Ces filtres peuvent être utilisés pour transformer les données (par exemple, transformer des attributs numériques réels en attributs discrets) et rendre possible l'effacement d'instances et d'attributs selon des critères spécifiques.

La figure III.4 illustre l'interface de l'onglet *Preprocess*

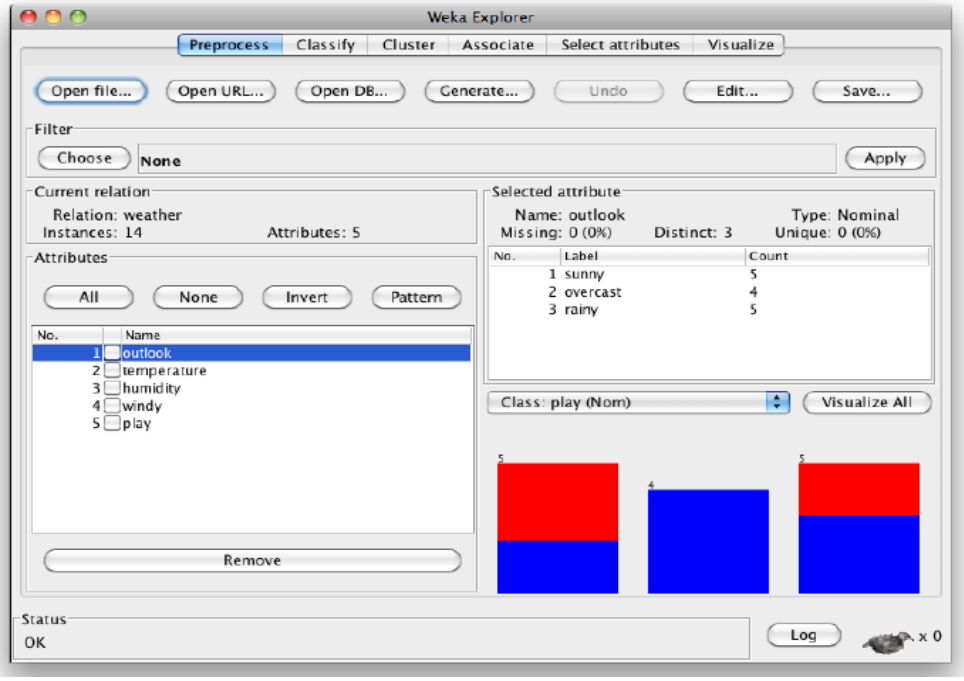

Figure III.4 : *Onglet Preprocess*

Une fois le fichier ARFF chargé, une liste d'attributs apparaît à gauche de la fenêtre

- **L'onglet Classify**

Une fois les données chargées, l'onglet *Classify* permet à l'utilisateur d'appliquer des méthodes de classification et des algorithmes de régression au jeu de données résultant, pour estimer la précision du modèle prédictif, et de visualiser les prédictions erronées ou le modèle lui-même (si le modèle est sujet à visualisation, comme un Arbre de décision).

La figure III.5 illustre l'interface de l'onglet *Classify*

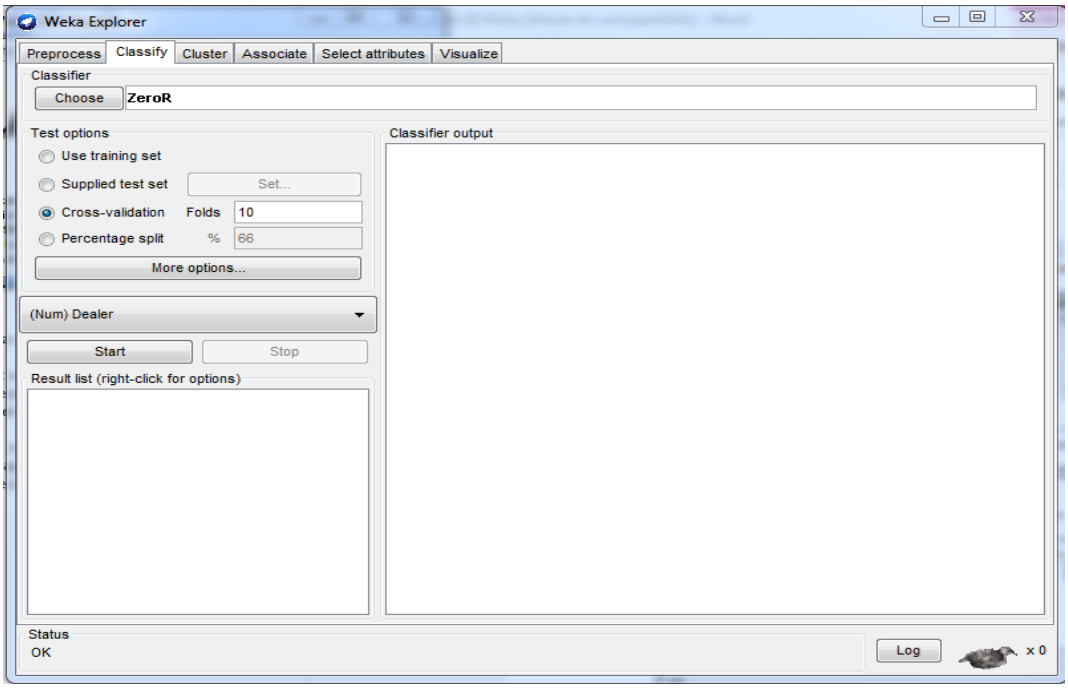

Figure III.5 : *Onglet Classify*

- **L'onglet Cluster**

L'onglet **Cluster** donne accès aux techniques de *clustering* (méthodes de segmentation) de Weka, comme l'algorithme *K-means*.

La figure III.6 illustre l'interface de l'onglet *Cluster*

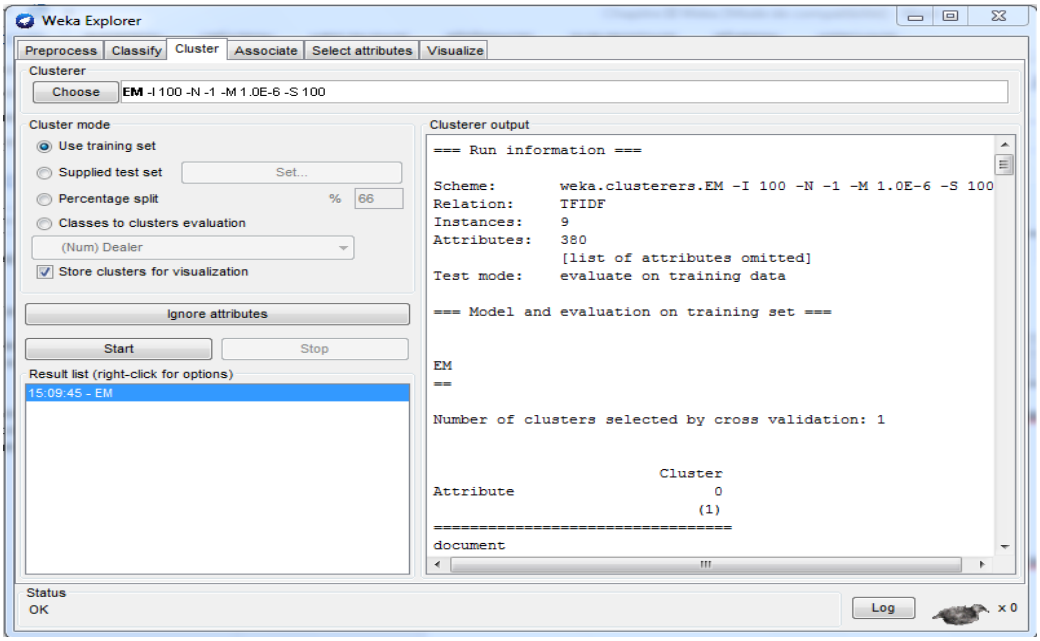

Figure III.6 : *Onglet Cluster*

- **L'onglet Associate :** contient les règles d'association.

La figure III.7 illustre l'interface de l'onglet *Associate*

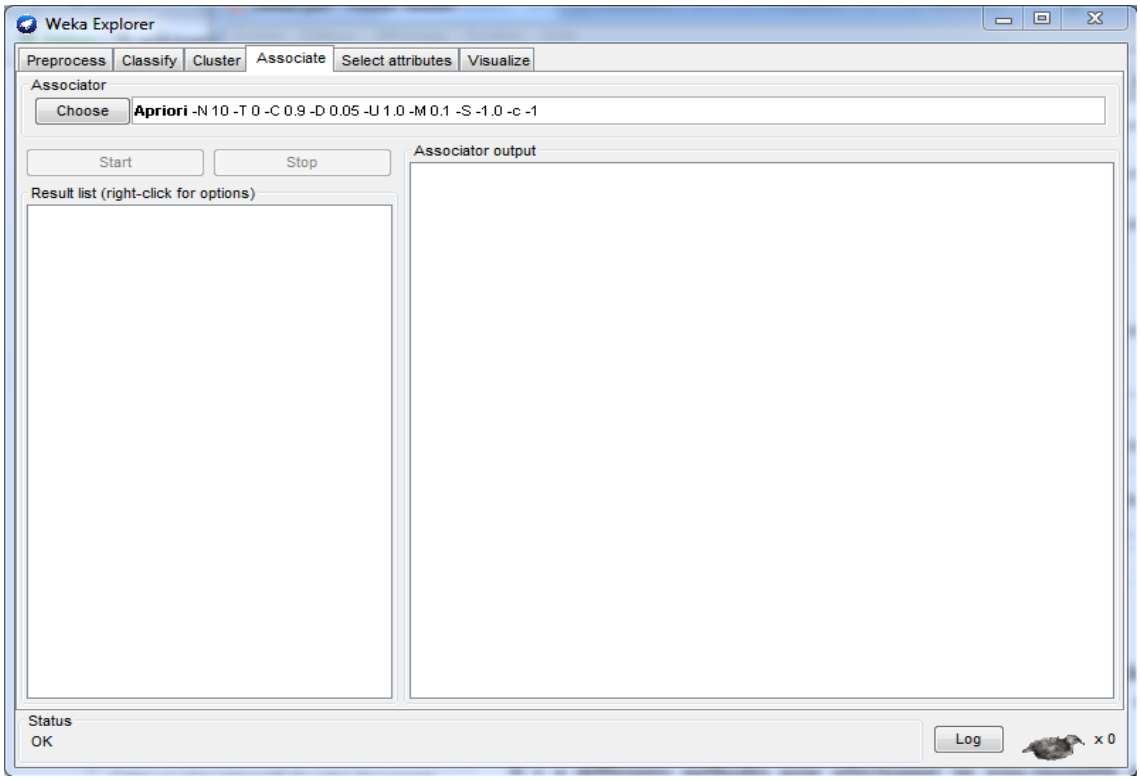

Figure III.7 : *Onglet Associate*

- **L'onglet Select attributes**

L'onglet *« Select attributes »* fournit des algorithmes pour l'identification des attributs les plus prédictifs dans un jeu de données et permet la sélection des attributs à utiliser pour la classification.

Il y a différentes méthodes pour sélectionner un sous-ensemble d'attributs à utiliser dans la classification. Ceci est très utile quand les données sont très bruitées, avec beaucoup d'attributs qui n'aident pas à la classification. Un nettoyage (sélection) est très bénéfique dans ce cas. Cette sélection aide aussi à accélérer les traitements. Pour cela, il faut choisir dans « Attribute Evaluator » la méthode InfoGainAttributeEval (la sélection basée sur le gain d'information), et dans « Search Method » la méthode Ranker – qui ordonne les attributs selon leur valeur. En cliquant sur Ranker, on peut préciser les critères de sélection, par exemple en fixant un seuil, ou en fixant un nombre d'attributs à garder.

La figure III.8 illustre l'interface de l'onglet *Select Attribute*

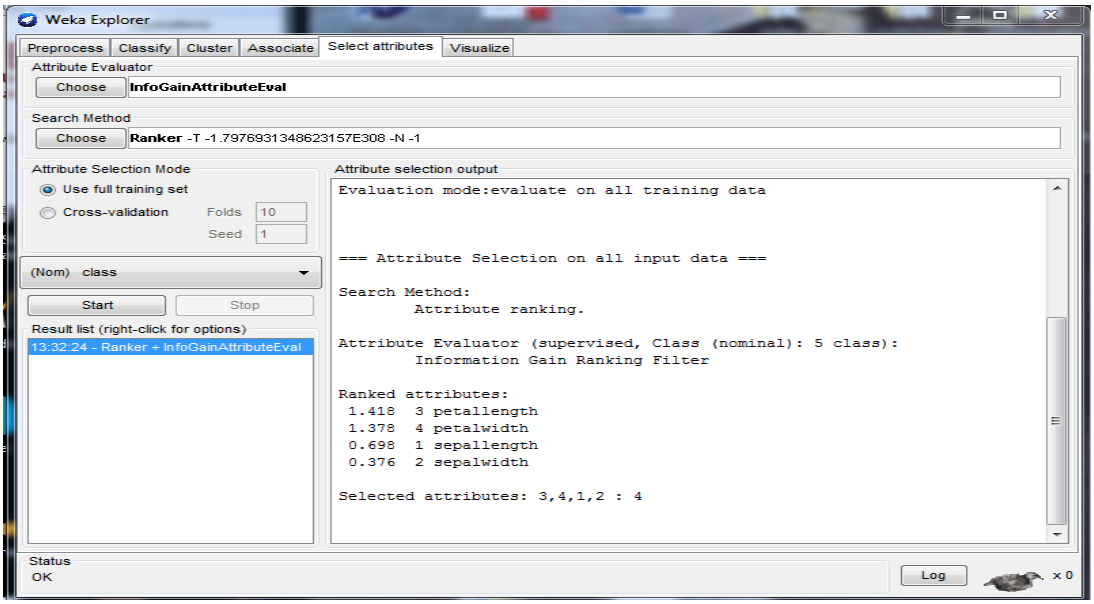

Figure III.8 : *Onglet Select attributes*

➢ L'onglet Visualize

Le dernier onglet « *Visualize* » montre une matrice de nuages de points, ou des nuages de points individuels peuvent être sélectionnés et élargis, et davantage analysés en utilisant divers opérateurs de sélection. La fenêtre Visualize dispose d'un ensemble de 25 graphiques, qui représentent chacun une vue sur l'ensemble d'exemples selon deux dimensions possibles, la couleur des points étant leur classe. Sur le graphique, chaque point représente un exemple : on peut obtenir le descriptif de cet exemple en cliquant dessus. La couleur d'un point correspond à sa classe. Au départ, le graphique n'est pas très utile, car les axes représentent le numéro de l'exemple.

