UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE
APPLIQUÉES

PAR

MEDARD SERIA

ANALYSE DU PANIER DE CONSOMMATION : RÈGLES D'ASSOCIATION, MÉTHODES ENSEMBLISTES ET AUTOENCODEURS

Juillet 2024

Résumé

Ce travail traite de la problématique de conception du système de recommandation de produit en faisant l'analyse du panier de consommation. En effet, l'analyse du panier de consommation est une technique de modélisation basée sur la théorie selon laquelle si vous achetez un certain groupe d'éléments, vous êtes plus susceptible d'acheter un autre groupe d'éléments. Il existe de nombreux algorithmes pour modéliser les données des transactions afin de prédire les produits qu'un client achètera à nouveau ou essayera pour la première fois.

L'extraction de règles d'association en utilisant Apriori ou FP-Growth est l'une des approches pour trouver des modèles dans les données de transactions. Cependant, les règles d'association peuvent être influencées par la taille des ensembles de données et ne prennent pas en compte les occurrences des produits en fonction des clients. Ainsi, pour capturer des relations complexes entre les variables d'entrée (historiques d'achats, caractéristiques d'éléments, préférences des clients, etc.), nous avons utilisé l'approche ensembliste de type XGBoost pour prédire avec fiabilité le prochain article susceptible d'être acheté. En complément de ces algorithmes, nous avons ajouté un autoencodeur débruiteur pour extraire les caractéristiques dans les données des clients, puis nous l'avons combiné à l'algorithme K-means pour regrouper les clients en fonction de leurs caractéristiques similaires.

L'objet de notre recherche est d'explorer des solutions possibles pour trouver le meilleur modèle afin de construire un système de recommandation de produits efficace et personnalisé. Pour ce faire, nous avons utilisé des données de 3 millions de commandes enregistrées sur la place de marché en ligne Instacart, rendues libres d'utilisation. Nos expérimentations démontrent que le modèle fiable à utiliser pour la prédiction de prochain produit à acheter est l'approche ensembliste d'arbre de décision

avec la variante XGBoost, et la combinaison de l'autoencodeur débruiteur et k-means est efficace pour la segmentation des clients en groupes homogènes.

Remerciements

Je tiens à exprimer mes sincères remerciements à mes collègues du laboratoire d'intelligence artificielle appliquée et à mes camarades qui ont partagé leurs idées, leurs connaissances et leur soutien au cours de ces années d'études.

Je souhaite exprimer ma profonde gratitude envers mon directeur de mémoire, M. Ismaïl Biskri. Ses conseils éclairés, sa patience et son dévouement ont été d'une valeur inestimable.

Enfin, je remercie ma famille et mes amis pour leur amour, leur soutien inconditionnel et leurs encouragements tout au long de ce parcours académique exigeant. Leurs mots d'encouragement ont été une source constante de motivation.

Merci à tous ceux qui ont contribué, de près ou de loin, à la réalisation de ce travail.

Table des matières

Liste des tableaux

Liste des figures

Liste des algorithmes

Chapitre 1. Introduction

1.1. Mise en contexte

De nombreuses entreprises commerciales accumulent de grandes quantités de données à partir de leurs opérations quotidiennes [1]. Par exemple, d'énormes quantités de données sur les achats des clients sont collectées quotidiennement aux caisses des épiceries ou sur les plateformes de vente en ligne. Il apparait la nécessité d'explorer et d'analyser ces données pour découvrir les connaissances qui s'y cachent. L'exploration de données et l'apprentissage automatique sont largement utilisés dans les domaines tels que les assurances, les banques, le commerce en ligne, la médecine, l'astronomie, etc.

Les avancées en intelligence artificielle, ont poussé les entreprises à tirer parti de la fouille de données et de l'apprentissage automatique. Ces disciplines peuvent résoudre facilement les problèmes les plus difficiles et les plus complexes [2]. En effet, la recherche de connaissances dans un ensemble de données permet d'automatiser certaines tâches ou d'extraire de l'information enfouie dans les données , ce qui permet de développer des applications informatiques. Des applications telles que la traduction automatique et les agents conversationnels intelligents sont issues du champ de recherche dans le domaine de l'intelligence artificielle [3].

Une des tâches les plus importantes de l'exploration de données est de trouver les règles d'association et de découvrir les modèles et les relations les plus intéressants et utiles dans un large volume de données. Actuellement, il existe plusieurs algorithmes proposés pour l'extraction des règles d'association, le plus connu et le plus exhaustif étant l'algorithme Apriori proposé par R. Agrawal et R. Srikant [4]. Les règles d'association permettent de rechercher les relations entre les objets fréquemment utilisés ensemble tels que les produits achetés par un client dans une épicerie. Bien que les règles d'associations apportent un atout en termes d'analyse, dans la pratique, la taille des ensembles éléments fréquents générés tient

difficilement dans la mémoire lorsqu'il s'agit de grands ensembles de données. Les approches heuristiques d'extraction des règles d'association tentent de résoudre les limitations des approches exhaustives en trouvant des solutions réalisables, pas nécessairement optimales.

L'apprentissage automatique se concentre sur le développement des modèles et d'algorithmes capables d'apprendre à partir des données pour faire des prédictions et aider à la prise de décision. Parmi ces techniques de modélisation, l'approche ensembliste est la plus connue pour avoir remporté plusieurs fois les compétitions en science des données sur la plateforme Kaggle, en particulier ceux qui utilisent des arbres de décision avec l'implémentation XGBoost (eXtreme Gradient Boosting) [5]. En effet, l'algorithme XGBoost a été introduit par Chen et Guestrin en 2016 [6] et utilise le concept de renforcement de l'arbre de gradient. L'avantage d'utiliser cet algorithme est sa vitesse fulgurante par rapport à d'autres algorithmes tels qu'AdaBoost ou CatBoost.

Dans notre contexte, qui consiste à construire un système de recommandation personnalisée, nous proposons également une approche novatrice qui consiste à combiner un algorithme d'apprentissage profond avec un algorithme de partitionnement pour effectuer un regroupement plus efficace des clients, En effet, avoir des groupes de clients ayant des habitudes de consommation similaires permettra à une entreprise d'établir une campagne de communication plus efficace. Pour ce faire, nous utilisons l'algorithme autoencodeur débruiteur pour extraire les caractéristiques importantes, puis K-Means pour effectuer le partitionnement en K groupes homogènes.

1.2. Objectif du mémoire

L'objectif de ce mémoire est d'explorer les modèles permettant d'effectuer l'analyse du panier de consommation à partir des données d'une épicerie en ligne.

Dans la première étape, nous avons construit des règles à partir d'Apriori et FP-Growth sur l'ensembles des données de transactions. Cependant, ces règles ne prennent pas en compte les occurrences de chaque article ni de chaque utilisateur.

Dans la deuxième étape nous avons développé un modèle prédictif XGBoost afin de cibler spécifiquement un seul utilisateur et prédire quel sera son futur achat. Cependant, cette approche rencontre une limite lorsqu'il s'agit de prédire le comportement d'un groupe d'utilisateurs, C'est pourquoi nous avons exploré les méthodes de partitionnement afin de résoudre ce problème.

Le présent mémoire est composé de 8 chapitres. Dans le premier chapitre, nous avons mis en avant le contexte, la problématique et l'objectif du projet. Le 2ème chapitre décrit le cadre théorique de l'extraction des règles d'association. Le chapitre 3 traite de l'apprentissage automatique par approche ensembliste, puis le chapitre 4 décrit l'apprentissage non supervisé et les réseaux de neurones artificiels. Le chapitre 5 présente la méthodologie adoptée pour développer notre solution. Une implémentation informatique de notre méthodologie est présentée dans le chapitre 6, puis nous présentons les résultats de nos expérimentations en chapitre 7 et tirer la conclusion en chapitre 8.

Chapitre 2. Règles d'association

Dans cette section nous allons présenter les règles d'associations ainsi que les différents algorithmes, à savoir Apriori et FP-Growth.

2.1. Introduction

Une des tâches plus importantes de l'exploration de données est de trouver les règles d'association et de découvrir les modèles et les relations les plus intéressantes et les plus utiles dans de grands ensembles de données [3]. Les relations les plus intéressantes peuvent être représentées sous la forme d'ensembles d'éléments présents dans de nombreuses transactions, appelés ensembles d'éléments fréquents ou règles d'association.

Nous pouvons utiliser des règles d'association dans n'importe quel jeu de données où les entités ne prennent que deux valeurs, c'est-à-dire 0 ou 1. Les applications populaires des règles d'associations sont :

- L'analyse du panier de consommation ;
- Détection d'anomalies ;
- Optimisation des stocks ;
- Analyse de données médicales.

Cependant qu'est-ce qu'une règle d'association ? Qu'est-ce qu'un ensemble d'éléments fréquents ?

2.2. Définition sur les règles d'association

2.2.1. Règle d'association

Une règle d'association est une structure conditionnelle (Si [condition] alors [résultat]) qui indique la relation entre un antécédent et un conséquent. A → B, où A est l'antécédent et B est le conséquent.

2.2.2. Transactions et itemset

Soient $T = \{I1, I2, \dots, Im\}$ l'ensemble d'attributs binaires appelés items. Soit T une base de données des transactions. Chaque transaction t est représentée comme un vecteur binaire, avec $t[k]$ = 1 si la transaction t achète l'item k, sinon $t[k]$ = 0. Chaque transaction est identifiée par un numéro unique dans la base de données des transactions. Un Itemset est un ensemble d'éléments dans une base de données des transactions.

Transactions	**Items**
T1	{Milk, Egg, Bread, Butter}
T2	{Milk, Egg, Butter, Ketchup}
T3	{Bread, Butter, Ketchup}
T4	{Milk, Bread, Butter}
T5	{Bread, Butter, Cookies}
T6	{Milk, Bread, Butter, Cookies}
T7	{Milk, Cookies}
T8	{Milk, Bread, Butter}
T9	{Egg, Bread, Butter, Cookies}
T10	{Milk, Bread, Butter}

Tableau 2.1: base de données des transactions

Dans le **Tableau 2.1**, on observe dix transactions d'une épicerie. Chaque transaction montre les articles achetés. La première transaction correspond à T1 =

{Milk, Egg, Bread, Butter}, une règle d'association pour cette transaction peut être définie par {Milk, Egg} → {Bread}.

Pendant le traitement, ces informations sont présentées sous forme binaire. Voici la représentation dans le **Tableau *2.2***.

Transactions	Milk	Egg	Bread	Butter	Ketchup	Cookies
t[1]	1	1	1	1	0	0
t[2]	1	1	0	1	1	0
t[3]	0	0	1	1	1	0
t[4]	1	0	1	1	0	0
t[5]	0	0	1	1	0	1
t[6]	1	0	1	1	0	1
t[7]	1	0	0	0	0	1
t[8]	1	0	1	1	0	0
t[9]	0	1	1	1	0	1
t[10]	1	0	1	1	0	0

Tableau 2.2: base de transactions binaires

2.2.3. Support et confiance

Soient A et B deux sous-ensembles d'items disjoints. Une règle d'association de la forme $A \rightarrow B$ possède deux métriques importantes pour mesurer sa force à savoir : le support et la confiance.

Le support d'un itemset est le nombre de transaction qui le contiennent, divisé par le nombre total des transactions dans la base de données D. Le support d'un itemset A, est défini mathématiquement par :

$$\sigma\,(\mathrm{A}\,) = \frac{\mathrm{Card(A)}}{\mathrm{Card(D)}}$$

Le support d'une règle d'association $A \rightarrow B$ est le support $\sigma\ (A \cup B)$, divisé par le nombre total des transactions de la base de données D.

$$\mathrm{s}\,(\mathrm{A}\ \rightarrow\ \mathrm{B}) = \frac{\mathrm{Card(A \cup B)}}{\mathrm{Card(D)}}\,.$$

La confiance d'une règle d'association $A \rightarrow B$ est le support σ $(A \cup B)$, divisé par le support $\sigma(A)$.

$$c\ (A\ \rightarrow\ B)\ = \frac{\sigma\ (A\ \cup\ B)}{\sigma(A)}$$

Exemple :

Considérons la règle {Milk, Egg} → {Bread}; *Card ({Milk, Egg, Bread})* étant 1, *Card ({Milk, Egg})* étant 2 et le nombre total de transactions étant 10.

Le support de la règle est :

$$s(\{\text{Milk, Egg}\} \rightarrow\ \{\text{Bread}\}) = \frac{1}{10}\text{=0.1}$$

La confiance de la règle est :

$$c\ (\{\text{Milk, Egg}\} \rightarrow\ \{\text{Bread}\}) = \frac{\sigma\ (\{\text{Milk, Egg}\} \rightarrow\ \{\text{Bread}\})}{\sigma(\{\text{Milk, Egg}\})} = \frac{0.1}{0.2} = 0.5$$

2.2.4. Support minimum (minsup)

Le support minimum d'une règle d'association est le seuil minimal défini par l'utilisateur que la règle d'association doit satisfaire pour être acceptée.

2.2.5. Confiance minimum (minconf)

La confiance minimum d'une règle d'association, représente le seuil minimal défini par l'utilisateur que la règle doit satisfaire pour être acceptée.

2.2.6. Lift

Le lift d'une règle d'association mesure l'amélioration apportée par la règle par rapport au jeu de transactions , où A et B seraient indépendants. Il est défini par :

$$\text{Lift}\ (A\ \rightarrow\ B)\ = \frac{s(A\ \cup\ B\ )}{s(A)\ \ x\ s(B)}$$

Une règle d'association avec un lift supérieur à 1 indique une association positive. Plus le lift est élevé, plus l'association est forte. Si le lift est égal à 1, cela

signifie une indépendance entre l'antécédent et le conséquent. Un lift inférieur à 1 indique une association négative.

Exemple :

$$\text{Lift}\ (\{\text{Milk}, \text{Egg}\} \rightarrow\ \{\text{Bread}\})\ = \frac{0.1}{0.2\ \ \text{x}\ \ 0.8} = 0.625$$

L'achat de {Milk, Egg}, diminue la probabilité d'achat de {Bread}.

2.2.7. Itemset fréquent

Un Itemset est fréquent si et seulement si son support est supérieur à un support minimum.

2.3. Extraction des règles d'association

La recherche de règles d'associations passe par trois étapes : la préparation des données, la recherche d'ensembles d'éléments fréquents et la génération des règles d'association. Le problème d'extraction des règles d'association peut être formellement énoncé comme suit : étant donné un ensemble de transactions T, trouver toutes les règles ayant un support ≥ minsup et une confiance ≥ minconf, où minsup et minconf sont les seuils de support et de confiance correspondants.

