RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ ABBES LAGHROUR- KHENCHELA

FACULTÉ DES SCIENCES ET DE LA TECHNOLOGIE
DÉPARTEMENT DE MATHÉMATIQUES ET D'INFORMATIQUE

POLYCOPIÉ

Introduction à la fouille de données

Cours, exercices et travaux pratiques

Réalisé par:

Dr. LEDMI Makhlouf

Année universitaire

2020/2021

POLYCOPIÉ

Sommaire

Liste des figures

Liste des tables

Liste des Algorithmes

Introduction

Ce polycopié de cours d'introduction à a fouille de données s'adresse aux étudiants de deuxième année master Génie logiciel et Systèmes distribués (GLSD) offert au département de Mathématiques et d'Informatique, de la faculté des sciences et de la technologie à l'université Abbès Laghrour de Khenchela.

La matière de fouille de données est programmée dans l'unité d'enseignement méthodologique UEM1 ayant 6 crédits et un coefficient égae à 3. Elle est programmée en 03 séances par semaine: un cours magistral, une séance de travaux dirigés (TD) et une séance de travaux pratiques (TP).

Le mot anglais "Data Mining" se traduit, en français, par plusieurs expressions comme la fouille de données ou l'exploration de données. Ou plus encore l'extraction des connaissances à partir de données, qui donne clairement le but de cette discipline, à savoir ,l'acquisition des connaissances à partir de grande volume de données.

L'objectif principal du cours est de présenter les différentes tâches de fouille de données en se concentrant sur l'aspect algorithmique et logiciel de la fouille de données, l'étude des notions d'apprentissage supervisé et non supervisé, et les algorithmes afférant et leur utilisation.

Le polycopié est organisé en cinq chapitres:

- Le premier chapitre intitulé "Extraction des connaissances et fouille de données" présente une définition de la fouille de données, il énumère les différent étapes du processus global d'extraction de connaissances à partir de données, il met l'accent sur les tâches principales de la fouille de données. Enfin, il positionne la fouille de données par rapport aux autres techniques ou disciplines dont la fouille de données fait appel pour trouver des modèles intéressants afin expliciter des connaissances cachées.
- Dans le deuxième chapitre, on présente une des tâches de la fouille de données: "la classification". On expose les différentes natures des attributs qu'on peut rencontrer dans les données, on donne une définition formelle de la problématique de la classification supervisée. Ensuite, on présente un algorithme pour la construction d'un classifieur basé sur le modèle des arbres de décision. Avant de terminer avec quelques exercices corrigés et proposés, on présente les techniques utilisées pour évaluer les performances des modèles de classification.
- Dans le troisième chapitre, on présente la tâche de " fouille de règles d'association". On définit les différents concepts et notions liés à cette tâche comme: une règle

d'association, un ensemble d'items fréquents, le support et la confiance. Ensuite, on présente l'algorithme Apriori utilisé pour l'extraction des ensembles d'items fréquents fréquents et la génération des règles d'association. Enfin, on termine par donner quelques exercices corrigés et proposés.

- Dans le quatrième chapitre, on présente une autre tâche de la fouille de données: "la segmentation " ou le clustring. On expose la problématique de la classification non supervisée et les différentes mesures de distance utilisées. Ensuite, on présente un algorithme populaire "k-means" utilisé pour regrouper les données. Enfin, on termine par donner quelques exercices corrigés et proposés.
- Le dernier chapitre intitulé "R et fouille de données" présente une étude en logiciel R des différentes tâches vues précédemment. Il commence par présenter la syntaxe, la manipulation des données, les structures de contrôle et les fonctions dans R. Ensuite, il présente des travaux dirigés détaillés pour chacune des trois tâches: classification, règles d'association et clustering.

Chapitre 1 Extraction des connaissances et fouille de données

La révolution numérique a rendu l'information facile à être capturer, traiter, stocker, distribuer et transmettre. Avec le progrès important et l'utilisation en pleine expansion des technologies informatiques dans les différents domaines de vie, de grandes quantités de données diverses continueront d'être collectées et stockées dans les bases de données. Si la quantité d'informations double tous les mois, la taille et le nombre de bases de données augmente probablement à un rythme similaire.

En effet, l'extraction des connaissances à partir de ce grand volume de données est un défi. Plus on a de données, plus il est difficile d'en tirer de la connaissance. La fouille de données, souvent appelée "data mining", est une tentative d'explorer et d'analyser cet énorme volume de données afin d'y découvrir des informations implicites. Ces informations peuvent être de différente nature, par exemple on recherchera des règles d'association, une classification ou une segmentation de population.

Aujourd'hui, la gestion des bases de données avancées a permis d'intégrer des différents types de données, tels que l'image, vidéo, texte, dans une base de données unique afin de faciliter le traitement multimédia. Ceci nécessite de développer des techniques de fouille de données plus avancées pour s'adapter à ce nouveau type de données.

Ce chapitre a pour but de présenter, en premier temps, une introduction à la fouille de données (data mining), ses tâches et quelques techniques utilisées, et citer quelques domaines d'applications du data mining.

1.1 Extraction des connaissances à partir de données (ECD)

La définition de l'extraction des connaissances à partir de données (**ECD**) a été enrichie au cours du temps et celle proposés par (Fayyad *et al*, 1996) est maintenant consensuelle:

Définition 1.1. Extraction des connaissances à partir des données

L'extraction des connaissances à partir des données est un processus non trivial d'identification des modèles valides, nouveaux, potentiellement utiles et au final compréhensibles, à partir de données. ♣

- **Nouveau**: Ce qui est recherché est non prévisible, ce n'est pas le bon sens ou de faits connus.
- **Valide**: La sélection ou la représentation de données inappropriées mèneront à des résultats incorrects. L'information extraite doit être soigneusement vérifiée par des experts du domaine et correcte dans le futur.
- **Utile**: L'information extraite doit pouvoir être utilisé dans un certain domaine de problème pour prendre des décisions.
- **Compréhensible**: L'information extraite doit être significative et facile à comprendre.

1.2 Processus d'extraction des connaissances à partir des données

Le processus global consiste à transformer les données de bas niveau en connaissances de haut niveau. Le processus ECD est décrit dans la figure 1.1. Il est interactif et itératif impliquant, plus ou moins, les étapes suivantes:

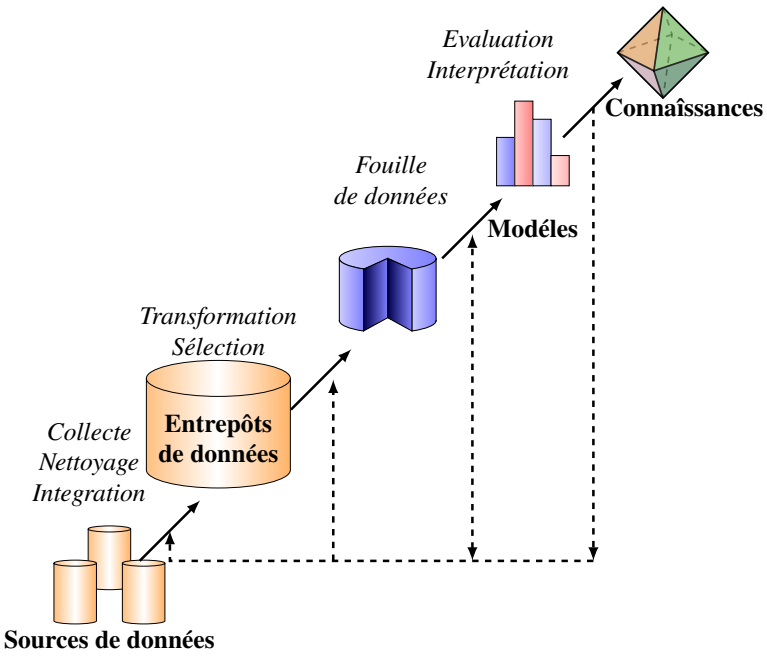


Figure 1.1: Schéma globale de l'ECD.

1. **La compréhension du domaine d'application**: Cela inclut la connaissance a priori, de l'application et des objectifs à atteindre.
2. **Extraction des données cibles**: C'est la sélection d'un ensemble de données ou de se concentrer sur un sous-ensemble de variables, en utilisant le classement et la sélection attributs (motifs).
3. **Prétraitements des données**: Ceci est nécessaire pour améliorer la qualité des données effectives pour la fouille. Ceci augmente également l'efficacité de l'extraction, en réduisant le temps nécessaire pour l'extraction des données prétraitées. Le prétraitement des données implique le nettoyage des données, la transformation de données, l'intégration de données, la réduction de données ou la compression de données, . . . etc.
   (a). *Nettoyage*: Elle se compose de quelques opérations de base, telles que la normalisation, la suppression du bruit et la manipulation des données manquantes, la réduction de la redondance, . . . etc. Les données obtenues à partir des sources du monde réel sont souvent erronées, incomplètes, et incompatibles, peut-être dû à une erreur opérationnelle, ou des failles d'implémentation du système. Ces données de faible qualité doivent être nettoyées avant la fouille de données.
   (b). *Intégration*: L'intégration joue un rôle important dans l'ECD. Cette opération comprend l'intégration de multiples ensembles de données hétérogènes, générées à partir de différents sources de données.
   (c). *La réduction et de la projection des données*: Cela inclue de trouver caractéristiques (motifs) utiles pour représenter les données (en fonction de l'objectif de la tâche à effectuer) en utilisant les méthodes de réduction de la dimensionnalité, de discrétisation et l'extraction de caractéristiques.
4. **Fouille de données**: Cela correspond à l'une ou plusieurs des tâches , à savoir, la classification, régression, le clustering, la découverte de règles d'association . . . etc.
5. **Interprétation**: Cela inclut l'interprétation des modèles découverts, ainsi que la visualisation possible (faible dimension) des modèles extraits. La visualisation est un outil important qui augmente compréhensible du point de vue de l'homme. On peut évaluer les modèles extraits d'une façon automatique ou semi-automatique pour identifier les modèles vraiment intéressants ou utiles pour l'utilisateur.
6. **Utilisation des connaissances découvertes**: elle comprend l'intégration de ces connaissances dans des systèmes performants et de les mettre à la disposition des décideurs.

1.3 Les tâches de fouille de données

Dans cette section, nous présentons quelques tâches typiques et populaires de la fouille de données à savoir la classification, la segmentation, l'association. En général, ces tâches peuvent être classées en deux catégories: descriptives et prédictives. Les tâches descriptives caractérisent les propriétés des données contenues dans un ensemble de données de cibles. Les tâches prédictives effectuent induction sur les données actuelles afin de faire des prédictions.

1.3.1 La classification

La classification (appelée aussi apprentissage supervisé) est le processus de recherche d'un modèle (ou une fonction) qui décrit et distingue des classes de données ou des concepts. Le modèle est établi en se basant sur l'analyse d'un ensemble de données d'apprentissage (ensemble de données ou objets pour lesquels les classes sont prédéfinies), il est utilisé pour prédire l'étiquette de la classe d'objets pour lesquels la classe est inconnue. Il est représenté sous différentes formes: règles de classification, arbres de décision ou réseaux de neurones (figure 1.2).

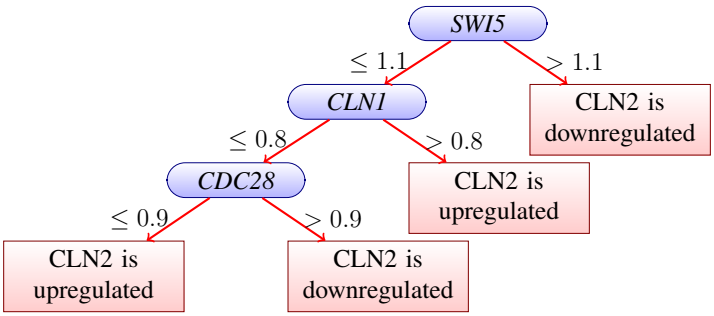
- Un arbre de décision est un organigramme comme la structure de l'arbre où chaque nœud représente un test sur une valeur d'attribut, chaque branche représente un résultat de test, et les feuilles représentent des classes ou des distributions de classe.
- Un réseau de neurones, lorsqu'ils sont utilisés pour la classification, est généralement une collection de neurones (des unités de traitement avec des connexions pondérées entre les unités).
- Les modèles probabilistes ou génératifs, qui calculent les probabilités pour des hypothèses basées sur le théorème de Bayes.
- Les classifieurs plus proches voisins, qui calculent la distance minimale à partir d'instances ou de prototypes.

Parmi les exemples d'application de la classification, on cite:

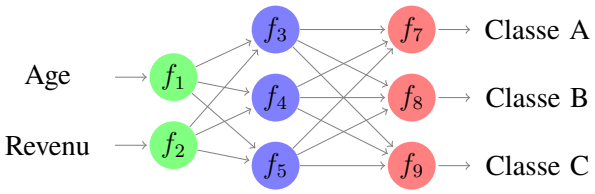
- Identification de signature dans le secteur bancaire ou dans le traitement des documents sensibles (correspondance, aucune correspondance).
- Identification d'empreinte digitale numérique dans des applications de sécurité (correspondance, aucune correspondance).
- Attribuer un crédit bancaire considérant de la qualité de la clientèle, et les possibilités financières (bon, moyen, mauvais).

Age(X,'*Jeune*') ET **Revenu**(X,'*Elevé*') → **Classe**(X,'*A*')
Age(X,'*Jeune*') ET **Revenu**(X,'*Bas*') → **Classe**(X,'*B*')
Age(X,'*Senior*') → **Classe**(X,'*C*')

(a) Règles de classification



(b) Arbres de décision.



(c) Réseaux de Neurones

Figure 1.2: Quelques modèles de classification

- L'efficacité du traitement d'un médicament en présence d'un ensemble de maladies symptômes (bon, moyen, mauvais).
- Détection de cellules suspectes dans une image numérique d'échantillons de sang (oui, non).

1.3.2 La segmentation (Clustering)

La segmentation se rapporte à la catégorisation d'un ensemble d'objets de données dans des clusters. Elle est aussi appelée classification non supervisée. Un cluster est une collection d'objets de données similaires les uns aux autres dans le même segment, mais différents des objets dans d'autres segments.

Une bonne méthode de segmentation produira des catégories avec une similarité intra-classe importante et une similarité inter-classe faible. La qualité d'un clustering dépend à la fois de la mesure de similarité utilisée par le la méthode et sa mise en œuvre. Les approches de clustering peuvent être classées comme suit:

- Méthode de partitionnement: Créer un partitionnement initial et ensuite utiliser une stratégie de contrôle itérative pour l'optimiser.
- Méthodes hiérarchiques: construire une hiérarchie de clusters (appelé dendrogramme), non seulement un partitionnement unique des objets en utilisant une condition de terminaison. (ex. Nombre de clusters).

- Méthodes basées sur la densité: utiliser les fonctions de densité de voisinage.

Certaines applications générales de regroupement incluent:

- La reconnaissance de formes et le traitement d'images.
- Analyse des données spatiales: créer des cartes thématiques dans les systèmes d'information géographique (SIG).
- Analyse médicale: la détection des croissances anormales de l'IRM.
- Bioinformatique: la détermination des groupes de signatures à partir d'une base de données de gènes.
- Web: clustering des fichiers log pour découvrir des modèles d'accès similaires.

1.3.3 L'association

La fouille de règles d'association se rapporte à la découverte des relations entre les attributs d'un ensemble de données appelé souvent ensemble des transactions. Une transaction est l'ensemble des articles achetés ensemble par les clients. Une règle est normalement exprimée sous la forme $A \Rightarrow B$, où A et B sont des ensembles d'attributs de l'ensemble de données. Cela implique que les transactions qui contiennent A contiennent B avec une grande probabilité. La règle peut s'écrire sous une autre forme:

SI <*certaines conditions satisfaites* > ALORS <*prédire les valeurs pour certains autres attributs*>,

ainsi l'association $X \Rightarrow B$ est exprimé en SI A ALORS B.

Une règle d'association $A \Rightarrow B$ peut être identifié lorsque le support et la confiance de la règle sont largement supérieurs aux seuils respectifs. Le support de la règle d'association est le rapport entre le nombre de transactions contenant à la fois A et B sur le nombre total de transactions dans la base de données. La confiance de la règle d'association est la proportion du nombre de transactions contenant à la fois A et B sur le nombre total de transactions contenant A seulement.

Par exemple, la règle:

$$\text{Age}(X, 20..29) \wedge \text{ revenu}(X, 40000..49000) \Rightarrow \text{ achète}(X, \text{Ordinatur portable})$$
$$(\text{support } 2\% \text{ , confiance} 60\%)$$

signifie que 2% des clients sont âgés de 20 à 29 ans ayant un revenu compris entre 40.000 et 49.000 et ont achetés un ordinateur portable. Il y a une probabilité de '60% qu'un client dans cet intervalle d'âge et de revenu va acheter un ordinateur portable.