La figure III.9 illustre l'interface de l'onglet *Visualize*

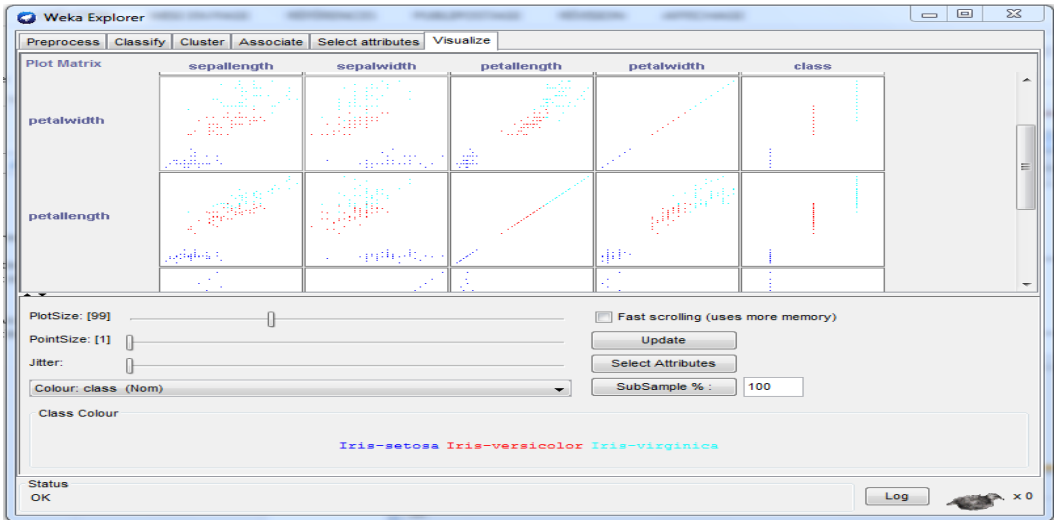

Figure III.9 : *Onglet Visualize*

III.4.2.2. Experimenter

L'experimenteur permet la comparaison systématique (taxinomique) des performances prédictives des algorithmes d'apprentissage automatique de Weka sur une collection de jeux de données.

La figure III.10 Montre l'interface *Experimenter* de Weka

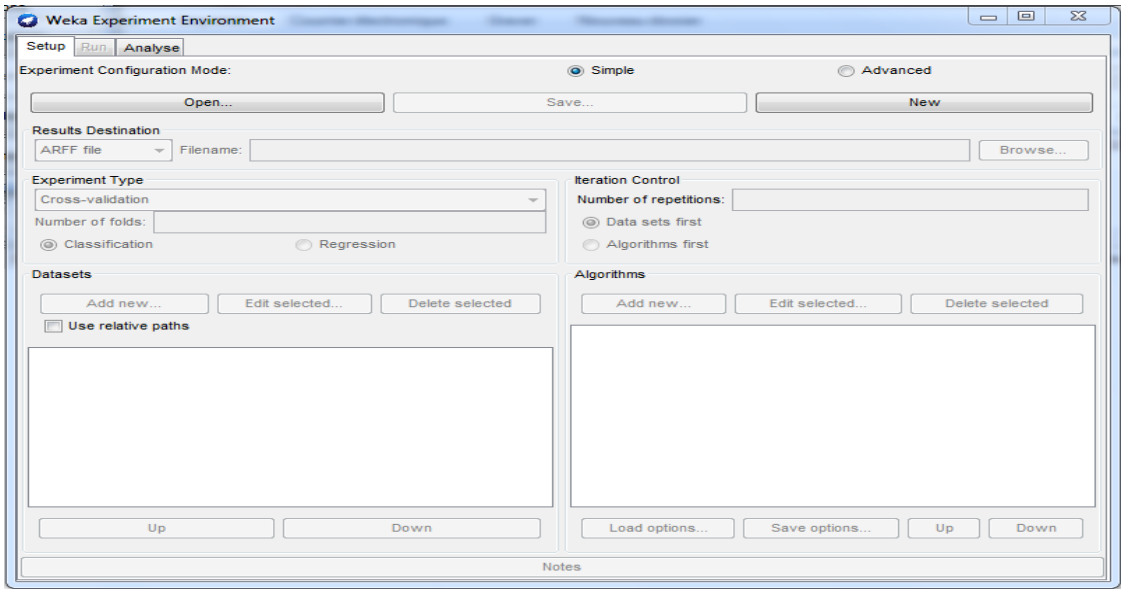

Figure III.10 : *Interface Experimenter*

III.4.2.3. Knowledge Flow

L'interface **Knowledge Flow**, ou « flux de connaissance » atteint à peu près les mêmes fonctionnalités que l'Explorer.

La figure suivante présente l'interface *Knowledge Flow* de Weka

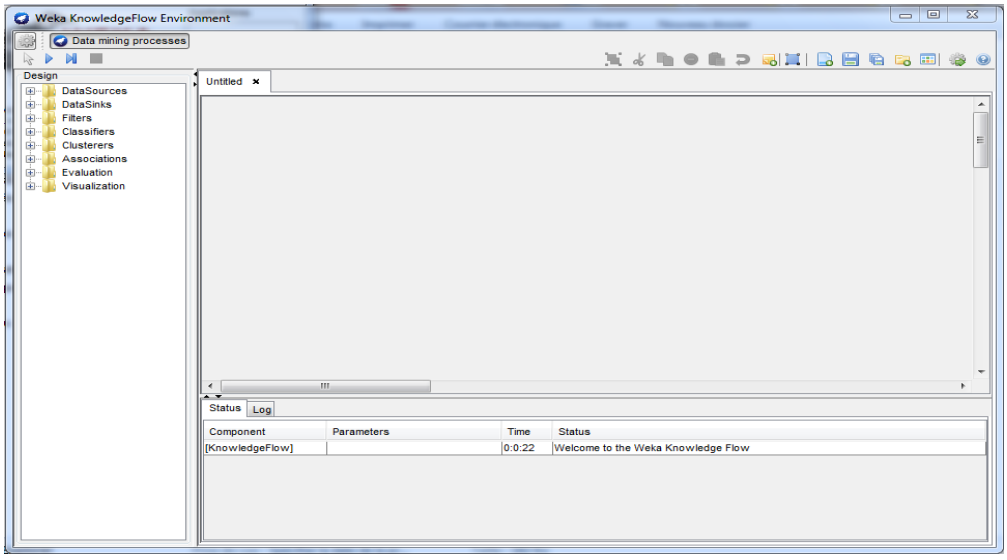

Figure III.11: *Interface Knowmedge Flow*

III.4.2.4. CLI

L'interface **SimpleCLI**, qui est un interpréteur de ligne de commande, elle est recommandée pour une utilisation plus poussée, elle offre des fonctionnalités supplémentaires et utilise beaucoup moins de mémoire.

La figure ci-dessous illustre l'interface *CLI* de Weka

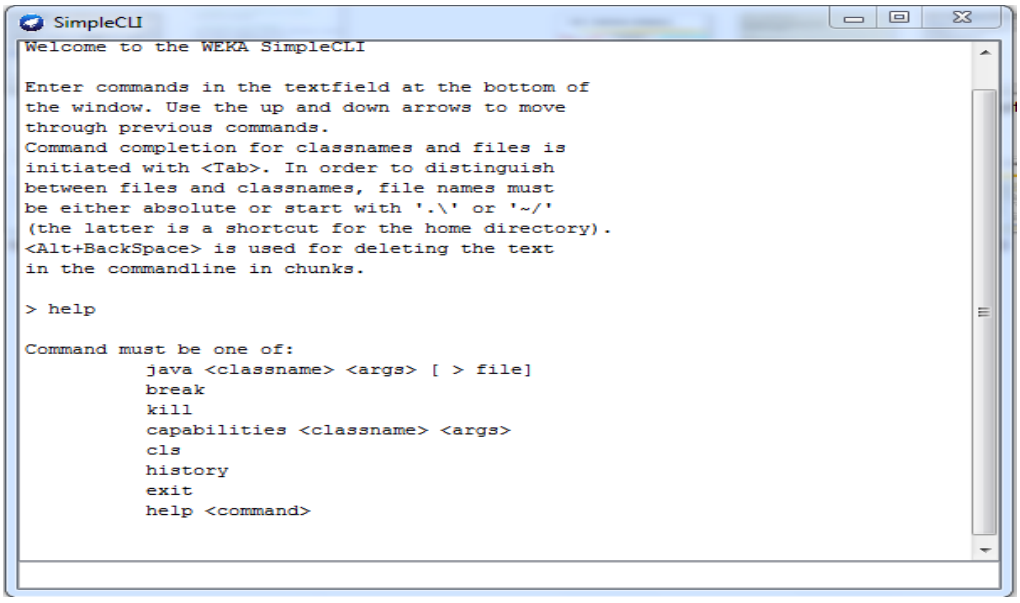

Figure III.12 : *Interface SimpleCLI*

III.4.3. Prétraitement des données

Les prétraitements dans Weka sont effectués grâce aux filtres, qui permettent de modifier les ensembles de données (supprimer ou ajouter des attributs, ré-échantillonner, supprimer des exemples, etc.). Weka propose toute une batterie de filtres (sous l'onglet *Preprocess*), et notamment les filtres d'attributs, souvent utiles pour traiter les données avant apprentissage. Parmi ces filtres, citons :

- **Attribute Selection Filter :** sélection d'attributs selon la classe ;
- **Ajout, suppression et copie d'attributs** (*Add*, *Remove* et *Copy*) ;
- **Ajout de bruit** (*AddNoise*) ;
- **Fusion d'attributs** (*Merge*);
- **Conversion** (*NominalToBinary*, *StringToBinary*, etc.): conversion d'attributs nominaux vers binaires et strings vers binaires ;
- **Numéric Transformation Filter** : transformation d'attributs numériques selon une méthode à préciser (racine carrée, val.absolue, etc.) ;
- **Anonymer** (*obfuscate*);
- **Méthode de discrétisation** (*Discretize Filter*) : discrétise un intervalle d'attributs numériques vers des attributs nominaux.

III.4.4. Classification

III.4.4.1. Algorithmes de classification

Weka implémente un très grand nombre de classificateurs (à base de règles, d'arbres, de réseaux bayésiens, etc.). Parmi ces classificateurs, il y'a :

- **ZeroR** : règle de la classe majoritaire ;
- **J48** : arbre de décision ;
- **NaiveBayes** : Bayes naïf ;
- **IBk** : KNN

III.4.4.2. Validation de modèles

Il existe différentes méthodes de validation de modèles dans Weka, qui se trouvent sur la partie *Test options* de l'onglet *Classify* comme illustrés ci-dessous

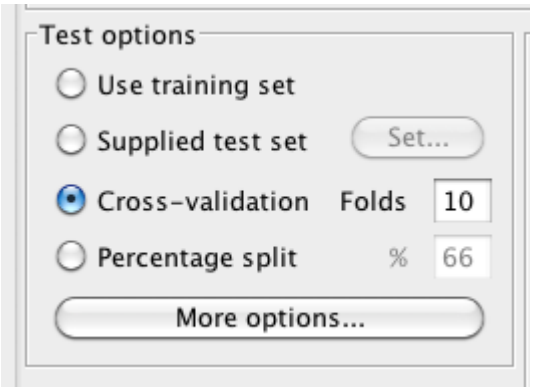

Figure III.13 : *Fenêtre Test options*

- **Use training set :** qui tente d'entraîner un arbre en utilisant tous les exemples d'entraînement, comme résultat un arbre est obtenu, ainsi que le résultat de classification sur ce même ensemble de données ;

- **Supplied test set** (avec le paramètre set – choix du jeu de données pour la validation) : consiste à évaluer le modèle sur un autre jeu de données (a priori différent de celui utilisé pour construire le modèle) ;

- **Cross-validation** (avec comme paramètre « folds ») **:** pour avoir l'effet de validation croisée, c'est-à-dire de découper les données en parties. Elle est utilisée lorsqu'on ne possède pas déjà une collection avec les sous-ensembles d'entraînement et de test déjà séparés. Elle consiste à diviser les données en **n** groupes. Les modèles sont construits sur **n-1** groupes et testés sur le nième groupe. Puis le groupe de test est changé, et le même procédé est répété jusqu'à avoir réalisé toutes les combinaisons. La moyenne des validations est alors considérée comme la validation finale.

- **Percentage split** (avec comme paramètre « pourcentage ») : consiste à utiliser un certain pourcentage des données pour construire le modèle et l'autre partie pour le valider.

III.4.4.3. Processus de classification

Chaque classificateur a sa propre liste de paramètres. Ces classificateurs peuvent être lancés sur un échantillon de données préalablement chargé (et éventuellement filtré par sélection d'attributs, discrétisation ou autre).

L'onglet *Classify* de la fenêtre de l'exploreur Weka permet d'exécuter des processus d'apprentissage, et d'observer les résultats et les performances estimées. Pour ce faire, il faut indiquer le mode d'évaluation (*cross validation*, *empirical evaluation*, etc.), sélectionner le classificateur choisi et indiquer ses paramètres (en cliquant sur la commande), puis lancer la classification. En bas de la fenêtre, la progression de l'apprentissage apparaît.