2.3.1. Préparation de données

Cette étape consiste à réduire la quantité de données en gardant seulement celles qui sont plus pertinentes, et à transformer par la suite ces dernières en un contexte d'extraction, c'est- à-dire une transformation en triplets constitués :

i. D'un ensemble d'objets,
ii. D'un ensemble d'Itemsets
iii. D'une relation binaire entre les deux.

Cette transformation est nécessaire afin qu'il soit possible d'appliquer les algorithmes d'extraction de règles d'association sur divers types de données (données relationnelles, transactionnelles, spatiales, multimédias, etc.), et de permettre de

réduire le temps d'extraction des itemsets fréquents et d'améliorer la qualité des règles d'association.

2.3.2. Recherche d'itemsets fréquents

L'extraction des itemsets fréquents permet d'isoler, depuis une base de données, des ensembles d'attributs qui satisfont un seuil de fréquence minimale minsup spécifié par l'utilisateur. Cette étape est très coûteuse en temps d'exécution. Une structure en treillis peut être utilisée pour énumérer la liste de tous les ensembles d'éléments possibles [7]. La **Figure 2.1** montre un treillis d'itemset pour I = {a, b, c, d, e}. En général, un ensemble de données qui contient K objets peut potentiellement générer jusqu'à 2^k – 1 ensembles d'éléments fréquents, exclut l'ensemble nul. En effet K peut être très grand dans de nombreuses applications, l'espace de recherche des ensembles d'éléments qui doivent être explorés est exponentiellement grand.

Une approche par force brute pour trouver des ensembles d'éléments fréquents consiste à déterminer le nombre de supports pour chaque ensemble d'éléments candidats dans la structure en treillis. Pour ce faire, nous devons comparer chaque candidat à chaque transaction, et son nombre de supports sera incrémenté. Par exemple, la prise en charge de { Milk, Egg } est incrémentée deux fois car l'itemset est contenu dans les transactions 1 et 2. Une telle approche peut être coûteuse car elle nécessite *O (N M ω)* comparaisons, où N est la largeur de transaction maximale. La largeur des transactions est le nombre d'éléments présents dans une transaction.

Il existe trois approches principales pour réduire la complexité de calcul de la génération d'itemsets fréquents.

i. **Réduire le nombre d'itemsets candidats (*M*)** : Le principe d'apriori décrit dans la section suivante est un moyen efficace d'éliminer certains des itemsets candidats sans compter leurs valeurs de support.
ii. **Réduire le nombre de comparaisons :** Au lieu de faire correspondre chaque ensemble d'éléments candidats à chaque transaction, nous pouvons réduire le nombre de comparaisons en utilisant des structures de

données plus avancées, soit pour stocker les ensembles d'éléments candidats, soit pour compresser l'ensemble de données.

iii. **Réduire le nombre de transactions (*N*) :** À mesure que la taille des ensembles d'éléments candidats augmente, moins de transactions seront prises en charge par les ensembles d'éléments. Par exemple, étant donné que la largeur de la septième transaction est de 2, il serait avantageux de supprimer cette transaction avant de rechercher des ensembles d'éléments fréquents de taille 3 et plus.

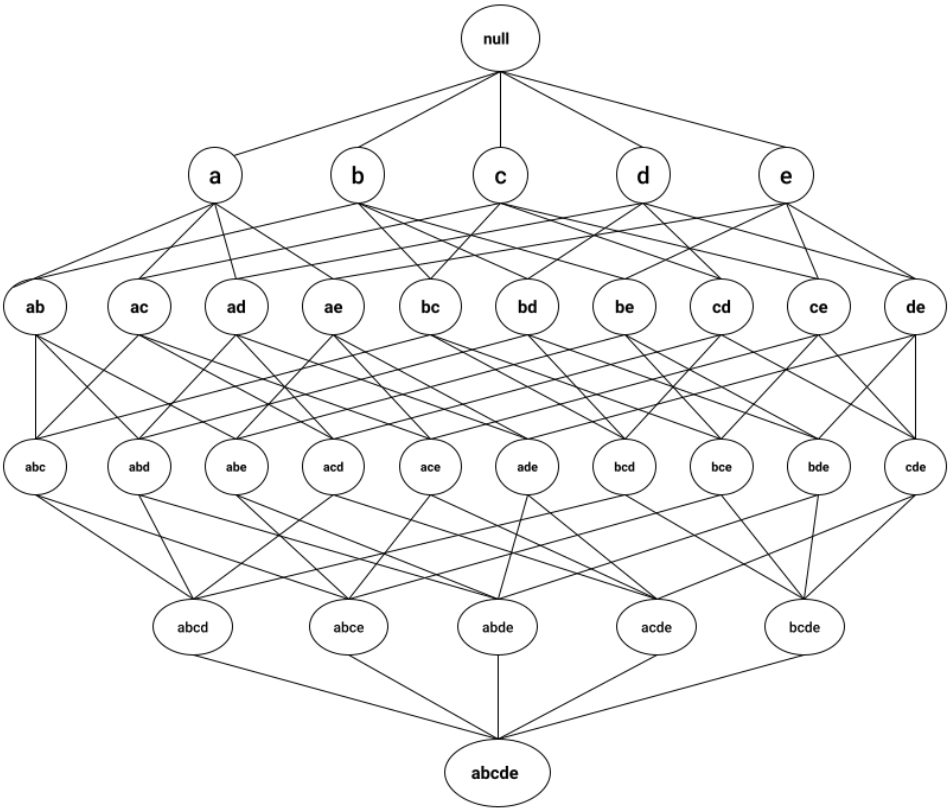

Figure 2.1: Génération d'itemsets fréquents

2.3.3. Génération des règles d'association

La génération des règles d'association consiste à trouver toutes les règles ayant un support supérieur ou égal à $minsup$ et une confiance supérieure ou égale à $minconf$, où

$minsup$ et $minconf$ sont des seuils, définis par l'utilisateur pour le support et la confiance respectivement. Cette étape est moins coûteuse en temps d'exécution par rapport à la recherche des ensembles d'éléments fréquents. Étant donné un ensemble d'éléments fréquents, chaque *k*- itemset peut produire jusqu'à $2^k - 2$ règles d'association, en ignorant les règles qui ont des antécédents ou des conséquents vides. Une règle d'association peut être extraite en partitionnant l'itemset Y en deux sous-ensembles non vides, X et $\mathrm{Y} - \mathrm{X}$, tel que $\mathrm{X} \rightarrow \mathrm{Y} - \mathrm{X}$ satisfait le seuil de confiance. Notons que toutes ces règles doivent déjà avoir atteint le seuil de prise en charge car elles sont générées à partir d'un jeu d'éléments fréquents.

Exemple : Soit $\mathrm{X} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$ un ensemble d'éléments fréquents. Il y a six règles d'associations candidates qui peuvent être générées à partir de X : $\{\mathrm{a}, \mathrm{b}\} \rightarrow \{\mathrm{c}\}$, $\{\mathrm{a}, \mathrm{c}\} \rightarrow \{\mathrm{b}\}$, $\{\mathrm{b}, \mathrm{c}\} \rightarrow \{\mathrm{a}\}$, $\{\mathrm{a}\} \rightarrow \{\mathrm{b}, \mathrm{c}\}$, $\{\mathrm{b}\} \rightarrow \{\mathrm{a}, \mathrm{c}\}$ et $\{\mathrm{c}\} \rightarrow \{\mathrm{a}, \mathrm{b}\}$. Comme leur support est identique au support de X , toutes les règles satisfont le seuil de support.

Le calcul de la confiance d'une règle d'association ne nécessite pas d'analyses supplémentaires de l'ensemble de données de transaction. Considérons la règle $\{1\ ,2\} \rightarrow \{3\}$, qui est générée à partir de l'itemset fréquent $\mathrm{X} = \{1, 2, 3\}$. La confiance pour cette règle est calculée comme $\frac{\sigma\{1,2,3\}}{\sigma(\{1,2\})}$, car $\{1, 2, 3\}$ est fréquent, la propriété anti-monotone du support assure que $\{1, 2\}$ doit également être fréquent. Étant donné que le nombre de supports pour les deux ensembles d'éléments a déjà été trouvé lors de la génération d'éléments fréquents, il n'est pas nécessaire de relire l'intégralité de l'ensemble de données.

La confiance ne montre pas la propriété anti-monotone de la même manière que la mesure de support.

2.4. Approche exhaustive

Les algorithmes d'extractions de règles d'association exhaustifs tentent de trouver de manière exhaustive toutes les solutions possibles, contrairement aux solutions heuristiques [6]. Cependant, cette stratégie peut rencontrer des problèmes dans les bases de données plus grandes et des espaces de recherche plus grands, et nécessite une puissance de calcul et un espace mémoire élevés. Nous présentons Apriori et FP-Growth.

2.4.1. L'algorithme Apriori

Proposé par Agrawal et Srikant en 1994, l'algorithme Apriori représente la base de tous les algorithmes de recherche des règles d'association [6]. L'algorithme Apriori utilise une approche itérative, où $\mathrm{k} - \mathrm{Itemsets}$ sont employés pour explorer les $(\mathrm{k}+1) - \mathrm{Itemsets}$. D'abord, les $1 - \mathrm{Itemsets}$ sont trouvés par balayage de la base de données pour calculer le support de chaque item, et la collecte de ces Itemsets qui ont un support supérieur ou égal à $minsup$. L'ensemble résultant est noté L_1, puis utilisé pour trouver L_2, les 2-itemsets, qui est utilisé pour trouver L_3, et ainsi de suite jusqu'à ce qu'aucun k-Itemsets puisse être trouvé. L'obtention de chaque L_k nécessite une analyse complète de la base de données.

La description complète de l'algorithme Apriori se résume dans les étapes suivantes [8]:

Entrées : un support minimum et une base de données de transactions.

Sorties : génération des Itemsets fréquents

1. $M_i = \emptyset,\ i = 0$
2. $C_1 =$ tous les 1-Itemsets dans la base de données
3. L_1 = tous les Itemsets fréquents de C_1

Tant que (M_i est non vide) **Faire**

1. $C_{i+1} =$ Candidate-gen (L_i)
2. $L_{i+1} =$ tous les Itemsets fréquents de C_{i+1}
3. $i++$
4. Retourner l'union des M_i

Fin tant que

Algorithme 2.1: Algorithme Apriori

Considérons les données du **Tableau 2.2**, il y a dix transactions dans la base de données (D), soit $|D| = 10$. Nous utilisons la **Figure 2.2** pour illustrer l'algorithme Apriori. Supposons que le support minimum requis est égal à 4, soit $minsup$ = 4.

1. Dans la première itération de l'algorithme (**Figure 2.2**), chaque item appartient à l'ensemble 1-itemsets, C_1. Ensuite l'algorithme calcule l'occurrence de chaque élément de C_1. Après avoir calculer les occurrences, il compare chaque valeur avec la valeur $minsup$ afin d'éliminer les items qui ne satisfont pas le $minsup$. Le résultat obtenu détermine l'ensemble fréquents, L_1.

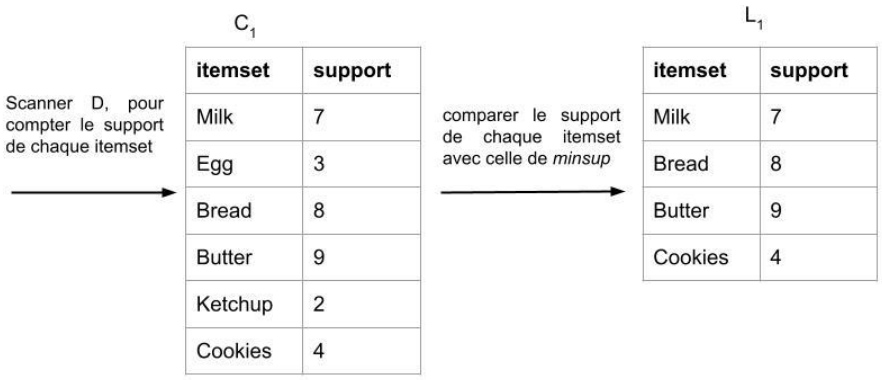

Figure 2.2: La première itération de l'algorithme Apriori

2. Dans la deuxième itération (**Figure 2.3**), l'algorithme utilise la jointure L_1 x L_1 pour générer l'ensemble des candidats 2-itemsets, C_2.Ensuite , il analyse les transactions dans le but de calculer le support de chaque candidat en C_2. Enfin, l'algorithme supprime les candidats avec lequel leur support est inférieur à $minsup$ pour obtenir l'ensemble fréquents, L_2.

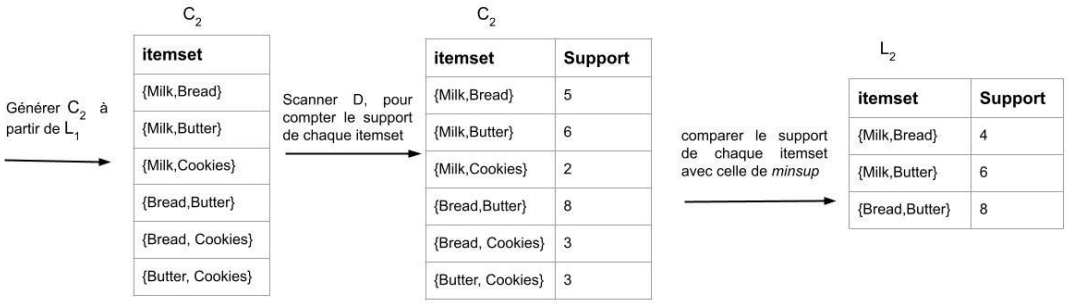

Figure 2.3: La deuxième itération de l'algorithme Apriori

3. Dans la troisième itération (**Figure 2.4**), la génération de C_3 est faite par la jointure $L_2 \text{ x } L_2$. L'algorithme scanne les transactions pour calculer l'occurrence de chaque item. L'ensemble fréquent, L_3, est déterminé par l'élimination des candidats qui ne satisfont pas le $minsup = 4$.

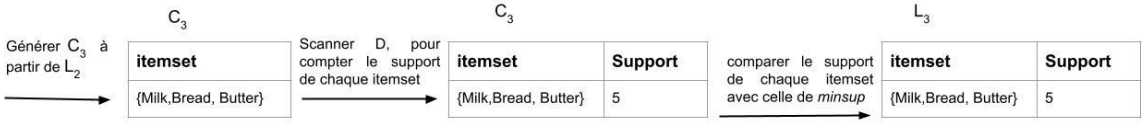

Figure 2.4: La troisième itération de l'algorithme Apriori

4. Enfin, Le résultat de L_3 est constitué d'un seul candidat, à savoir {Milk, Bread, Butter}, qui satisfait le $minsup$. Ainsi l'algorithme se termine, à ce stade-là, en affichant tous les Itemsets fréquents trouvés.