De plus, les règles d'association peuvent être utilisées pour extraire des règles de classification, l'objectif est de prévoir un attribut de classe qui ne peut jamais apparaître

dans la partie antécédente d'une règle. Les règles générées sont utilisées pour prédire l'attribut de classe d'un ensemble de test. Elles peuvent être aussi utilisées pour modéliser les dépendances entre, les attributs objectifs étant choisi parmi un ensemble prédéfini d'attributs peuvent apparaître dans les deux parties d'une règle, tandis que les attributs non-objectifs peuvent se produire que dans la partie antécédente.

1.4 Les techniques de fouille de données

La fouille de données fait appel aux différentes techniques permettant de trouver des modèles intéressants qui aident à expliciter des connaissances cachées dans les données. Parmi les problèmes fondamentaux liés à la découverte de connaissances:

- Comment représenter les connaissances?
- Quels attributs à sélectionner ou à choisir?
- Comment faire face aux données manquantes, bruitées et rares?
- Comment visualier et interpréter les modèles découverts "intéressants" et "utiles".

Dans cette section, nous donnons des exemples de certaines disciplines qui influent fortement sur le développement de méthodes de a fouille de données. Nous allons nous concentrer sur les bases de données, les statistiques, l'apprentissage automatique et la recherche d?information. Il s'en suit que, d'un point de vue technique, la fouille de données peut être vue comme se trouvant à l'intersection de ses sciences parentes, comme illustré par la figure 1.3.

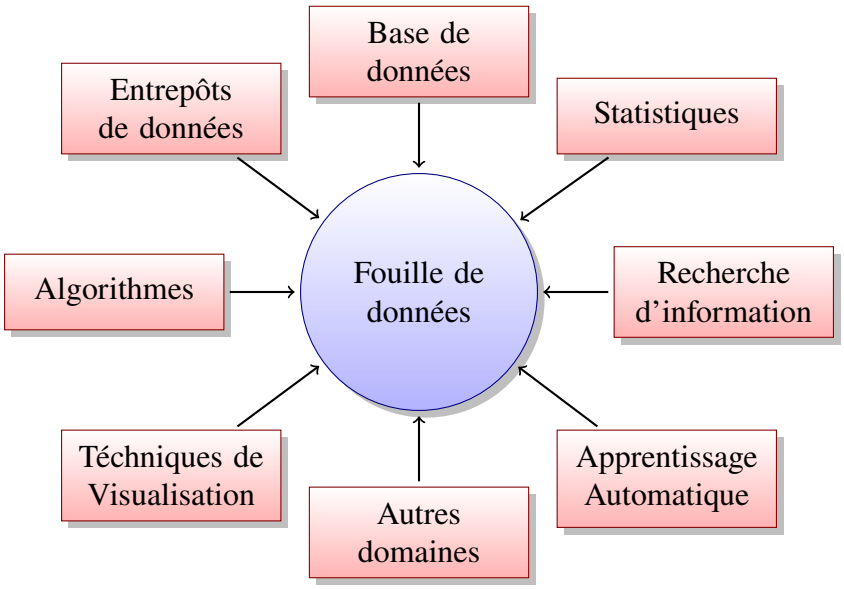


Figure 1.3: Techniques de la fouille de données

1.4.1 Statistiques:

Bien que les méthodes statistiques soient largement utilisées dans la fouille de données, elles ont deux philosophies différentes. Dans les statistiques, les hypothèses sont généralement faites d'abord et elles sont avérées valides ou non en appliquant des théories sur les données. Au contraire, la fouille de données ne sait pas exactement ce qu'elle cherche. De plus, l'application de méthodes statistiques dans la fouille de données est loin d'être trivial. Souvent, un sérieux défi est de savoir comment appliquer à grande échelle une méthode statistique, ayant souvent une grande complexité de calcul, sur un grand ensemble de données qui peut être également distribué sur des sites logiques ou physiques multiples. Ce défi devient encore plus difficile pour les applications en ligne, comme les suggestions de requête en ligne dans les moteurs de recherche, où la fouille de données est nécessaire pour gérer en continu, les flux de données en temps réel.

1.4.2 Machine Learning:

L'apprentissage automatique est une technique très utilisée dans l'intelligence artificielle. Tout d'abord, un modèle approprié est construit en définissant certains paramètres et une mesure d'erreur. Ensuite, une phase d'apprentissage, dont le but est d'adapter un ensemble de données à ce modèle, est utilisée pour ajuster les paramètres en fonction de la mesure prédéfinie. Enfin, le modèle peut être utilisé pour prédire ou classifier des nouveaux échantillons de données. Il y a beaucoup de similitudes entre la fouille de données et l'apprentissage automatique. Pour les tâches de classification et de clustering, l'apprentissage automatique se concentre souvent sur la précision du modèle. En plus de la précision, les recherches en fouille de données mettent fortement l'accent sur l'efficacité et l'extensibilité de méthodes de fouille de données sur de grands ensembles de données, ainsi que sur les façons de gérer les types de données complexes et d'explorer de nouvelles méthodes alternatives.

1.4.3 Bases de données et entrepôts de données:

Les recherches sur les systèmes de base de données se concentrent sur la création, la maintenance et l'utilisation des bases de données pour les organisations et les utilisateurs finaux. En particulier, ces recherches ont établi des technologies très reconnus dans les modèles de données, langages de requêtes, les méthodes de traitement des requêtes et d'optimisation, de stockage de données, et l'indexation et modalités d'accès. Bien que

la fouille de données et les bases de données aient des objectifs différents, la fouille de données peut se bénéficier du développement des techniques des bases de données dans le stockage et le prétraitement des collectes de données. De plus, les tâches de la fouille de données peuvent être utilisées pour étendre les capacités des systèmes de base de données existants à satisfaire les exigences de leurs utilisateurs en matière d'analyse sophistiquée de données. Un entrepôt de données intègre des données provenant de sources multiples et différents délais. Il est généralement modélisée par une structure de données multidimensionnelle, appelé un cube de données, dans lequel chaque dimension correspond à un attribut ou un ensemble d'attributs dans le schéma. Un cube de données fournit une vue multidimensionnelle des données qui permet le calcul préalable et l'accès rapide des données.

1.4.4 La recherche d'information

La recherche d'information (IR) est la science de la recherche de documents ou d'informations dans les documents. Les documents peuvent être textuels ou multimédias, et peuvent se trouver sur localement ou sur le Web. A l'égard des systèmes de gestion des bases de données, les systèmes traditionnels de recherche d'information supposent que les données sous la recherche ne sont pas structurées, et les requêtes sont formées principalement par mots-clés et n'ont pas de structures complexes (à la différence des requêtes SQL dans la base de données). De plus en plus de grandes quantités de données textuelles et multimédias ont été recueillies et deviennent disponibles en ligne en raison du développement rapide du web, de l'Internet, et des applications telles que les bibliothèques numériques, les gouvernements numériques et des systèmes d'information de santé. Une recherche et une analyse efficace ont soulevé de nombreux enjeux dans la fouille de données. Par conséquent, la fouille de texte et la fouille de données multimédia, intégrées avec les méthodes de recherche d'information, sont devenues de plus en plus importantes.

Chapitre 2 Classification

Tout au long de ce chapitre, nous considérons les notations suivantes:

- On notera $\mathcal{X}$ un ensemble de données.
- Chaque donnée est décrite par un ensemble $\mathcal{A}$ d'attributs.
- Chaque attribut $a \in \mathcal{A}$ prend sa valeur dans un certain ensemble de valeurs $\mathcal{V}_a$
- Ainsi, on peut considérer l'ensemble des données x dont les coordonnées balayent toutes les valeurs possibles des attributs: c'est l'espace des données que nous noterons $\mathcal{D}$.
- Si l'on note $a_1, a_2 \ldots a_P$ les P attributs,$\mathcal{D} = V_{a1} \times V_{a2} \times \ldots \times V_{aP}$.
- Toute donnée appartient à cet ensemble et on a $\mathcal{X} \subset \mathcal{D}$.

2.1 Natures d'attributs

2.1.1 Donnée

Une donnée est un:

- **Enregistrement** au sens des bases de données,
- **Individu** (terminologie issue des statistiques).
- **Instance** (terminologie orientée objet en informatique) .
- **Tuple** (terminologie base de données)

Note *Une données est caractérisée par un ensemble de champs, de caractères, ou encore d'attributs.*

2.1.2 Attribut qualitatif *vs*. quantitatif

Un attribut peut être de nature *qualitative* ou *quantitative* en fonction de l'ensemble des valeurs qu'il peut prendre:

- **Attribut qualitatif**: Sa valeur est d'un type défini en extension (une couleur, une marque de voiture, . . . etc.), et on ne peut pas en faire une moyenne.
- **Attribut quantitatif**: L'attribut est de nature quantitative: un entier, un réel, . . . etc. Il peut représenter un salaire, une surface, un nombre d'habitants, . . . etc. On peut appliquer les opérateurs arithmétiques habituels.

2.1.3 Valeur d'un attribut

La valeur d'un attribut est censée représenter une certaine mesure d'une quantité. Selon cette valeur, on distingue plusieurs types:

Définition 2.1. Attribut nominal

Les valeurs sont des symboles (des noms), aucune relation (ordre ou distance) entre les nominaux n'existe, et seulement des tests d'égalité peuvent être exécutés. ♣

Exemple 2.1

- Les valeurs de Temps (Ensoleillé, Pluvieux, Neigeux, Gris)
- La couleur (Vert, Bleu, Rouge)
- Si $Temps = Pluvieux$ alors $Jouer = Non$

Définition 2.2. Attribut ordinal

Une notion d'ordre s'impose sur les ordinaux, mais il n'est pas possible de calculer directement des distances entre des valeurs ordinales. Les opérations d'addition et de soustraction ne sont pas possibles. ♣

Exemple 2.2

- La température est décrite par les adjectifs chaud, froid, moyen, et chaud > moyen > froid
- Si $temprature > froid$ alors $Jouer = Oui$

Définition 2.3. Attribut de type intervalle

Les intervalles impliquent une notion d'ordre, et les valeurs sont mesurées dans des unités spécifiques et fixées. La somme, la différence et le produit de deux intervalles ne sont pas possibles. ♣

Exemple 2.3

- La température exprimée en degrés *Celsius* ou *Fahrenheit*
- L'attribut année: (2000,2010,2014).

Définition 2.4. Attribut de type rapport (ratio)

outes les opérations mathématiques sont autorisées sur les attributs de ce type. ♣

Exemple 2.4

L'attribut distance:

- On peut comparer deux distances.
- On peut additionner deux distances.
- La distance entre un objet et lui-même est nulle.

2.2 Classification supervisée

On dispose d'un ensemble $\mathcal{X}$ de N exemples, i.e. des couples (donnée, étiquette). Chaque donnée $x_i \in \mathcal{D}$ est caractérisée par P attributs et par sa classe $y_i \in \mathcal{Y}$. Dans un problème de classification supervisée, la classe prend sa valeur parmi un ensemble $\mathcal{Y}$ fini. Le problème consiste alors, en s'appuyant sur l'ensemble d'exemples $X = (x_i, y_i) \quad i \in \{1 \dots N\}$, à prédire la classe de toute nouvelle donnée $x \in \mathcal{D}$.

Définition 2.5. Classifieur

Un classifieur est une procédure (un algorithme) qui, à partir d'un ensemble d'exemples, produit une prédiction de la classe de toute donnée. ♣

Remarque

- D'une manière générale, un classifieur procède par *induction*: à partir d'exemples (donc de cas particuliers), on construit une connaissance plus générale.
- Parmi les modèles, on distingue ceux qui peuvent être interprétés par un humain (arbre de décision) et ceux qui ne le peuvent pas (réseau de neurones).

Note

- *On parle de classification **binaire** quand le nombre de classes y_i est deux .*
- *Il peut naturellement être quelconque. Dans tous les cas, il s'agît d'un attribut qualitatif pouvant prendre un nombre fini de valeurs.*
- *Dans l'absolu, une donnée peut appartenir à plusieurs classes: c'est alors un problème **multi-classes**.*
- *Ici, on considère que chaque donnée appartient à une et une seule classe.*
- *On utilise le mot étiquette comme synonyme de classe.*

2.2.1 Arbres de décision

Soit un ensemble $\mathcal{X}$ de N exemples notés x_i dont les P attributs sont quantitatifs ou qualitatifs. Chaque exemple x est étiqueté, c'est-à-dire qu'il lui est associée une *classe* ou un *attribut cible* que l'on note $y \in \mathcal{Y}$.

Définition 2.6. Arbre de décision

*A partir de ces exemples, on construit un **arbre** dit **de décision** tel que:*

- *Chaque noeud correspond à un test sur la valeur d'un ou plusieurs attributs;*
- *Chaque branche partant d'un nœud correspond à une ou plusieurs valeurs de ce test;*
- *A chaque feuille est associée une valeur de l'attribut cible.* ♣

Pour construire un arbre de décision, plusieurs algorithmes ont été proposés, notamment:

- **CART** proposé par Breiman dans les années 1980 .
- **ID3** de Quinlan proposé en 1986 qui a été raffiné par la suite (**C4.5** puis **C5**) .

L'algorithme **ID3** prend en compte que des attributs nominaux. Son successeur, **C4.5**, prend également en charge des attributs quantitatifs.

2.3 Algorithme ID3

2.3.1 Principe de l'algorithme ID3

Dans l'algorithme **ID3** (Algorithme 1), les tests placés dans un noeud concernent exclusivement le test de la valeur d'un et seul attribut. **ID3** fonctionne récursivement:

- Il détermine un attribut à placer en racine de l'arbre.
- Cette racine possède autant de branches que cet attribut prend de valeurs.
- A chaque branche est associé un ensemble d'exemples dont l'attribut prend la valeur qui étiquette cette branche;
- On accroche alors au bout de cette branche l'arbre de décision construit sur ce sous-ensemble des exemples et en considérant tous les attributs excepté celui qui vient d'être mis à la racine.
- Par cette procédure, l'ensemble des exemples ainsi que l'ensemble des attributs diminuent petit à petit au long de la descente dans l'arbre.

Ayant l'idée de l'algorithme, il reste à résoudre une question centrale: quel attribut placer en racine?. Une fois cette question résolue, on itérera le raisonnement pour les sous-arbres.

Pour répondre à cette question, on tente de réduire l'hétérogénéité à chaque noeud:

- Les données qui atteignent un certain noeud de l'arbre de décision doivent être plus homogènes que les données atteignant un noeud ancêtre.
- Pour cela, on a besoin de pouvoir mesurer l'homogénéïté d'un ensemble de données.
- En physique, on mesure l'homogénéité par l'**entropie**.

2.3.2 L'entropie

Soit un ensemble $\mathcal{X}$ d'exemples dont une proportion p_+ sont positifs et une proportion p_- sont négatifs. (Bien entendu, $p_+ + p_- = 1$.)

Définition 2.7. Entropie

L'entropie de ' ensemble $\mathcal{X}$ est définie comme suit:

$$H(\mathcal{X}) = -p_+ log_2(p_+) - p_- log_2(p_-) \tag{2.1}$$

Remarque

1. $0 \leq H(\mathcal{X}) \leq 1$;
2. Si $p_+ = 0$ ou $p_- = 0$, alors $H(\mathcal{X}) = 0$: ainsi, si tous exemples sont soit tous positifs, soit tous négatifs, l'entropie est nulle;
3. Si $p_+ = p_- = 0.5$, alors $H(\mathcal{X}) = 1$: ainsi, s'il y a autant de positifs que de négatifs, l'entropie est maximale.

La définition précédente de l'entropie se généralise aisément à un attribut pouvant prendre plus de deux valeurs distinctes: Pour une classe prenant n valeurs distinctes (numérotées de 1 à n) , notons p_i la proportion d'exemples dont la valeur de cet attribut est i dans l'ensemble d'exemples considéré $\mathcal{X}$. L'entropie de l'ensemble d'exemples $\mathcal{X}$ est:

$$H(\mathcal{X}) = -\sum_{i=1}^{n} p_i log_2(p_i) \tag{2.2}$$

Avoir des sous-sensembles dont l'entropie est minimale est intéressant: cela signifie que l'attribut placer à la racine discrimine les exemples en fonction de leur classe. De pus, il est naturel de sommer ces entropies en les pondérant en fonction de la proportion d'exemples dans chacun des sous-ensembles. Cela nous amène à utiliser la différence entre l'entropie de l'ensemble d'exemples initial (utilisé pour déterminer la racine) et cette somme pondérée pour trouver l'attribut le plus intéressant à placer dans la racine. Donc, l'attribut qui maximise cette différence sera l'attribut qui discrimine le mieux les exemples en fonction de leur classe. Cette différence porte le nom de **gain d'information**.

2.3.3 Gain d'information

Soit une population d'exemples $\mathcal{X}$.