La fenêtre de droite indique les performances estimées du type de classificateur appris. Il existe ainsi plusieurs méthodes d'estimation de ces performances (qu'il faut sélectionner puis éventuellement paramétrer), mais au final, les mêmes indicateurs sont fournis, parmi lesquels le taux de bonne classification, ou le temps moyen d'apprentissage. En réalisant un clic droit sur le processus terminé dans la liste des résultats (bas-gauche), nous pouvons accéder à la visualisation du modèle, lorsque cette dernière est faisable.

Le modèle appris peut être ensuite sauvegardé, pour être ultérieurement visualisé, et pour évidemment, être entraîné sur de nouvelles données.
La figure qui suit illustre un exemple de classification sous Weka et ses résultats

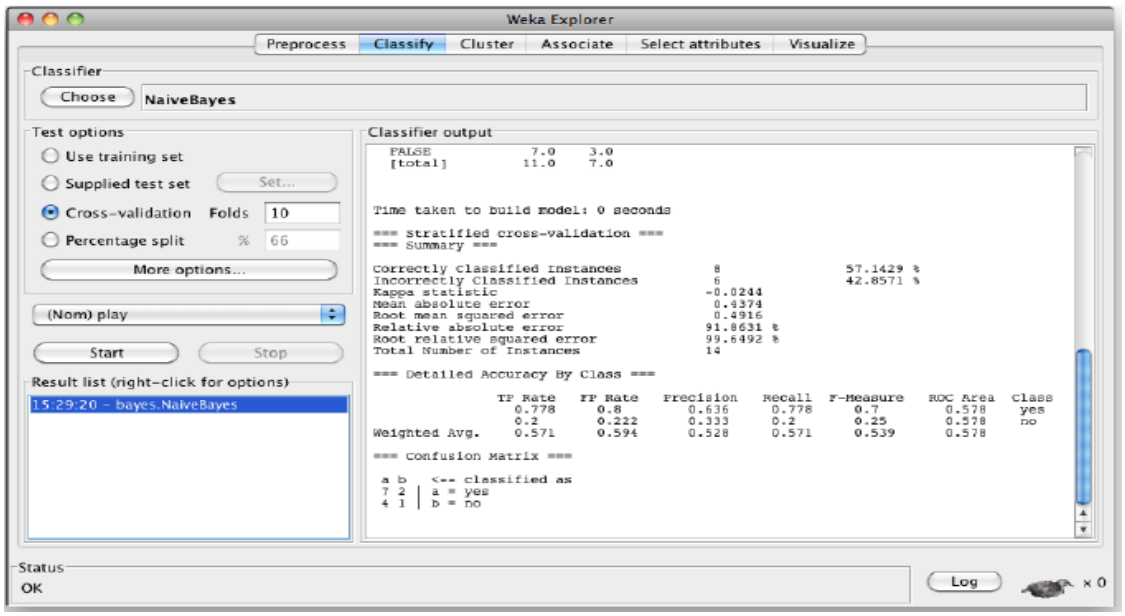

Figure III.14 : *Exemple de classification*

III.5. Conclusion

Weka est un des très rares logiciels à proposer un dispositif assez facile d'accès grâce à son interface conviviale. Il propose beaucoup d'algorithmes classiques en apprentissage automatique (supervisé ou non), avec les fonctionnalités de prétraitement des données, analyse et évaluation des résultats.

Dans ce chapitre nous avons eu un aperçu des fonctionnalités de classification de Weka ainsi que ces méthodes de base. Le but était de nous familiariser avec ce logiciel, car bien qu'on puisse aisément comprendre certaines fonctionnalités de base de Weka, pour en faire une utilisation réellement utile, il s'avère absolument nécessaire d'avoir une compréhension des modèles mathématiques proposés, voire d'utiliser son propre modèle.

CHAPITRE IV

Implémentation et interprétation des résultats

IV.1. Introduction

Ce chapitre est essentiellement consacré aux grandes lignes qui visent à réaliser l'objectif de ce thème, et les outils exploités pour le développement tels que le choix du langage de programmation, l'environnement de programmation et les logiciels utilisés. Ensuite, nous discuterons et interpréterons les résultats, à savoir les résultats de la classification supervisée effectué sur deux collections de documents textuels. Enfin, conclure avec une comparaison des différentes méthodes utilisées et de leurs performances respectives.

IV.2. Technologies et outils de développement

Dans cette section, nous allons présenter l'environnement de développement de notre programme, le langage de programmation utilisé ainsi que les outils d'indexation que nous avons utilisé.

IV.2.1. Langage de programmation « Java »

Pour le choix de programmation de notre système nous avons opté pour le langage JAVA et cela pour de nombreuses raison :

- Java est un langage orienté objet simple, qui réduit le risque d'erreurs d'incohérence.
- Il permet d'accéder d'une manière simple aux fichiers et aux réseaux.
- Il est caractérisé aussi par la réutilisation de son code ainsi que la simplicité de sa mise en œuvre.
- Il possède une riche bibliothèque de classes comprenant des fonctions divers telles que les fonctions standards, le système de gestion de fichiers ainsi que beaucoup de fonctionnalités qui peuvent être utilisées pour développer des applications diverses.

IV.2.2. Environnement de développement « NetBeans » [21]

NetBeans est un environnement de développement intégré (EDI), placé en Open Source par Sun en juin 2000 sous licence CDDL et GPLv2 (*Common Development and Distribution License*). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous **Windows**, **Linux**, **Solaris** (**sur x86 et SPARC**), **Mac OS X** ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

L'environnement de base comprend les fonctions générales suivantes :

- Configuration et gestion de l'interface graphique des utilisateurs,
- Support de différents langages de programmation,
- Traitement du code source (édition, navigation, formatage, inspection,…),

- Fonctions d'import/export depuis et vers d'autres IDE, tels qu'*Eclipse* ou *JBuilder*,
- Accès et gestion de bases de données, serveurs Web, ressources partagées,
- Gestion des tâches
- Documentation intégrée.

La version utilisée dans notre projet est **NetBeans IDE 8.0.2**

La figure suivante illustre l'interface de l'environnement de développement NetBeans IDE 8.0.2

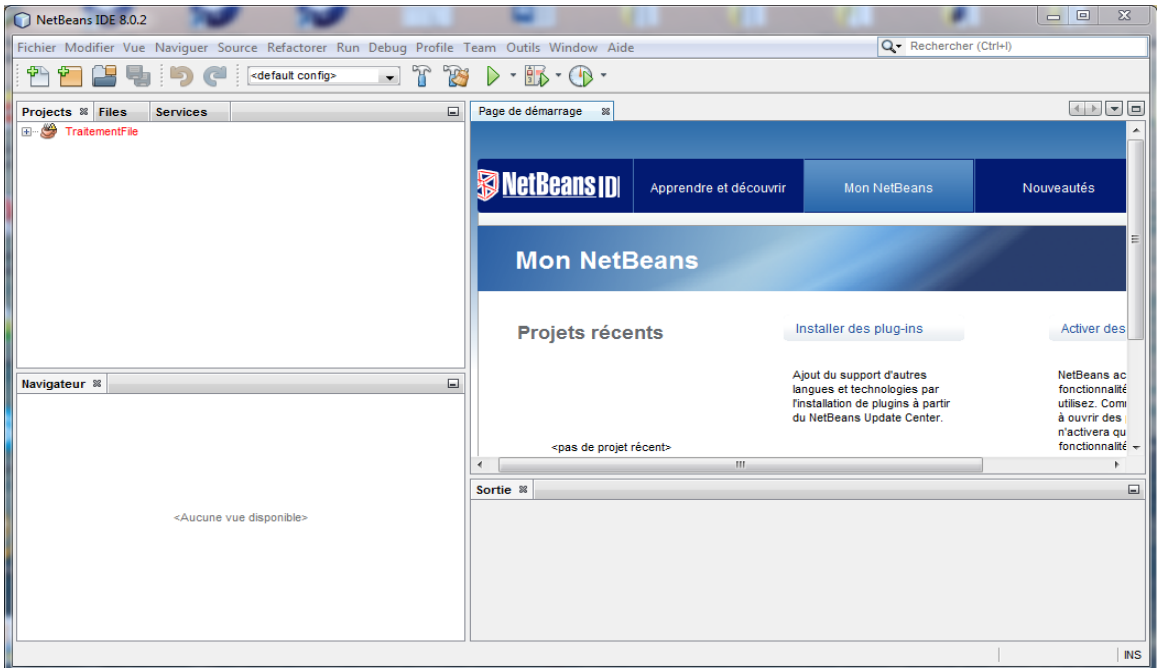

Figure IV.1: *Interface NetBeans IDE 8.0.2*

IV.2.3. Stanford POS Tagger [22]

Stanford POS Tagger (*Part Of Speech Tagger*) est un étiqueteur grammatical développé par l'Université de Stanford. Son processus consiste à lire un bout de texte dans une langue donnée et d'assigner les informations grammaticales correspondantes à chaque mot prit séparément, tel que les noms, verbes, adjectifs, etc.

Il a été développé avec le langage Java par **Kristina Toutanova**, étudiante à l'Université de Stanford puis chercheuse dans le domaine du traitement automatique du langage naturel (*Natural Language Processing* (**NLP**)). Il est adapté à plusieurs langues autres que l'anglais, tel que le chinois, le français, l'allemand, l'Arabe. Il dispose d'un corpus anglais pré-étiqueté appelé « *The Penn Treebank tag set* » contenant les abréviations des informations grammaticales utilisées par le Stanford Tagger.

Il a été développé sous la licence *GNU General Public License* (v2 or later). Le package Stanford POS Tagger inclut différents composants pour une utilisation sous ligne de commande, pour une utilisation autant que server, ou encore, autant qu'une API Java.

Il peut être utilisé avec plusieurs langages dont, Ruby, Python, C#, Gate, XML-RPC, Javascript, PHP, Matlab et ceci simplement, en intégrant les bibliothèques de Stanford Tagger au programme.
Le site **nlp.stanford.edu/software/tagger.shtml** dispose des différentes versions de Stanford POS Tagger téléchargeable avec une documentation intégrée.

La figure suivante représente les étiquettes grammaticales utilisées dans Stanford POS Tagger

Tag	Description
JJ	Adjective
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
POS	Possessive ending
PRP	Personal pronoun
RB	Adverb
RP	Particle
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle

Figure IV.2 : *Etiquettes grammaticales dans Stanford POS Tagger*

IV.2.4. **RiTa** [23]

Conçu pour soutenir la création de nouvelles œuvres de littérature informatique, la bibliothèque RiTa fournit des outils pour les artistes et les écrivains qui travaillent avec le langage naturel dans les médias programmables. La bibliothèque est conçue pour être simple tout en permettant une gamme de fonctionnalités puissantes telles que la *tokenisation*, *l'étiquetage grammatical*, la *racinisation* (*stemming*), *l'élimination des ponctuations*, etc.

RiTa peut également être intégrée avec son propre lexique personnalisable par l'utilisateur, ou avec la base de données WordNet. Elle est mise en œuvre à la fois avec *Java* et *Javascript*, et est libre et Open Source, elle fonctionne dans un certain nombre d'environnements de programmation populaires tels que *NetBeans*, *Eclipse*, *JBuilder*, et y compris *Android*, *Traitement*, *Node* et *p5.js*.

IV.2.5. **Weka** [24]

Weka (*Waikato Environment for Knowledge Analysis*), autrement dit, environnement Waikato pour l'analyse de connaissances, est un package Open Source très populaire, en d'autres termes, un ensemble de classes et d'algorithmes développé sous Java, à *l'Université de Waikato* en *Nouvelle-Zélande*. Il propose différents algorithmes d'apprentissage automatique (supervisée

ou non), à savoir, *Naïve Bayes*, *Arbre de décision*, *SVM*, *réseau de neurones*, etc. avec les fonctionnalités de prétraitement des données, analyse et évaluation des résultats. Il offre une interface GUI conviviale pour manipuler et inspecter les données et visualiser les résultats. Ce package peut fonctionner sur les plateformes Linux, Windows et Mac.

La version weka utilisé dans notre travail est **Weka 3.7**

La figure ci-dessous présente la fenêtre d'invite au lancement de Weka

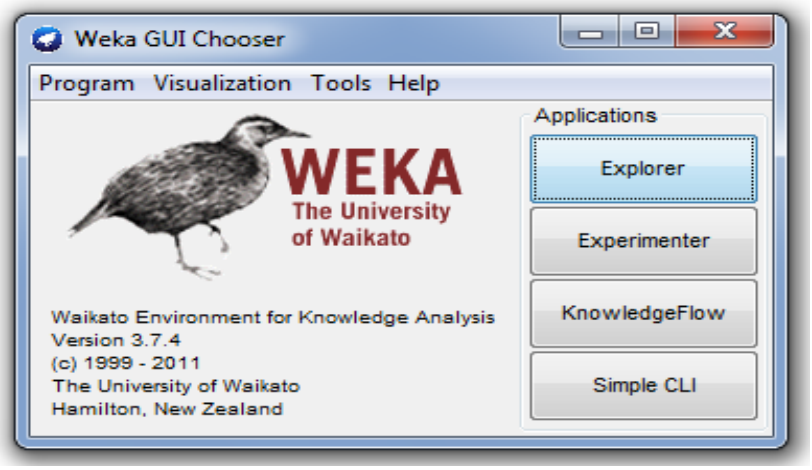

Figure IV.3 : *Fenêtre d'invite Weka 3.7*

IV.3. Collection utilisée pour l'évaluation

Afin de pouvoir évaluer les systèmes de classification il est nécessaire d'utiliser un corpus. De nombreux types de corpus ont vu le jour ces dernières années qui s'organisent en différentes typologies.

Quelques bases de textes ont donc émergées comme « corpus de référence » pour la classification de textes. Ils doivent regrouper un certain nombre de documents qui sont tenus d'être de diverses utilisations. Ainsi une dimension suffisante et la diversité des documents sont deux caractéristiques principales d'un corpus pour qu'il soit qualifié de « référence ».