- **Génération des règles d'association à partir d'itemsets fréquents :**

Supposons l'itemset fréquent $l = \{ Milk, Bread, Butter \}$, les sous-ensembles non vides de l: $\{ Milk, Bread \}, \{Milk, Butter\}, \{Bread, Butter\}, \{Milk\}, \{Bread\}, \{Butter\}$. Les règles d'associations résultants sont indiquées ci-dessous :

1. $\{ Milk, Bread \} \rightarrow \{ Butter\}$, Confiance = 100 %

2. $\{ Milk, Butter \} \rightarrow \{ Bread \}$, Confiance = 83,33 %
3. $\{ Bread, Butter \} \rightarrow \{ Milk \}$, Confiance = 62,50 %
4. $\{ Milk \} \rightarrow \{ Bread, Butter \}$, Confiance = 71, 43 %
5. $\{ Bread \} \rightarrow \{ Milk, Butter \}$, Confiance = 62, 50 %
6. $\{ Butter \} \rightarrow \{ Milk, Bread \}$, Confiance = 55, 55 %

Si le $minconf$ est de 65 %, alors la première, la deuxième, et la quatrième règle seront affichées en sortie.

2.4.2. L'algorithme FP-Growth

FP-Growth est un algorithme (**Algorithme *2.2***) très connu et inventé par Jiawei Han [9]. Son innovation la plus intéressante est qu'il génère des ensembles fréquents sans générer d'itemsets candidats. Il n'analyse la base de données que deux fois. Cette approche utilise une structure appelée arbre à motifs fréquents ou arbre FP. La description complète de l'algorithme FP-Growth se résume dans les étapes suivantes [10] :

1. La première étape consiste à analyser la base de données pour trouver les occurrences des ensembles d'éléments. Cette étape est la même que la première étape d'Apriori.
2. La deuxième étape consiste à construire l'arborescence FP. Pour cela, on crée la racine de l'arbre. La racine est représentée par nul.
3. L'étape suivante consiste à analyser à nouveau la base de données et à examiner les transactions. On examine la première transaction pour découvrir l'ensemble d'éléments qu'elle contient. L'ensemble d'éléments avec le nombre maximum est pris en haut, l'ensemble d'éléments suivant avec un nombre inférieur et ainsi de suite. Cela signifie que la branche de l'arborescence est construite avec des ensembles d'éléments en ordre décroissant.
4. La transaction suivante dans la base de données est examinée. Les ensembles d'éléments sont classés par ordre décroissant de nombre. Si

un ensemble d'éléments de cette transaction est déjà présent dans une autre branche (par exemple dans la 1ère transaction), alors cette branche de transaction partagerait un préfixe commun à la racine (Cela signifie que l'ensemble d'éléments commun est lié au nouveau nœud d'un autre ensemble d'éléments dans cette transaction).

5. De plus, le nombre de l'ensemble d'éléments est incrémenté au fur et à mesure qu'il se produit dans les transactions. Le nombre de nœuds communs et de nouveaux nœuds est augmenté de 1 à mesure qu'ils sont créés et liés en fonction des transactions.
6. L'étape suivante consiste à exploiter l'arbre FP créé. Pour cela, le nœud le plus bas est examiné en premier ainsi que les liens des nœuds les plus bas. Le nœud le plus bas représente la longueur du modèle de fréquence 1. À partir de là, On parcourt le chemin dans l'arborescence FP. Ce ou ces chemins sont appelés bases de modèles conditionnels (La base de modèles conditionnels est une sous-base composée de chemins de préfixes dans l'arborescence FP se produisant avec le nœud le plus bas).
7. Construire un arbre FP conditionnel, qui est formé par un nombre d'ensembles d'éléments dans le chemin. Les ensembles d'éléments répondant au seuil de prise en charge sont pris en compte dans l'arborescence FP conditionnelle.
8. Les modèles fréquents sont générés à partir de l'arborescence FP conditionnelle.

Algorithme 2.2: Description d'algorithme FP-Growth

Exemple :

Nous allons construire les règles d'associations à partir de la base de transactions **Tableau *2.3*** en utilisant FP-Growth [10]. Prenons $minsup = 50\%$ et $minconf = 60\%$.

Transactions	Items
T1	{I1, I2, I3}
T2	{I2, I3, I4}
T3	{I4, I5}
T4	{I1, I2, I4}
T5	{I1, I2, I3, I5}
T6	{I1, I2, I3, I4}

Tableau 2.3: Exemple base de données de transactions

On a : Seuil de support = 50% => 0,5*6 = 3 => $minsup = 3$.

1. Etape 1 : Compter le support de chaque élément (voir **Tableau *2.4***).

Items	Support
I1	4
I2	5
I3	4
I4	4
I5	2

Tableau 2.4: Support d'items

2. Etape 2 : Trier l'ensemble d'éléments satisfaisant au seuil par ordre décroissant(voir **Tableau *2.5***).

Items	Support
I2	5
I1	4
I3	4
I4	4

Tableau 2.5: Ensemble d'éléments fréquents triés

3. Etape 3 : Construire l'arbre FP (voir **Figure 2.5**)
    3.1. Considérant le nœud racine Null.
    3.2. La première analyse de la transaction T1 : I1, I2, I3 contient trois éléments {I1 :1}, {I2 :1}, {I3 :1}, où I2 est lié en tant qu'enfant à la racine, I1 est lié à I2 et I3 est lié à I1.
    3.3. T2 : I2, I3, I4 contient I2, I3 et I4, où I2 est lié à la racine, I3 est lié à I2 et I4 est lié à I3. Mais cette branche partagerait le nœud I2 commun qui est déjà utilisé dans T1.
    3.4. Incrémenter le nombre de I2 de 1 et I3 est lié en tant qu'enfant à I2, I4 est lié en tant qu'enfant à I3. Le décompte est {I2:2}, {I3:1}, {I4:1}.
    3.5. T3 : I4, I5. De même, une nouvelle branche avec I5 est liée à I4 lors de la création d'un enfant.
    3.6. T4 : I1, I2, I4. La séquence sera I2, I1 et I4. I2 est déjà lié au nœud racine, il sera donc incrémenté de 1. De même I1 sera incrémenté de 1 car il est déjà lié à I2 dans T1, donc {I2 :3}, {I1 :2}, {I4 : 1}.
    3.7. T5: I1, I2, I3, I5. La séquence sera I2, I1, I3 et I5. Ainsi {I2 :4}, {I1 :3}, {I3 :2}, {I5 :1}.
    3.8. T6: I1, I2, I3, I4. La séquence sera I2, I1, I3 et I4. Ainsi {I2 :5}, {I1 :4}, {I3 :3}, {I4 1}.

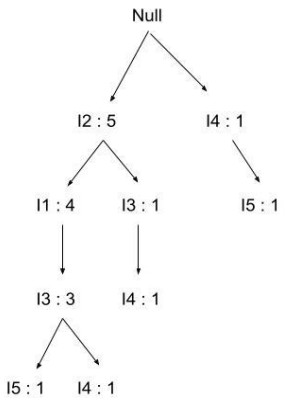

Figure 2.5: Arbre FP

4. Etape 4 : Générer les modèles fréquents (voir **Tableau *2.6***)

L'exploitation de l'arbre FP pour générer des modèles fréquents se résume comme suit :

4.1. L'élément de nœud le plus bas I5 n'est pas pris en compte car il n'a pas de nombre de support minimum, il est donc supprimé.

4.2. Le nœud inférieur suivant est I4. I4 se produit en 2 branches, {I2, I1, I3, I4:1}, {I2, I3, I4:1}. Par conséquent, en considérant I4 comme suffixe, les chemins de préfixe seront {I2, I1, I3 :1}, {I2, I3 : 1}. Cela constitue la base du modèle conditionnel.

4.3. La base de modèles conditionnels est considérée comme une base de données de transactions, un arbre FP est construit. Celui-ci contiendra {I2 :2, I3 :2}, I1 n'est pas pris en compte car il ne répond pas au nombre minimum de support.

4.4. Ce chemin générera toutes les combinaisons de motifs fréquents : {I2, I4 :2}, {I3, I4 :2}, {I2, I3, I4 :2}.

4.5. Pour I3, le chemin du préfixe serait : {I2, I1 :3}, {I2 :1}, cela générera un arbre FP à 2 nœuds : {I2 :4, I1 :3} et des modèles fréquents sont générés : {I2, I3 :4}, {I1 , I3 :3}, {I2, I1, I3 :3}.

4.6. Pour I1, le chemin du préfixe serait : {I2 :4}, cela générera un arbre FP à nœud unique : {I2 :4} et des modèles fréquents seront générés : {I2, I1 :4}.

Items	**Base de modèle conditionnel**	**Arbre FP conditionnel**	**Modèles fréquents générés**
I4	{I2, I1, I3 :1}; {I2, I3 :1}	{I2 :2, I3 :2}	{I2, I4 :2}; {I3, I4 :2}; {I2, I3, I4 :2}
I3	{I2, I1 : 3}; {I2 : 1}	{I2 :4, I1 : 3}	{I2, I3 :4}; {I1, I3 :3}; {I2, I1, I3 :3}
I1	{I2 : 4}	{I2 : 4}	{I2, I1 : 4}

Tableau 2.6: Modèles fréquents générés

2.4.3. Avantages et inconvénients

L'avantage majeur des approches exhaustives d'extraction des règles d'association est qu'elles tentent de trouver toutes les solutions possibles. Le premier inconvénient d'Apriori est le processus complexe de génération de candidats qui utilise la plupart du temps de l'espace mémoire. Un autre inconvénient est qu'il nécessite plusieurs analyses de la source de données, tandis que FP-Growth n'a besoin d'analyser la base de données que deux fois. Ensuite, l'inconvénient de FP-Growth est qu'il doit élaborer des bases de modèles conditionnels et construire de manière récursive des arbres. Le **Tableau *2.7*** met en perspective Apriori et FP-Growth.

Apriori	**FP-Growth**
Génération de modèles	
Apriori génère un motif en associant les éléments en singletons, paires et triplets.	FP-Growth génère un modèle en construisant un arbre FP.
Génération de candidats	
Apriori utilise la génération de candidats.	Il n'y a pas de génération de candidats.
Processus	
Le processus est comparativement plus lent que FP-Growth, le temps d'exécution augmente de façon exponentielle avec l'augmentation du nombre d'ensembles d'éléments.	Le processus est plus rapide que celui d'Apriori. La durée d'exécution du processus augmente linéairement avec l'augmentation du nombre d'ensembles d'éléments.
Utilisation de la mémoire	
Les combinaisons candidates sont enregistrées en mémoire.	Une version compacte de la base de données est enregistrée.

Tableau 2.7: Comparaison Apriori et FP-Growth

2.5. Approche heuristique

Les algorithmes d'extractions des règles d'association heuristiques tentent d'optimiser certaines des fonctionnalités des approches exhaustives. Une heuristique

est un algorithme qui fournit rapidement (en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation NP-difficile [10]. Une heuristique, ou méthode approximative, est donc le contraire d'un algorithme exact qui trouve une solution optimale pour un problème donné. Il existe plusieurs algorithmes heuristiques pour extraire les règles d'association. Nous présentons dans ce document quatre groupes : les algorithmes génétiques (GA), l'optimisation des essaims d'abeilles (BSO), l'optimisation des essaims de particules
(PSO) et l'optimisation des colonies de fourmis (ACO).

2.5.1. Algorithmes génétiques (genetic algorithms : GA)

Les algorithmes génétiques ont été initialement développés par Holand (1975) [11]. GA est un algorithme de recherche stochastique basé sur les principes de sélection naturelle et de génétique naturelle, qui a été appliqué avec succès à de nombreux problèmes d'apprentissage automatique et d'optimisation, pour générer des solutions utiles.

L'algorithme génétique fonctionne de manière itérative en générant de nouvelles populations de chaînes à partir des anciennes. Les AG utilisent un vocabulaire similaire à celui de la génétique naturelle. L'algorithme part d'une population d'individus générés aléatoirement et se déroule sur plusieurs générations. À chaque génération, la condition physique de chaque individu de la population est évaluée, plus les individus en forme sont sélectionnés dans la population actuelle et des opérateurs génétiques sont appliqués pour former une nouvelle population. La nouvelle population est ensuite utilisée dans la prochaine itération de l'algorithme. L'algorithme se termine lorsqu'un nombre maximum de générations a été produit ou qu'un niveau de condition physique satisfaisant a été atteint pour la population. Un algorithme génétique standard (voir **Algorithme *2.3***) utilise trois opérateurs génétiques :

- **Reproduction (ou sélection)** : La sélection concerne la survie probabiliste du plus apte, dans laquelle les chromosomes aptes sont choisis pour survivre. Il existe de nombreuses méthodes pour

sélectionner les meilleurs chromosomes, comme la sélection à la roulette, la sélection par classement, la sélection par tournoi.

- **Croisement :** Il prélève des chromosomes individuels des parents et les combine pour en former de nouveaux. Le croisement peut être un croisement à point unique, un croisement multipoint.
- **Mutation :** La mutation modifie les nouvelles solutions afin d'obtenir les meilleures solutions. Il retourne chaque bit chez l'individu avec une probabilité de mutation prédéfinie (0 devient 1, 1 devient 0).

1. **[Commencer]** Générer une population aléatoire de n chromosomes;
2. **[Aptitude]** Évaluer la fonction de fitness f(x) de chaque chromosome x dans la population;
3. **[Nouvelle population]** Créer une nouvelle population en répétant les étapes suivantes jusqu'à ce que la nouvelle population soit terminée :

   Sélection: Sélectionner deux chromosomes parents dans une population en fonction de leur forme physique (plus la forme physique est bonne, plus grandes sont les chances d'être sélectionné)

   Croisement : Avec une probabilité de croisement, croiser les parents pour former une nouvelle progéniture (enfants). Si aucun croisement n'a été effectué, la progéniture est une copie exacte des parents.

   Mutation: Avec une probabilité de mutation, une nouvelle progéniture mute à chaque locus (position dans le chromosome).

   Acceptant: Placer une nouvelle progéniture dans une nouvelle population.
4. [**Remplacer**] utiliser la nouvelle population générée pour une nouvelle exécution de l'algorithme.
5. [**Test**] Si la condition finale est satisfaite, arrêtez et renvoyez la meilleure solution dans la population actuelle

6. [**Boucle**] Passer à l'étape 2.