Définition 2.8. Gain d'information

e gain d'information de $\mathcal{X}$ par rapport à un attribut a_j donné est la variation d'entropie causée par la partition de $\mathcal{X}$ selon a_j:

$$Gain(\mathcal{X}, a_j) = H(\mathcal{X}) - \sum_{v \in valeurs(a_j)} \frac{|\mathcal{X}_{a_j=v}|}{|\mathcal{X}|} H(\mathcal{X}_{a_j=v}) \tag{2.3}$$

Table 2.1: Jeu de données *jouer au tennis ?*

Jour	Ciel	Température	Humidité	Vent	Jouer au tennis ?
1	Ensoleillé	Chaude	Elevée	Faible	Non
2	Ensoleillé	Chaude	Elevée	Fort	Non
3	Couvert	Chaude	Elevée	Faible	Oui
4	Pluie	Tiède	Elevée	Faible	Oui
5	Pluie	Fraîche	Normale	Faible	Oui
6	Pluie	Fraîche	Normale	Fort	Non
7	Couvert	Fraîche	Normale	Fort	Oui
8	Ensoleillé	Tiède	Elevée	Faible	Non
9	Ensoleillé	Fraîche	Normale	Faible	Oui
10	Pluie	Tiède	Normale	Faible	Oui
11	Ensoleillé	Tiède	Normale	Fort	Oui
12	Couvert	Tiède	Elevée	Fort	Oui
13	Couvert	Chaude	Normale	Faible	Oui
14	Pluie	Tiède	Elevée	Fort	Non

où:

- $\mathcal{X}_{a_j=v} \subset \mathcal{X}$ est l'ensemble des exemples dont l'attribut considéré a_j prend la valeur v,
- La notation $|\mathcal{X}|$ indique le cardinal de l'ensemble $\mathcal{X}$.

Exemple 2.5 Supposons que $\mathcal{X}$ soit constitué de 14 exemples, 9 positifs et 5 négatifs. Parmi ces exemples, 6 positifs et 2 négatifs prennent la valeur *oui* pour l'attribut a, tandis que les autres exemples prennent la valeur *non* pour cet attribut.

$$
\begin{aligned}
Gain(\mathcal{X}, a) =& H(\mathcal{X}) - \sum_{v \in \{oui, non\}} \frac{|\mathcal{X}_{a=v}|}{|\mathcal{X}|} H(\mathcal{X}_{a=v}) \\
Gain(\mathcal{X}, a) =& H(\mathcal{X}) - \frac{8}{14} H(\mathcal{X}_{a=oui}) - \frac{6}{14} H(\mathcal{X}_{a=non}) \\
H(\mathcal{X}_{a=oui}) =& -\left(\frac{6}{8}\log_2\frac{6}{8} + \frac{2}{8}\log_2\frac{2}{8}\right) \approx 0.811 \\
H(\mathcal{X}_{a=non}) =& -\left(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) = 1.0 \\
Gain(\mathcal{X}, a) \approx& 0.940 - \frac{8}{14} 0.811 - \frac{6}{14} \approx 0.048
\end{aligned}
$$

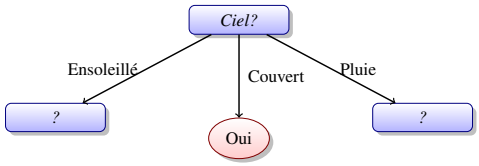
2.3.4 Déroulement de l'algorithme ID3:

On considère l'ensemble des exemples suivant (Table 2.1), l'attribut cible est "*Jouer au tennis ?*"

1. On va déterminer l'attribut qui maximise le gain parmi les attributs (Ciel, Humidité, Vent et Température):

Attribut	Gain
Ciel	**0.246**
Humidité	0.151
Vent	0.048
Température	0.029

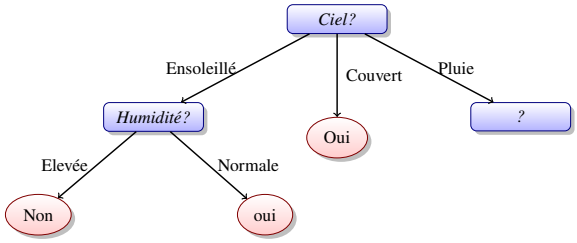
Donc, l'attribut est Ciel.



2. Pour l'attribut "Ciel=Ensoleilllé", on va déterminer l'attribut qui maximise le gain parmi les attributs (Humidité, Vent et Température):

Attribut	Gain
Humidité	**0.970**
Tempétarure	0.570
Vent	0.019

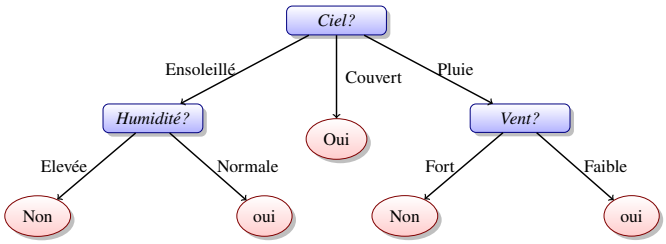
Donc, l'attribut est Humidité.



3. Pour l'attribut "Ciel=Pluie", on va déterminer l'attribut qui maximise le gain parmi les attributs (Humidité, Vent et Température):

Attribut	Gain
Vent	**0.970**
Humidité	0.0.019
Tempétarure	0.019

Donc, l'attribut est Vent.



Algorithme 1 : Algorithme **ID3**

Entrées : ensemble d'exemples $\mathcal{X}$, ensemble d'attributs $\mathcal{A}$

Sorties : racine d'un noeud de l'arbre de décision

Créer un noeud racine;
si *tous les élements de* $\mathcal{X}$ *sont positifs* **alors**
    racine. étiquette $\leftarrow \oplus$;
    retourner racine;
si *tous les élements de* $\mathcal{X}$ *sont négatifs* **alors**
    racine. étiquette $\leftarrow \ominus$;
    retourner racine;
si $\mathcal{A} = \emptyset$ **alors**
    racine. étiquette $\leftarrow$ valeur la plus présente de la classe parmi les $\mathcal{X}$;
    retourner racine;
$a^* \leftarrow \arg\max_{a \in \mathcal{A}} Gain(\mathcal{X}, a)$;
racine. étiquette $\leftarrow a^*$;
pour *toutes les valeurs* v_i *de* a^* **faire**
    ajouter une branche à racine correspondant à la valeur v_i;
    former $\mathcal{X}_{a^*=v_i} \subseteq \mathcal{X}$ dont l'attribut a^* vaut v_i;
    si $\mathcal{X}_{a^*=v_i} = \emptyset$ **alors**
        a l'extrémité de cette branche, mettre une feuille étiquetée avec la valeur la plus présente de la classe parmi les $\mathcal{X}$;
    sinon
        a l'extrémité de cette branche, mettre ID3($\mathcal{X}_{a^*=v_i}$, $\mathcal{A} - a^*$);
    retourner racine;

2.4 Apprentissage, test et validation

2.4.1 Ensemble d'apprentissage et de test

La tâche de classification repose sur la disponibilité d'un corpus initial $\Omega = d_1, \ldots, d_{|\Omega|} \subset D$ des exemples pré-classifiés sous $C = \{c_1, \ldots, c_{|C|}\}$. Les valeurs de la fonction $\Phi : D \times C \rightarrow \{T, F\}$ sont connues pour chaque paire $\langle d_j, c_i \rangle$:

- Un exemples d_j est un exemple positif de c_i si $\Phi(d_j, c_i) = T$,
- Un exemple négatif de c_i si $\Phi(d_j, c_i) = F$.

Avant la construction du classifieur, le corpus initial est divisé en deux séries, pas nécessairement de taille égale: un ensemble d'apprentissage et un ensemble de test. Une fois qu'un classifieur Φ a été construit il est souhaitable d'évaluer son efficacité.

Définition 2.9. Ensemble d'apprentissage

$$EA = \{d_1, \ldots, d_{|EA|}\}$$

Le classifieur Φ pour les catégories $C = \{c_1, \ldots, c_{|C|}\}$ est construit en observant les caractéristiques de ces exemples. ♣

Définition 2.10. Ensemble de test

$$ET = \{d_{|EA+1|}, \ldots, d_{|\Omega|}\}$$

Il est utilisé pour tester l'efficacité des classifieurs. Chaque $d_j \in ET$ est donné au classifieur, et les décisions du classifieur $\Phi(d_j, c_i)$ sont comparées avec les décisions d'expert. ♣

Remarque

- Les documents de ET ne peuvent pas participer d'une façon quelconque à la construction d'induction du classement.
- Si cette condition n'était satisfaite, les résultats expérimentaux obtenus seraient probablement trop bons, et l'évaluation n'aurait donc pas de caractère scientifique.
- La validation est une phase indispensable à tout processus d'apprentissage.
- Elle consiste à vérifier que le modèle construit sur l'ensemble d'apprentissage permet de classer tout individu avec le minimum d'erreurs possible.

2.4.2 Validation

Les résultats de l'évaluation seraient une estimation pessimiste de la performance réelle, si L'ensemble d'apprentissage permet de générer le modèle, l'ensemble de test permet d'évaluer l'erreur réelle du modèle sur un ensemble indépendant évitant ainsi un biais d'apprentissage. S'il s'agit de tester plusieurs modèles et de les comparer, on peut sélectionner le meilleur modèle selon ses performances sur l'ensemble de validation et ensuite évaluer l'erreur réelle sur l'ensemble de test.

Dans le cas de la validation croisée, les k différents classifieurs $\Phi_1, \cdots, \Phi_k$ sont construits par le partitionnement initial du corpus en k ensembles disjoints $ET_1, \cdots, ET_k$. La validation par test est ensuite appliquée de façon itérative sur les paires $\langle EA_i = \Omega - ET_i, ET_i \rangle$. L'efficacité finale est obtenue par le calcul individuel de l'efficacité de

$\Phi_1, \cdots, \Phi_k$. La validation croisée ne construit pas de modèle utilisable, elle estime juste l'erreur réelle. En général le nombre k de parties est fixé à 10.

2.4.3 Mesures de d'efficacité

L'efficacité de classification se mesure généralement par les paramètres classiques du domaine de la recherche d'information:

- La précision,
- Le rappel.

Définition 2.11. Précision

La précision c'est un ratio entre le nombre de documents pertinents trouvés et le nombre total de documents trouvés. ♣

Note *La précision mesure le bruit, et plus elle est proche de* 100%*, moins il y a de bruit, et donc meilleure est la réponse.*

Définition 2.12. Rappel

Le rappel est un ratio entre le nombre de documents pertinents trouvés et le nombre de documents pertinents présents dans la base. ♣

Note *Plus le rappel est proche de* 100%*, moins il y a de silence, et meilleure est la réponse.*

Ces mesures peuvent être estimées par un tableau qui montre toutes les possibilités d'un classifieur. Ce tableau et appelé Matrice de confusion (Table 2.2).

Table 2.2: Matrice de confusion

Catégorie c_i		Classement de l'Expert	
		Vrai	**Faux**
Jugement du Classifieur	**Positif**	TP_i	FP_i
	Négatif	FN_i	TN_i

Où:

- FP_i: le nombre de documents de test mal classés dans la catégorie c_i ,
- TN_i: le nombre de documents bien classés qui n'appartiennent pas à la catégorie c_i,
- TP_i: le nombre de documents bien classés qui appartiennent à la catégorie c_i,et
- FN_i: le nombre de documents de catégorie c_i non classés par le classifieur.

La précision et le rappel peuvent être exprimés localement de la façon suivante:

$$\pi_i = \frac{TP_i}{TP_i + FP_i} \quad , \quad \rho_i = \frac{TP_i}{TP_i + FN_i}$$

Deux approches sont adoptées pour exprimer globalement la précision et le rappel:

- Macro-moyenne: donner un poids égal à toutes les classes.
- Micro-moyenne: Donner un poids proportionnel à la fréquence de chaque classe.

La micro-moyenne *(ang: micro-averaging)* calcule les mesures rappel et précision de façon globale: si l'on considère les tables de contingences associées à chaque catégorie, cela revient à sommer les cases TP et FP de chaque catégorie pour obtenir la table de contingence globale. Les différentes mesures comme le π et ρ sont ensuite calculées à partir des valeurs cumulées.

$$\pi^{\mu} = \frac{TP}{TP+FP} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i+FP_i)} \quad , \quad \rho^{\mu} = \frac{TP}{TP+FN} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i+FN_i)}$$

La macro-moyenne *(ang: macro-averaging)* évalue d'abord indépendamment chaque catégorie. Ensuite, la performance globale du classifieur est calculée en faisant la moyenne des mesures individuelles. Les différentes catégories ont alors la même importance. La précision et le rappel macro-moyenne sont calculés comme suit:

$$\pi^{M} = \frac{\sum_{i=1}^{|C|} \pi_i}{|C|} \quad , \quad \rho^{M} = \frac{\sum_{i=1}^{|C|} \rho_i}{|C|}$$

Table 2.3: Table de contingence globale

Catégories $C = \{c_i, \cdots, c_{\|C\|}\}$		Classement de l'Expert	
		Vrai	**Faux**
Jugement du Classifieur	**Positif**	$TP = \sum_{i=1}^{\|C\|} TP_i$	$FP = \sum_{i=1}^{\|C\|} FP_i$
	Négatif	$FN = \sum_{i=1}^{\|C\|} FN_i$	$TN = \sum_{i=1}^{\|C\|} TN_i$

D'autres mesures peuvent être utilisés pour évaluer la performance 'un classifieur:

- F-measure F_1 peut être considéré comme le degré relatif d'importance attribué à la précision et au rappel:
$$F_1 = \frac{2\pi\rho}{\pi+\rho}$$
- Le taux de succès *(ang: accuracy rate)* désigne le pourcentage d'exemples bien classés par le classifieur:
$$A = \frac{TP+TN}{TP+TN+FP+FN}$$
- Taux d'erreur *(ang: error rate)* désigne le pourcentage d'exemples mal classés.
$$E = \frac{FP+FN}{TP+TN+FP+FN} = 1 - A$$

2.5 Exercices

Exercice 1

On considère l'ensemble de données $\mathcal{X}$ suivant:

N	a_1	a_2	a_3	Classe	N	a_1	a_2	a_3	Classe
1	V	V	1.0	+	6	F	V	3.0	−
2	V	V	6.0	+	7	F	F	8.0	−
3	V	F	5.0	−	8	V	F	7.0	+
4	F	F	4.0	+	9	F	V	5.0	−
5	F	V	7.0	−					

(a) Calculer l'entropie de $\mathcal{X}$ pour l'attribut cible Classe.

(b) Est-il possible d'appliquer directement l'algorithme **ID3** sur l'ensemble $\mathcal{X}$? Argumenter votre réponse.

(c) Calculer le gain d'information des attributs a_1 et a_2 relatif à l'ensemble $\mathcal{X}$.

(d) Calculer le gain d'information de l'attribut a_3 pour chaque point de répartition suivant: $\{2.0, 3.5, 4.5, 5.5, 6.5, 7.5\}$, et déduire la meilleure répartition.

(e) Déduire l'attribut à choisir comme racine de l'arbre de décision. .

Voir Corrigé, page 26.

Exercice 2

On considère l'ensemble de données $\mathcal{X}$ suivant ayant les attributs A, B, C et D:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
A	a_1	a_1	a_1	a_2	a_2	a_2	a_1	a_2	a_3	a_3
B	b_1	b_2	b_2	b_1	b_2	b_2	b_1	b_1	b_1	b_2
C	c_1	c_2	c_3	c_1	c_1	c_1	c_1	c_2	c_3	c_2
D	d_2	d_2	d_1	d_1	d_1	d_2	d_1	d_2	d_1	d_2
Classe	+	+	−	−	−	+	+	−	+	+

(a) Construire l'arbre de décision correspondant à l'ensemble $\mathcal{X}$ en utilisant l'algorithme ID3.

(b) Prédire la classe de l'exemple ayant les attributs (a_2,b_1,c_3,d_1).

Voir Corrigé, page 27.

Exercice 3

Le tableau suivant contient des données sur les résultats obtenus par des étudiants de Tronc Commun. Chaque étudiant est décrit par 3 attributs: Redoublant ou non, la série et la mention du Baccalauréat obtenu. Les étudiants sont répartis en deux classes: Admis et Non Admis. On veut construire un arbre de décision à partir des données du tableau, pour rendre compte des éléments qui influent sur les résultats des étudiants en Tronc Commun. Les lignes de 1 à 12 sont utilisées comme données

d'apprentissage. Les lignes restantes (de 13 à 16) sont utilisées comme données de tests.