L'utilisation de ces corpus standards permet ainsi une comparaison plus aisée des performances des différentes techniques de classification.

Dans ce présent travail nous avons eu recours à deux corpus : le corpus Reuters et un corpus que nous nommerons corpus Master conçu par notre encadreur.

IV.3.1. Corpus 1

IV.3.1.1 Présentation générale du corpus Reuters

Reuters est un corpus de dépêches en langue anglaise qui a été proposé par l'agence de presse Reuters en 1987. Il correspond à une problématique de classification en plusieurs classes (un document appartient à une ou plusieurs classes).

Deux qualités principales caractérisent les documents Reuters c'est qu'ils sont courts et plutôt homogènes avec un vocabulaire riche (environ 17000 mots).

Les diverses expérimentations des chercheurs sur la base ont fait de ce corpus une référence dans le domaine de la classification supervisée de textes.

IV.3.1.2 Historique

Ce corpus initialement nommé Reuters-22173 comportait 22173 dépêches de presse qui ont été publiés en 1987. Tous les articles ont été rassemblés et indexés à des catégories par le personnel de l'agence et le groupe Carnegie (Carnegie Group Inc – CGI).

En 1990, les documents ont été mis à disposition du laboratoire de recherche d'information (Université de Massachusetts à Amherst), par Reuters et CGI, pour des fins de recherche. Le formatage des documents et la génération de fichiers associés ont été réalisés en 1990 par David D.Lewis et Stephen Harding au même laboratoire.

Un autre formatage et génération des fichiers associés ont été fait en 1991 et 1992 par David D.Lewis et Peter Shoemaker au centre d'information et études du langage (Université de Chiacgo). Cette base de fichiers a donné naissance le 01/01//1993, à la première version dénommé « Reuters-22173 ».

IV.3.1.4. Corpus Reuters-21578

Le corpus Reuters-21578 est composé de 21578 documents extraits du journal « Reuters » en 1987. Il contient 21578 documents dans 90 catégories. Ce corpus est souvent utilisé comme base de comparaison entre les différents outils de classification de documents, ceci d'une part. D'autre part, on retiendra que le Reuters-21578 est souvent qualifié de « corpus difficile » pour des traitements complexes.

- **Les différentes catégories du corpus Reuters-21578**

Les mises à jour de Reuters-21578, ont été obtenues par la suppression des documents non étiquetés que comportait la version précédente (textes ambigus), elles ont permis de supprimer les documents redondants, de corriger des erreurs typographiques, et la conservation des catégories ayant au moins un document dans la base d'apprentissage et un dans la base de test pour se ramener à moins d'une centaine de catégories. Il en résulte donc 90 catégories avec 9586 documents pour l'ensemble d'apprentissage et 3745 pour l'ensemble de test.

Ces 90 catégories sont classées par ordre décroissant du nombre de documents associés à la base d'apprentissage. Le nombre de documents disponibles pour effectuer l'apprentissage décroît rapidement ; dès la vingt-sixième catégorie, ce nombre est inférieur à cinquante. Il faut noter que les documents sont à peu près également répartis sur les deux bases, c'est-à-dire que les catégories ayant beaucoup (respectivement peu) de documents associés à la base d'apprentissage ont aussi beaucoup (respectivement peu) de documents associés à la base de test.

Dans le tableau ci-dessous on présente 10 catégories parmi les 90 issues du découpage des ensembles d'apprentissage et de tests, ainsi que le nombre de documents associés disponibles sur chaque base.

Tableau IV.1 : *Les 10 catégories de corpus Reuters avec le nombre de documents associés*

	Catégorie	**Apprentissage**	**Test**
1	*earn*	2877	1087
2	*Acquisition*	1650	719
3	*Money-fx*	538	179
4	*Grain*	433	149
5	*Crude*	389	189
6	*Trade*	369	118
7	*Interest*	347	131
8	*Wheat*	212	71
9	*Ship*	197	89
10	*Corn*	192	56

IV.3.2. Corpus 2

Ce corpus regroupe une panoplie de 110 documents de catégories différentes, à savoir, des documents dans les domaines : politiques, économie, sport, musique et droit. Il a été réalisé et mis au point par notre encadrant M^{elle} Iltache Samia à partir du corpus Brown et de Wikipédia.

IV.4. Implémentation

Les tests que nous présentons ici concernent l'application de systèmes de classification automatique (méthodes de classification automatique) pour la catégorisation d'un ensemble de documents.

Lorsqu'il est possible d'attribuer une ou plusieurs catégories à un ensemble de documents, ces systèmes peuvent, à partir d'un corpus ou la catégorie de chaque document est connue a priori, « apprendre » une définition de ces catégories et ensuite les attribuer automatiquement à des nouveaux documents.

IV.4.1. Partie Indexation

Cette partie a consisté en la détermination et l'extraction des termes représentatifs de notre collection de données qui nous permettra ainsi, de passer d'un document textuel à un document qui peut être utilisé dans la partie classification automatique, à savoir, une représentation

vectorielle des textes. Ainsi, chaque document est représenté par un vecteur, où chaque terme du document se voit attribué un poids basé sur le calcul TF * IDF.

Notre processus d'indexation a consisté en plusieurs étapes qui sont :

IV.4.1.1. Analyse lexicale (tokenisation)

C'est la première phase de l'indexation, elle consiste à découper le texte en unités élémentaires. Pour cela, nous avons utilisé la bibliothèque ***RiTa*** qui fournit un outil de tokenisation (la méthode ***tokenize***).

IV.4.1.2. Elimination des ponctuations

Cette phase consiste à éliminer tout symbole qui ne correspond pas à une lettre de l'alphabet car ils ne sont pas liés au contenu des documents. Et pour cela, nous avons utilisé la bibliothèque ***RiTa*** sui fournit aussi, un outil de dé-ponctuation (la méthode ***StripPunctuation***).

IV.4.1.3. L'étiquetage grammatical

C'est le processus qui permet d'associer des catégories morphosyntaxiques aux termes du texte. Dans cette phase, nous avons utilisé l'étiqueteur grammatical ***Stanford POS Tagger***.

IV.4.1.4. Elimination des mots vides

Ça consiste à filtrer et éliminer les mots vides de sens (article, proposition, conjoncture, etc.). Pour réaliser cette phase nous avons extrait de nos documents les verbes, les noms et les adjectifs, et nous avons éliminé le reste.

IV.4.1.5. Normalisation

Qui consiste à représenter les différents termes de notre collection sous leur forme radicale. Et afin de réaliser cette phase, nous avons opté encore une fois pour la bibliothèque RiTa qui offre un outil de stemmatisation (la méthode ***RiTa.stem***), qui s'inspire de l'algorithme de Porter [9].

IV.4.1.6. Choix des descripteurs

Cette phase consiste en la sélection des termes les plus représentatifs de nos textes, à savoir, les mots clés. Dans notre cas, nous avons sélectionné les verbes, les noms et les adjectifs comme descripteurs de notre corpus.

IV.4.1.7. Représentation vectorielle des documents textuels

Pour notre utilisation ultérieure (classification), il est nécessaire de pouvoir représenter nos documents textuels sous forme vectorielle. Et pour réaliser cette phase, nous avons utilisé le modèle de pondération des termes de *Salton* [10], qui combine un facteur de pondération local TF, qui mesure la fréquence du terme dans un document (plus un terme est fréquent dans un document, plus il est important dans la description de ce document), et un second facteur de pondération global IDF, mesurant l'importance (fréquence) du terme dans toute la collection (les termes qui apparaissent dans peu de documents de la collection sont plus représentatifs du contenu). Enfin, le poids s'obtient en multipliant les deux mesures TFIDF.

Chaque unité documentaire peut alors faire l'objet d'une représentation vectorielle : les coordonnées représentant les fréquences des mots représentatifs. Et on obtient au final, la matrice représenté ci-dessous


```
'C08_partie1.music'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'C08_partie2.music'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
'C08_partie3.music'0,0,0,0,0,0,0,0,0.017,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'C09_partie2.music'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'C10_partie4.music'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'C11_partie1.music'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
'C11_partie4.music'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'C12_partie3.music'0,0,0,0,0,0,0,0.014,0.01,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.014,0,0,0,0,0,
'eco13_ecnomscale.economy'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'eco14_microecnomy.economy'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.018,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'eco15.market.economy'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'eco16.market2.economy'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.023,0,0,0,
'J37.politic'0,0,0,0.005,0,0,0,0,0,0,0,0,0,0.005,0,0,0,0,0,0,0.009,0,0,0,0,0,0,0,0,0,0,0,0,0.005,0,0,0,0,0,
'J39.economy'0,0,0,0,0,0,0,0,0,0,0,0,0,0.005,0.005,0,0,0,0,0,0,0.004,0,0,0,0,0,0,0,0,0,0,0,0,0.005,0.002,0,0,
'J40.economy'0.017,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.002,0,0,0,0,0,0,0,0,0,0,0,0,0,0.005,0,0,0,0,0,0,
'j41.economy'0,0,0,0,0,0,0.003,0.002,0.002,0.003,0,0,0,0,0,0,0,0,0,0,0,0.002,0,0,0,0,0,0,0,0,0,0.005,0.003,0
'J50.economy'0,0,0,0,0,0,0.003,0.002,0,0,0,0,0,0,0,0,0,0,0,0,0,0.002,0,0,0,0,0,0,0,0,0,0,0.003,0.002,0,0,0,0,
```


Figure IV.4 : *Représentation vectorielle des documents*

IV.4.2. Classification avec Weka

Cette partie consiste à traiter la matrice obtenu ci-dessus avec les algorithmes de classification que nous allons utiliser, à savoir, l'algorithme SVM, K-PPV, Naïve Bayes et l'arbre de décision, pour cela, nous allons utiliser l'outil WEKA, qui implémente ces différents algorithmes. Le choix de cet outil est dû au fait qu'il est très utilisé dans le domaine de l'apprentissage et de la fouille de données et il est aussi facile à manipuler. Mais avant de soumettre la matrice à ces différents algorithmes, il est nécessaire de représenter cette matrice sous le format adapté au package WEKA, qui est le format ARFF.

La figure qui suit représente le fichier ARFF obtenu :


```
@attribute 'AllStar' real
@attribute 'sagging' real
@attribute 'CONSERVATION' real
@attribute 'flew' real
@attribute 'Grayson' real
@attribute 'carried' real
@attribute 'hurrying' real
@attribute 'BAM' real
@attribute 'response' real
@attribute 'Minnie' real
@attribute 'delegation' real
@attribute 'rival' real
@attribute 'straightaway' real
@attribute class {law,politic,economy,music,sport}
@data
'A01_partie1.law'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.005,0,0,0,0,0,0,0,0,0,0,0,0,0.007,0,0,
'A01_partie2.politic'0,0,0,0,0,0,0.024,0,0,0.025,0,0,0,0,0,0.019,0,0.028,0.022,0,0,0,0,0,0,0,0,0,0,
'A01_partie3.politic'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.019,0,0,0,0,0,0,0,0,
'A01_partie5.politic'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.017,0,0,
'A01_partie6.politic'0,0,0,0,0,0,0.031,0,0.029,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.031,0,
'A01_partie7.politic'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'A01_partie8.politic'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.019,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'A02_partie2.politic'0,0,0,0,0,0,0,0,0,0.008,0,0,0,0,0,0,0,0,0.007,0,0,0,0.009,0,0,0,0.011,0,0,0,0,
'A02_partie3.politic'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.032,
'A02_partie4.politic'0,0,0,0,0,0,0,0,0,0,0,0.028,0,0,0,0.014,0,0,0,0,0,0,0,0,0,0,0,0,0.017,0,0,0,0,
'A02_partie5.politic'0,0,0,0,0,0,0,0.018,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.018,0,0,
'A02_partie8.politic'0,0,0,0,0,0,0,0.013,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.013,0,0,
'A03_partie1.law'0,0,0,0,0,0,0.024,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.024,0,0,0,0,0,
'A03_partie2.politic'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'A03_partie3.law'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.008,0,0,0.01,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.01,0,
```


Figure IV.5 : *fichier ARFF*

Après avoir représenté notre matrice dans un fichier en format ARFF, l'étape suivante consiste à ouvrir ce fichier avec le logiciel Weka afin de passer à l'étape de classification.

La figure suivante illustre le fichier ARFF lu avec le logiciel Weka

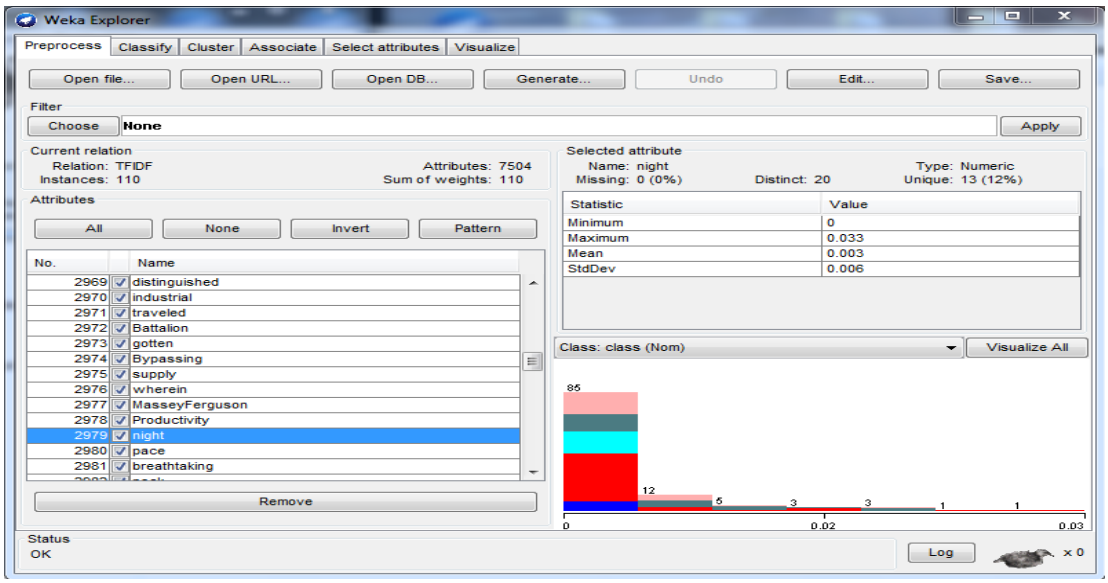

Figure IV.6 : *Lecture du fichier ARFF sous Weka*

IV.4.2.1. Algorithmes de classification appliqués

Les algorithmes de classification testés dans notre projet sont :

- **Machine à vecteurs de support (SVM) :** qui est désigné par le nom **SMO** dans le logiciel WEKA ;
- **K plus proches voisins (KNN) :** qui est lui désigné par le nom **Ibk** dans WEKA ;
- **Naïve Bayes :** qui garde le même nom sous WEKA
- **Arbre de décision :** qui est désigné par l'appellation **J48** sous WEKA.