Algorithme 2.3: Algorithme génétique de base

Un organigramme d'algorithme génétique est illustré à la **Figure *2.6***. Une itération de l'algorithme est appelée génération.

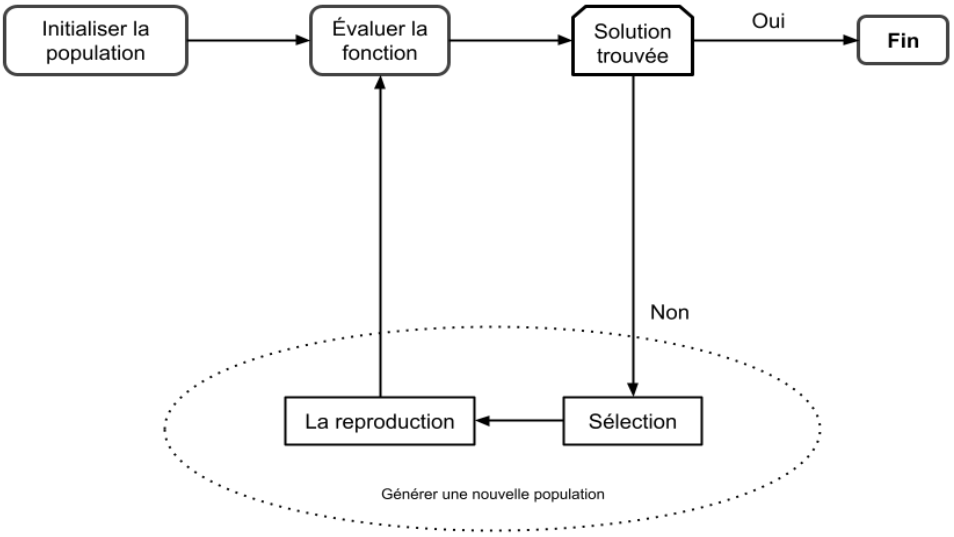

Figure 2.6: Organigramme d'algorithme génétique [10]

2.5.1.1. Optimisation des règles d'association par algorithme génétique

i. Encoder les données

L'algorithme génétique ne fonctionne pas directement sur les données brutes, il faut coder sous la forme d'une technique de représentation binaire (0 et 1).

ii. Fonction de fitness (fonction d'aptitude)

La partie la plus importante de l'algorithme génétique est la conception de la fonction de fitness :

$$f(x) = \frac{Support(x)}{minsupport} \begin{cases} p(support(x) > minsupport) \\ q(support(x) < minsupport) \end{cases}$$

Le support est le support de la nouvelle règle générée par une opération génétique.

iii. Stratégie de sélection

La stratégie de sélection basée sur la forme physique individuelle et la concentration Pi est la probabilité de sélection d'un individu, dont la valeur de forme physique est la plus grande, supérieur à 1 et f(α) est une valeur dont l'adéquation est inférieure à 1 mais plus proche de la valeur de 1.

On a donc :

$$Pi = \frac{f(X_i)}{\sum_1^M f(X_j)}\ e^{-\alpha f(\alpha)}$$

Où α est un facteur d'ajustement.

iv. Opération génétique

On détermine la capacitée de recherche et convergence avec la sélection, le croisement et la mutation.

2.5.1.2. Avantages des algorithmes génétiques

Les algorithmes génétiques présentent plusieurs avantages tels que :

1. Leurs convergences ne dépendent pas de la valeur initiale.
2. Ils permettent de déterminer l'optimum global de la fonction objectif.
3. Ils représentent des méthodes génériques qui peuvent optimiser une large gamme de problèmes différents.
4. Leur capacité à faire plusieurs calculs en parallèle.

2.5.1.3. Inconvénients des algorithmes génétiques

Les algorithmes génétiques ont aussi des inconvénients tels que :

1. Le temps de calcul est énorme puisqu'ils manipulent plusieurs solutions en parallèle.

2. La recherche pour la solution optimale se limite généralement autour d'un minimum qui n'est pas forcément l'optimum attendu. On parle dans ce cas de convergence prématurée.
3. L'efficacité d'un algorithme génétique dépend beaucoup de la méthode de croisement et du type de codage choisis.

2.5.2. Optimisation des essaims d'abeilles (Bee swarm optimization : BSO)

L'objectif de cette approche est d'identifier des règles d'association significatives dans de grandes bases de données transactionnelles en s'inspirant du comportement des abeilles exerçant des responsabilités différentes. Ces abeilles doivent explorer au hasard l'espace du problème de recherche pour trouver les sources de nourriture possibles (solutions). La qualité de chaque source de nourriture peut être évaluée par sa distance à la ruche. Cependant on définit un essaim comme un ensemble ordonné de N unités décrites par N composantes d'un vecteur, chacune des N unités pouvant mettre à jour le vecteur, à son propre temps, en utilisant une fonction des composantes du vecteur [12].

2.5.2.1. Principe d'optimisation des essaims d'abeilles

Le principe d'optimisation des essaims d'abeilles est un paradigme d'optimisation qui a été adapté pour résoudre des problèmes d'optimisations, le principe d'optimisation des essaims d'abeilles repose sur les idées suivantes:

1. **Population d'individus** : Tout comme une ruche d'abeilles est composée d'une population d'individus (les abeilles), l'optimisation des essaims d'abeilles crée une population d'individus artificiels pour résoudre un problème donné. Ces individus sont appelés "abeilles artificielles" et représentent des solutions potentielles au problème.
2. **Communication indirecte** : Dans une colonie d'abeilles, la communication entre abeilles se fait principalement de manière indirecte, par le biais de phéromones. Les abeilles déposent des phéromones sur leur chemin pour

marquer les sources de nourriture ou les meilleures voies. Dans l'optimisation des essaims d'abeilles, des "phéromones artificielles" sont utilisées pour modéliser l'information partagée entre les individus.

3. **Exploration et exploitation** : Les abeilles réelles équilibrent l'exploration de nouvelles sources de nourriture (exploration) avec l'exploitation de sources connues (exploitation). De même, dans l'optimisation des essaims d'abeilles, les individus équilibrent l'exploration de nouvelles solutions avec l'exploitation des solutions déjà trouvées pour atteindre l'optimum du problème.
4. **Mise à jour des solutions** : Les abeilles artificielles mettent à jour leurs solutions en fonction de la qualité des solutions déjà explorées et des phéromones déposées. Les solutions les mieux adaptées laissent généralement plus de phéromones, attirant ainsi davantage d'abeilles artificielles vers ces solutions.
5. **Convergence** : Au fil des générations, les abeilles artificielles convergent vers une solution optimale ou quasi-optimale. Les phéromones déposées deviennent plus fortes sur la meilleure solution, attirant davantage d'abeilles artificielles vers cette solution.

L'optimisation des essaims d'abeilles est utilisée pour la recherche de règles d'association, dont l’algorithme le plus souvent utilisé est « colonies d'abeilles artificielles ».

L'algorithme de colonie d'abeilles artificielles a été introduit par Dervis Karaboga, voici une description de cet algorithme [13].

Phase d'initialisation : Tous les vecteurs de la population des sources alimentaires $\underset{X_m}{\rightarrow}$, sont initialisés $(m = 1, \dots, SN, | SN: taille\ de\ population)$ par les abeilles éclaireuses et les paramètres de contrôle sont définis. Puisque chaque source de nourriture, $\underset{X_m}{\rightarrow}$,est un vecteur de solution au problème d'optimisation, chaque $\underset{X_m}{\rightarrow}$: le vecteur tient n variables $(x_{mi}, i = 1 \dots N)$, qui doivent être optimisés de manière à minimiser la fonction objective. La définition suivante peut être utilisée initialiser :

$$x_{mi} = l_i + rand(0,1) * (u_i - l_i)$$

Où l_i et u_i sont respectivement les limites inférieure et supérieure du paramètre x_{mi}.

Phase des abeilles employées: Les abeilles employées recherchent de nouvelles sources de nourriture $(\underset{v_m}{\rightarrow})$, pour avoir plus de nectar à proximité de la source de nourriture $(\underset{X_m}{\rightarrow})$ en leur mémoire . Ils trouvent une source de nourriture à proximité et évaluent ensuite sa rentabilité (fitness). Par exemple, ils peuvent déterminer la source de nourriture d'un voisin $\underset{v_m}{\rightarrow}$ en utilisant la formule donnée par l'équation suivante :

$$v_{mi} = x_{mi} + \varphi_{mi}(x_{mi} - x_{ki})$$

Où $\underset{X_k}{\rightarrow}$ est une source de nourriture sélectionnée au hasard, i est un indice de paramètre choisis aléatoirement et φ_{mi} est un nombre choisi aléatoirement dans $[-\alpha, \alpha]$. La fonction de fitness de la solution $fit_m(\underset{x_m}{\rightarrow})$ peut être calculé pour les problèmes de minimisation en utilisant la formule suivante :

$$fit_m\left(\underset{x_m}{\rightarrow}\right) = \begin{cases} \dfrac{1}{1 + f_m\left(\underset{x_m}{\rightarrow}\right)}, & si\ f_m\left(\underset{x_m}{\rightarrow}\right) \geq 0 \\ 1 + \left|f_m\left(\underset{x_m}{\rightarrow}\right)\right|, & si\ f_m\left(\underset{x_m}{\rightarrow}\right) < 0 \end{cases}$$

Où $f_m\left(\underset{x_m}{\rightarrow}\right)$ est une fonction objective de la solution $\underset{x_m}{\rightarrow}$.

Phase des abeilles spectatrices : Les abeilles au chômage se composent de deux groupes d'abeilles : les abeilles spectatrices et les éclaireuses. Les abeilles employées partagent leurs informations sur leurs sources de nourriture avec les abeilles spectatrices qui attendent dans la ruche, puis les abeilles spectatrices choisissent de manière probabiliste leurs sources de nourriture en fonction de ces informations. Dans CAA, une abeille spectatrice choisit une source de nourriture en fonction des valeurs de probabilité calculées à l'aide des valeurs de condition physique fournies par les abeilles employées. À cette fin, une technique de sélection basée sur la condition physique peut être utilisée, telle que la méthode de sélection à la roulette (Goldberg, 1989) [13].

La valeur de probabilité p_m avec laquelle $\underset{X_m}{\longrightarrow}$ est choisi par une abeille spectatrice peut être calculé en utilisant l'expression donnée dans l'équation :

$$p_m = \frac{fit_m\left(\underset{x_m}{\rightarrow}\right)}{\sum_{m=1}^{SN} fit_m\left(\underset{x_m}{\rightarrow}\right)}$$

Phase des abeilles scoutes : Les abeilles au chômage qui choisissent leurs sources de nourriture au hasard sont appelées éclaireuses. Les abeilles employées dont les solutions ne peuvent être améliorées par un nombre prédéterminé d'essais, spécifié par l'utilisateur de l'algorithme CAA et appelés ici « limite » ou « critères d'abandon », deviennent des éclaireuses et leurs solutions sont abandonnées. Ensuite, les éclaireurs convertis se mettent à chercher de nouvelles solutions, au hasard. Par exemple, si la solution $\underset{X_m}{\longrightarrow}$ a été abandonnée, la nouvelle

solution découverte par l'éclaireur qui était l'abeille employée de $\underset{X_m}{\longrightarrow}$ peut être définit à la phase d'initialisation. Par conséquent, les sources qui sont initialement pauvres ou qui ont été rendues pauvres par l'exploitation sont abandonnées et un comportement de rétroaction négative apparaît pour contrebalancer la rétroaction positive.

Algorithme 2.4: Algorithme des colonies d'abeilles artificielles (CAA)

2.5.2.2. Avantages des algorithmes d'optimisation des essaims d'abeilles

Les algorithmes d'optimisation des essaims d'abeilles présentent plusieurs avantages tels que :

1. **Efficacité globale** : Les algorithmes d'optimisation des essaims d'abeilles sont souvent très efficaces pour résoudre des problèmes d'optimisation combinatoire, y compris la recherche de règles d'association.
2. **Équilibrage entre exploration et exploitation** : Les algorithmes d'optimisation des essaims d'abeilles parviennent à équilibrer efficacement l'exploration de nouvelles solutions avec l'exploitation des solutions déjà identifiées, ce qui favorise la convergence vers des solutions optimales.
3. **Applicabilité en temps réel** : Ils sont utilisés dans des applications en temps réel, notamment dans la planification de tournées de véhicules, le routage, et les systèmes de recommandation.

2.5.2.3. Inconvénients des algorithmes d'optimisation des essaims d'abeilles

1. **Paramètres sensibles :** Le réglage des paramètres de ces algorithmes peut être délicat, et une mauvaise configuration peut nuire à leur performance.
2. **Convergence lente :** Dans certains cas, ces algorithmes peuvent avoir une convergence lente, en particulier pour des problèmes très complexes.

3. **Dépendance à l'initialisation** : La qualité de la solution trouvée dépend souvent de l'initialisation de l'algorithme, ce qui signifie qu'une initialisation appropriée est essentielle.

2.5.3. Optimisation des essaims de particules (Particle Swarm Optimization : PSO)

L'algorithme PSO est basé sur le comportement du troupeau d'oiseaux, où chaque oiseau est une solution unique et a une valeur de fitness. Sur la base de cet algorithme, les oiseaux savent que des sources de nourriture existent, mais ils ne savent pas exactement où se trouvent ces sources , ainsi chaque oiseau essaiera de trouver la meilleure source de nourriture [6]. La PSO s'inspire de ce comportement afin d'approximer des solutions des problèmes d'optimisation globale, mais peut également être adapté pour le contexte des règles d'association[12]. L'optimisation des essaims de particules peut être utilisée pour rechercher les meilleures combinaisons de prédicteurs ou d'éléments qui maximisent la qualité des règles d'association générées. L'algorithme PSO est décrit comme suit (voir **Algorithme *2.5***):

1. **Initialisation** :
    - Créez un ensemble de particules, où chaque particule est représentée par un vecteur de paramètres c'est-à-dire des prédicteurs pour générer des règles.
    - Chaque particule a une position initiale dans l'espace de recherche, représentée par le vecteur de paramètres : $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$, où i représente l'indice de la particule et d est la dimension de l'espace de recherche (le nombre de prédicteurs).
    - Chaque particule a une vitesse initiale, représentée par le vecteur de vitesses : $v_i = (v_{i1}, v_{i2}, \dots, v_{id})$.
2. **Évaluation de la qualité** :
    - Pour évaluer la qualité de chaque particule, on utilise une fonction d'évaluation $f(x_i)$. Cette fonction attribue une valeur de qualité à la

particule en fonction de la pertinence des règles d'association qu'elle génère.