N	Redoub.	Série	Mention	Classe	N	Redoub.	Série	Mention	Classe
1	Non	Maths	A Bien	Admis	7	Oui	Sciences	Passable	Non Admis
2	Non	Techniques	A Bien	Admis	8	Oui	Maths	Passable	Non Admis
3	Oui	Sciences	A Bien	Non Admis	9	Oui	Techniques	Passable	Non Admis
4	Oui	Sciences	Bien	Admis	10	Oui	Maths	T Bien	Admis
5	Non	Maths	Bien	Admis	11	Oui	Techniques	T Bien	Admis
6	Non	Techniques	Bien	Admis	12	Non	Sciences	T Bien	Admis
13	Oui	Maths	Bien	Admis	15	Non	Maths	TBien	Admis
14	Non	Sciences	A Bien	Non Admis	16	Non	Maths	Passable	Non Admis

(a) Utiliser les données des lignes de 1 à 12 pour construire l'arbre en utilisant l'algorithme ID3. Montrez toutes les étapes de calcul (En cas d'égalité, prendre toutes les décisions possibles). Dessinez l'arbre final.

(b) Quels sont les résultats de test de l'arbre obtenu sur les données des lignes de 13 à 16 ? Que suggérez-vous?

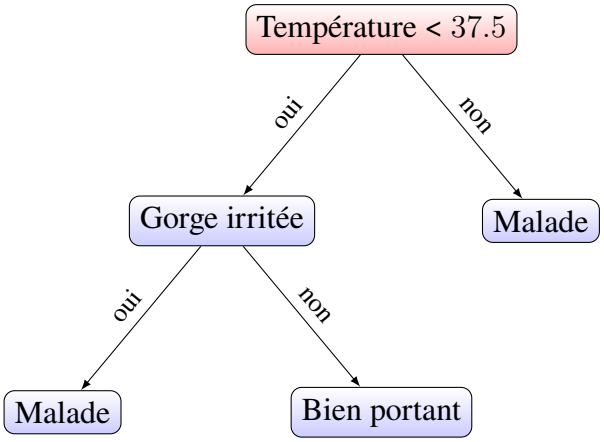
Voir Corrigé, page 28.

Exercice 4

On dispose d'un échantillon de 200 patients. On sait que 100 sont malades (m), et les 100 autres sont bien portants (bp). On dispose en outre des informations suivantes:

	gorge irritée		gorge non irrité	
température < 37.5	6 bp	37 m	91 bp	1 m
température >= 37.5	2 bp	21 m	1 bp	41 m

Soit l'arbre de décision de la figure suivante:



(a) Calculer, pour l'arbre de décision donné , le gain d'information à chaque nœud en utilisant la fonction d'entropie.

(b) Considérons l'arbre vide. Nous avons le choix entre choisir l'attribut "température < 37.5" et l'attribut "gorge irritée". Lequel doit-on choisir pour maximiser le gain?

Exercice 5

Soit un jeu de données de 9 patients contenu dans le tableau suivant. Les attributs sont "Fièvre", "Douleur" et "Toux". La classe est "Maladie".

Fièvre	Douleur	Toux	Maladie
oui	Abdomen	non	Appendicite
non	Abdomen	oui	Appendicite
oui	gorge	non	rhume
oui	gorge	oui	rhume
non	gorge	oui	mal de gorge
oui	non	non	aucune
oui	non	oui	rhume
non	non	oui	refroidissement
non	non	non	aucune

(a) Déterminez l'arbre de décision de ces exemples. Détaillerez à chaque fois le calcul de l'entropie et du gain.

Chapitre 2: Corrigé des exercices

Exercice 1

On considère l'ensemble de données $\mathcal{X}$ suivant:

N	a_1	a_2	a_3	Classe	N	a_1	a_2	a_3	Classe
1	V	V	1.0	+	6	F	V	3.0	−
2	V	V	6.0	+	7	F	F	8.0	−
3	V	F	5.0	−	8	V	F	7.0	+
4	F	F	4.0	+	9	F	V	5.0	−
5	F	V	7.0	−					

(a) Calculer l'entropie de $\mathcal{X}$ pour l'attribut cible Classe.

Solution:

$$H(\mathcal{X}) = -\frac{4}{9}log_2(\frac{4}{9}) - \frac{5}{9}log_2(\frac{5}{9}) = 0.9911$$

(b) Est-il possible d'appliquer directement l'algorithme **ID3** sur l'ensemble $\mathcal{X}$? Argumenter votre réponse.

Solution:

Ce n'est pas possible d'appliquer directement l'algorithme **ID3** sur l'ensemble $\mathcal{X}$ du fait que l'attribut a_3 est continu, et l'algorithme **ID3** ne prend en compte que des attributs nominaux.

(c) Calculer le gain d'information des attributs a_1 et a_2 relatif à l'ensemble $\mathcal{X}$.

Solution:

$$Gain(\mathcal{X}, a_j) = H(\mathcal{X}) - \sum_{v \in valeurs(a_j)} \frac{|\mathcal{X}_{a_j=v}|}{|\mathcal{X}|} H(\mathcal{X}_{a_j=v})$$

Pour l'attribut a_1:

a_1	+	-
V	3	1
F	1	4

$$\begin{aligned} Gain(\mathcal{X}, a_1) &= H(\mathcal{X}) - \left[\frac{4}{9}H(\mathcal{X}_{a_1=V}) + \frac{5}{9}H(\mathcal{X}_{a_1=F})\right] \\ &= (0.9911) - \left[\frac{4}{9}\left(-\frac{1}{4}log_2\frac{1}{4} - \frac{3}{4}log_2\frac{3}{4}\right) + \frac{5}{9}\left(-\frac{1}{5}log_2\frac{1}{5} - \frac{4}{5}log_2\frac{4}{5}\right)\right] \\ &= 0.2294 \end{aligned}$$

Pour l'attribut a_2:

a_2	+	-
V	2	3
F	2	2

$$
\begin{aligned}
Gain(\mathcal{X}, a_1) &= H(\mathcal{X}) - \left[\frac{5}{9}H(\mathcal{X}_{a_2=V}) + \frac{4}{9}H(\mathcal{X}_{a_2=F})\right] \\
&= (0.9911) - \left[\frac{5}{9}\left(-\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5}\right) + \frac{4}{9}\left(-\frac{2}{4}log_2\frac{2}{4} - \frac{2}{4}log_2\frac{2}{4}\right)\right] \\
&= 0.0072
\end{aligned}
$$

(d) Calculer le gain d'information de l'attribut a_3 pour chaque point de répartition suivant: $\{2.0, 3.5, 4.5, 5.5, 6.5, 7.5\}$, et déduire la meilleure répartition.

Solution:

Pour l'attribut a_3:

Pt. Répartition	Entropie	Gain	Pt. Répartition	Entropie	Gain
2.0	0.8484	0.1427	5.5	0.9839	0.0072
3.5	0.9885	0.0026	6.5	0.9728	0.0183
4.5	0.9183	0.0728	7.5	0.8889	0.1022

La meilleure répartition est au point 2.0

(e) Déduire l'attribut à choisir comme racine de l'arbre de décision. .

Solution:

Ll'attribut à choisir comme racine de l'arbre de décision est a_1.

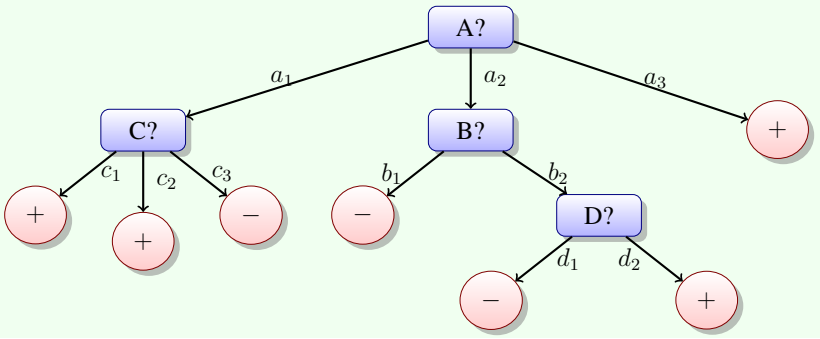
Exercice 2

On considère l'ensemble de données $\mathcal{X}$ suivant ayant les attributs A, B, C et D:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
A	a_1	a_1	a_1	a_2	a_2	a_2	a_1	a_2	a_3	a_3
B	b_1	b_2	b_2	b_1	b_2	b_2	b_1	b_1	b_1	b_2
C	c_1	c_2	c_3	c_1	c_1	c_1	c_1	c_2	c_3	c_2
D	d_2	d_2	d_1	d_1	d_1	d_2	d_1	d_2	d_1	d_2
Classe	$+$	$+$	$-$	$-$	$-$	$+$	$+$	$-$	$+$	$+$

(a) Construire l'arbre de décision correspondant à l'ensemble $\mathcal{X}$ en utilisant l'algorithme ID3.

Solution:



(b) Prédire la classe de l'exemple ayant les attributs (a_2,b_1,c_3,d_1).

Solution:

L'exemple ayant les attributs (a_2,b_1,c_3,d_1) est classé dans la classe négative.

Exercice 3

Le tableau suivant contient des données sur les résultats obtenus par des étudiants de Tronc Commun. Chaque étudiant est décrit par 3 attributs: Redoublant ou non, la série et la mention du Baccalauréat obtenu. Les étudiants sont répartis en deux classes: Admis et Non Admis. On veut construire un arbre de décision à partir des données du tableau, pour rendre compte des éléments qui influent sur les résultats des étudiants en Tronc Commun. Les lignes de 1 à 12 sont utilisées comme données d'apprentissage. Les lignes restantes (de 13 à 16) sont utilisées comme données de tests.

N	Redoub.	Série	Mention	Classe	N	Redoub.	Série	Mention	Classe
1	Non	Maths	A Bien	Admis	7	Oui	Sciences	Passable	Non Admis
2	Non	Techniques	A Bien	Admis	8	Oui	Maths	Passable	Non Admis
3	Oui	Sciences	A Bien	Non Admis	9	Oui	Techniques	Passable	Non Admis
4	Oui	Sciences	Bien	Admis	10	Oui	Maths	T Bien	Admis
5	Non	Maths	Bien	Admis	11	Oui	Techniques	T Bien	Admis
6	Non	Techniques	Bien	Admis	12	Non	Sciences	T Bien	Admis
13	Oui	Maths	Bien	Admis	15	Non	Maths	TBien	Admis
14	Non	Sciences	A Bien	Non Admis	16	Non	Maths	Passable	Non Admis

(a) Utiliser les données des lignes de 1 à 12 pour construire l'arbre en utilisant l'algorithme ID3. Montrez toutes les étapes de calcul (En cas d'égalité, prendre toutes les décisions possibles). Dessinez l'arbre final.

Solution:

On remarque que sur les 12 lignes des données d'apprentissage, 8 correspondent à la classe "Admis" et 4 à la classe "Non Admis". L'entropie de l'ensemble S (à la racine de l'arbre) est donc égale à:

$$\begin{aligned} H(S) &= -(8/12) * Log_2(8/12) - (4/12) * Log_2(4/12) \\ &= 0.92 \end{aligned}$$

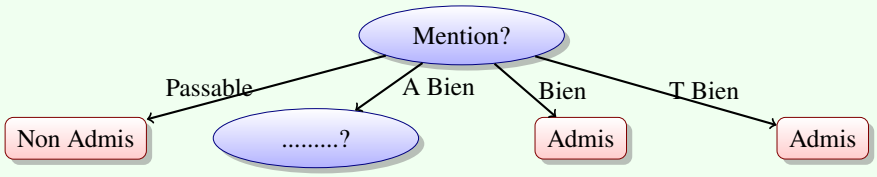
Pour connaitre quel attribut on doit choisir comme test au niveau de la racine de l'arbre, il faut calculer le gain d'entropie sur chacun des attributs: "Redoublant", "Série" et "Mention":

$$\begin{aligned} Gain(S, Red) &= H(S) - 7/12 * H(S_{Red=Oui}) - 5/12 * H(S_{Red=Non}) \\ &= 0.34 \end{aligned}$$

$$\begin{aligned} Gain(S, \textit{Série}) &= H(S) - (4/12) * H(S_{Srie=Maths}) \\ &\quad - (4/12) * H(S_{\textit{Série}=Techniques}) - (4/12) * H(S_{\textit{Série}=Sciences}) \\ &= 0.04 \end{aligned}$$

$$\begin{aligned} Gain(S, Mention) &= H(S) - (3/12) * H(S_{Mention=Pass}) - (3/12) * H(S_{Mention=A.Bien}) \\ &\quad - (3/12) * H(S_{Mention=Bien}) - 3/12 * H(S_{Mention=T.Bien}) \\ &= 0.69 \end{aligned}$$

On constate que le plus grand gain d'entropie est obtenu sur l'attribut "Mention". C'est donc cet attribut qui est choisi comme test à la racine de l'arbre.



On voit que mettre l'attribut "Mention" à la racine de l'arbre permet d'obtenir 4 branches dont 3 produisent des noeuds finaux. Il ne reste à traiter que le noeud présentant un mélange correspondant à la branche "A Bien". Ce noeud comporte un ensemble (que nous noterons $S2$) ayant 2 individus appartenant à la classe "Admis" et 1 individu de la classe "Non Admis". L'entropie de l'ensemble $S2$ est donc égale à:

$$\begin{aligned} H(S2) &= -(2/3) * Log_2(2/3) - (1/3) * Log_2(1/3) \\ &= 0.92 \end{aligned}$$

Pour connaitre quel attribut on doit choisir comme test au niveau du noeud, il

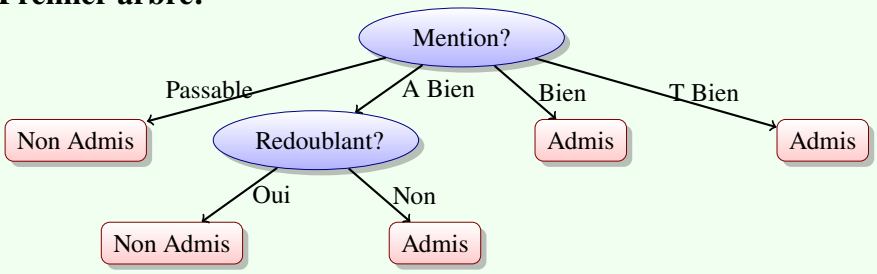
faut calculer le gain d'entropie sur chacun des attributs restants: "Redoublant" et "Série".

$$Gain(S2, Red) = H(S2) - 1/3 * H(S2_{Red=Oui}) - 2/3 * H(S2_{Red=Non})$$
$$= 0.92$$

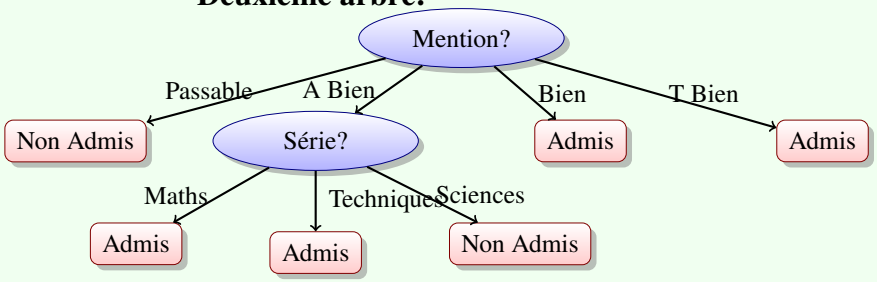
$$Gain(S2, \textit{Série}) = H(S2) - (1/3) * H(S2_{Srie=Maths})$$
$$- (1/3) * H(S2_{\textit{Série}=Techniques}) - (1/3) * H(S2_{\textit{Série}=Sciences})$$
$$= 0.92$$

On constate que les deux attributs "Redoublant" et "Série" procurent le même gain d'entropie. Nous pouvons donc choisir l'un ou l'autre comme test au niveau du noeud courant. Nous avons donc deux arbres de décision possibles:

Premier arbre:



Deuxième arbre:



(b) Quels sont les résultats de test de l'arbre obtenu sur les données des lignes de 13 à 16 ? Que suggérez-vous?

Solution:

N	Red	Série	Mention	Classe	Arbre1 Prédiction	Observation	Arbre2 Prédiction	Observation
13	Oui	Maths	Bien	Admis	Admis	Correct	Admis	Correct
14	Non	Sciences	A Bien	Non Admis	Admis	Erreur	Non Admis	Correct
15	Non	Maths	T Bien	Admis	Admis	Correct	Admis	Correct
16	Non	Maths	Passable	Non Admis	Non Admis	Correct	Non Admis	Correct

On remarque que l'arbre 1 a donné un taux d'erreur de 1/4 soit 25%, alors que l'arbre 2 présente un taux de succès de 100%. Cela suggère de retenir en définitif l'arbre 2.