IV.4.2.2. Expérience et résultats

Comme on l'a déjà mentionné précédemment, on va s'intéresser à la classification. Dans Weka, on peut trouver tous les algorithmes correspondant dans l'onglet *Classify*. Après avoir chargé les données dans l'outil, on applique les différentes fonctions destinées pour cette étude. Weka offre quatre options pour faire l'évaluation de la performance du modèle appris, dans cette expérience, on ne va se pencher que sur l'une d'elle, à savoir, la validation croisée (*Cross-validation*). L'ensemble d'apprentissage va être subdivisé en 10 parties (**fold = 10**) ; l'algorithme va apprendre 10 fois sur 9 parties et la dernière partie sert à évaluer le modèle puis les 10 évaluations sont combinés pour calculer une moyenne de résultats.

Les résultats de cette expérience de classification pour chaque algorithme et dans chaque collection de documents sont représentés dans ce qui suit :

A. Corpus 1

➢ Algorithme SVM (SMO sous Weka)

La figure IV.7 représente les résultats de la classification avec l'algorithme **SVM**

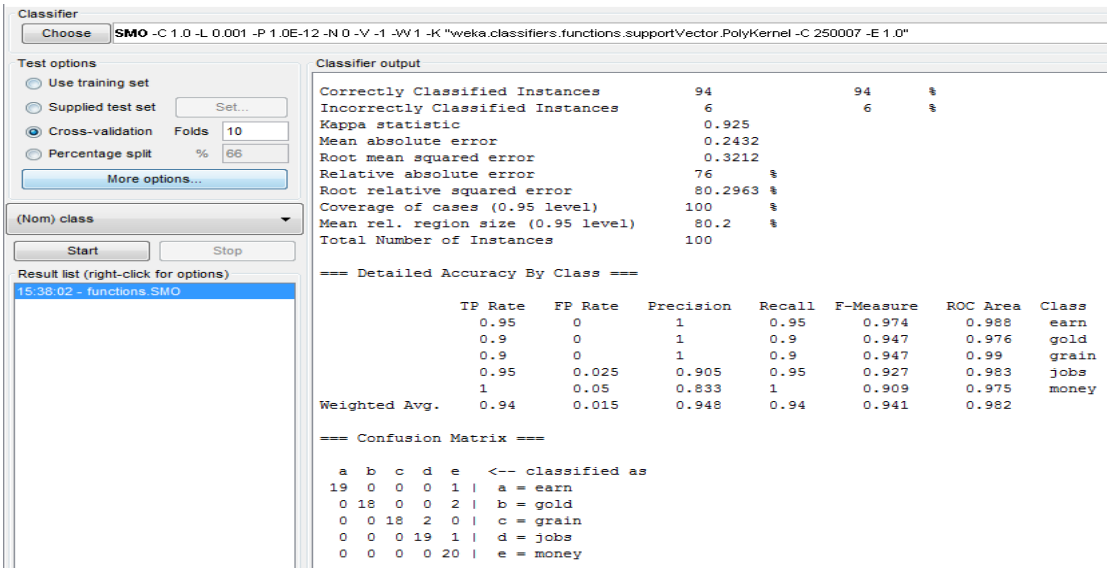

Figure IV.7 : *classification avec l'algorithme SVM*

D'après cette figure, on voit que 94% des documents ont été classés correctement. La matrice de confusion en bas, indique que les quelques erreurs ont concerné les classes « *earn*, *gold*, *grain* et *jobs* ». Les classes *earn* et *jobs* ont 19 documents sur 20 correctement classés et les classes *gold* et *grain* ont quant à elles 18 documents sur 20 correctement classés.

Le tableau suivant présente les mesures d'exactitude par classe pour la méthode SVM, on les trouve dans la partie « Detailed Accuracy By Class ».

Tableau IV.2 : *Mesures d'exactitude par classe pour la méthode SVM*

Classe	TP Rate	FP Rate	Precision	Recall	F-Measure
earn	0,95	0	1	0,95	0.974
gold	0,9	0	1	0,9	0,947
grain	0,9	0	1	0,9	0,947
jobs	0,95	0,025	0,905	0,95	0,927
money	1	0,05	0,833	1	0,909

- **TP Rate :** c'est le rapport des vrais positifs. Il correspond à :
  Nombre de vrais positif / Nombre d'exemple de cette classe
  C'est donc le rapport entre le nombre de bien classé et le nombre total d'éléments qui devraient être bien classé.

- **FP Rate :** *Nombre de faux positifs / Nombre d'exemples n'étant pas de cette classe*

  Les données des taux TP Rate et FP Rate permettent de reconstruire la matrice de confusion pour une classe donnée. Symétriquement, la matrice de confusion permet de calculer TP Rate et FP Rate.

- **Precision :** c'est le rapport entre le nombre de vrais positifs et la somme des vrais positifs et des faux positifs. Une valeur de 1 exprime le fait que tous les exemples classés positifs l'étaient vraiment.

- **Recall :** un Recall de 1 signifie que tous les exemples positifs ont été trouvés.

- **F-Measure :** cette quantité permet de regrouper en un seul nombre les performances du classifieur (pour une classe donnée).
  *F-Measure = (2 * Recall * Precision) / (Recall * Precision)*

➢ Algorithme KNN (Ibk sous Weka)

La figure IV.8 représente les résultats de la classification avec l'algorithme **KNN**

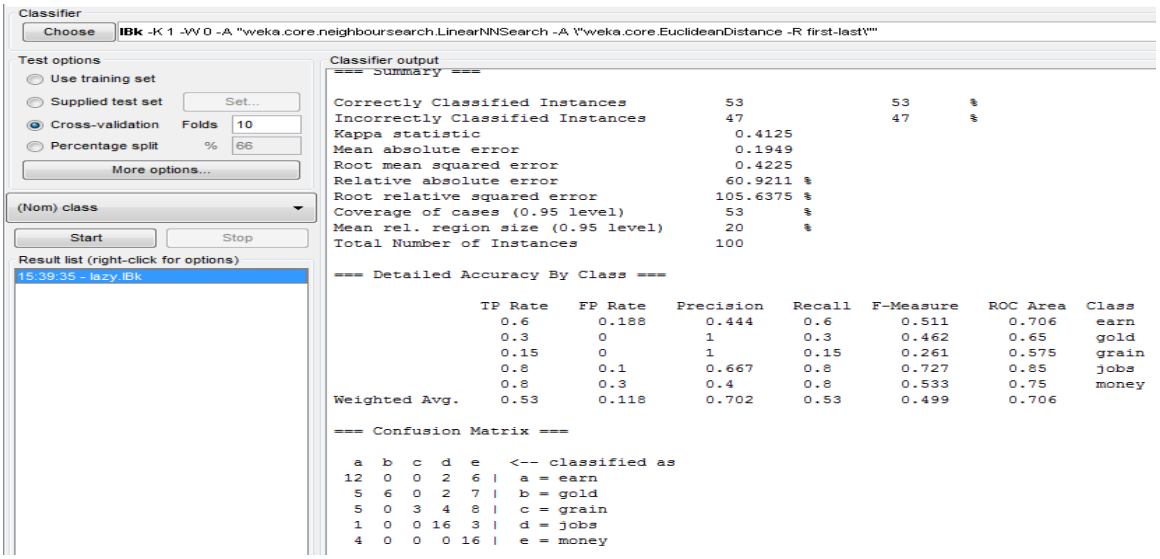

Figure IV.8 : *classification avec l'algorithme KNN*

Dans cette figure, on remarque que 53% des documents ont été classés correctement. La matrice de confusion, indique que les erreurs ont concerné toutes les classes. Par exemple, on voit que la classe grain a eu seulement 3 documents sur 20 correctement classés et le reste a été distribué sur les classes *earn* (5 documents), *jobs* (4 documents) et *money* (8 documents).

Le tableau suivant présente les mesures d'exactitude par classe pour la méthode KNN, on les trouve dans la partie « Detailed Accuracy By Class ».

Tableau IV.3 : *Mesures d'exactitude par classe pour la méthode KNN*

Classe	TP Rate	FP Rate	Precision	Recall	F-Measure
earn	0,6	0,188	0,444	0,6	0,511
gold	0,3	0	1	0,3	0,462
grain	0,15	0	1	0,15	0,261
jobs	0,8	0,1	0,667	0,8	0,727
money	0,8	0,3	0,4	0,8	0,533

- **Algorithme Naive Bayes**

La figure IV.9 montre les résultats issus de la classification avec l'algorithme **Naïve Bayes**

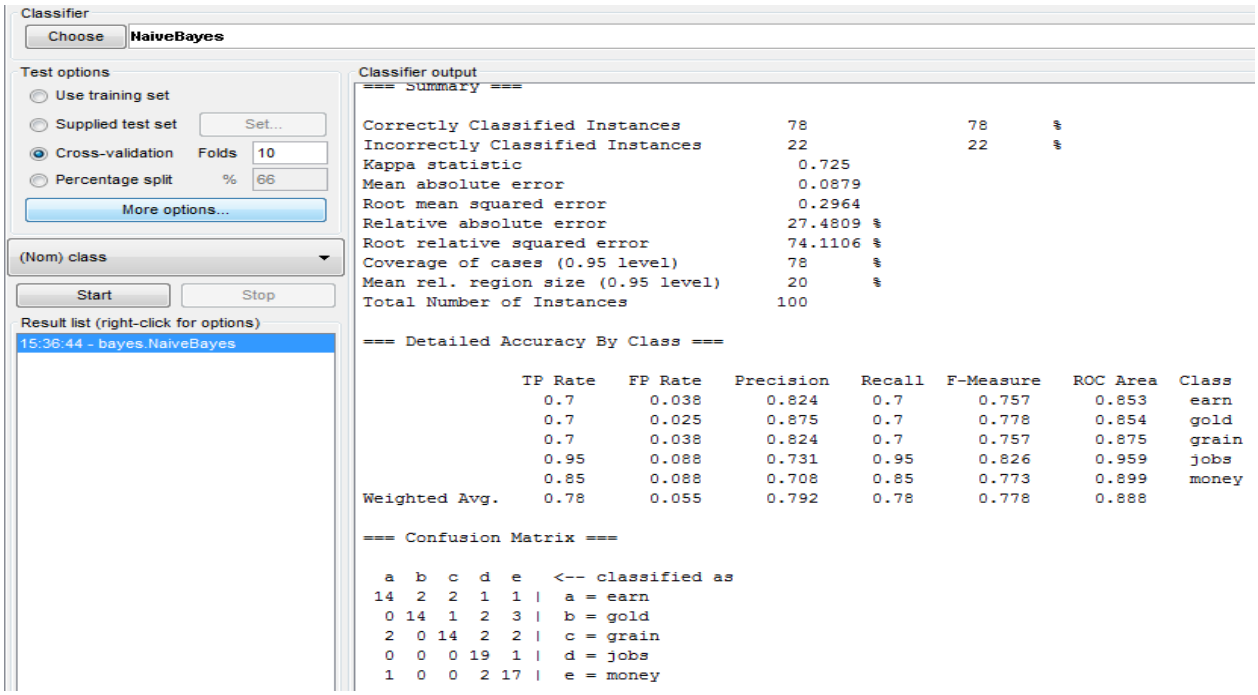

Figure IV.9 : *classification avec l'algorithme Naïve Bayes*

D'après la figure ci-dessus, on constate que 78% des documents ont été classés correctement. La matrice de confusion montre que les erreurs ont concerné toutes les classes mais elles ne dépassent pas trois erreurs par classe.

Le tableau suivant présente les mesures d'exactitude par classe pour la méthode Naïve Bayes, on les trouve dans la partie « Detailed Accuracy By Class ».