3. **Mise à jour des meilleures positions :**
   - Chaque particule conserve sa meilleure position personnelle $(pbest_i)$ jusqu'à présent, c'est-à-dire la position à laquelle elle a obtenu la meilleure valeur d'évaluation. On met à jour $pbest_i$ si la qualité actuelle de la particule est meilleure.
   - On identifie également la meilleure position globale $(gbest)$ parmi toutes les particules de l'essaim. C'est la position associée à la meilleure qualité parmi toutes les particules.
4. **Mise à jour de la vitesse et de la position** :
   - Pour mettre à jour la vitesse de chaque particule, on utilise les équations suivantes :

$$v_{ij}^{(t+1)} = \omega v_{ij}^{(t)} + c_1 r_1 \left(pbest_{ij} - x_{ij}^{(t)}\right) + c_2 r_2 \left(gbest_j - x_{ij}^{(t)}\right)$$

   Où $v_{ij}^{(t+1)}$ est la vitesse de la particule i dans la dimension j à l'itération $t+1, \omega$ est le coefficient d'inertie, $c_1 et\, c_2$ sont des coefficients d'accélération, $r_1 et\, r_2$ sont des nombres aléatoires dans l'intervalle $[0, 1]$.
   - Pour mettre à jour la position de chaque particule, on utilise les équations suivantes :

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}$$

   Où $x_{ij}^{(t+1)}$ est la position de la particule i dans la dimension j à l'itération $t+1$
5. **Répétition :**
   - Les étapes 2 à 4 sont répétées pour un certain nombre d'itérations ou jusqu'à ce qu'un critère d'arrêt soit atteint.
6. **Résultats :**

- Les particules finales contiennent les meilleures combinaisons de prédicteurs pour générer des règles d'association pertinentes, et ces règles d'association peuvent être extraites à partir des positions des particules finales.

Algorithme 2.5: Algorithme d'optimisation des essaims de particules (PSO)

L'algorithme PSO s'appuie sur des concepts de dynamique des particules et d'optimisation stochastique pour explorer l'espace à la recherche des meilleures combinaisons. Il est relativement simple à mettre en place , sa convergence est rapide et se prête bien à des implémentations parallèles. Cependant il dépend fortement de plusieurs paramètres et il peut être piégé dans des optima locaux.

2.5.4. Optimisation des colonies de fourmis (Ant Colony Optimization : ACO)

ACO est un algorithme d'optimisation métaheuristique qui peut être utilisé pour trouver le meilleur chemin dans un graphe. Il est inspiré par le comportement des fourmis lors de la recherche de nourriture. Il a été introduit par Marco Dorigo dans les années 1990. Cependant une variante de ACO, c'est-à-dire "Ant Colony System" (ACS) a été développé par Kuo et Shih (2007) pour résoudre des problèmes d'optimisation combinatoire dont la recherche des règles d'association [6].

Le premier algorithme de colonies de fourmis vise notamment à résoudre le problème du voyageur de commerce, où le but est de trouver le plus court chemin permettant de relier un ensemble de villes [14]. Une description de base est illustrée par L'**Algorithme 2.6**:

Commencer
Initialiser
Bien que le critère d'arrêt ne soit pas satisfait, **faites**
Positionnez chaque fourmi dans un nœud de départ
Répéter
Pour chaque fourmi, **faites**
Choisissez le nœud suivant en appliquant la règle de transition d'état
Appliquer la mise à jour étape par étape des phéromones
Fin pour
Jusqu'à ce que chaque fourmi ait construit une solution
Mettre à jour la meilleure solution
Appliquer la mise à jour des phéromones hors ligne
Terminer
Fin

Algorithme 2.6: Pseudo-code de l'algorithme ACS

Dans l'ACO, chaque fourmi construit une solution possible au problème, en se déplaçant à travers une séquence finie d'états voisins. Les mouvements sont sélectionnés en appliquant une recherche locale stochastique dirigée par l'état interne de la fourmi, les informations heuristiques spécifiques au problème et les informations partagées sur la phéromone. Les fourmis utilisent une règle de décision appelée règle proportionnelle pseudo-aléatoire. Une fourmi k dans le nœud i choisira le nœud j pour se déplacer comme suit :

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{u \in J^{k}(j)}\{[\tau_{iu}(t)]^{\alpha}[\eta_{iu}(t)]^{\beta}\}} & si\ j \in J^{k}(i) \\ 0 & si\ j \notin J^{k}(i) \end{cases}$$

Où $\eta_{ij}(t)$ représente l'information heuristique; $\tau_{ij}(t)$ représente la phéromone totale déposée sur le chemin ij à l'instant t. α et β sont des paramètres qui déterminent l'importance relative de la phéromone.

La règle de transition d'état est la suivante : le prochain nœud j que la fourmi k choisit d'atteindre est donné comme ,

$$J = \begin{cases} \max\limits_{u \in J_k(i)} \{[\tau_{iu}(t)]^{\alpha}[\eta_{iu}(t)]^{\beta}\} & si\ q \leq q_0 \\ J & si\ q > q_0 \end{cases}$$

Où q est un nombre aléatoire uniformément distribué dans$[0,1]$; q_0 est un paramètre définit par $0 \leq q_0 \leq 1$.

A chaque itération de l'algorithme, chaque fourmi construit progressivement une solution, en en utilisant la règle de transition des probabilités. La piste de phéromones est mise à jour à la fois localement et globalement.

Il convient de noter que l'ACO est une application qui dépend des problèmes. Ainsi, pour appliquer l'algorithme, il faut une représentation appropriée du problème et une heuristique appropriée dans la construction de sa solution.

2.6. Conclusion

Avec l'augmentation spectaculaire des tendances à extraire des connaissances des données, de nombreuses techniques d'exploration de données ont été explorées. Les règles d'association occupent une place très importante dans le domaine de l'analyse des données. On trouve plusieurs applications dans le marketing ou dans la recherche médicale [3]. Dans ce travail, nous utilisons ces algorithmes pour construire un système de recommandation des produits dans une place de marché et présenter en chapitre 7.

Les algorithmes tels qu'Apriori permettent d'extraire les règles de manière exhaustive, mais connaissent des limitent dans des grands ensembles de données car la taille des éléments fréquents générés ne tient pas dans la mémoire, d'où l'exploration d'autres approches dites heuristiques.

Chapitre 3. Méthodes ensemblistes

Imaginons que nous posions une question complexe à des milliers de personnes choisies au hasard, et que nous combinons leur réponse. Souvent, nous découvrirons que la synthèse de ces réponses est meilleure que celle d'un expert. De la même façon, si nous agrégeons les prédictions d'un groupe de prédicteurs, nous obtiendrons souvent une meilleure prédiction qu'avec le meilleur des prédicteurs pris individuellement [15]. C'est ce que nous allons découvrir dans cette section.

3.1. Introduction

Les méthodes ensemblistes sont des méthodes très puissantes en pratique, reposant sur l'idée que combiner de nombreux apprenants faibles permet d'obtenir une performance largement supérieure aux performances individuelles de ces apprenants faibles, car leurs erreurs se compensent mutuellement [16]. Il est important de noter que les méthodes ensemblistes s'insèrent dans l'apprentissage automatique et sont regroupées en deux sous-familles principales : les méthodes parallèles et celles dites séquentielles.

Dans ce travail, nous allons faire usage des apprenants faibles appelés arbres de décision.

3.2. Apprentissage automatique

L'apprentissage automatique est la science de la programmation des ordinateurs de manière à ce qu'ils puissent apprendre à partir des données [17]. Etant donné une tache T et une mesure de performance P, on dit qu'un programme informatique apprend à partir d'une expérience E si les résultats de la tâche T, mesurés par P, s'améliorent avec l'expérience E. Par exemple un filtre anti-spam est un programme

d'apprentissage automatique qui peut apprendre à identifier les courriels frauduleux à partir d'exemples de spam et de messages normaux (ou ham).

Il existe tellement de types de systèmes d'apprentissage automatique différents qu'il est utile de les classer en grandes catégories [17]:

- Selon que l'apprentissage s'effectue ou non sous supervision humaine : Apprentissage supervisé, non supervisé , semi-supervisé ou avec renforcement;
- Selon que l'apprentissage s'effectue ou non progressivement, au fur et à mesure : apprentissage en ligne ou apprentissage groupé;
- Selon qu'il se contente de comparer les nouvelles données à des données connues, ou qu'il détecte au contraire des éléments de structuration dans les données d'entrainement et construit un modèle prédictif à la façon d'un scientifique : apprentissage à partir d'observation ou à partir d'un modèle.

3.2.1. Apprentissage supervisé

L'approche de l'apprentissage supervisé consiste à utiliser des données étiquetées qui entraînent des algorithmes pour classer les nouvelles données ou prédire des résultats avec précision. Le modèle exploite les données étiquetées pour mesurer la pertinence des différentes caractéristiques afin d'affiner progressivement l'ajustement du modèle en fonction du résultat connu. Il existe deux grandes catégories d'apprentissage supervisé : la classification et la régression [18].

- **La classification :** Un problème de classification utilise des algorithmes pour classer les données dans des segments spécifiques. Voici quelques algorithmes de classification courants : la régression logistique, la méthode des k plus proches voisins, la classification naïve bayésienne, l'algorithme du gradient stochastique et l'arbre de décision.
- **La régression :** Il s'agit d'une méthode statistique qui s'appuie sur des algorithmes pour mesurer la relation entre une variable dépendante et une

ou plusieurs variables indépendantes. La régression ridge, le lasso, et la régression logistique sont des algorithmes de régression courants.

3.2.2. Apprentissage non supervisé

Dans le cas de l'apprentissage non supervisé, des algorithmes sont utilisés pour examiner et regrouper des jeux de données non étiquetés. Ces algorithmes peuvent révéler des schémas inconnus dans les données sans aucune supervision humaine. Il existe trois principales catégories d'algorithmes : le clustering ou le partitionnement, l'association, la réduction de la dimensionnalité [18].

- **Le partitionnement :** Les données non étiquetées sont regroupées à l'aide de techniques de clustering en fonction de leurs similitudes ou de leurs différences. K-means et DBSCAN sont les algorithmes les plus utilisés dans cette catégorie.
- **L'association :** La méthode d'association de l'apprentissage non supervisé est intéressante pour trouver des relations entre les variables d'un jeu de données. Les algorithmes courants sont : Apriori, FP-Growth et Éclat.
- **La réduction de dimensionnalité :** Il arrive qu'un jeu de données comporte un nombre exceptionnellement élevé de caractéristiques. La réduction de la dimensionnalité permet de réduire ce nombre sans compromettre l'intégrité des données.

3.3. Arbre de décision

Un arbre de décision est un modèle hiérarchique qui se comporte comme une série successive de tests conditionnels, dans laquelle chaque test dépend de ses antécédents [16]. On appelle arbre de décision un modèle de prédiction qui peut être représenté sous la forme d'un arbre. Chaque nœud de l'arbre teste une condition sur une variable et chacun de ses enfants correspond à une réponse possible à cette condition. Les feuilles de l'arbre correspondent à une étiquette. Pour prédire l'étiquette d'une observation, on « suit » les réponses aux tests depuis la racine de l'arbre, et on

retourne l'étiquette de la feuille à laquelle on arrive. Un arbre de décision peut servir à décrire les étapes d'un diagnostic ou d'un choix de traitement pour un médecin. La **Figure 3.1** présente un tel arbre de décision.

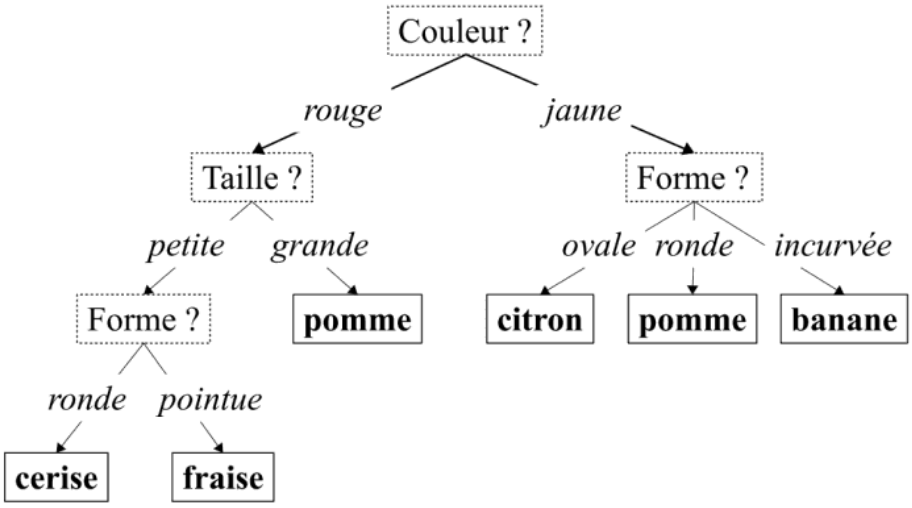

Figure 3.1: Exemple d'arbre de décision pour étiqueter un fruit.

La **Figure 3.1** nous permet d'illustrer trois propriétés des arbres de décision [16]:

- Ils permettent de traiter des attributs discrets (comme ici la forme, la taille et la couleur), sans nécessiter une notion de similarité ou d'ordre sur ces attributs (on parle d'apprentissage non métrique).
- Ils permettent de traiter un problème de classification multi-classe sans passer par des classifications binaires.
- Ils permettent de traiter des classes multimodales (comme ici pour l'étiquette « pomme », qui est affectée à un fruit grand et rouge ou à un fruit jaune et rond).

3.3.1. Structure , définitions et concepts de base d'arbre de décision

Le fonctionnement d'un arbre de décision repose sur des heuristiques construites selon des techniques d'apprentissage supervisé [19]. Les arbres de décision ont une

structure hiérarchique et sont composés de **nœuds** et de feuilles (aussi appelées nœuds terminaux) reliées par des **branches**. Dans leur représentation graphique, la racine est placée en haut et les feuilles en bas. Les nœuds internes sont appelés des nœuds de décision. Ils peuvent contenir une ou plusieurs **règles** (aussi appelées tests ou conditions).

Les valeurs qu'une **variable** peut prendre dans un arbre de décision sont appelées instances ou attributs. Les nœuds terminaux contiennent la classe, également appelée **classe à prédire** ou variable cible. Après sa construction, un arbre de décision peut être traduit sous la forme d'un ensemble de règles de décision.