Chapitre 3 Règles d'association

La fouille de règles d'association se rapporte à la découverte des relations entre les attributs d'un ensemble de données appelé souvent ensemble des transactions:

- Une transaction est l'ensemble des articles achetés ensemble par les clients.
- Une règle est normalement exprimée sous la forme $A \Rightarrow B$, où A et B sont des ensembles d'attributs de l'ensemble de données. Cela implique que les transactions qui contiennent A contiennent B avec une grande probabilité.
- La règle peut s'écrire sous une autre forme:

SI <*certaines conditions satisfaites* > ALORS <*prédire les valeurs pour certains autres attributs*>,

3.1 Règle d'association et ensemble d'items fréquents

3.1.1 Notations

- On dispose de N données x_i , chacune décrites par P attributs $x_{i,j}$ dénote la valeur de l'attribut a_j de la donnée x_i.
- Dans de nombreuses applications, chaque attribut correspond à un *item* et la valeur de cet attribut dans une donnée particulière indique sa quantitée dans cette donnée.
- Un cas particulier est celui où les attributs sont à valeur binaire et indiquent la présence ou l'absence d'un item.

3.1.2 Règle d'association

Définition 3.1. Règle d'association

Une règle d'association est de la forme:

$$(a_i = v_i, a_j = v_j, \ldots, a_m = v_m) \Rightarrow (a_\alpha = v_\alpha, a_\beta = v_\beta, \ldots$$

Ce qui s'interprète par: si les attributs a_i, a_j, ... , a_m ont une certaine valeur, alors l'attribut a_α prend généralement une certaine valeur v_α , a_β une certaine valeur v_β, ...

♣

Remarque

- La difficulté consiste notamment à trouver des règles qui soient significatives et non seulement le résultat du hasard.

- Les valeurs de N et P sont généralement trés grandes ($N = 10^6$ et $P = 10^5$ par exemple).

On s'intéresse au cas où les attributs prennent une valeur binaire, indiquant donc la présence ou l'absence d'un item. On présente une approche qui s'appuie sur la notion d'ensemble d'items fréquents (**EIF**), c'est-à-dire, des items qui sont souvent présents ensemble dans une même donnée. Après avoir détecté ces **EIF**, on génère ensuite des règles d'association.

Dans ce qui suit, on part des individus suivants (Table 3.1):

Table 3.1: Table des transations

	Item A	Item B	Item C	Item D
Individu 1	x	x		
Individu 2	x		x	
Individu 3		x		
Individu 4	x		x	x
Individu 5		x		

3.1.3 Ensemble d'items fréquents

3.1.3.1 Support

Définition 3.2. Support

*On définit le **support** d'un ensemble d'items comme la fréquence d'apparition simultanée des items figurant dans l'ensemble.* ♣

Exemple 3.1

- $Supp(A, B) = \frac{1}{5}$ car A et B n'apparaissent simultanément que dans l'individu 1;
- $Supp(A, C) = \frac{2}{5}$ car A et C apparaissent simultanément dans les individus 2 et 4.

3.1.3.2 Ensemble d'items fréquents

Définition 3.3. Ensemble d'items fréquents

On dit qu'un ensemble d'items I est un ensemble d'items fréquents si le support de cet ensemble d'items est supérieur à un certain seuil $minsup$ (10% par exemple):

$$I \text{ est fréquent } \Leftrightarrow Supp(I) \geq minsup$$

♣

Exemple 3.2 Avec $minsup = 30\%$,

- L'ensemble $'(A, C)$ est fréquent car $Supp(A, C) = \frac{2}{5} > minsup$;
- L'ensemble (A, B) n'est pas fréquent car $Supp(A, B) = \frac{1}{5} < minsup$.

Proposition 3.1

Si S est un ensemble d'items fréquents, alors tout sous-ensemble de S est également un ensemble d'items fréquents.

♠

Note

- *Un ensemble d'items fréquents M est maximal si tout sur-ensemble de M n'est pas un **EIF**.*
- *Un ensemble d'items fréquents F est fermé si tout sur-ensemble de F n'a pas le même support que F.*

3.1.3.3 Confiance

Définition 3.4. Confiance

*La confiance d'une règle **si** condition **alors** conclusion est le rapport:*

$$\frac{\text{nombre de données où les items de la condition et de la conclusion apparaissent simultanément}}{\text{nombre de données où les items de la condition apparaissent simultanément}}$$

♣

Exemple 3.3

- $confiance(A \Rightarrow B) = \frac{1}{3}$ car A et B apparaissent simultanément dans 1 individu et A apparaît dans 3 individus,
- $confiance(A \Rightarrow C) = \frac{2}{3}$ ar A et C apparaissent simultanément dans 2 individu et A apparaît dans 3 individus.

On définit un seuil de confiance comme la valeur minimale que la confiance doit avoir pour que l'apparition simultanée des items considérés ne puisse pas être simplement due au hasard. On ne s'intéresse qu'aux règles ayant une confiance maximale.

Proposition 3.2

Si la règle: ***Si*** *a et b* ***Alors*** *c et d à une confiance supérieure à un seuil fixé, alors les deux règles:*

- *Si a et b et d Alors c*
- *Si a et b et c Alors d ont une confiance supérieure à ce même seuil.*

♠

3.2 Algorithme Apriori

L'algorithme Apriori (Agrawal et Srikant, 1994) est le premier algorithme proposé pour extraire les règles d'association présentes dans un jeu de données, pour un *seuil de support* et un *seuil de confiance* fixés. Cet algorithme fonctionne en deux phases:

- Tout d'abord on recherche tous les ensembles d'items fréquents (**EIF**);
- Ensuite, on utilise ces **EIF** pour déterminer les règles d'association dont la confiance est supérieure au seuil fixé.

3.2.1 Extraction des EIFs

On note L_k l'ensemble des ensembles des items fréquents de taille k et C_k l'ensemble des ensembles des items candidats de taille k.

- L'algorithme commence par l'extraction des ensembles d'items fréquents de taille 1: L_1.
- Ensuite, il génère itérativement l'extraction des ensembles d'items fréquents de taille k à partir des ensembles d'items fréquents de taille $k-1$ obtenus à l'étape précédente.

La fonction apriori_gen génère les ensembles d'items candidats en ajoutant aux itemsets fréquents de taille $k-1$ des items fréquents.

3.2.2 Génération des règles d'association à partir des EIF

Disposant des **EIF**, il nous faut maintenant les transformer en règles. Supposons que L_3 contienne le triplet (a, b, c). Plusieurs règles d'association peuvent être engendrées par ce triplet:

- si a et b alors c
- si a et c alors b
- si a alors b et c
- si b et c alors a

- si b alors a et c
- si c alors a et b

On adoptons une approche itérative:

- En effet, il est tout à fait envisageable de tester toutes les règles candidates dont la conclusion ne contient qu'un seul item.
- Ensuite, on va appliquer la proposition précédente sur la confiance:
  - Supposons que $L4 = \{(a, b, c, d)\}$.
  - Supposons que la confiance des règles: *si a et b et c alors d* et *si a et b et d alors c* soit supérieure au seuil.
  - Dans ce cas, on peut affirmer que la confiance de la règle *si a et b alors c et d* est également supérieure au seuil.
- Determiner les règles ayant un seul item en conclusion et dont la confiance est supérieure au seuil.
- Engendrer les règles ayant deux items en conclusion dont la confiance est supérieure au seuil.
- Itérer vers les règles ayant 3 items en conclusion, puis 4, . . . etc.

3.2.3 Application

Sur l'exemple vu plus haut, on prend un support minimal de 2. On a:

- $C_1 = \{A, B, C, D\}$
- $L_1 = \{A, B, C\}$
- $C_2 = \{(A, B), (A, C), (B, C)\}$
- $L_2 = \{(A, C)\}$
- $C_3 = \emptyset$

Les règles suivantes sont examinées:

- si A et B alors C
- si A et C alors B
- si B et C alors A

dont les confiances sont nulles. On n'examine donc pas de règles ayant deux items en con-

Algorithme 2 : Algorithme APriori

Entrées : ensemble d'exemples D, un seuil de support min_sup

Sorties : **EIF** disponibles L

```
1  L_1 ← 1-itemsets fréquents;
2  k ← 2;
3  tant que L_{k-1} ≠ ∅ faire
4      C_k =apriori_gen(L_{k-1});
5      pour chaque t ∈ D faire
6          C_t = subset(C_k, t);
7          pour chaque candidat c ∈ C_t faire
8              c.count++;
9      L_k = {c ∈ C_k | c.count ≥ min_sup};
10     k++;
11 Fonction apriori_gen(L_{k-1}:k-1 itemsets fréquents)
12     pour chaque itemset l_1 ∈ L_{k-1} faire
13         pour chaque itemset l_2 ∈ L_{k-1} faire
14             si
               (l_1[1] = l_2[1]) ∧ ... ∧ (l_1[k-2] = l_2[k-2]) ∧ (l_1[k-1] < l_2[k-1])
               alors
15                 c = l_1 ⋈ l_2;
16                 si has_infrequent_subset(c,L_{k-1}) alors
17                     supprimer c;
18                 sinon
19                     ajouter c à C_k;
20     retourner C_k;
21 Fonction has_infrequent_subset(c: k-itemset candidat,L_{k-1}:k-1 itemsets
   fréquents)
22     pour chaque (k-1)- subset s ∈ c faire
23         si s ∉ L_{k-1} alors
24             retourner Vrai;
25     retourner Faux;
```

clusion.

3.3 Exercices

Exercice 1

Soient A, B, C des ensembles d'items, et p, q et r des items:

(a) Calculer la confiance des règles $\emptyset \rightarrow A$, $A \rightarrow \emptyset$.

(b) Soient c_1, c_2 et c_3 les confiances respectives des règles $\{p\} \rightarrow \{q\}$, $\{p\} \rightarrow \{q, r\}$ et $\{p, r\} \rightarrow \{q\}$:

1. Si on suppose que c_1, c_2 et c_3 ont des valeurs différentes, Déterminer les relations qui peuvent exister entre c_1, c_2 et c_3 ? Déduire la règle ayant une confiance minimale.
2. Si on suppose que la confiance des règles $A \rightarrow B$ et $B \rightarrow C$ est supérieure à un seuil $minconf$. Est-il possible que la règle $A \rightarrow C$ aura une confiance inférieure à $minconf$?

Voir Corrigé, page 40.

Exercice 2

Dans un supermarché, on dispose de la base de transactions suivante:

TID	Items	TID	Items
T1	Lait, Jus, Couches	T6	Lait, Couches, Pain, Beurre
T2	Pain, Beurre, Lait	T7	Pain, Beurre, Couches
T3	Lait, Couches, Sucre	T8	Jus, Couches
T4	Pain, Beurre, Sucre	T9	Lait, Couches, Pain, Beurre
T5	Jus, Sucre, Couches	T10	Jus, Sucre

En utilisant l'algorithme **Apriori** avec un support minimum de 30% et une confiance minimale de 90%, trouver:

(a) Les ensembles des items fréquents (EIF): L_1, L_2, L_3 et L_4.

(b) Les règles d'association de type: $A, B \Rightarrow C$

Voir Corrigé, page 40.

Exercice 3

(a) Une règle spécifique est une règle de la forme $\{p\} \rightarrow \{q_1, q_2, \ldots, q_n\}$ où l'antécédent de la règle contient un seul item. Soit ζ la confiance minimale de toutes les règles spécifiques générées à partir d'un itemset donné.

$$\zeta\Big(\{p_1, p_2, \ldots, p_k\}\Big) = min\Big[c\Big(\{p_1\} \rightarrow \{p_2, p_3, \ldots, p_k\}\Big), \ldots, c\Big(\{p_k\} \rightarrow \{p_1, p_2, \ldots, p_{k-1}\}\Big)\Big]$$

Dites si ζ est monotone, anti-monotone ou non-monotone?

NB: Faire un comparaison entre $\zeta(\{A, B\})$ et $\zeta(\{A, B, C\})$

(b) Une règle sélective est une règle de la forme $\{p_1, p_2, \ldots, p_n\} \rightarrow \{q\}$ où le conséquent de la règle contient un seul item. Soit η la confiance minimale de toutes les règles sélectives générées à partir d'un itemset donné.

$$\eta\Big(\{p_1, p_2, \ldots, p_k\}\Big) = min\Big[c\Big(\{p_2, p_3, \ldots, p_k\} \rightarrow \{p_1\}\Big), \ldots, c\Big(\{p_1, p_2, \ldots, p_{k-1}\} \rightarrow \{p_k\}\Big)\Big]$$

Dites si η est monotone, anti-monotone ou non-monotone?

NB: Faire un comparaison entre $\eta(\{A, B\})$ et $\eta(\{A, B, C\})$

Voir Corrigé, page 41.

Exercice 4

Le but de cet exercice est de dérouler l'algorithme **Apriori** sur les données d'une épicerie suivante, en utilisant comme seuils de support et de confiance *minsup=33,3%* et *minconf=60%*.

(a) Calculez d'abord les ensembles de k-itemsets candidats et fréquents, pour chaque étape de l'algorithme.

(b) Montrez ensuite l'ensemble des règles d'associations généré, en les triant par confiance décroissante.

TID	Items
T1	Fromage, Pain, Ketchup
T2	Fromage, Pain
T3	Fromage, Coca, Chips
T4	Chips, Coca
T5	Chips, Ketchup
T6	Fromage, Coca, Chips

Exercice 5

Soit l'ensemble d'items $I = \{A, B, C, D, E\}$. Supposons que l'algorithme **Apriori** soit appliquée sur les données ci-dessous avec le seuil *minsup=30%*.

(a) Dessinez le treillis des itemsets. Pour chaque noeud du treillis, joignez lui la lettre suivante, selon le cas:

- **N**: si l'itemset ne fait pas partie des itemsets candidats (s'il n'est jamais généré par Apriori).
- **F**: si c'est un itemset fréquent.
- **R**: si c'est un itemset candidat, qui s'est révélé non-fréquent (itemset rare).

(b) Quel est le pourcentage d'itemsets qui sont fréquents ?

TID	Items
T1	A, B, D, E
T2	B, C, D
T3	A, B, D, E
T4	A, C, D, E
T5	B, C, D, E
T6	B, D, E
T7	C, D
T8	A, B, C
T9	A, D, E
T10	B, D

(c) Quel est le *«taux d'élagage»* d'Apriori sur ces données (le pourcentage d'itemsets non candidats) ?

(d) Quel est le *«taux de rejet»* (le pourcentage d'itemsets non-fréquents parmi les candidats) ?

Chapitre 3: Corrigé des exercices

Exercice 1

Soient A, B, C des ensembles d'items, et p, q et r des items:

(a) Calculer la confiance des règles $\emptyset \rightarrow A$, $A \rightarrow \emptyset$.

Solution:

$c(\emptyset \rightarrow A) = s(\emptyset \rightarrow A) = s(A)$. $c(A \rightarrow \emptyset) = 100\%$

(b) Soient c_1, c_2 et c_3 les confiances respectives des règles $\{p\} \rightarrow \{q\}$, $\{p\} \rightarrow \{q, r\}$ et $\{p, r\} \rightarrow \{q\}$:

1. Si on suppose que c_1, c_2 et c_3 ont des valeurs différentes, Déterminer les relations qui peuvent exister entre c_1, c_2 et c_3 ? Déduire la règle ayant une confiance minimale.
2. Si on suppose que la confiance des règles $A \rightarrow B$ et $B \rightarrow C$ est supérieure à un seuil $minconf$. Est-il possible que la règle $A \rightarrow C$ aura une confiance inférieure à $minconf$?

Solution:

1. $$c_1 = \frac{s(p \cup q)}{s(p)} \qquad c_2 = \frac{s(p \cup q \cup r)}{s(p)} \qquad c_3 = \frac{s(p \cup q \cup r)}{s(p \cup r)}$$

   On a: $s(p) \geq s(p \cup q) \geq s(p \cup q \cup r)$ d'où: $c_1 \geq c_2$ et $c_3 \geq c_2$.

   Donc, c_2 est la confiance minimale.
2. Oui, ça depend du support des itemsets A, B, et C. Par exemple:

   $$s(A, B) = 60\% \qquad s(A) = 90\%$$
   $$s(A, C) = 20\% \qquad s(B) = 70\%$$
   $$s(B, C) = 50\% \qquad s(C) = 60\%$$

   On suppose que $minconf = 50\%$ alors:

   $$c(A \rightarrow B) = 66\% > minconf$$
   $$c(B \rightarrow C) = 71\% > minconf$$

   Mais $c(A \rightarrow C) = 22\% < minconf$

Exercice 2

Dans un supermarché, on dispose de la base de transactions suivante:

TID	Items	TID	Items
T1	Lait, Jus, Couches	T6	Lait, Couches, Pain, Beurre
T2	Pain, Beurre, Lait	T7	Pain, Beurre, Couches
T3	Lait, Couches, Sucre	T8	Jus, Couches
T4	Pain, Beurre, Sucre	T9	Lait, Couches, Pain, Beurre
T5	Jus, Sucre, Couches	T10	Jus, Sucre

En utilisant l'algorithme **Apriori** avec un support minimum de 30% et une confiance minimale de 90%, trouver:

(a) Les ensembles des items fréquents (EIF): L_1, L_2, L_3 et L_4.