Tableau IV.4 : *Mesures d'exactitude par classe pour la méthode Naïve Bayes*

Classe	TP Rate	FP Rate	Precision	Recall	F-Measure
earn	0,7	0,036	0,824	0,7	0,853
gold	0,7	0,025	0,875	0,7	0,854
grain	0,7	0,036	0,824	0,7	0,875
jobs	0,95	0,088	0,731	0,95	0,959
money	0,85	0,088	0,708	0,85	0,899

- **Algorithme Arbre de décision (J48 sous Weka)**

La figure IV.10 illustre les résultats de classification avec l'algorithme **Arbre de décision**

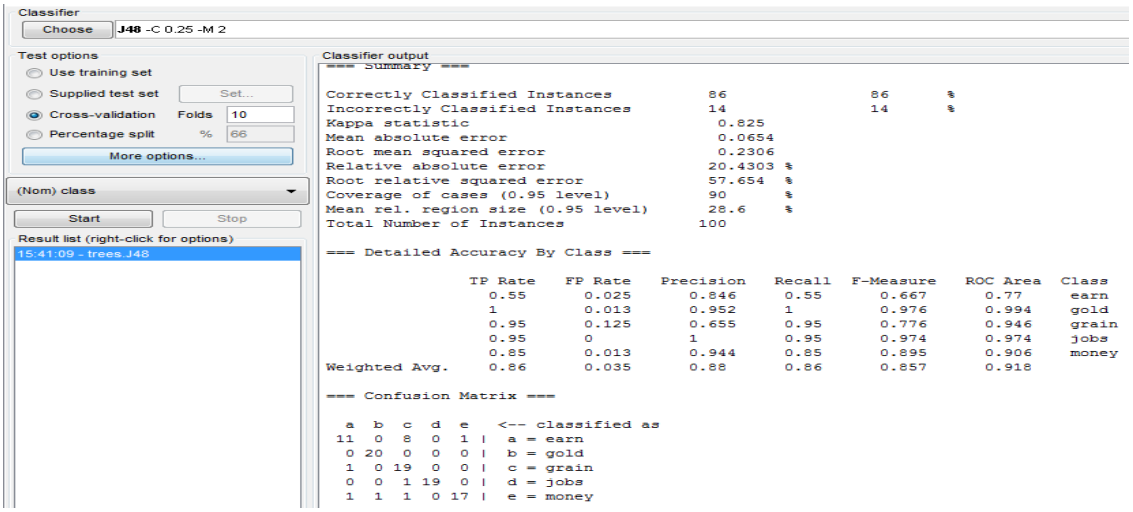

Figure IV.10 : *classification avec l'algorithme J48*

Cette figure nous indique que 86% des documents ont été classés correctement. La matrice de confusion en bas, indique que les erreurs ont concerné les quatre classes « *earn*, *grain*, *jobs* et *money* », et particulièrement la classe *earn* qui a enregistré 11 documents sur 20 correctement classés.

Le tableau suivant présente les mesures d'exactitude par classe pour la méthode J48, on les trouve dans la partie « Detailed Accuracy By Class ».

Tableau IV.5 : *Mesures d'exactitude par classe pour la méthode J48*

Classe	TP Rate	FP Rate	Precision	Recall	F-Measure
earn	0,55	0,025	0,846	0,55	0,77
gold	1	0,013	0,952	1	0,994
grain	0,95	0,125	0,655	0,95	0,946
jobs	0,95	0	1	0,95	0,947
money	0,85	0,013	0,944	0,85	0,906

Dans le tableau suivant nous allons énumérer les quatre algorithmes avec le nombre de documents correctement classifiés et le nombre de documents incorrectement classifiés pour chacun d'eux.

Tableau IV.6 : *Evaluation des performances de classification du corpus Reuters*

Algorithmes de classification	Nombre de documents correctement classifiés	Nombre de documents incorrectement classifiés
SVM	94	6
KNN	53	47
Naïve Bayes	78	22
Arbre de décision	86	14

Le tableau qui suit illustre les pourcentages de documents correctement classifiés pour les différents algorithmes

Tableau IV.7 : *Pourcentage de bonne classification pour les différents algorithmes (corpus Reuters)*

Algorithmes de classification	Pourcentage de classification correcte
SVM	94%
Arbre de décision	86%
Naïve Bayes	78%
KNN	53%

D'après les résultats de classification de la première collection, on remarque que la méthode SVM enregistre le meilleur pourcentage de documents correctement classifiés (94%) puis viens l'arbre de décision et la méthode Naïve Bayes avec 86% et 78% respectivement. Enfin, la méthode KNN enregistre le pourcentage le plus faible avec 53% de documents correctement classifiés.

B. Corpus 2

➢ Algorithme SVM (SMO sous Weka)

La figure IV.11 représente les résultats de la classification avec l'algorithme **SVM**

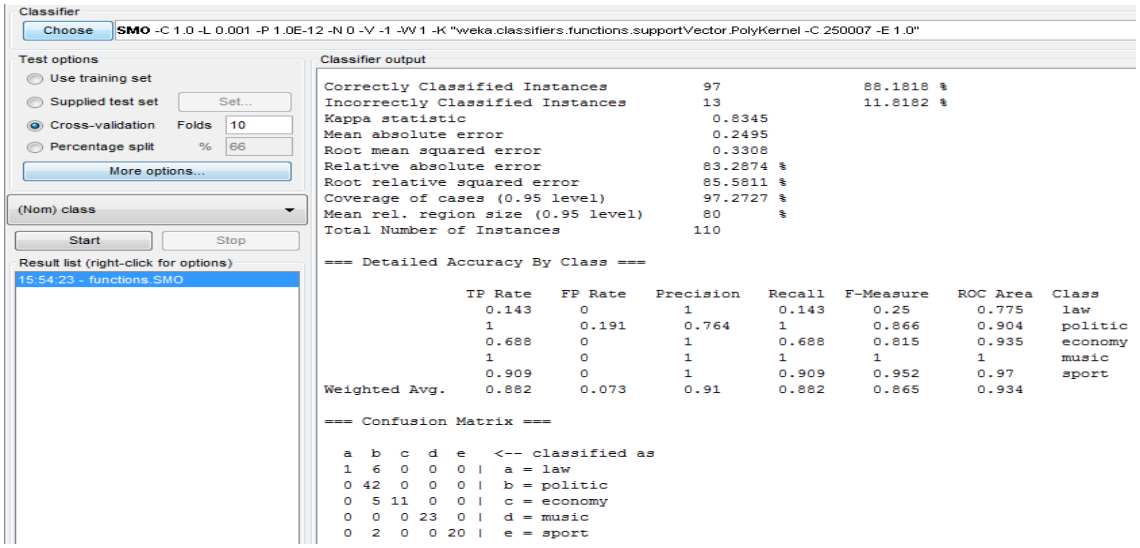

Figure IV.11 : *classification avec l'algorithme SVM*

D'après cette figure, on voit que 88,18% des documents ont été classés correctement. La matrice de confusion en bas, indique que les quelques erreurs ont concerné la classe *law*, où seul 1 document sur 7 est correctement classé, la classe *economy* où on a eu 11 documents bien classés sur 16. Et enfin, la classe *sport* avec 20 document sur 22 correctement classés.

Le tableau suivant présente les mesures d'exactitude par classe pour la méthode SVM, on les trouve dans la partie « Detailed Accuracy By Class ».

Tableau IV.8 : *Mesures d'exactitude par classe pour la méthode SVM*

Classe	TP Rate	FP Rate	Precision	Recall	F-Measure
law	0,143	0	1	0,143	0,25
politic	1	0,191	0,764	1	0,866
economy	0,688	0	1	0,688	0,815
music	1	0	1	1	1
sport	0,909	0	1	0,909	0,952

➢ Algorithme KNN (Ibk sous Weka)

La figure IV.12 représente les résultats de la classification avec l'algorithme **KNN**

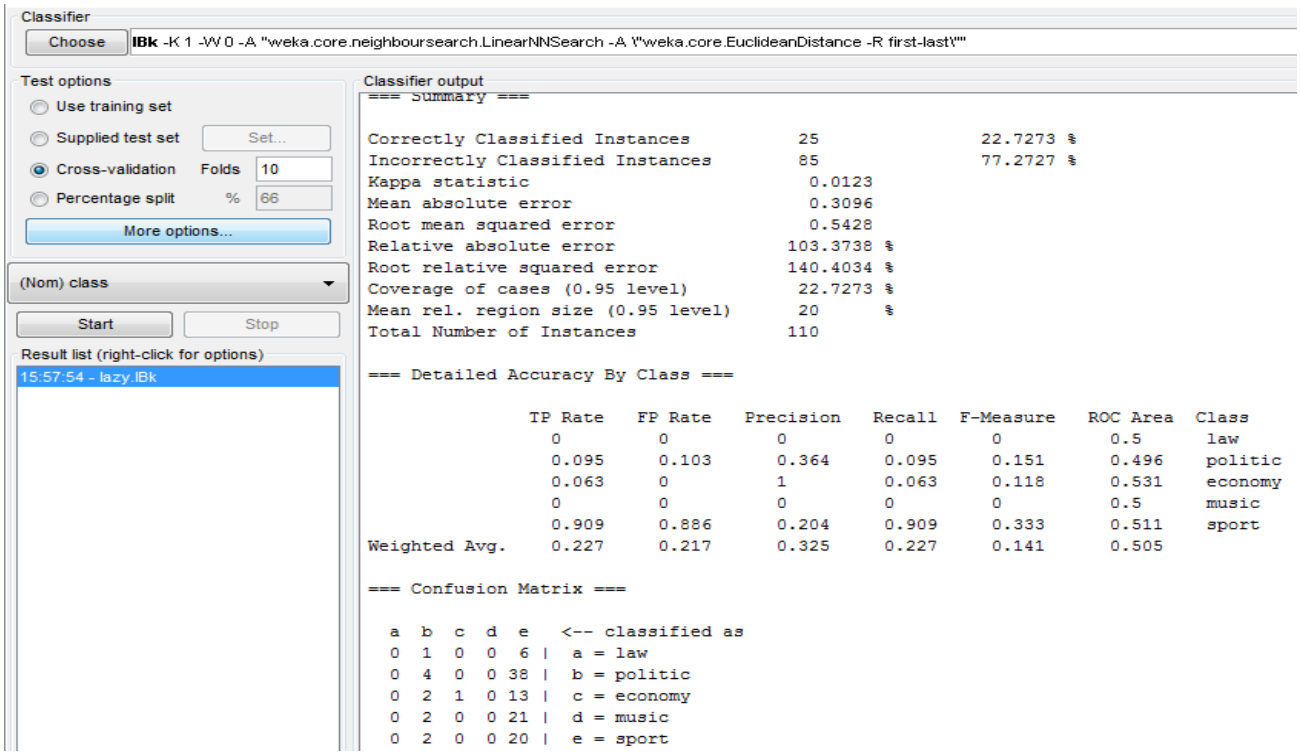

Figure IV.12 : *classification avec l'algorithme KNN*

D'après cette figure, on voit que seulement 22,72% des documents ont été classés correctement. La matrice de confusion en bas, indique que les classes ont toutes des erreurs, plus particulièrement les classes *law* et *music* qui n'ont aucun document correctement classé.

Le tableau suivant présente les mesures d'exactitude par classe pour la méthode KNN, on les trouve dans la partie « Detailed Accuracy By Class ».

Tableau IV.9 : *Mesures d'exactitude par classe pour la méthode KNN*

Classe	TP Rate	FP Rate	Precision	Recall	F-Measure
law	0	0	0	0	0
politic	0,095	0,103	0,364	0,095	0,151
economy	0,063	0	1	0,063	0,118
music	0	0	0	0	0
sport	0,909	0,886	0,204	0,909	0,333

- **Algorithme Naive Bayes**

La figure IV.13 montre les résultats issus de la classification avec l'algorithme **Naive Bayes**

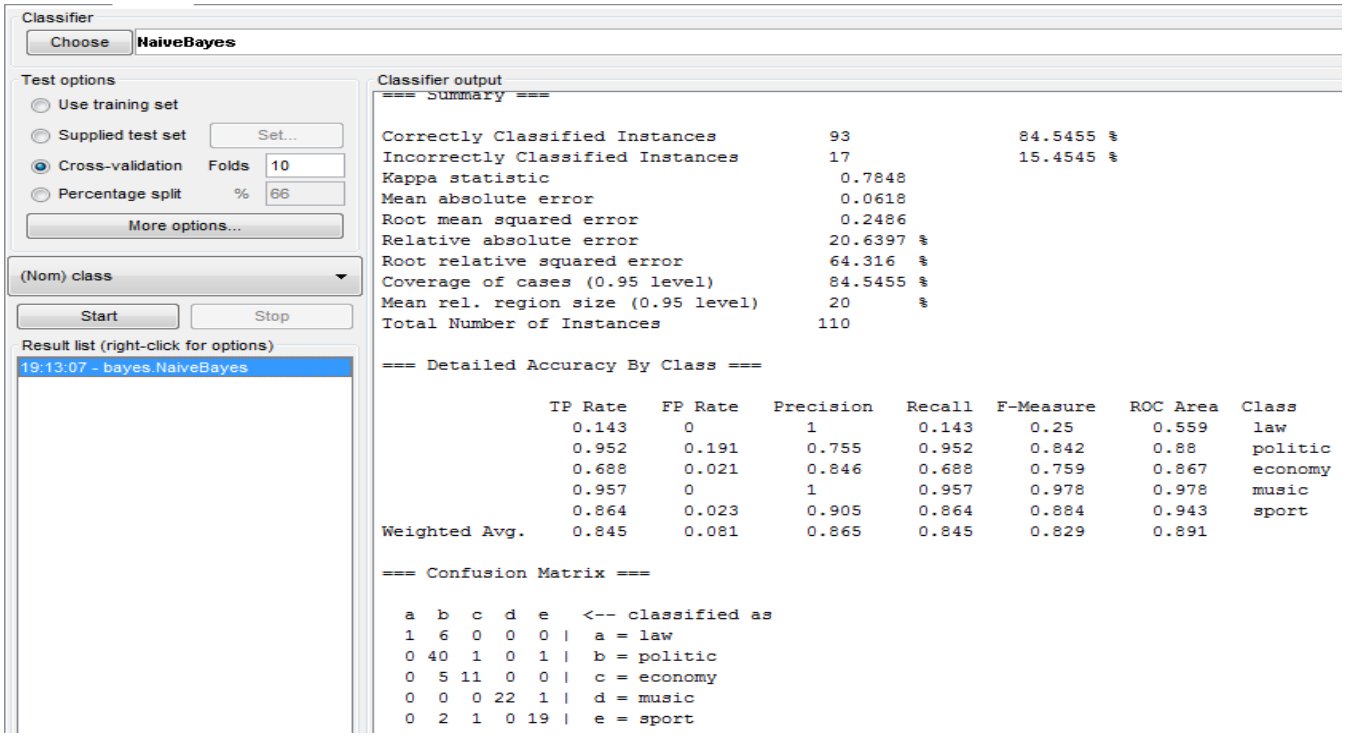

Figure IV.13 : *classification avec l'algorithme Naïve Bayes*

D'après la figure précédente, on remarque que 84,54% des documents ont été classés correctement. La matrice de confusion en bas, indique que les quelques erreurs ont concerné toutes les classes mais avec un taux faible.