Lorsque la classe à prédire est une variable qualitative, nous avons un **arbre de classification**. Si la classe à prédire est une variable quantitative, nous avons un **arbre de régression**.

3.3.2. Algorithme de base

L'algorithme d'arbre de décision peut être décrit selon le pseudo-code ci-dessous (**Algorithme 3.1**) [20]. Il permet de générer itérativement l'arbre en prenant à chaque itération une variable et en lui créant ses nœuds et ses feuilles. Cet algorithme procède de manière descendante et ne prend pas en considération l'évaluation de la variable choisie. Plusieurs méthodes de construction d'arbres de décision permettent d'évaluer les différentes variables en se basant sur l'entropie ou le gain d'informations.

1. Initialiser l'arbre courant à l'arbre vide ; la racine est le nœud courant
2. **Répéter**
3. Décider si le nœud courant est terminal (une feuille).
4. **Si** le nœud est terminal **Alors**
5. Affecter une classe au nœud courant
6. **Sinon**
7. Sélectionner un test et créer autant de nouveaux fils qu'il y'a de possibilités de réponses
8. **FinSi**
9. Passer au nœud suivant
10. **Jusqu'à** l'obtention de l'arbre de décision

Algorithme 3.1: Pseudo-code d'algorithme de base d'un arbre de décision

L'idée derrière la construction de l'arbre de décision est de sélectionner l'attribut qui offre le plus grand gain d'information, c'est-à-dire qui réduit l'entropie de l'ensemble de données initial de la manière la plus significative.

3.3.3. Entropie et gain d'informations

L'entropie et le gain d'information sont en effet des concepts essentiels dans la construction d'arbres de décision.

3.3.3.1. Entropie

L'entropie est une mesure de l'incertitude au sein d'un ensemble de données.

Soit une distribution de probabilité $P = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n)$. L'entropie de la distribution P est la quantité d'information qu'elle peut apporter. Elle est donnée par l'équation :

$$E(P) = -\sum_{i=1}^{n} \mathcal{P}_i \log \mathcal{P}_i \tag{3.1}$$

Où $\log$ désigne la fonction logarithme en base 2 [20].

Soit un ensemble de données T caractérisé par classes $(C_1, C_2, \ldots, C_n)$ selon la variable cible. La quantité d'informations nécessaire pour identifier la classe d'un individu de l'ensemble T correspond à l'entropie $E(P)$, Où P est la distribution de probabilité de la partition $(C_1, C_2, \ldots, C_n)$:

$$P = \left(\frac{|C_1|}{|T|}, \frac{|C_2|}{|T|}, \ldots, \frac{|C_n|}{|T|}\right) \tag{3.2}$$

Où $|C_i|$ désigne le cardinal de la classe i, c'est-à-dire nombre d'éléments de la classe i.

L'entropie de T est déduite à partir de l'´équation (3.1), qui donne :

$$E(T) = -\sum_{i=1}^{n} \frac{|C_i|}{|T|} \log \frac{|C_i|}{|T|} \tag{3.3}$$

Supposons que l'ensemble des données T est également partitionné $(T_1, T_2, \ldots, T_m)$, où m correspond au nombre de valeurs que peut prendre l'attribut X considéré. Dans ce cas, l'information nécessaire pour identifier la classe d'un individu de T_i devient :

$$E(X,T) = -\sum_{j=1}^{m} \frac{|T_j|}{|T|} E(T_j) \tag{3.4}$$

3.3.3.2. Gain d'informations

Le gain d'information mesure la réduction d'entropie obtenue en divisant un ensemble de données en fonction d'un attribut de l'ensemble de données. Soit un ensemble de données T, le gain d'informations de T par rapport à une partition T_j donnée est la variation d'entropie causée par la partition de T selon T_j.

$$Gain(X,T) = E(T) - E(X,T) = E(T) - \sum_{j=1}^{m} \frac{|T_j|}{|T|} E(T_j)$$

L'idée sous-jacente à la construction de l'arbre de décision est de choisir l'attribut offrant le gain d'information le plus élevé, ce qui signifie qu'il réduit l'entropie de l'ensemble de données initial de manière significative. L'attribut avec le gain d'information le plus élevé est sélectionné comme le nœud de décision principal pour diviser l'ensemble de données. Ce processus est ensuite répété de manière récursive pour construire l'arbre de décision complet.

3.3.3.3. Indice de Gini

L'indice de Gini a été introduit par Breiman en 2001 [21]. Cet indice mesure l'impureté, un concept très utile dans la construction des arbres de décision : la qualité d'un nœud et son pouvoir discriminant peuvent être évalués par son impureté. L'indice de Gini est donné par la relation suivante :

$$Gini(T) = 1 - \sum_{j=1}^{m} \left(\frac{|T_j|}{|T|} \right)^2$$

Une fois que l'ensemble des données T est caractérisé par une variable X partitionnée ainsi : $(T_1, T_2, \ldots, T_m)$.

$$Gain\ de\ Gini(X,T) = \sum_{j=1}^{m} \frac{|T_j|}{|T|} Gini(T_j)$$

3.3.3.4. Algorithme CART

L'algorithme couramment utilisé pour entrainer un arbre de décision est appelé CART, pour « *Classification And Regression Tree* ». Il s'agit d'un algorithme de partitionnement de l'espace par une approche gloutonne, récursive et divisive. En effet, cet algorithme permet de construire l'arbre en maximisant l'indice de Gini, il génère des arbres de décision binaires comme l'illustre la **Figure 3.2**.

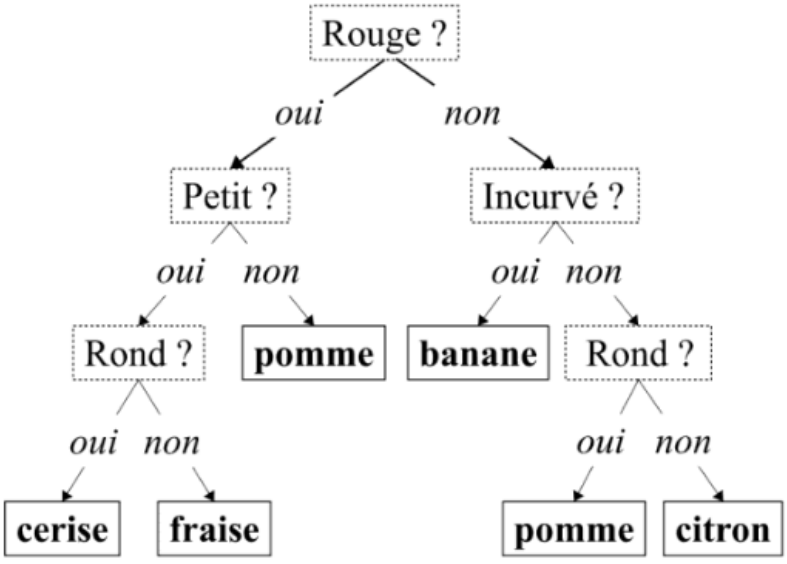

Figure 3.2: Version binaire de l'arbre de décision de figure 3.1 [16]

L'algorithme CART est capable de gérer des données qualitatives et quantitatives, des ensembles de données de grande taille, et il est robuste face aux valeurs manquantes. Une fois l'arbre de décision construit, il peut être utilisé pour

effectuer des prédictions pour de nouvelles données en traversant l'arbre de la racine aux feuilles en fonction des valeurs des attributs.

3.3.4. Avantages et limites des arbres de décision

Les arbres de ´décision sont utilisés dans plusieurs domaines allant du domaine médical au marketing, en raison des nombreux avantages qu'ils présentent. En effet, ces structures sont facilement compréhensibles par les utilisateurs et offrent une classification rapide et visuelle des données à travers le parcours d'un chemin dans l'arbre. De plus, il s'agit d'un outil disponible dans la plupart des environnements d'exploration de données [19].

Cependant, ces structures présentent deux limites importantes. En effet, ces approches sont sensibles au nombre de classes ; ainsi, si le nombre de classes devient important, les résultats se dégradent. De plus, ces structures ne sont pas très flexibles : si les données changent avec le temps, il devient nécessaire de reconstruire l'arbre de décision. Heureusement, il est possible d'y remédier grâce aux méthodes ensemblistes.

3.4. Techniques d'apprentissage d'ensemble

Les méthodes ensemblistes sont des méthodes qui consistent à combiner plusieurs modèles afin de générer un nouveau modèle potentiellement plus performant, mais aussi plus robuste [22]. Cette combinaison s'effectue généralement de deux façons : via du Bagging ou du Boosting.

3.4.1. Bagging

Les méthodes de bagging (**b**ootstrap **agg**regat**ing**) consistent à combiner des modèles appris à partir d'informations différentes, c'est-à-dire à partir d'échantillons différents. Ces modèles sont ensuite combinés, généralement par sommation, afin de créer un modèle unique et performant [22]. Le processus commence par la creation de plusieurs sous-ensembles d'apprentissage en utilisant un échantillonnage

bootstrap. Le bootstrap est une méthode d'échantillonnage aléatoire avec remplacement.

On considère un échantillon d'apprentissage S et on effectue la procédure suivante T fois, où T représente le nombre de modèles que l'on souhaite apprendre :

i. On tire un échantillon S_t avec remise à partir de S ;
ii. On apprend un modèle h_t en en utilisant l'échantillon S_t.

Une fois que les T modèles sont appris, on se retrouve avec un modèle global H_t que l'on écrira :

$$H_T = \frac{1}{T}\sum_{t=1}^{T} h_t$$

Un algorithme emblématique reposant sur du bagging est celui des forêts aléatoires.

3.4.1.1. Forêts aléatoires

Comme son nom l'indique, une forêt aléatoire consiste à agréger des arbres de classification ou de régression. Une forêt aléatoire est donc une agrégation d'arbres dépendants de variables aléatoires. Une famille de forêt aléatoire se distingue parmi les autres, notamment par la qualité de ses performances sur de nombreux jeux de données [23]. Il s'agit de forêt aléatoire dont les arbres sont construits avec l'algorithme CART. Le pseudo-code est décrit par **Algorithme 3.2**.

Entrées :

-x l'observation à prévoir ;

-d_n l'échantillon ;

- B le nombre d'arbres ;

- m $\in$ N^* le nombre de variables candidates pour découper un nœud.

Pour $k = 1, \dots, B$:

1. Tirer un échantillon bootstrap d_n.
2. Construire un arbre CART sur cet échantillon bootstrap, chaque coupure est sélectionnée en minimisant la fonction de cout de CART sur un ensemble de m variables choisies au hasard parmi les p. On note $h(., \theta_k)$ l'arbre construit.

Sortie : L'estimateur $h(x) = \frac{1}{B} \sum_{k=1}^{B} h(x, \theta_k)$

Algorithme 3.2: Algorithme forêt aléatoire

Remarque :

- Si nous sommes dans un contexte de classification, l'étape finale d'agrégation dans l'algorithme consiste à faire voter les arbres pour déterminer la classe prédite.
- On retrouve un compromis biais-variance dans le choix de m:
  - Lorsque m diminue, la tendance est à se rapprocher d'un choix "aléatoire" des variables de découpe des arbres. Dans le cas extrême où $m = 1$, les axes de la partition des arbres sont choisis au "hasard", seuls les points de coupure utiliseront l'échantillon. Ainsi, si m diminue, la corrélation entre les arbres va avoir tendance à diminuer également, ce qui entraînera une baisse de la variance de l'estimateur agrégé. En revanche, choisir les axes de découpe des arbres de manière (presque) aléatoire va se traduire par une moins bonne qualité d'ajustement des arbres sur l'échantillon d'apprentissage, d'où une augmentation du biais pour chaque arbre ainsi que pour l'estimateur agrégé.
  - Lorsque m augmente, les phénomènes inverses se produisent.

On déduit de cette remarque que le choix de m est lié aux paramètres de l'arbre, notamment au choix du nombre d'observations dans ses nœuds terminaux. En effet, si ce nombre est petit, chaque arbre aura un biais faible mais une forte variance. Dans ce cas, Il faudra donc s'attacher à diminuer cette variance, ce qui conduira à choisir une valeur de m relativement faible. A l'inverse, si les arbres ont un grand nombre d'observations dans leurs nœuds terminaux, ils posséderont moins de variance mais un biais plus élevé. Dans ce cas, la procédure d'agrégation peut se révéler moins efficace.

3.4.2. Boosting

Si dans bagging, les modèles sont appris de manière indépendante les uns des autres, l'idée du boosting est de rompre cette indépendance.

On suppose que les modèles appris ont un faible pouvoir prédictif et on souhaite créer une combinaison de ces modèles afin que cette dernière soit plus performante. Pour faire cela, on va apprendre nos modèles de façon itérative et cela, de sorte que le modèle appris à l'itération actuelle soit capable de corriger les erreurs effectuées par le modèle précédent.

3.4.2.1. Algorithme Adaboost

Le premier algorithme de boosting qui a été développé et qui répond à cette problématique est l'algorithme Adaboost [24]. La présentation de cette procédure est donnée par **Algorithme 3.3**:

Entrées : Echantillon d'apprentissage S de taille m, un nombre T de modèles
Sortie : Un modèle $H_T = \sum_{t=0}^{T} \alpha_t h_t$
Début :

Distribution uniforme $\omega_i^{(0)} = \frac{1}{m}$

Pour $t = 1, \ldots, T$ **faire**

Apprendre un classifieur h_t à partir d'un algorithme $\mathcal{A}$
Calculer l'erreur ε_t de l'algorithme
Si $\varepsilon_t > ½$ **alors**
Arrêt
Sinon
Calculer $\alpha_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
$\omega_i^{(t)} = \omega_i^{(t-1)} \frac{\exp(-\alpha_t y_i h_t(X_i))}{Z_t}$

Poser $H_T = \sum_{t=0}^{T} \alpha_t h_t$

Retourner : H_T

Algorithme 3.3: Algorithme Adaboost

On dispose initialement de notre échantillon S avec nos m exemples (X_i, Y_i) et toutes les données ont le même poids, c'est-à-dire la même importance.

On va maintenant regarder comment une hypothèse h_{t+1} est apprise en fonction des performances du classifieur h_t .