Solution:

L_1

Items	Supp
Lait	5
Jus	4
Pain	5
Beurre	5
Couches	7
Sucre	4

L_2

Items	Supp
(Lait , Pain)	3
(Lait , Beurre)	3
(Lait , Couches)	4
(Jus , Couches)	3
(Pain , Beurre)	5
(Pain , Couches)	3
(Beurre , Couches)	3

L_3

Items	Supp
(Lait , Pain , Beurre)	3
(Pain , Beurre , Couches)	3

$L_4 = \emptyset$

(b) Les règles d'association de type: $A, B \Rightarrow C$

Solution:

1. Lait , Beurre $\Rightarrow$ Pain
2. Lait , Pain $\Rightarrow$ Beurre
3. Beurre , Couches $\Rightarrow$ Pain
4. Pain , Couches $\Rightarrow$ Beurre

Exercice 3

(a) Une règle spécifique est une règle de la forme $\{p\} \rightarrow \{q_1, q_2, \ldots, q_n\}$ où l'antécédent de la règle contient un seul item. Soit ζ la confiance minimale de

toutes les règles spécifiques générées à partir d'un itemset donné.

$$\zeta\Big(\{p_1, p_2, \ldots, p_k\}\Big) = min\Big[c\Big(\{p_1\} \rightarrow \{p_2, p_3, \ldots, p_k\}\Big), \ldots, c\Big(\{p_k\} \rightarrow \{p_1, p_2, \ldots, p_{k-1}\}\Big)\Big]$$

Dites si ζ est monotone, anti-monotone ou non-monotone?

NB: Faire un comparaison entre $\zeta(\{A, B\})$ et $\zeta(\{A, B, C\})$

Solution:

ζ est anti-monotone parce que

$$\zeta\Big(\{p_1, p_2, \ldots, p_k\}\Big) \geq \zeta\Big(\{p_1, p_2, \ldots, p_k, p_{k+1}\}\Big)$$

Par exemple, on peut comparer les valeurs de ζ pour $\{A, B\}$ et $\{A, B, C\}$.

$$\begin{aligned}
\zeta\Big(\{A, B\}\Big) &= min\Big(c(A \rightarrow B), c(B \rightarrow A)\Big) \\
&= min\Big(\frac{s(A, B)}{s(A)}, \frac{s(A, B)}{s(B)}\Big) \\
&= \frac{s(A, B)}{max\Big(s(A), s(B)\Big)}
\end{aligned}$$

$$\begin{aligned}
\zeta\Big(\{A, B, C\}\Big) &= min\Big(c(A \rightarrow \{B, C\}), c(B \rightarrow \{A, C\}), c(C \rightarrow \{A, B\})\Big) \\
&= min\Big(\frac{s(A, B, C)}{s(A)}, \frac{s(A, B, C)}{s(B)}, \frac{s(A, B, C)}{s(C)}\Big) \\
&= \frac{s(A, B, C)}{max\Big(s(A), s(B), s(C)\Big)}
\end{aligned}$$

Puisque $s(A, B, C) \leq s(A, B)$ et $max\Big(s(A), s(B), s(C)\Big) \geq max\Big(s(A), s(B)\Big)$, alors

$\zeta(\{A, B\}) \geq \zeta(\{A, B, C\})$.

(b) Une règle sélective est une règle de la forme $\{p_1, p_2, \ldots, p_n\} \rightarrow \{q\}$ où le conséquent de la règle contient un seul item. Soit η la confiance minimale de toutes les règles sélectives générées à partir d'un itemset donné.

$$\eta\Big(\{p_1, p_2, \ldots, p_k\}\Big) = min\Big[c\Big(\{p_2, p_3, \ldots, p_k\} \rightarrow \{p_1\}\Big), \ldots, c\Big(\{p_1, p_2, \ldots, p_{k-1}\} \rightarrow \{p_k\}\Big)\Big]$$

Dites si η est monotone, anti-monotone ou non-monotone?

NB: Faire un comparaison entre $\eta(\{A, B\})$ et $\eta(\{A, B, C\})$

Solution:

η est non-monotone, on peut le montrer en comparant la valeur de $\eta\Big(\{A, B\}\Big)$

par rapport à $\eta\Big(\{A, B, C\}\Big)$.

$$\begin{aligned}\eta\Big(\{A, B\}\Big) &= min\Big(c(A \to B), c(B \to A)\Big)\\ &= min\Big(\frac{s(A, B)}{s(A)}, \frac{s(A, B)}{s(B)}\Big)\\ &= \frac{s(A, B)}{max\Big(s(A), s(B)\Big)}\end{aligned}$$

$$\begin{aligned}\eta\Big(\{A, B, C\}\Big) &= min\Big(c(\{B, C\} \to A), c(\{A, C\} \to B), c(\{A, B\} \to C)\Big)\\ &= min\Big(\frac{s(A, B, C)}{s(B, C)}, \frac{s(A, B, C)}{s(A, C)}, \frac{s(A, B, C)}{s(A, B)}\Big)\\ &= \frac{s(A, B, C)}{max\Big(s(A, B), s(A, C), s(B, C)\Big)}\end{aligned}$$

Puisque $s(A, B, C) \leq s(A, C)$ et $max\Big(s(A, B), s(A, C), s(B, C)\Big) \leq max\Big(s(A), s(B)\Big)$, alors $\eta(\{A, B, C\})$ peut-être inférieure, supérieure ou égale à $\eta(\{A, B\})$.

Chapitre 4 Clustering

La segmentation, aussi appelée classification non supervisé, se rapporte à la catégorisation d'un ensemble d'objets de données dans des clusters. Un cluster est une collection d'objets de données similaires les uns aux autres dans le même segment et différents des objets dans d'autres segments.

4.1 Problématique

4.1.1 Définition

Soit $\mathcal{P}$ une polpulation d'instances de données à N attributs, trouver un partitionnement en K clusters $\{C_1, C2, \ldots C_K\}$ de $\mathcal{P}$ tel que:

$$\bigcup_{k=1}^{K} C_k = \mathcal{P}$$

Où les clusters C_k soient:

1. Homogènes que possible (similaires au sein d'un même groupe).
2. Distincts que possible (dissimilaires quand ils appartiennent à des groupes différents).

Note

- *K peut être donné, ou "découvert".*
- *Le clustering est plus difficile que la classification car les classes ne sont pas connues à l'avance.*

Une bonne méthode de clustering produira des clusters d'excellente qualité avec:

- Similarité intra-classe importante.
- Similarité inter-classe faible.

4.1.2 Qualité d'un clustering

La qualité d'une méthode de clustering est évaluée par son abilité à découvrir certains ou tous les "pattern" cachés. Elle dépend de:

- La mesure de similarité utilisée.
- L'implémentation de la mesure de similarité.

Pour cela, il faut faire un bon choix de cette mesure de similarité suivant a nature des attributs.

4.2 Distance et Dissimilarité

4.2.1 Définition

Définition 4.1. Distance

On appelle distance sur un ensemble E, une application $d : E \times E \leftarrow \mathbb{R}^+$ telle que:

1. *Séparation:* $\forall (x, y) \in E^2 : d(x, y) = 0$ *ssi* $x = y$
2. *Symétrie:* $\forall (x, y) \in E^2 : d(x, y) = d(y, x)$
3. *Inégalité triangulaire:* $\forall (x, y, z) \in E^3 : d(x, z) \leq d(x, y) + d(y, z)$

Note

- *Une dissimilarité est une application qui a les propriétés de la distance sauf éventuellement l'inégalité triangulaire.*

4.2.2 Choix d'une Distance

4.2.2.1 Données numériques

Souvent, les distances classiques sont utilisées: Soient $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$

- Distance **Euclidienne**:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Distance de **Minkowski** (généralisation de la distance euclidienne):

$$(x, y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q}$$

  Avec q entier positif non nul.

- Distance de **Manhattan** (Cas particuer $q = 1$):

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

En pratique, On ai amené à normaliser ou standardiser les valeurs des données pour égaliser le poids des variables pour assurer l'indépendance par rapport aux unités de mesures:

- Calculer l'écart absolu moyen:

$$S_f = \frac{1}{n} \left(|x_{1f} - M_f| + |x_{2f} - M_f| + \cdots + |x_{nf} - M_f| \right)$$

où

$$M_f = \frac{1}{n}(x_{1f} + x_{2f} + \cdots + x_{nf})$$

- Calculer la mesure standardisée (z-score):

$$z_{if} = \frac{x_{if} - M_f}{S_f}$$

Exemple 4.1

On veut calculer les distances $d(P1, P2)$ et $d(P1, P3)$:

	Age	Salaire
P1	50	11000
P2	70	11100
P3	60	11122
P4	60	11074

- Sans standardiser les données:
  - d(P1,P2)=120
  - d(P1,P3)=132

  $\Rightarrow$ Conclusion: P1 ressemble plus à P2 qu'à P3

	Age	Salaire
P1	-2	-2
P2	2	0.7
P3	0	1.3
P4	0	0

- Après avoir standardisé les données::
  - d(P1,P2)=6.7
  - d(P1,P3)=4.3

  $\Rightarrow$ Conclusion: P1 ressemble plus à P3 qu'à P2

4.2.2.2 Données binaires

Dans le cas très particulier où toutes les données sont binaires (présence, absence de caractéristiques), de nombreux indices de similarité ont été proposés en se basant sur les quantités suivantes définis pour deux objets i et j distincts avec p attributs (voir Table 4.1):

Table 4.1: Table de dissimilarité

		Objet J		
		1	0	Somme
Objet I	1	a	b	$a+b$
	0	c	d	$c+d$
	Somme	$a+c$	$b+d$	p

- a = nombre de caractères communs à i et j sur les p considérés,
- b = nombre de caractères possédés par i mais pas par j,
- c = nombre de caractères possédés par j mais pas par i,
- d = nombre de caractères que ne possèdent ni i ni j.
- Bien sûr, $a + b + c + d = p$.

Les indices de similarité ou ressemblance les plus courants sont:

1. **Coefficient de correspondance simple**: (similarité invariante, si la variable binaire est *symétrique*):
$$d(i, j) = \frac{b + c}{a + b + c + d}$$
2. **Coefficient de Jaccard**: (similarité non invariante, si la variable binaire est *asymétrique*):
$$d(i, j) = \frac{b + c}{a + b + c}$$

Exemple 4.2 Soit la table 4.2 des patients suivante:

Table 4.2: Table de patients

Nom	Fièvre	Toux	Test-1	Test-2	Test-3	Test-4
Salim	Oui	N	P	N	N	N
Karima	Oui	N	P	N	P	N
Ali	Oui	P	N	N	N	N

Calculer la distance entre patients, basée sur le coefficient de Jaccard?

$$d(Salim, Karima) = \frac{0 + 1}{2 + 0 + 1} = 0.33$$
$$d(Salim, Ali) = \frac{1 + 1}{1 + 1 + 1} = 0.67$$
$$d(Karima, Ali) = \frac{2 + 1}{1 + 2 + 1} = 0.75$$

4.2.2.3 Données énumératives

Généralisation des variables binaires, avec plus de 2 états: rouge, jaune, bleu, vert . . . etc.

- *Méthode 1*: **Correpondance simple** m: # de correspondances, p: # total de variables
$$d(i, j) = \frac{p - m}{p}$$
- *Méthode 2*: Utiliser un grand nombre de variables binaires:
  - Créer une variable binaire pour chaque modalité (ex: variable rouge qui prend les valeurs vrai ou faux).

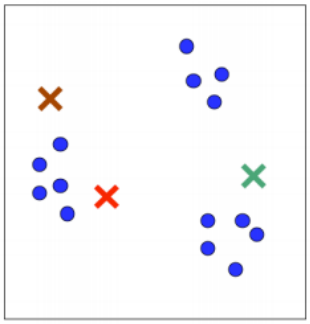
4.3 Algorithme *k-Means*

k-Means (MacQueen'67) est l'un des algorithmes de clustering les plus répandus vu sa simplicité conceptuelle et d'être très facile à comprendre et à mettre en œuvre.

En pratique l'algorithme associe chaque donnée à son centre le plus proche, afin de créer des clusters. Après avoir initialisé ses centres en prenant des données au hasard dans le jeu de données, *k-means* réitère plusieurs fois ces deux étapes pour optimiser les centres et leurs clusters jusqu'à ce que les objets ne changent plus de cluster:

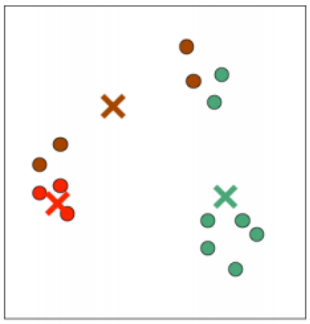
- Placer les objets dans le groupe de centre le plus proche.
- Recalculer le centre de gravité de chaque groupe.

Un exemple illustratif est donné par la Figure 4.1.

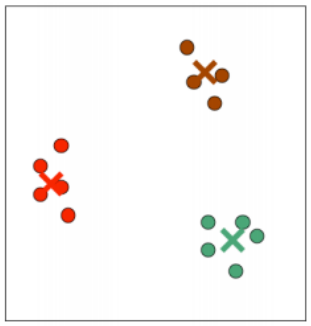


(a) Choix aléatoire de k objets, centres initiaux et calcul des clusters

(b) Calcul des centres des clusters et recalcul des clusters



(c) Calcul des centres des clusters et recalcul des clusters



(d) On stoppe lorsque les clusters sont stables

Figure 4.1: *k-means*: Illustration

Algorithme 3 : Algorithme *k-Means*

Entrées : un ensemble de m objets $x_1, \ldots, x_m$

1 Choisir k centres initiaux $c_1, \ldots, c_k$;

2 Répartir chacun des m objets dans le groupe i dont le centre c_i est le plus proche.;

3 Si aucun élément ne change de groupe alors arrêt et sortir les groupes;

4 Calculer les nouveaux centres: pour tout i, c_i est la moyenne des éléments du groupe i.;

5 Aller en 2.;

Exemple 4.3

- On a un ensemble de 8 points $A, B, \ldots, H$ de l'espace euclidien 2D. On fixe $k = 2$ (2 groupes)
- On tire aléatoirement 2 centres: B et D choisi.

Table 4.3: Exemple d'application de *k-means*

Point	Centre B(2,2) D(2,4)	Centre D(2,4) I(27/7,17/7)	Centre J(7/4,12/4) K(22/4,9/4)
A(1,3)	B	D	J
B(2,2)	B	D	J
C(2,3)	B	D	J
D(2,4)	D	D	J
E(4,2)	B	I	K
F(5,2)	B	I	K
G(6,2)	B	I	K
H(7,3)	B	I	K

4.4 Exercices

Exercice 1

Soit l'ensemble D des entiers suivants:

$$D = \{2, 5, 8, 10, 11, 18, 20\}$$

On veut répartir les données de D en trois (3) clusters, en utilisant l'algorithme *k-means*. La distance d entre deux nombres a et b est la valeur absolue de a moins b calculée ainsi:

$$d(a, b) = |a - b|$$

(a) Appliquez *k-means* en choisissant comme centres initiaux des 3 clusters respectivement: 8, 10 et 11. Montrez toutes les étapes de calcul.

(b) Donnez le résultat final et précisez le nombre d'itérations qui ont été nécessaires.

(c) Peut-on avoir un nombre d'itérations inférieur pour ce problème ? Discutez.

Voir Corrigé, page 52.

Exercice 2

Soit X le tableau de données suivant:

X	1	2	9	12	20

(a) Appliquez l'algorithme *K-means* en utilisant la distance euclidienne, avec les valeurs de k et les points de départ suivants:

1. $k = 2, \mu_1 = 1, \mu_2 = 20$
2. $k = 3, \mu_1 = 1, \mu_2 = 12, \mu_3 = 20$
3. $k = 4, \mu_1 = 1, \mu_2 = 9, \mu_3 = 12, \mu_4 = 20$

(b) On aimerait maintenant comparer la qualité de ces regroupements. Pour cela, on recommence par regarder l'inertie intra-cluster $J_w = \sum_{k=1}^{K} \sum_{i \in C_k} D^2(x_i, \mu_k)$.

1. Calculer cette valeur pour les 3 regroupements précédents.
2. En utilisant ce critère, quel serait le meilleur regroupement possible ? est-ce que cela vous parait réaliste ?

Voir Corrigé, page 55.

Exercice 3

Soit les points A (2, 10) ; B (2,8) ; C (8,4) ; D (5,8) ; E (7,5) ; F (6,4) ; G (1,2) ; H (4,9)

(a) Donner la répartition géométrique de ces points. Quels sont les clusters qu'on peut identifier visuellement ?