Le tableau suivant présente les mesures d'exactitude par classe pour la méthode Naïve Bayes, on les trouve dans la partie « Detailed Accuracy By Class ».

Tableau IV.10 : *Mesures d'exactitude par classe pour la méthode Naïve Bayes*

Classe	TP Rate	FP Rate	Precision	Recall	F-Measure
law	0,143	0	1	0,143	0,25
politic	0,952	0,191	0,755	0,952	0,842
economy	0,688	0,021	0,846	0,688	0,759
music	0,957	0	1	0,957	0,978
sport	0,864	0,023	0,905	0,864	0,884

- **Algorithme Arbre de décision (J48 sous Weka)**

La figure IV.14 illustre les résultats de classification avec l'algorithme **Arbre de décision**

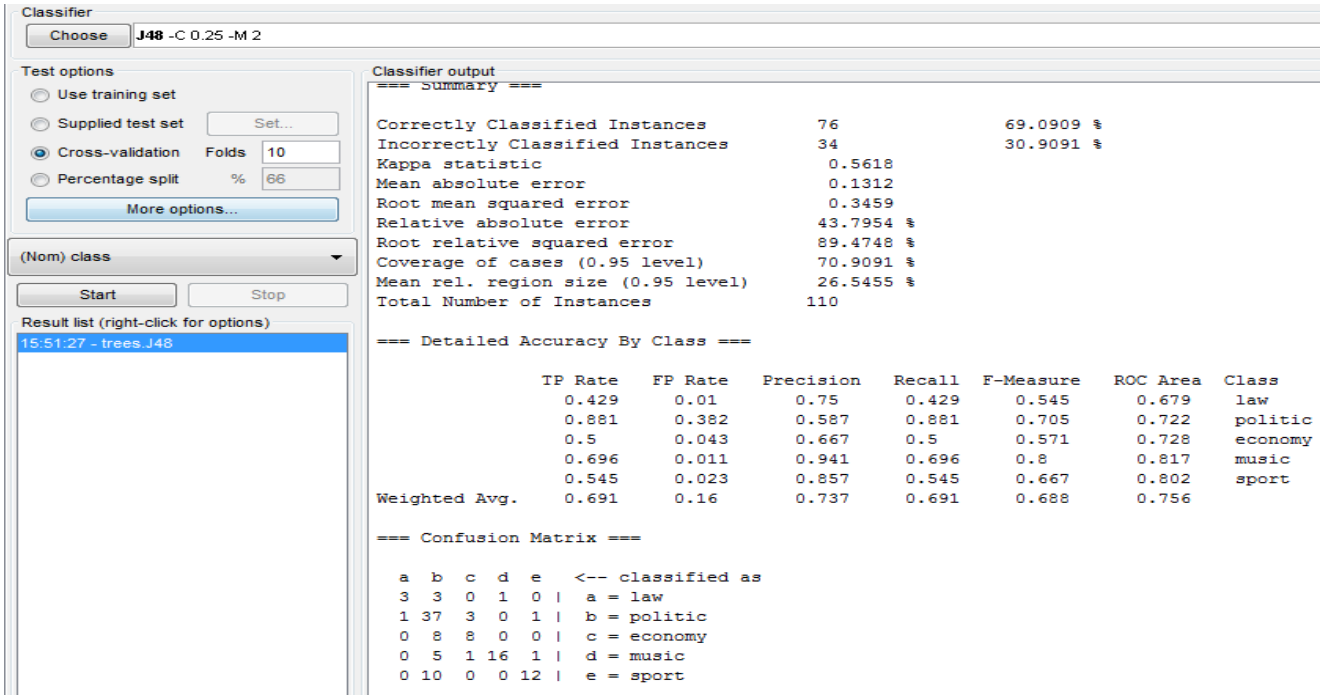

Figure IV.14 : *classification avec l'algorithme J48*

D'après cette figure, on voit que 69% des documents ont été classés correctement. La matrice de confusion, indique que les quelques erreurs ont concerné toutes les classes.

Le tableau suivant présente les mesures d'exactitude par classe pour la méthode J48, on les trouve dans la partie « Detailed Accuracy By Class ».

Tableau IV.11 : *Mesures d'exactitude par classe pour la méthode J48*

Classe	TP Rate	FP Rate	Precision	Recall	F-Measure
law	0,429	0,01	0,75	0,429	0,545
politic	0,881	0,382	0,587	0,881	0,705
economy	0,5	0,043	0,667	0,5	0,571
music	0,696	0,011	0,941	0,696	0,8
sport	0,545	0,023	0,887	0,545	0,667

Dans le tableau suivant nous allons énumérer les quatre algorithmes avec le nombre de documents correctement classifiés et le nombre de documents incorrectement classifiés pour chacun d'eux.

Tableau IV.12 : *Evaluation des performances de classification avec le corpus 2*

Algorithmes de classification	Nombre de documents correctement classifiés	Nombre de documents incorrectement classifiés
SVM	97	13
KNN	25	85
Naïve Bayes	93	17
Arbre de décision	76	34

Le tableau qui suit représente les pourcentages de documents correctement classifiés pour les différents algorithmes

Tableau IV.13 : *Pourcentage de bonne classification des différents algorithmes (corpus 2)*

Algorithmes de classification	Pourcentage de classification correcte
SVM	88,18%
Naïve Bayes	84,54%
Arbre de décision	69,09%
KNN	22,72%

Après l'évaluation du second corpus, nous constatons que les deux méthodes SVM et Naïve Bayes réalisent un taux de réussite de 88,18% et 84,54% respectivement. Quant à la méthode Arbre de décision, elle réalise un taux de 69%. Et enfin la méthode KNN réalise un taux très faible égal à 22,72%.

IV.4.2.3. Comparaison des performances des différents algorithmes

Au terme de l'expérimentation de classification des deux collections de données, nous avons pu constater que l'efficacité de la méthode SVM est supérieure à celle des autres méthodes de classification. Effectivement, nous avons enregistré un taux de documents correctement classifiés largement bon de 94% et 88% pour le premier et le second corpus respectivement.

La méthode Naïve Bayes a réalisée elle aussi des pourcentages assez grands durant la classification des deux corpus, à savoir, 78% pour le corpus Reuters et 84,54% pour le second corpus.

En ce qui concerne l'algorithme d'Arbre de décision, il réalise un taux élevé de 86% avec le corpus Reuters, et un taux moins bon mais non négligeable de 69% avec le second corpus.

La méthode KNN réalise quant à elle le taux le moins bon pour les deux collections de documents. Pour le corpus Reuters, elle enregistre un taux de 53% et pour le second corpus, un taux très faible de 22,72%.

On peut clairement constater que les meilleurs résultats sont obtenus avec le corpus 1, cela s'explique par la nature du corpus lui-même (textes journalistiques) qui est peu bruité et ou chaque document est assez riche en terme de nombre de mots. Le choix des descripteurs, à savoir, les verbes, les noms et les adjectifs jouent aussi un rôle dans les résultats de la classification (résultats satisfaisants pour les eux corpus).

Les deux corpus utilisés pour cette expérimentation ne sont pas très grands et pour des données assez petites, les exemples apprennent mal puisque le taux des instances mal classées est beaucoup plus grand tandis que pour des données de grande taille, ce taux diminue. Cette constatation conduit à conclure que plus l'exemple d'échantillon est riche, plus les apprenants apprennent mieux, ce qui influence les résultats négatifs enregistrés durant le processus de classification.

La méthode SVM reste la meilleure en termes de taux de documents correctement classés dans les deux cas traités. Elle est considérée comme la méthode la mieux adaptée à la classification automatique de textes. Le taux d'erreurs enregistré avec cette méthode peut être dû au fait que Weka ne fournit qu'un SVM linéaire ce qui peut limiter son efficacité.

La méthode KNN reste la moins bonne, avec des taux de classification assez faibles, particulièrement pour le second corpus. Cela peut être dû au paramètre k défini par défaut dans Weka (k=1).

Pour conclure, on constate que dans tous les cas, les trois algorithmes SVM, Naïve Bayes et l'Arbre de décision réalisent des performances assez impressionnantes.

IV.5. Conclusion

Plusieurs méthodes sont proposées pour le problème général de la classification. Ils différent par les mesures de proximité qu'ils utilisent, la nature des données qu'ils traitent et l'objectif final de la classification. Chacune de ces méthodes possède ses points forts et ses points faibles.

Au final, on ne peut pas affirmer qu'un tel algorithme est meilleur par rapport à un autre, le choix dépend du problème où il va être appliqué, de ses caractéristiques. Malgré cela, les méthodes SVM, Naïve Bayes et l'arbre de décision restent les meilleures dans notre cas.

Donc, le choix d'une méthode appropriée dépend fortement de l'application, la nature des données et les ressources disponibles. Une analyse attentive des données aide à bien choisir le meilleur algorithme. Il n'existe pas un algorithme qui peut répondre à toutes les demandes.

CONCLUSION GÉNÉRALE

Conclusion générale

La recherche accorde ces dernières années, beaucoup d'importance au traitement des données textuelles. Ce présent travail, intitulé « Comparaison des différentes approches de classification automatique des documents textuels » traite comme problématique la classification des documents textuels. Le système accomplit la phase de prétraitement (indexation), et puis applique les algorithmes d'apprentissage.

Nous avons présentés dans ce mémoire, les différentes approches d'analyse intelligentes de documents textuels qui utilisent les techniques d'apprentissage en s'intéressant particulièrement à la classification supervisés afin de comparer les principaux algorithmes d'apprentissage supervisé. Le but de la recherche sur la classification automatique des textes est donc de trouver un algorithme permettant d'assigner un texte à une classe avec le plus grand taux de réussite possible.

Au terme de ce travail, on a pu constater que les méthodes SVM, Naïve Bayes et Arbre de décisions enregistrent des taux de documents correctement classés extraordinaires pour les deux collections. La méthode SVM reste particulièrement la meilleure dans les deux cas. Quant à la méthode KNN, elle réalise le taux le plus faible pour les deux corpus, elle est donc considérée comme la moins bonne.

La classification supervisée est un champ de recherche fertile, qui ne dispose pas encore d'une théorie figée. Elle se nourrit donc régulièrement d'avancées dans d'autres domaines. Il n'existe pas encore de théorie unifiée permettant de définir l'algorithme optimal pour ranger les documents dans les classes suivant une problématique donné. Cependant, de nombreuses et diverses approches sont disponibles, à savoir, les algorithmes par apprentissage et le traitement automatique de la langue.

La classification supervisée de documents a fait beaucoup de progrès ces dernières années. Lorsque les documents pré-classifiés sont nombreux, les résultats sont comparables à ceux obtenus par des experts. Dans le cas d'une nouvelle application, l'étude manuelle préalable de milliers de documents coûterait trop chère. Les experts cherchent actuellement à diminuer ce coût en sélectionnant les documents qui apportent le plus d'information pour obtenir la même qualité avec beaucoup moins de documents pré-classifiés. Parallèlement, nous espérons que les progrès réalisés dans les techniques linguistiques pourront apporter un réel gain dans la qualité des classifications automatiques.

Suite à notre travail quelques perspectives sont envisagées dont, l'utilisation de corpus plus grand pour les tests, l'amélioration du processus d'indexation de documents textuels, l'application des approches de classification sémantique, qui pourraient améliorer les performances des algorithmes, et ensuite, la comparaison de celles-ci avec les approches de classification classiques. L'autre perspective envisagée est liée aux techniques de classification, en d'autres termes, nous souhaitons expérimenter d'autres types d'algorithmes de classification.

BIBLIOGRAPHIE

[1]. Salton, G., & McGill. M , Introduction to Modern Information Retrieval, McGraw-Hill, New York, 1983

[2]. Association des professionnels de l'information et de la documentation

[3]. Jean-Louis Baraggioli, B. Desnoues, Indexation Dewey et Rameau, Médiadix, 2005

[4]. Mustapha Baziz, Indexation conceptuelle guidée par ontologie pour la Recherche d'Information, Thèse de doctorat de l'Université Paul Sebatier, 2005

[5].Jacquemin.C., Daille, B., Royanté, J., and Polanco, X. 2002. In vitro evaluation of a program for machine-aided indexing. *Inf. Process. Manage.* 38, 6 (Nov. 2002), 765-792

[6].William B. Frakes, Ricardo A. Baeza-Yates, Information Retrieval: Data Structures & Algorithms, Prentice-Hall, 1992.

[7]. Fox & C.Buckley, la pondération des termes d'indexation, Institut de recherche en informatique de Toulouse, 1992

[8]. Helmut Schmid, Probabilistic Part-of-Speech Tagging Using Decision Trees, Proceedings of International Conference on New Methods in Language Processing, Sept. 1994.

[9]. Martin Porter, An algorithm for suffix stripping. Program, 14(3):130-137, July, 1980.

[10]. K. Sparck Jones, Automatic keywords classification for information retrieval. 1971.

[11].Zipf, George K, Human behavior and the principle of least effort, Cambridge, Addison-Wesley 1949.

[12]. Trésor de la Langue Française Informatisé

[13]. www.enssib.fr/le-dictionnaire/classification

[14]. D. Bensalem, C. Bounouar, Z. Boudia, Classification automatique de documents: de la classification classique à la classification utilisant une ressource externe, Département Informatique de l'UMMTO, 2014.

[15]. www.maths.cnam.fr

[16]. www.fr.wiképédia.

[17]. www.math.unice.fr

[18]. J.J. Rocchio, Jr, The SMART Retrieval System: Experiments in Automatic Document Processing, Prentice-Hall, 1971.

[20]. www.cs.waikato.ac.nz/ml/weka/

[21]. https://netbeans.org/downloads/

[22]. http://nlp.stanford.edu/software/tagger.shtml

[23]. https://rednoise.org/rita/download.php

[24]. http://www.cs.waikato.ac.nz/ml/weka/

Liste des abréviations

ADBS : Association des professionnels de l'information et de la documentation.

TALN : Traitements Automatiques de la Langue Naturelle.