Pour cela, plaçons-nous à une étape t de notre algorithme où les exemples ont un poids égal à $\omega_i^{(t)}$. Une hypothèse h_t est alors apprise et nous pouvons évaluer son taux d'erreur en classification ϵ_t .

$$\epsilon_t = \sum_{i=1}^{m} \omega_i^{(t)} 1_{\{h_t(x_i) y_i < 0\}}$$

A partir de cette erreur, nous allons déterminer une quantité α_t définie par :

$$\alpha_t = \frac{1}{2} ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

Et qui va permettre de quantifier l'importance de l'hypothèse h_t dans la décision finale, c'est-à-dire on va définir un poids sur le classifieur appris.

Le reste de la procédure consiste ensuite à trouver une bonne repondération des exemples de façon que l'hypothèse qui sera apprise à l'itération suivante, puis se focaliser sur les erreurs commises par l'hypothèse actuelle, cela se fait par la mise à jour suivante :

$$\omega_i^{(t+1)} = \omega_i^{(t)} \frac{exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Où Z_t est un facteur de normalisation permettant d'avoir une distribution sur les poids des exemples.

3.4.2.2. XGBoost

XGBoost pour « eXtreme Gradient Boosting » est une variante de l'algorithme de boosting de gradient qui a été utilisée par de nombreux lauréats de compétitions en science de données. Il s'agit d'une généralisation du boosting dans laquelle la fonction de perte est utilisée de manière similaire à une descente de gradient. Le gradient boosting a émergé lorsque des chercheurs ont reformulé l'algorithme Adaboost dans un cadre statistique plus formel, où il n'était qu'une optimisation numérique visant à minimiser une fonction de perte de manière successive, à la manière d'une descente de gradient [25]. En effet, la descente de gradient minimise une fonction objective en suivant la pente de cette fonction à l'aide de son gradient ou bien d'une approximation de celui-ci. XGBoost utilise une variante de gradient appelée descente de gradient fonctionnelle.

L'algorithme se présente comme suit (voir **Algorithme *3.4***):

1. Initialisation $f_0 = h_0$.
2. Pour $b = 1, \dots, B$
   (a) Ajuster un arbre h_b à T feuilles qui minimise

$$\sum_{i=1}^{n}\left[l_i^{(1)}h_b(x_i) + \frac{1}{2}l_i^{(2)}h_b^{(2)}(x_i)\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T}\omega_j$$

   (b) Mettre à jour

$$f_b(x) = f_{b-1}(x) + h_b(x).$$

3. Sortie : la suite d'algorithmes $(f_b)_b$

Algorithme 3.4: Algorithme XGBoost

Où $l_i^{(1)}$: est la fonction de perte, c'est-à-dire une fonction des apprenants CART à l'itération i .

h_b : est un arbre d'indice b .

ω_j : les poids à un nœud j.

$\gamma\ et\ \lambda$: ce sont les facteurs de régularisations pour capturer la complexité du modèle.

Il est facile de voir que l'objectif XGBoost est fonction de fonctions (c'est-à-dire que l est une fonction des apprenants CART, une somme des arbres additifs actuels et précédents).

Pour résumer, l'algorithme du gradient boosting a besoin de 3 éléments principaux :

- Une fonction de perte à optimiser, qui doit être différentiable, qui permet de résoudre le problème.
- Un apprenant faible pour effectuer des prédictions. On peut utiliser des arbres un peu plus grands.
- Un modèle additif pour combiner nos apprenants faibles afin de minimiser notre fonction de perte. C'est à dire avancer dans notre fonction de perte

en suivant le gradient, ce qui pourra être effectué en ajoutant un arbre de décision supplémentaire. On effectue cette procédure en paramétrant l'arbre, et ensuite en modifiant ces paramètres en allant dans la direction du gradient.

❖ Avantages de XGBoost

- **Performances :** XGBoost a prouvé son efficacité en produisant des résultats de haute qualité dans diverses tâches d'apprentissage automatique. Il a particulièrement excellé dans les compétitions Kaggle, où il a été largement adopté comme méthode de prédilection pour les solutions gagnantes.
- **Gestion des valeurs manquantes :** XGBoost intègre une gestion native des valeurs manquantes, ce qui simplifie l'utilisation des données réelles fréquemment marquées par des absences de valeurs.
- **Interprétabilité :** Contrairement à certains algorithmes d'apprentissage automatique, souvent complexes à interpréter, XGBoost offre des indications sur l'importance des caractéristiques, permettant ainsi de mieux identifier les variables les plus déterminantes pour les prédictions.

❖ Inconvénients de XGBoost

- **Surajustement :** XGBoost peut être sujet au surajustement, notamment lorsqu'il est entraîné sur de petits ensembles de données ou lorsqu'un nombre excessif d'arbres est utilisé dans le modèle.
- **Exigences en matière de mémoire :** XGBoost peut se révéler très gourmand en mémoire, surtout lorsqu'il est appliqué à de grands ensembles de données, ce qui le rend moins approprié pour les systèmes disposant de ressources mémoire limitées.
- **Interprétabilité :** Bien que XGBoost fournisse des informations sur l'importance des caractéristiques, il reste moins interprétable que des

modèles plus simples, comme les arbres de décision classiques ou les régressions linéaires.

3.5. Conclusion

En résumé, les méthodes d’ensemble offrent différentes manières de combiner plusieurs modèles afin d’améliorer la précision prédictive. Le choix de la méthode dépend du problème à résoudre, de la diversité des modèles de base et des objectifs spécifiques. En comprenant et en utilisant efficacement ces techniques d'ensemble, il est possible de créer un modèle d'apprentissage automatique plus robuste et plus précis.

Chapitre 4. Apprentissage non supervisé et réseaux de neurones artificiels

Nous allons découvrir dans cette partie les différentes tâches d'apprentissage non supervisé et quelques réseaux de neurones artificiels.

4.1. Introduction

L'objectif de l'apprentissage non supervisé n'est pas de faire des prédictions, mais plutôt de modéliser la structure ou la distribution sous-jacente dans les données qui sont étudiées afin d'en apprendre davantage sur celles-ci [15]. On étudiera deux tâches principales à savoir la réduction de dimensionnalité et le partitionnement.

4.2. Techniques d'apprentissage non supervisé

4.2.1. La réduction de dimensionalité

Pourquoi réduire la dimension de nos données ? Nous pouvons assez facilement représenter des données en 2 ou 3 dimensions. Mais au-delà, on est obligé de regarder les variables paires par paires (ou triplet par triplet). Cela devient rapidement difficile, voire impossible, quand le nombre de dimensions augmente. Si l'on pouvait représenter nos données avec 2 ou 3 dimensions, cela serait beaucoup plus simple [15]. Représenter les données en quelques dimensions, permet de réduire le coût en espace mémoire, de réduire la complexité des algorithmes d'apprentissage et donc le temps de calcul.

4.2.1.1. Analyse en composantes principales (ACP)

L'ACP est une méthode de réduction de dimensions des données pour faciliter leur utilisation.

L'ACP a deux objectifs principaux, elle permet d'étudier [26]:

- La variabilité entre les individus, c'est-à-dire les différences et les ressemblances entre les individus;

- Les liaisons entre les variables, c'est-à-dire les corrélations entre les variables qui peuvent être regroupés en de nouvelles variables synthétiques.

Le principe de l'ACP est de trouver une nouvelle base orthonormée dans laquelle représenter les données de sorte que la variance des données selon ces nouveaux axes soit maximisée [15].

❖ **Calcul des composantes principales**

Supposons p variables représentées par un vecteur $\omega \in \mathbb{R}^p$ et n observations rassemblées dans une matrice $X \in \mathbb{R}^{p\ x\ n}$.

La projection des données $X\ sur\ \omega$ est : $\mathrm{Z}_1 = \omega_1^T X$ supposons que X soit centrée ; la variance de $\omega_1^T X$ est : $Var(\omega_1^T X) = E[(\omega_1^T X - \mathrm{E}[\omega_1^T X])^2]$.

Comme $E(X) = 0$ car X est centrée, on a donc $Var(\omega_1^T X) = \omega_1^T \mathrm{E}(X X^T)\ \omega_1$.

On note Σ la matrice de taille $p\ x\ p$ est égale à $X X^T$, Σ est la matrice de covariance des données. Nous allons donc chercher ω_1 tel que :

- $\omega_1^T \Sigma\ \omega_1$ soit maximale;
- $\| \omega_1 \| = 1$

On sait que $\omega_1^T \mathrm{E}(X X^T)\ \omega_1 > 0$ donc Σ est une matrice symétrique définie positive par conséquent elle est donc diagonalisable par un changement de base orthonormée : $\Sigma = Q^T \Lambda\ Q$, où $\Lambda \in \mathbb{R}^{p\ x\ p}$ est une matrice diagonale dont les valeurs diagonales sont les valeurs propres de Σ, qui sont toutes positives.

On a donc $\omega_1^T \Sigma\ \omega_1 = \omega_1^T Q^T \Lambda\ Q\ \omega_1 = (Q\ \omega_1)^T \Lambda (Q\ \omega_1)$

On pose $V = Q\ \omega_1$, on cherche donc à maximiser : $\sum_{j=1}^{p} v_j^2 \lambda_j$

Comme $\| \omega_1 \| = 1$ et Q est orthonormée, $\| V_1 \| = 1$ alors $\sum_{j=1}^{p} v_j^2 \lambda_j = 1$.

La somme $\sum_{j=1}^{p} v_j^2 \lambda_j$ est donc maximisée pour un vecteur V qui a une seule entrée à 1 et les autres à 0, l'entrée 1 étant pour la valeur maximale des λ_j, autrement dit la plus grande valeur propre de la matrice Σ. Autrement dit, λ_1 est le vecteur propre correspondant à la plus grande valeur propre de Σ, c'est la première composante principale de X.

La deuxième composante principale de X doit vérifier les mêmes critères que ω_1, c'est-à-dire la norme égale 1 et maximiser $\omega_2^T \Sigma\, \omega_2$, mais lui est orthogonale. Il s'agit donc du vecteur propre de Σ correspondant à la deuxième plus grande valeur propre. Et on calcule ainsi de suite les autres composantes principales. On utilise comme nouvelles dimensions, la base formée par les vecteurs propres de la matrice des données.

- **Comment choisir le nombre de composantes principales ?**

On remarque que par cette méthode, on vient de construire autant de composantes principales que Σ a de vecteurs propres, soit autant que le nombre de descripteurs p des données. Nous n'avons pas encore réduit la dimention des données. Pour ce faire, on regarde la proportion de variance expliquée par chacune des composantes principales.

La variance totale est donnée par la somme des termes diagonaux de la matrice de covariance Σ, autrement dit sa trace $Tr(\Sigma) = \lambda_1 + \lambda_2 + \ldots + \lambda_p$

La variance totale est donnée par $\lambda_1 + \lambda_2 + \ldots + \lambda_p$ et celle expliquée par les k premières composantes principales, par $\lambda_1 + \lambda_2 + \ldots + \lambda_k$. La proportion de variance expliquée par les k premières composantes principales est donc $\frac{\lambda_1 + \lambda_2 + \ldots + \lambda_k}{\lambda_1 + \lambda_2 + \ldots + \lambda_p}$

On note aussi que:

- Retenir tous les p facteurs équivaut à garder toute l'information initiale, et donc à ne pas simplifier la structure de liaisons entre les variables;

- Inversement, ne garder qu'un petit nombre de facteurs peut revenir à n'expliquer qu'un pourcentage trop faible de variance totale, et donc à résumer l'information de manière excessive.

Voici 2 des critères permettant de faire apparaitre la décision du choix de critères comme plus objective et moins arbitraire.

Premier critère : (Critère de Joliffe) : S'arrêter dès que la proportion expliquée de la variance globale atteint une proportion fixée, par exemple 90%;

Deuxième critère : (Critère de Catell) faire le diagramme des valeurs propres consistant à représenter les λ_α en fonction de α , souvent, un coude apparaît, marquant un changement de régime dans la décroissance des valeurs propres. Il ne faut alors que garder les valeurs propres avant l'apparition du coude.

4.2.2. Partitionnement

Le partitionnement est une technique utilisant des entités de données similaires, lesquelles sont regroupées en fonction de certaines variables. Comme en classification, chaque observation est affectée à un groupe. Cependant ,contrairement à la classification, le partitionnement est une tâche non supervisée [17]. Dans cette section nous allons décrire deux algorithmes de partitionnement bien connu, K-means et DBSCAN.

4.2.2.1. K-means

K-means est une technique basée sur le centroïde qui divise l'ensemble de données en *k* clusters (groupes) distincts, où chaque point de données appartient au cluster dont le centre est le plus proche. C'est l'une des méthodes de partitionnement les plus simples et les plus efficaces. Cet algorithme a été initialement introduit par Stuart Lloyd en 1957 [27], c'est pourquoi il est souvent appelé algorithme de Lloyd. K-means a été utilisé avec succès dans de nombreuses applications dans divers domaines tels que la segmentation des clients, la compression d'images, le regroupement de documents et la détection d'anomalies.

L'algorithme commence par choisir k centroïdes initiaux $c_1, \ldots, c_2$. Cela peut être fait de manière aléatoire en sélectionnant k points de données dans l'ensemble de données ou en utilisant diverses autres méthodes d'initialisation.

Ensuite, l'algorithme alterne entre les deux étapes (voir **Algorithme *4.1***):

i. **Étape d'affectation :** attribuez chaque point de données au cluster avec le centroïde le plus proche, c'est-à-dire le centroïde avec la distance euclidienne la plus petite au carré :

$$S_i^{(t)} = \left\{ X_p : \left\| X_p - C_i^{(t)} \right\|^2 \leq \left\| X_p - C_j^{(t)} \right\|^2 \ \forall\, j, 1 \leq j \leq k \right\}$$

Où $S_i^{(t)}$ est l'ensemble des points qui appartiennent au cluster i à l'itération t et $C_i^{(t)}$ est le centroïde du cluster i à l'itération t.

ii. **Étape de mise à jour :** pour chaque cluster, calculez un nouveau centroïde en calculant la moyenne des points de données attribués à ce cluster :

$$C_i^{(t+1)} = \frac{1}{\left| S_i^{(t)} \right|} \sum_{X_j \in S_i^{(t)}} X_j$$

Algorithme 4.1: Algorithme de K-means

Ces deux étapes sont répétées jusqu'à ce que les affectations de cluster ne changent plus ou qu'un nombre spécifié d'itérations soit atteint.

L'algorithme est assuré de converger vers une solution. Cependant, la solution pourrait être un minimum local, et pas nécessairement l'optimum global. En raison de la sensibilité de l'algorithme au choix initial des centroïdes, il est courant d'exécuter les k-means plusieurs fois avec des initialisations différentes et de sélectionner le résultat avec la somme des carrés des distances la plus basse .

$$E = \sum_{i=1}^{k} \sum_{X_j \in S_i} \left\| X_j - C_i \right\|^2$$

Il existe plusieurs types de distance pour calculer la distance entre les objets et les k centroïdes. Parmi ces distances, on peut citer :

- La distance Euclidienne.
- La distance de Minkowsky.
- La distance de Manhattan.