(b) En prenant comme centroïdes initiaux les points A B et C, appliquer l'algorithme *K-means* pour regrouper les points en trois clusters (utiliser la distance de Manhattan).

(c) Est-il possible de minimiser le nombre d'itérations par un autre choix des centroides initiaux ? Justifier la réponse.

Exercice 4

Utiliser l'algorithme *k-means* et a distance euclidienne pour regroupere points suivants en 3 clusters:
A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9).
Supposons que les centres initiaux sont A1, A4 et A7. Exécutez l'algorithme *k-means* pour une seule itération. A la fin de cette étape, Montrer:

(a) Les nouveaux clusters (c'est-à-dire les points appartenant à chaque cluster).

(b) Les centres des nouveaux clusters.

(c) Dessinez un espace 10×10 avec les 8 points et montrez les clusters après la première itération et les nouveaux centroïdes.

(d) Combien d'itérations supplémentaires sont nécessaires pour converger ? Dessinez le résultat pour chaque étape.

Exercice 5

Soit Un jeu de données constitué de cinq (5) produits dont les quantités vendues dans deux régions sont indiquées ci-dessous:

	Produit	Region 1	Region 2
1	A	22	21
2	B	19	20
3	C	18	22
4	D	1	3
5	E	4	2

(a) Regroupez ces produits en deux groupes en utilisant l'algorithme *k-means*, la distance euclidienne, et les produits A et E comme centres initiaux.

(b) Regroupez ces produits en deux groupes en utilisant l'algorithme *k-means*, la distance de Manhattan, et les produits A et E comme centres initiaux.

Chapitre 4: corrigé des exercices

Exercice 1

Soit l'ensemble D des entiers suivants:

$$D = \{2, 5, 8, 10, 11, 18, 20\}$$

On veut répartir les données de D en trois (3) clusters, en utilisant l'algorithme *k-means*. La distance d entre deux nombres a et b est la valeur absolue de a moins b calculée ainsi:

$$d(a, b) = |a - b|$$

(a) Appliquez *k-means* en choisissant comme centres initiaux des 3 clusters respectivement: 8, 10 et 11. Montrez toutes les étapes de calcul.

Solution:

- Initialisation des centres des clusters: $c_1 = 8$, $c_2 = 10$, $c_3 = 11$.
  $C_1 = \emptyset$, $C_2 = \emptyset$, $C_3 = \emptyset$
- Itération 1:
  - Calcul des distances:

Nombre	Distances	Affectation
2	$d(2, c_1) = \|2 - 8\| = 6$ $d(2, c_2) = \|2 - 10\| = 8$ $d(2, c_3) = \|2 - 11\| = 9$	C_1
5	$d(5, c_1) = \|5 - 8\| = 3$ $d(5, c_2) = \|5 - 10\| = 5$ $d(5, c_3) = \|5 - 11\| = 6$	C_1
8	$d(8, c_1) = \|8 - 8\| = 0$ $d(8, c_2) = \|8 - 10\| = 2$ $d(8, c_3) = \|8 - 11\| = 3$	C_1
10	$d(5, c_1) = \|10 - 8\| = 2$ $d(5, c_2) = \|10 - 10\| = 0$ $d(5, c_3) = \|10 - 11\| = 1$	C_2
11	$d(11, c_1) = \|11 - 8\| = 3$ $d(1, c_2) = \|11 - 10\| = 1$ $d(11, c_3) = \|11 - 11\| = 0$	C_3
18	$d(18, c_1) = \|18 - 8\| = 10$ $d(18, c_2) = \|18 - 10\| = 8$ $d(18, c_3) = \|18 - 11\| = 7$	C_3
20	$d(20, c_1) = \|20 - 8\| = 12$ $d(20, c_2) = \|20 - 10\| = 10$ $d(20, c_3) = \|20 - 11\| = 9$	C_3

  - Mise à jour des clusters: $C_1 = \{2, 5, 8\}$, $C_2 = \{10\}$, $C_3 =$

$\{11, 18, 20\}$

- Recalcul des centres des clusters: $c_1 = (2+5+8)/3 = 5$, $c_2 = 10$, $c_3 = (11 + 18 + 20)/3 = 16.33$.

- Itération 2:

  - Calcul des distances:

Nombre	Distances	Affectation
2	$d(2, c_1) = \|2 - 5\| = 3$ $d(2, c_2) = \|2 - 10\| = 8$ $d(2, c_3) = \|2 - 16.33\| = 14.33$	C_1
5	$d(5, c_1) = \|5 - 8\| = 3$ $d(5, c_2) = \|5 - 10\| = 5$ $d(5, c_3) = \|5 - 11\| = 6$	C_1
8	$d(8, c_1) = \|8 - 5\| = 3$ $d(8, c_2) = \|8 - 10\| = 2$ $d(8, c_3) = \|8 - 16.33\| = 8.33$	C_2
10	$d(10, c_1) = \|10 - 5\| = 5$ $d(10, c_2) = \|10 - 10\| = 0$ $d(10, c_3) = \|10 - 16.33\| = 6.33$	C_2
11	$d(11, c_1) = \|11 - 5\| = 6$ $d(11, c_2) = \|11 - 10\| = 1$ $d(11, c_3) = \|11 - 16.33\| = 5.33$	C_2
18	$d(18, c_1) = \|18 - 5\| = 3$ $d(18, c_2) = \|18 - 10\| = 2$ $d(18, c_3) = \|18 - 16.33\| = 81.67$	C_3
20	$d(20, c_1) = \|20 - 5\| = 5$ $d(20, c_2) = \|20 - 10\| = 10$ $d(20, c_3) = \|20 - 16.33\| = 3.67$	C_3

  - Mise à jour des clusters: $C_1 = \{2, 5\}$, $C_2 = \{8, 10, 11\}$, $C_3 = \{18, 20\}$
  - Recalcul des centres des clusters: $c_1 = (2 + 5)/2 = 3.5$, $c_2 = (8 + 10 + 11)/3 = 9.66$, $c_3 = (18 + 20)/2 = 19$.

- Itération 3:

  - Calcul des distances:

Nombre	Distances	Affectation
2	$d(2, c_1) = \|2 - 3.5\| = 1.5$ $d(2, c_2) = \|2 - 9.66\| = 7.66$ $d(2, c_3) = \|2 - 19\| = 17$	C_1
5	$d(5, c_1) = \|5 - 3.5\| = 1.5$ $d(5, c_2) = \|5 - 9.66\| = 4.66$ $d(5, c_3) = \|5 - 19\| = 14$	C_1
8	$d(8, c_1) = \|8 - 3.5\| = 4.5$ $d(8, c_2) = \|8 - 9.66\| = 1.66$ $d(8, c_3) = \|8 - 19\| = 11$	C_2
10	$d(10, c_1) = \|10 - 3.5\| = 6.5$ $d(10, c_2) = \|10 - 9.66\| = 0.34$ $d(10, c_3) = \|10 - 19\| = 9$	C_2
11	$d(11, c_1) = \|11 - 3.5\| = 7.5$ $d(11, c_2) = \|11 - 9.66\| = 1.34$ $d(11, c_3) = \|11 - 19\| = 8$	C_2
18	$d(18, c_1) = \|18 - 3.5\| = 14.5$ $d(18, c_2) = \|18 - 9.66\| = 8.34$ $d(18, c_3) = \|18 - 19\| = 1$	C_3
20	$d(20, c_1) = \|20 - 3.5\| = 16.5$ $d(20, c_2) = \|20 - 9.66\| = 10.34$ $d(20, c_3) = \|20 - 19\| = 1$	C_3

- Mise à jour des clusters: $C_1 = \{2, 5\}$, $C_2 = \{8, 10, 11\}$, $C_3 = \{18, 20\}$
- Recalcul des centres des clusters: $c_1 = (2 + 5)/2 = 3.5$, $c_2 = (8 + 10 + 11)/3 = 9.66$, $c_3 = (18 + 20)/2 = 19$.
- Stabilité: Les centres n'ont pas changé. L'algorithme s'arrête.

(b) Donnez le résultat final et précisez le nombre d'itérations qui ont été nécessaires.

Solution:

- Les clusters résultats: $C_1 = \{2, 5\}$, $C_2 = \{8, 10, 11\}$ et $C_3 = \{18, 20\}$
- Nombre d'itérations = 3

(c) Peut-on avoir un nombre d'itérations inférieur pour ce problème ? Discutez.

Solution:

Dans ce problème, les données sont ordonnées et restreintes dans un intervalle (de 2 à 20). Comme on veut construire 3 clusters, on est sûr que la borne inférieure (2) sera dans le cluster 1, et la borne supérieure (20) sera dans le

cluster 3. Il est donc intéressant de choisir comme centres initiaux: $c_1 = 2$, $c_3 = 20$ et $c_2 = 9$. Avec une telle initialisation, l'algorithme convergera après seulement 2 itérations.

Exercice 2

Soit X le tableau de données suivant:

X	1	2	9	12	20

(a) Appliquez l'algorithme *K-means* en utilisant la distance euclidienne, avec les valeurs de k et les points de départ suivants:

1. $k = 2$, $\mu_1 = 1$, $\mu_2 = 20$

Solution:

- Itération 1:

	1	2	9	12	20	
$d(x, \mu_1)$	**0**	**1**	**64**	121	361	$\mu_1 = (1+2+9)/3 = 4$
$d(x, \mu_2)$	361	324	121	**64**	**0**	$\mu_2 = (12+20)/2 = 16$

- Itération 2:

	1	2	9	12	20	
$d(x, \mu_1)$	**9**	**4**	**25**	64	256	$\mu_1 = (1+2+9)/3 = 4$
$d(x, \mu_2)$	225	196	49	**16**	**16**	$\mu_2 = (12+20)/2 = 16$

μ_1 et μ_2 ne changent pas $\Rightarrow$ convergence

2. $k = 3$, $\mu_1 = 1$, $\mu_2 = 12$, $\mu_3 = 20$

Solution:

- Itération 1:

	1	2	9	12	20	
$d(x, \mu_1)$	**0**	**1**	64	121	361	$\mu_1 = (1+2)/2 = 1.5$
$d(x, \mu_2)$	121	100	**9**	**0**	64	$\mu_2 = (9+12)/2 = 10.5$
$d(x, \mu_3)$	361	324	121	64	**0**	$\mu_3 = 20$

- Itération 2:

	1	2	9	12	20	
$d(x, \mu_1)$	**0.25**	**0.25**	56.25	110.25	342.25	$\mu_1 = (1+2)/2 = 1.5$
$d(x, \mu_2)$	90.25	72.25	**2.25**	**2.25**	90.25	$\mu_2 = (9+12)/2 = 10.5$
$d(x, \mu_3)$	361	324	121	64	**0**	$\mu_3 = 20$

μ_1, μ_2 et μ_3 ne changent pas $\Rightarrow$ convergence

3. $k = 4$, $\mu_1 = 1$, $\mu_2 = 9$, $\mu_3 = 12$, $\mu_4 = 20$

Solution:

- Itération 1:

	1	2	9	12	20	
$d(x,\mu_1)$	**0**	**1**	64	121	361	$\mu_1 = (1+2)/2 = 1.5$
$d(x,\mu_2)$	64	49	**0**	9	121	$\mu_2 = 9$
$d(x,\mu_3)$	121	100	9	**0**	64	$\mu_3 = 12$
$d(x,\mu_4)$	361	324	121	64	**0**	$\mu_4 = 20$

- Itération 2:

	1	2	9	12	20	
$d(x,\mu_1)$	**0.25**	**0.25**	56.25	110.25	342.25	$\mu_1 = (1+2)/2 = 1.5$
$d(x,\mu_2)$	64	49	**0**	9	121	$\mu_2 = 9$
$d(x,\mu_3)$	121	100	9	**0**	64	$\mu_3 = 12$
$d(x,\mu_4)$	361	324	121	64	**0**	$\mu_4 = 20$

μ_1, μ_2 , μ_3 et μ_4 ne changent pas $\Rightarrow$ convergence

(b) On aimerait maintenant comparer la qualité de ces regroupements. Pour cela, on recommence par regarder l'inertie intra-cluster $J_w = \sum_{k=1}^{K} \sum_{i \in C_k} D^2(x_i, \mu_k)$.

1. Calculer cette valeur pour les 3 regroupements précédents.

Solution:

$$J_w(k=2) = 9 + 4 + 25 + 16 + 16 = 70$$

$$J_w(k=3) = 0.25 + 0.25 + 2.25 + 2.25 + 0 = 5$$

$$J_w(k=4) = 0.25 + 0.25 + 0 + 0 + 0 = 0.5$$

2. En utilisant ce critère, quel serait le meilleur regroupement possible ? est-ce que cela vous parait réaliste ?

Solution:

En poursuivant le raisonnement, avec un regroupement en 5 clusters (1 par point), on obtient $J_w(k=5) = 0$. Ce critère ne nous permet donc pas de trouver le meilleur regroupement possible.

Chapitre 5 R et fouille de données

5.1 Introduction en R

5.1.1 Démarrer R

R est un logiciel pour l'analyse statistique. C'est un logiciel libre ; il est disponible gratuitement et tourne sur différent système (PC Linux, PC Windows, Mac).

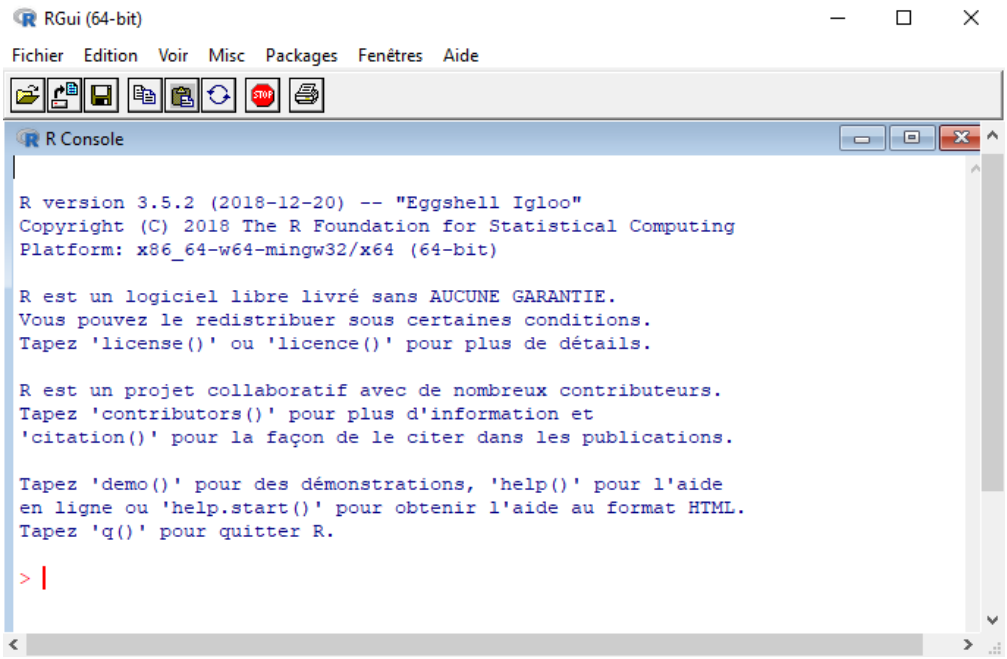


Figure 5.1: Ecran d'acceuil du logiciel R

- Vous pouvez le télécharger ici: http://cran.r-project.org/
- Le signe > en début de ligne est l'invite de commande de R.
- Pour quitter R: > `q()`

Note *Lorsque l'on utilise R, il est possible de reprendre la ligne de commande tapée précédemment en appuyant sur la flèche du haut du clavier. Appuyer plusieurs fois permet de récupérer des lignes plus anciennes.*

5.1.2 Syntaxe et manipulation de données

5.1.2.1 Commentaires

Dans R, tout ce qui suit le caractère # (dièse) est un commentaire et n'est pas pris en compte par R:

```
>#  Ceci est du baratin qui n'est pas pris en compte par R !
```

5.1.2.2 variables

R manipule que les tableaux de données. Ces tableaux sont stockés dans des *variables*, ce qui permet de leur donner un nom.

- Le nom des variables doit commencer par une lettre et peut contenir des lettres, des chiffres, des points et des caractères de soulignement (_), mais surtout pas d'espace.
- L'opérateur = est utilisé pour donner une valeur à une variable; il peut se lire *prend la valeur de*
- on trouve aussi l'opérateur <- qui a exactement la même signification.

```
> age = 28
```

- Pour afficher la valeur de la variable âge, il suffit de taper le nom de la variable:

```
> age
[1] 28
```

5.1.2.3 Types de donnée

R peut manipuler des nombres entiers, des flottants (= nombres à virgule), des chaînes de caractère et des booléens (valeur vraie ou fausse):

```
> age = 28 # Entier
> poids = 64.5 # Flottant
> nom = "Ngo␣bo" # Chaine de caractere
> enseignant = TRUE # booleen
> etudiant = FALSE # booleen
> telephone = "01␣48␣38␣73␣34" # Chaine de caractere
```

Note *La valeur spéciale NA (not available) est utilisée lorsqu'une donnée est manquante.*

5.1.2.4 Vecteurs

Un vecteur est un tableau à une dimension. Toutes les cases du vecteur doivent contenir des données du même type (des entiers, des chaînes de caractère,...).