PDF : Portable Document format.

HTML: Hyper Text Markup Language.

XML: eXtensible Markup Language.

TEI: Text Encoding Initiative.

POS : Part Of Speech.

IMS: Institute for Micro-electronics Stuttgart.

LIA: Laboratoire Informatique d'Avignon.

TF : Term Frequency.

IDF : Inverse of Document Frequency.

TLFI: Trésor de la Langue Française Informatisé.

FCM : Fuzzy C-means.

KNN: K-Nearest Neighbors.

SVM: Support Vector Machine.

WEKA: Waikato Environment for Knowledge Analysis.

GUI: Graphical User Interface.

SIGKDD: Data Mining and Knowledge Discovery Service Award.

BDD: Base De Données.

SQL: Structured Query Language.

CSV: Comma-Separated Values.

URL: Uniform Resource Locator.

ARFF: Attribute Relation File Format.

CLI: Command-Line Interface.

EDI: Electronic Data Interchange.

CDDL: Common Development and Distribution License.

GPLv2: General Public License v2.

PHP: Hypertext Preprocessor.

SPARC: Scalable Processor ARChitecture.

JDK: Java Development Kit.

IDE: Integrated Development Environment.

NLP: Natural Language Processing.

CGI: Carnegie Group Inc.

ANNEXE

ANNEXE

A. Présentation de la base de données Iris

Iris est une base de données très célèbre, introduite en 1936 par Ronald Aylmer Fisher comme un exemple d'analyse discriminante. Cet ensemble de données comporte 150 exemples de fleurs décrites par 5 attributs à valeur continue et appartenant à trois classes différentes (*Iris-setosa, Iris-virginica et Iris-versicolor*).

B. Fichier ARFF d'Iris

La figure suivante représente le fichier ARFF de l'ensemble de données Iris

```
@RELATION iris

@ATTRIBUTE sepallength   REAL
@ATTRIBUTE sepalwidth    REAL
@ATTRIBUTE petallength   REAL
@ATTRIBUTE petalwidth    REAL
@ATTRIBUTE class     {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
```

Figure 1 : *Fichier ARFF Iris*

C. Lecture et visualisation du fichier ARFF Iris sous Weka

La figure qui suit illustre le fichier ARFF Iris visualisé sous le logiciel Weka

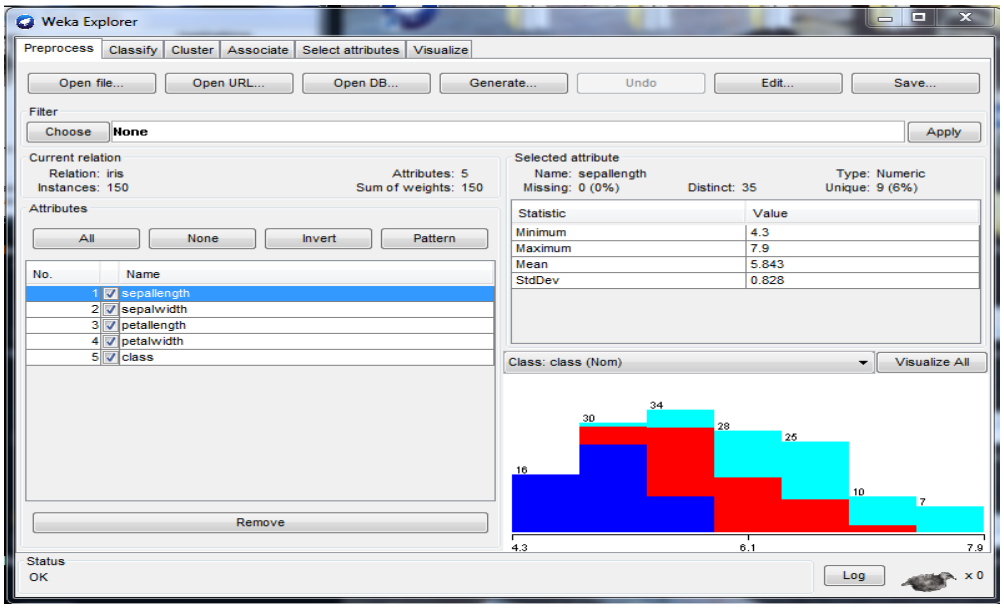

Figure 2: *fichier ARFF Iris lu sous Weka*

D. Exécution du processus d'apprentissage

Pour ce faire, l'onglet *Classify* de Weka offre les algorithmes nécessaires au processus d'apprentissage supervisé (SVM, KNN, Naive Bayes, Arbre de décision) avec le mode d'évaluation « *Cross-validation* ».

➢ Méthode SVM (SMO sous Weka)

La figure suivante illustre le résultat de la classification de l'ensemble de données Iris sous le logiciel Weka avec la méthode SVM.

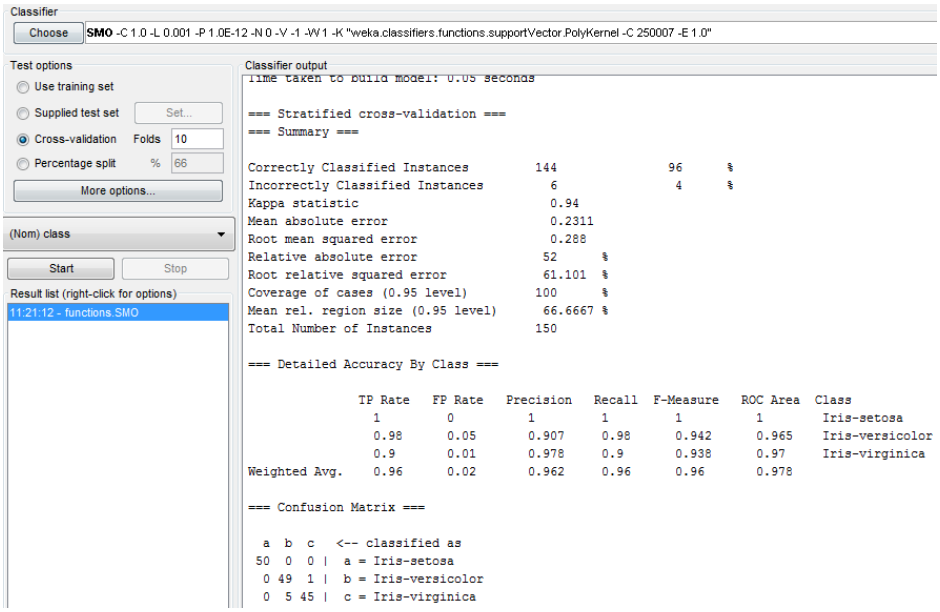

Figure 3 : *Résultat de la classification avec l'algorithme SVM*

➢ Algorithme KNN (Ibk sous Weka)

La figure suivante illustre le résultat de la classification de l'ensemble de données Iris sous le logiciel Weka avec la méthode KNN.

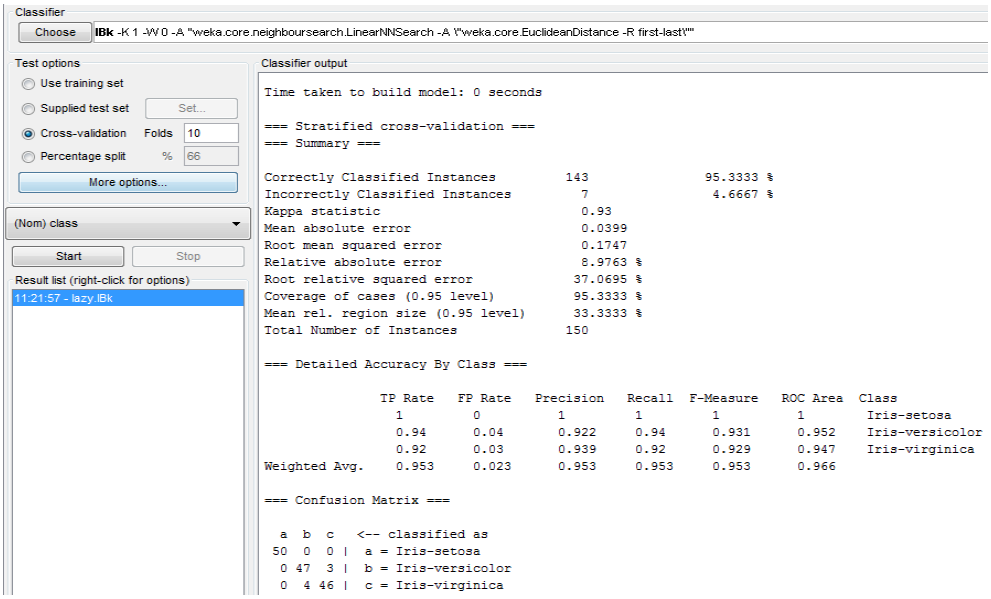

Figure 4: *Résultat de la classification avec l'algorithme KNN*

- **Algorithme Naïve Bayes**

La figure suivante illustre le résultat de la classification de l'ensemble de données Iris sous le logiciel Weka avec la méthode Naïve Bayes.

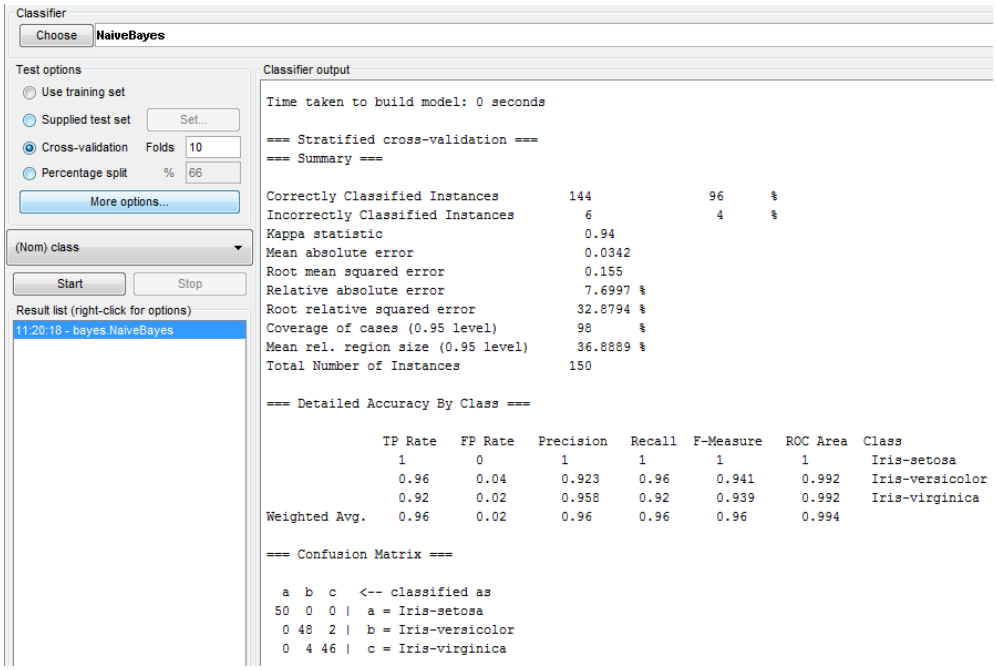

Figure 5 : *Résultat de la classification avec l'algorithme Naïve Bayes*

- **Algorithme Arbre de décision (J48 sous Weka)**

La figure suivante illustre le résultat de la classification de l'ensemble de données Iris sous le logiciel Weka avec la méthode Arbre de décision.

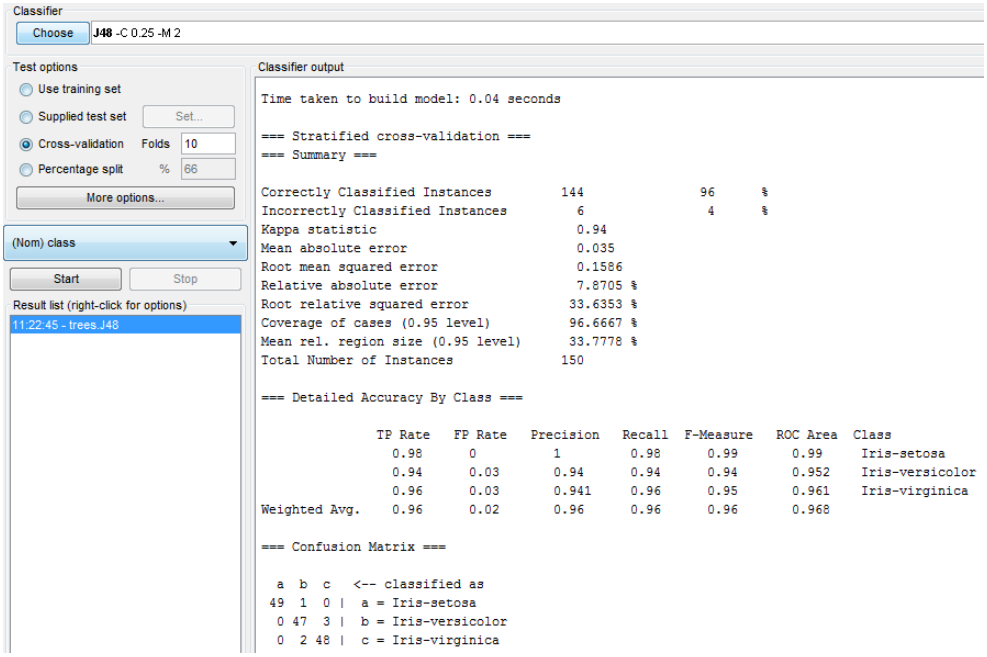

Figure 6 : *Résultat de la classification avec l'algorithme Arbre de décision*

E. Evaluation des résultats de la classification

Tableau : *Pourcentages de documents correctement classés pour les différents algorithmes de classification*

Algorithmes de classification	Pourcentage de documents correctement classés
SVM	**96%**
KNN	**95,33%**
Naïve Bayes	**96%**
Arbre de décision	**96%**

Les quatre méthodes de classification supervisée enregistrent des taux de classification correcte excellents.