La fonction `c()` permet de créer un vecteur:

```
> ages = c(28, 25, 23, 24, 26, 23, 21, 22, 24, 29, 24, 26, 31, 28, 27,
    24, 23, 25, 27, 25, 24, 21, 24, 23, 25, 31, 28, 27, 24, 23)
> ages
[1] 28 25 23 24 26 23 21 22 24 29 24 26 31 28 27 24 23
[17] 25 27 25 24 21 24 23 25 31 28 27 24 23
```

Note

- *[1] indique que ce qui suit est la première valeur du vecteur.*
- *[17] que la deuxième ligne commence à la 17ème valeur.*
- *[1] et **[17]** ne sont pas des éléments du vecteur.*

La fonction `c()` permet aussi de concaténer (mettre bout à bout) des vecteurs:

```
> poids_groupe_temoin = c(75.0, 69.2, 75.4, 87.3)
> poids_groupe_intervention = c(70.5, 64.2, 76.4, 81.6)
> poids = c(poids_groupe_temoin, poids_groupe_intervention)
> poids
[1] 75.0 69.2 75.4 87.3 70.5 64.2 76.4 81.6
```

Il est possible d'accéder à un élément du vecteur avec des crochets. Par exemple pour accéder au second élément du vecteur poids:

```
> poids[2]
[1] 69.2
```

Il est aussi possible d'accéder à l'ensemble des poids répondant à une condition, par exemple l'ensemble des poids supérieurs à 70:

```
> poids[poids > 70.0]
[1] 75.0 75.4 87.3 70.5 76.4 81.6
```

Enfin, la fonction `length()` permet de récupérer le nombre d'éléments d'un tableau:

```
> length(poids)
[1] 8
> length(poids[poids > 70.0])
[1] 6
```

Il est possible de créer un vecteur contenant une suite de nombres entiers avec la fonction `seq`:

```
> seq(1, 10)
```

```
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1, 10, by = 0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0
[14] 7.5 8.0 8.5 9.0 9.5 10.0
```

5.1.2.5 Matrices

Une matrice est un tableau à deux dimensions, toutes les cases doivent contenir des données du même type.

Une matrice est créée à partir d'un vecteur contenant les valeurs, et d'un nombre de ligne (nr, pour Number of Row) et de colonne (nc, pour Number of Column):

```
> ma_matrice = matrix(c(1.5, 2.1, 3.2, 1.6, 1.4, 1.5),nr=3, nc=2)
> ma_matrice
    [,1]  [,2]
[1,]  1.5   1.6
[2,]   2.1   1.4
[3,]   3.2   1.5
```

Les éléments de la matrice peuvent être accédé en donnant entre crochets le numéro de la ligne puis celui de la colonne:

```
> ma_matrice[1, 1]
[1] 1.5
```

Il est aussi possible de récupérer une ligne ou une colonne entière, en omettant le numéro correspondant:

```
> ma_matrice[1,]
[1] 1.5 1.6
```

5.1.2.6 Listes, Data frames

Une liste est un tableau à une dimension, qui peut contenir des données de différents types (contrairement au vecteur).

```
> ma_liste = list("JB", 28)
```

Un tableau de donnée est un tableau où chaque colonne correspond à un attribut différents (âge, taille, poids par exemple) et chaque ligne à un individu différent.

Il est possible de créer des tableaux de données dans R, cependant il est beaucoup plus facile de les charger à partir d'un fichier. On utilise pour cela la fonction `read.table()`. Voici le fichier **taille_poids.csv**:

```
nom,taille,poids
Ngo bo,1.70,64.0
Bo bo,1.80,63.0
M. X,1.67,70.5
M. Y,1.69,95.0
M. Z,1.75,NA
```

Ce fichier peut être chargé ainsi dans **R**:

```
> t = read.table("taille_poids.csv", sep=",", header=TRUE)
```

Les noms des colonnes sont disponibles via la fonction `names()`:

```
> names(t)
[1] "nom" "taille" "poids"
```

Comme pour les matrices, il est possible d'accéder aux cases, lignes et colonnes d'un tableau. Les noms des colonnes peuvent être utilisés à la place de leurs index:

```
> t[1, 2] # taille du premier individu
[1] 1.7
> t[1, "taille"] # Pareil
[1] 1.7
> t[1,] # Premiere ligne
nom    taille  poids
1 Ngo bo 1.70  64.0
> t[,"taille"] # Colonne taille
[1] 1.70 1.80 1.67 1.69 1.75
> t$taille # Notation raccourci pour la colonne taille
[1] 1.70 1.80 1.67 1.69 1.75
```

Il est aussi possible d'indexer avec une condition: par exemple, pour obtenir un tableau avec seulement les individus dont la taille est supérieure à 1m70 (ne pas oublier la virgule, qui sert à indiquer que l'on veut récupérer toutes les colonnes !):

```
> t[t\$taille > 1.7,]
    nom taille poids
  2 Bo bo 1.80 63
  5 M. Z 1.75  NA
```

5.1.2.7 Opérateurs et calcul

R permet de réaliser la plupart des opérations courantes à l'aide des opérateurs suivants: + (addition), - (soustraction), * (multiplication), / (division), ˆ(puissance).

```
> 2 * 3 + 1
[1] 7}
```

Les opérations sont réalisées sur chaque élément des tableaux. Par exemple, pour calculer l'Indice de Masse Corporelle (IMC) sur chaque individu du tableau de donnée chargé précédemment, en appliquant la formule $IMC = \frac{poid}{taille^2}$:

```
> t$poids / (t$taille\^2)
[1] 22.14533 19.44444 25.27878 33.26214 NA
```

Il est possible d'ajouter une quatrième colonne avec l'IMC à notre tableau de la manière suivante:

```
> t\$IMC = t$poids / (t$taille^2)
> t
  nom    taille poids  IMC
1 Ngo bo  1.70 64.0 22.14533
2 Bo bo  1.80  63.0  19.44444
3 M. X     1.67  70.5  25.27878
4 M. Y     1.69  95.0  33.26214
5 M. Z     1.75  NA   NA
```

5.1.2.8 Comparaison

Les comparaisons se font avec les opérateurs **<**, **>**, **<=** (inférieur ou égal) , **>=** (supérieur ou égal), **==** (égal), **!=** (différent de). Ils retournent une (ou plusieurs) valeur(s) booléenne(s).

```
> 1 < 3
[1] TRUE
> t\$IMC > 25
1] FALSE FALSE TRUE TRUE NA
```

Note *Il est possible de combiner plusieurs comparaisons avec des* **&** *(et) ou des* **|** *(ou)*

5.1.2.9 Fonctions

R définit un grand nombre de fonctions; nous en avons déjà vu quelques unes. La fonction **help()** permet d'obtenir de l'aide sur une fonction:

```
>help(mean)}
```

Voici une liste des principales fonctions:

- **summary(tableau)** affiche un résumé du tableau.
- **length(vecteur)** retourne le nombre de case d'un vecteur.
- **ncol(tableau)** retourne le nombre de colonne d'un tableau.
- **nrow(tableau)** retourne le nombre de ligne d'un tableau.
- **q()** quitte R
- **min(vecteur)** retourne la plus petite valeur d'un vecteur.
- **max(vecteur)** retourne la plus grande valeur d'un vecteur.
- **sort(vecteur)** retourne une copie d'un vecteur après l'avoir trié.
- **mean(vecteur)** calcule la moyenne.
- **median(vecteur)** calcule la médianne.
- **var(vecteur)** calcule la variance.
- **sd(vecteur)** calcule la déviation standard.
- **sqrt(nombre)** retourne la racine carré d'un nombre.
- **sum(vecteur)** retourne la somme de toutes les valeurs du vecteur.
- **round(flottant)** arrondit un nombre.

5.1.3 Structures de contrôle et Fonctions

5.1.3.1 Structures de contrôle

Structures de contrôle sont des commandes qui contrôlent l'ordre dans lequel les différentes instructions d'un programme informatique sont exécutées.

- Alternatives (énoncés conditionnels): **if ... else**,
- Boucles (énoncés itératifs): **for**, **while**, etc.

Les alternatives ont pour but d'exécuter des instructions seulement si une certaine condition est satisfaite, elles ont la forme d'écriture générale suivante:

```
if (condition) {
  instructions
  } else {
    instructions
```

```
  }
}
```

Exemple 5.1

```
  if (x>0) y=x*log(x) else y=0
```

Les boucles ont pour but de répéter des instructions à plusieurs reprises. Elles ont plusieurs formes d'écriture générale:

```
for (i in ensemble ) {
  instructions
}
```

Exemple 5.2

```
for (i in 1:10 ) {
print(i)
}
```

```
while (condition ) {
instructions
}
```

Exemple 5.3

```
  while (i <11 ) {
  print(i)
  i=i+1
  }
```

```
repeat {
instructions
if (condition) break
}
```

Exemple 5.4

```
  repeat { print(i)
  if (i<10) { i=i+1} else {break}
  }
```

5.1.3.2 Fonctions

Pour créer une fonction en R, on utilise la fonction nommée `function` en respectant la syntaxe suivante:

```
nomFonction <- function(arg1, arg2, arg3) {
# corps de la fonction
}
```

- Les accolades { et } définissent le début et la fin de la fonction.
- La dernière instruction contient le nom de l'objet retourné par la fonction.
- Exécuter la fonction: `myname(...)`
- On peut donner des valeurs par défaut aux paramètre

Exemple 5.5

```
exemple=function(n=10)
{   sample=runif(n)
  m = mean(sample); v =var(sample)
  list(serie=sample,moyenne=m, variance=v)
}
```

Les trois commandes suivantes: `exemple(10)` ou `exemple(n=10)` ou `exemple()` retournent le même résultat.

5.2 Arbres de décision

5.2.1 Installation et chargement

1. Installer le package rpart:

```
> install.packages("rpart", dep=TRUE)
```

2. Charger le package rpart:

```
> library(rpart)
```

3. Charger les données kyphosis:

```
> data(kyphosis)
```

   - La variable Kyphosis indique si l'enfant a subi une déformation de la colonne vertébrale après une opération chirurgicale;
   - La variable Age donne l'âge de l'enfant (en mois);

- La variable Number indique le nombre de vertèbres concernées par l'opération;
- La variable Start indique le numéro de la première vertèbre opérée.

5.2.2 Construction de l'arbre

On peut se demander si l'âge de l'enfant, le nombre de vertèbres opérées et la position des vertèbres opérées permettent d'évaluer le risque de survenue d'une cyphose à l'issue de l'opération. On va construire un modèle d'arbre de décision grâce à la commande:

```
arbre = rpart(Kyphosis ~., method="class", minsplit=20, xval=81, data=
    kyphosis)
```

- **arbre** est le nom choisi pour l'objet R qui contiendra les résultats;
- **Kyphosis** ~: La **tilde** signifie " est à expliquer en fonction de ": on met donc à gauche de la tilde la variable que l'on souhaite expliquer (ici, Kyphosis) et à droite les variables explicatives séparées par des **+**;
- **method = "class"**: optionnel, mais il permet de rappeler à R que la variable à expliquer est qualitative;
- **minsplit=20** signifie qu'il faut au moins 20 individus dans un nœud pour tenter un split;
- l'argument **xval** permet de régler la stratégie de validation croisée;
- L'argument **data=kyphosis** indique le nom du jeu de données utilisé.

5.2.3 Affichage de l'arbre

Les commandes suivantes permettent d'afficher l'arbre de décision sous forme graphique:

```
> plot(arbre, uniform=TRUE, margin=0.1, main="Decision␣Tree")
> text(arbre, fancy=TRUE, use.n=TRUE, pretty=0, all=TRUE)
```

5.2.4 Elagage de l'arbre

On peut souhaiter élaguer l'arbre de décision. Pour cela, il est conseillé d'exécuter la commande `plotcp(arbre)` pour déterminer la taille optimale (ou `printcp(arbre)`).

```
> arbre2 = rpart(Kyphosis ~., cp=0.059, data=kyphosis)
```

5.2.5 Prédiction

On peut être conduit à appliquer ultérieurement ces règles sur de nouveaux enfants qui vont être opérés, de manière à estimer avant l'opération leur risque de développer une cyphose. On construit arbitrairement un tableau de données correspondant à trois enfants sur le point d'être opérés:

```
> inco = data.frame(c(60,30,90), c(1,2,3), c(5,12,16))
> colnames(inco) = c("Age", "Number", "Start")
```

La prédiction de la variable Kyphosis pour ces trois individus s'opère alors par la commande `predict` (dont le premier argument est le nom du modèle utilisé, et le second, newdata, est le nom du data frame contenant les nouveaux individus):

```
> predict(arbre2, newdata=inco, type="class")
```

5.3 Règles d'association

5.3.1 Package arules

Nous allons utiliser une implantation en R de l'algorithme Apriori disponible dans le package arules de R.

1. Installer le package **arules** via la commande:

   ```
   > install.packages("arules", dep=TRUE)
   ```

2. Charger le package **arules**:

   ```
   > library(arules)
   ```

Note *Il peut être nécessaire d'installer aussi le package **Matrix**.*

5.3.2 Jeu de transactions

1. Nous allons travailler sur le jeu de transaction **Adult**:

   ```
   data("Adult")
   inspect(Adult[1:2])
   ```

   - Combien y a-t-il de transactions ?
   - Combien y a-t-il d'items ?
   - Quel est la longueur des transactions ?

- Pourquoi certaines transactions ont-elles moins de 13 items ?

2. Regardez l'aide sur les fonctions `itemFrequency()` et `itemFrequencyPlot()`.
   - Affichez le graphique des items de support supérieur à 0.2.
   - Affichez le graphique des 10 items les plus fréquents.

5.3.3 La fonction apriori()

1. Regardez l'aide sur cette fonction `apriori()`.
2. Par exemple pour obtenir les règles ayant un support d'au moins 1% et une confiance supérieure à 60%, il suffit de lancer la commande:

```
rules<-apriori(Adult,parameter=list(support=0.01,confidence=0.6))
```

3. Comprenez les messages affichés lors de l'exécution de l'algorithme. Vous pouvez utiliser les fonctions `summary()` et `inspect()` pour répondre aux questions suivantes:
   - Combien de règles ?
   - Pour combien de transactions ?
   - Affichez les dix premières règles obtenues.
4. Diminuez la valeur de la confiance (par exemple 0.1) et du support (par exemple 0.001, 0.0001, 0.00001) et regardez l'évolution du nombre de règles.

5.4 Clustering

5.4.1 Package cluster

Nous allons utiliser une implantation en R de l'algorithme k-moyennes disponible dans le package cluster de R.

1. Installer le package **cluster** via la commande:

```
> install.packages("cluster", dep=TRUE)
```

2. Charger le package **cluster**:

```
> library(cluster)
```

3. Regardez l'aide sur cette fonction `kmeans()`. Que fournit cette commande ?
4. Visualisez les données clustering fournies avec ce TP, combien de clusters pensez-vous observer ?
5. Visualiser le résultat de la segmentation avec k=2, à l'aide de couleurs.

5.4.2 Trouver K optimal

On s'intéresse maintenant à trouver le nombre de groupes optimal. Pour cela, on va essayer plusieurs valeurs de K qui semblent raisonnables.

1. Essayez toutes les valeurs entre 2 et 10.
2. Placez ces segmentations dans une liste.
3. Calculez l'inertie intraclasse de ces 9 segmentations.
4. Faites-en un graphe.
5. Quelle segmentation est la meilleure?

Références

1. Han, J., Kamber, M. & Pei, J. (2012). Data mining concepts and techniques, *Third edition Morgan Kaufmann Publishers*.
2. T-Larose, D. (2005). Des données à la connaissance, une introduction au data-mining. *Vuibert edition, Paris*, 2005.
3. M, Jambu. (1999). Introduction au Data Mining, Analyse intelligente des données, *Eyrolles edition*.
4. Ph. Preux, (2009). Fouille de données, Notes de cours, *Université de Lille 3*.
5. Agrawal, R. & Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of the 20th International Conference on Very Large Data Bases (pp. 487–499)*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.. ISBN: 1-55860-153-8
6. Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine, 17(3), 37*.
7. Quinlan, J.R. (1986) Induction of Decision Trees. *Machine Learning*, 1, 81-106.
8. MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1, 281-297.
9. Systèmes Zucker, J. (2012). Intelligents & Multimédia TP. `http://cours.zucker.fr/R/TDTP-R.html`