La méthode K-means est rapide, simple à appliquer et s'adapte à de larges bases de données. Cependant, elle a de difficulté de comparer la qualité des clusters obtenus et les résultats peuvent être influencés par le choix du paramètre k.

4.2.2.2. DBSCAN

L'algorithme DBSCAN, proposé en 1996 par Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiaowei Xu (Ester et al., 1996) [16], partitionne les données en créant des clusters de points atteignables par densité les uns depuis les autres. C'est un algorithme de partitionnement populaire, qui a obtenu en 2014 une distinction de contribution scientifique ayant résisté à l'épreuve du temps, le *test of time award* de la prestigieuse conférence KDD[1].

On appelle DBSCAN, pour Density-Based Spatial Clustering of Applications with Noise, ou partitionnement dans l'espace par densité pour des applications bruitées, la procédure de partitionnement suivante :

- Initialisation : Un ensemble d'éléments visités V = ∅, une liste de clusters K = ∅, une liste d'observations aberrantes A = ∅.
- Pour chaque élément $\vec{x} \in D \setminus V$:
    i. Construire $N_\epsilon(\vec{x})$
    ii. Si $|N_\epsilon(\vec{x})| < n_{min}$: considérer (temporairement) $\vec{x}$ comme une observation aberrante :

$$A \leftarrow A \cup \{\vec{x}\}$$

[1] https://kdd.org/conferences

Sinon :

- Créer un cluster $C \leftarrow \{\vec{x}\}$
- Augmenter ce cluster par la procédure *grow_cluster(C,* $N_{\in}(\vec{x}), \epsilon, n_{min}$ *)*

iii. Ajouter C à la liste des clusters : $K \leftarrow K \cup C$

iv. Marquer tous les éléments de C comme visités : $V \leftarrow V \cup C$

La procédure *grow_cluster(C,* $N_{\in}(\vec{x}), \epsilon, n_{min}$ *)* est définie comme suit :

Pour tout $\vec{u} \in N \setminus V$:

- Créer $N' \leftarrow N_{\in}(\vec{u})$
- Si $|N'| \geq n_{min}$: actualiser N de sorte à prendre les éléments de N' en considération : $N \leftarrow N \cup N'$
- Si $\vec{u}$ n'appartient à aucun autre cluster, l'ajouter à C : $C \leftarrow C \cup \{\vec{u}\}$
- Si $\vec{u}$ était précédemment cataloguée comme aberrante, l'enlever de la liste des observations aberrantes : $A = A \setminus \{\vec{u}\}$

Algorithme 4.2: Algorithme DBSCAN

Un des avantages de DBSCAN est sa robustesse aux données aberrantes, qui sont identifiées lors de la formation des clusters. Le fléau de la dimension rend DBSCAN difficile à appliquer en très grande dimension : les $\epsilon - \text{voisinages}$ auront tendance à ne contenir que leur centre. De plus, la densité étant définie par les paramètres ϵ et n_{min}, DBSCAN ne pourra pas trouver de clusters de densité différente. Cependant, DBSCAN ne requiert pas de prédéfinir le nombre de clusters, et est efficace en temps de calcul.

4.2.2.3. t-SNE (t-distributed Stochastic Neighbor Embedding)

t-SNE est un algorithme d'apprentissage non supervisé connu notamment pour sa capacité à faciliter la visualisation des données non linéaires ayant de nombreux

descripteurs. En d'autres mots, le rôle de cet algorithme est la réduction de la dimensionnalité non linéaire. Cela signifie qu'il permet de séparer les données qui ne peuvent pas être séparées par une ligne droite. Il donne une idée ou une intuition de la façon dont les données sont organisées dans un espace de grande dimension et par conséquent, il produit des groupes séparés et bien définis [28]. Il fonctionne en calculant les similarités entre les points dans l'espace d'origine, puis en cartographiant ces points dans un espace de dimension réduite où les similarités sont représentées de manière plus proche de la réalité. t-SNE se distingue en préservant la structure locale et globale des données, ce qui signifie qu'il essaie de maintenir efficacement les relations entre les points proches et éloignés respectivement dans la visualisation finale. Cela en fait un outil particulièrement utile pour visualiser des ensembles de données à haute dimensionnalité en deux ou trois dimensions.

Cependant, t-SNE présente certaines limitations, telles que sa sensibilité au choix des paramètres et l'interprétation directe des distances dans l'espace de sortie réduit. De plus, pour des ensembles de données très volumineux, il peut être coûteux en termes de calcul.

4.2.2.4. Indice de Davies-Bouldin

L'Indice Davies-Bouldin est une métrique de validation utilisée pour évaluer les modèles de clustering. Il est calculé comme la mesure moyenne de similarité de chaque cluster, le cluster lui étant le plus similaire. Dans ce contexte, la similarité est définie comme le rapport entre les distances inter-clusters et intra-clusters. En tant que tel, cet indice classe les clusters bien séparés avec moins de dispersion comme ayant un meilleur score [29].

4.3. Réseaux de neurones artificiels

Un réseau de neurones artificiels (RNA) est un modèle d’apprentissage automatique inspiré des réseaux de neurones biologiques que l’on trouve dans notre cerveau. Cependant, même si les avions ont les oiseaux pour modèle, ils ne battent

pas des ailes. De façon comparable, les RNA sont progressivement devenus assez différents de leurs cousins biologiques.

Les RNA sont au cœur de l'apprentissage profond. Ils sont polyvalents, puissants et extensibles, ce qui les rend parfaitement adaptés aux tâches d'apprentissage automatique extrêmement complexes, comme la classification de milliards d'images (Exemple : Google Images), la reconnaissance vocale (exemple : Apple Siri), la recommandation de vidéos auprès de centaines de millions d'utilisateurs (p. exemple : YouTube) où l'apprentissage nécessaire pour battre le champion du monde du jeu de go (AlphaGo de DeepMind) [30].

Dans cette partie, nous allons comprendre différentes architectures des RNA, dont le perceptron et les autoencodeurs.

4.3.1. Perceptron multicouche

Le perceptron, inventé en 1957 par Frank Rosenblatt [31], est l'une des architectures de RNA les plus simples. Il se fonde sur un neurone artificiel légèrement différent (**Figure 4.1**), appelé unité logique à seuil (TLU, Threshold Logic Unit) ou parfois unité linéaire à seuil (LTU, Linear Threshold Unit) [30]. Les entrées et la sortie sont à présent des nombres (à la place de valeurs binaires, actif/inactif) et chaque connexion en entrée possède un poids. Le TLU calcule une somme pondérée des entrées ($z = \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n = X^T W$) , puis il applique une fonction échelon (step) à cette somme et produit le résultat : $h_1(x) = step(z) = step(X^T W)$.

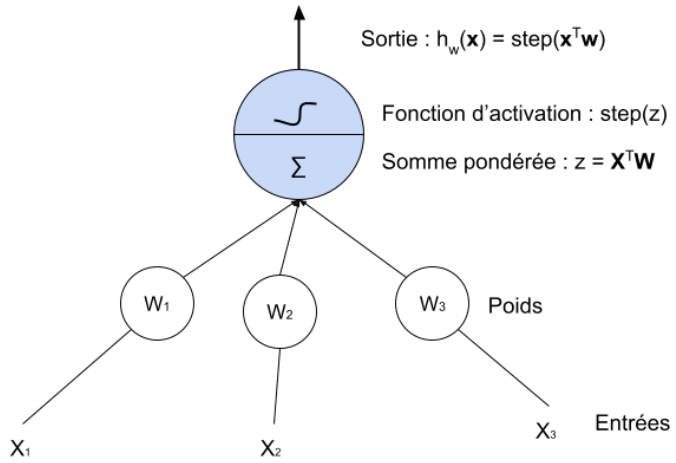

Figure 4.1 : Une unité logique à seuil

Dans les perceptrons, la fonction échelon la plus utilisée est la fonction de Heaviside ou la fonction signe :

$$heaviside(z) = \begin{cases} 0 & si\ z < 0 \\ 1 & si\ z \geq 0 \end{cases} \qquad sign(z) = \begin{matrix} -1 & si\ z < 0 \\ 0 & si\ z = 0 \\ +1 & si\ z > 0 \end{matrix}$$

Un perceptron est constitué d'une seule couche de TLU, chaque TLU étant connecté à toutes les entrées. Lorsque tous les neurones d'une couche sont connectés à chaque neurone de la couche précédente (c'est-à-dire les neurones d'entrée), la couche est une *couche intégralement connectée*, ou *couche dense*. Les entrées du perceptron sont transmises à des neurones intermédiaires particuliers appelés neurones d'entrée: ils se contentent de sortir l'entrée qui leur a été fournie. L'ensemble des neurones d'entrée forment la *couche d'entrée.*

Exemple : Un perceptron doté de deux entrées et de trois sorties est représenté à la **Figure 4.2**.

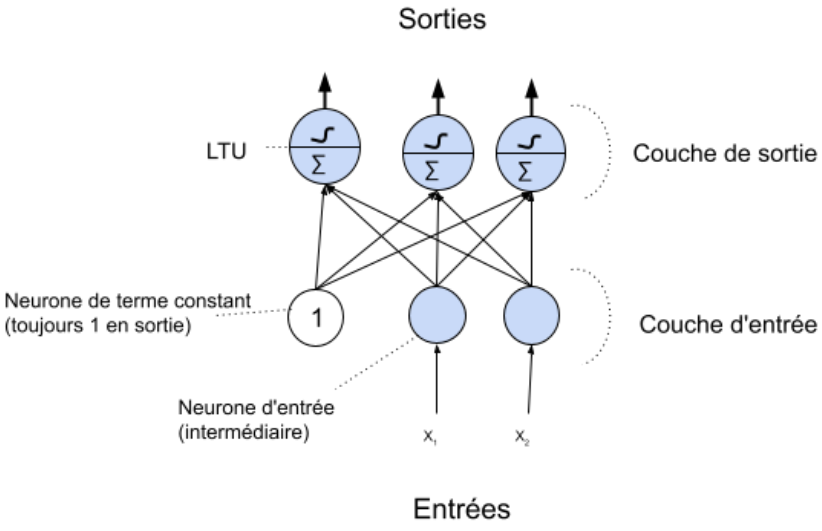

Figure 4.2: Architecture de perceptron

Comment entraîne-t-on un perceptron? L'algorithme d'entraînement du perceptron proposé par Rosenblatt se fonde largement sur la règle de Hebb. Dans son ouvrage The Organization of Behavior publié en 1949 [32], Donald Hebb suggérait que, si un neurone biologique déclenche souvent un autre neurone, alors la connexion entre ces deux neurones se renforce. Autrement dit, le poids d'une connexion entre deux neurones tend à augmenter lorsqu'ils s'activent simultanément. Cette règle d'apprentissage est illustrée par l'équation suivante :

$$\omega_{i,j}^{(étape\ suivante)} = \omega_{i,j} + \eta(y_j - \widehat{y_j})x_i$$

Dans cette équation, on a:

- $\omega_{i,j}$ correspond au poids de la connexion entre le i^{ième} neurone d'entrée et le j^{ième} neurone de sortie;
- x_i est la i^{ième} valeur d'entrée de l'instance d'entraînement courante;
- $\widehat{y_j}$ est la sortie du j^{ième} neurone de sortie pour l'instance d'entraînement courante ;

- y_j est la sortie souhaitée pour le j^{ième} neurone de sortie pour l'instance d'entraînement courante;
- η est le taux d'apprentissage.

4.3.1.1. Architecture de perceptron multicouche

Un Perceptron multicouche(PCM) comporte des couches d'entrée et de sortie, ainsi qu'une ou plusieurs couches cachées avec de nombreux neurones empilés ensemble [33].

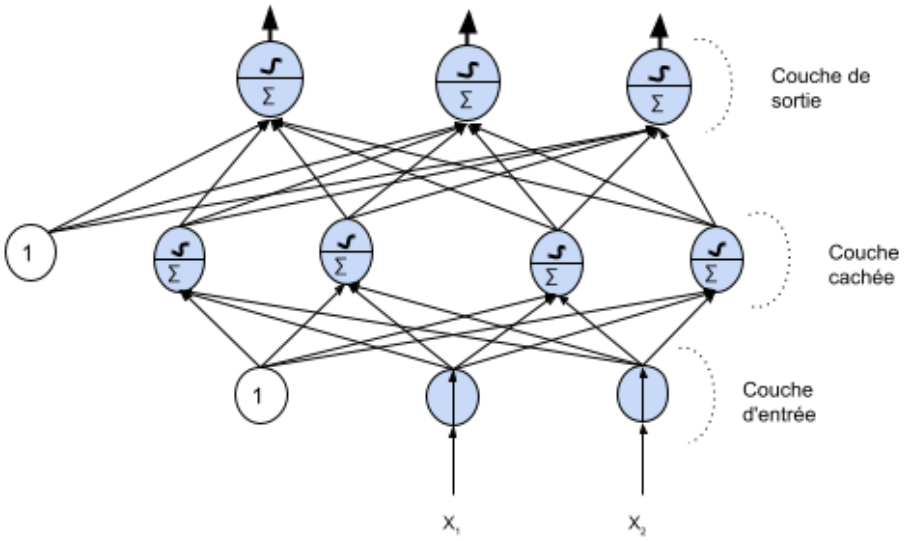

Figure 4.3: Architecture perceptron multicouche

La **Figure 4.3** illustre un perceptron multicouche avec deux entrées, une couche cachée de quatre neurones et trois neurones de sortie. Puisque le signal va dans une seule direction (des entrées vers les sorties), cette architecture est un exemple de réseau de neurones non bouclé (FNN, Feedforward Neural Network). Lorsqu'un RNA possède un grand nombre de couches cachées, on parle de réseau de neurones profond (RNP). Cependant comment entraine-t-on un PCM ? C'est là que l'algorithme de rétropropagation entre en jeu.

4.3.1.2. Rétropropagation

La rétropropagation est le mécanisme d'apprentissage qui permet au PCM d'ajuster de manière itérative les poids dans le réseau, dans le but de minimiser la fonction de coût [33]. Cet algorithme fût introduit en 1986, par David Rumelhart, Geoffrey Hinton et Ronald Williams, et toujours employé aujourd'hui [34]. Examinons cet algorithme (voir **Figure 4.4**).

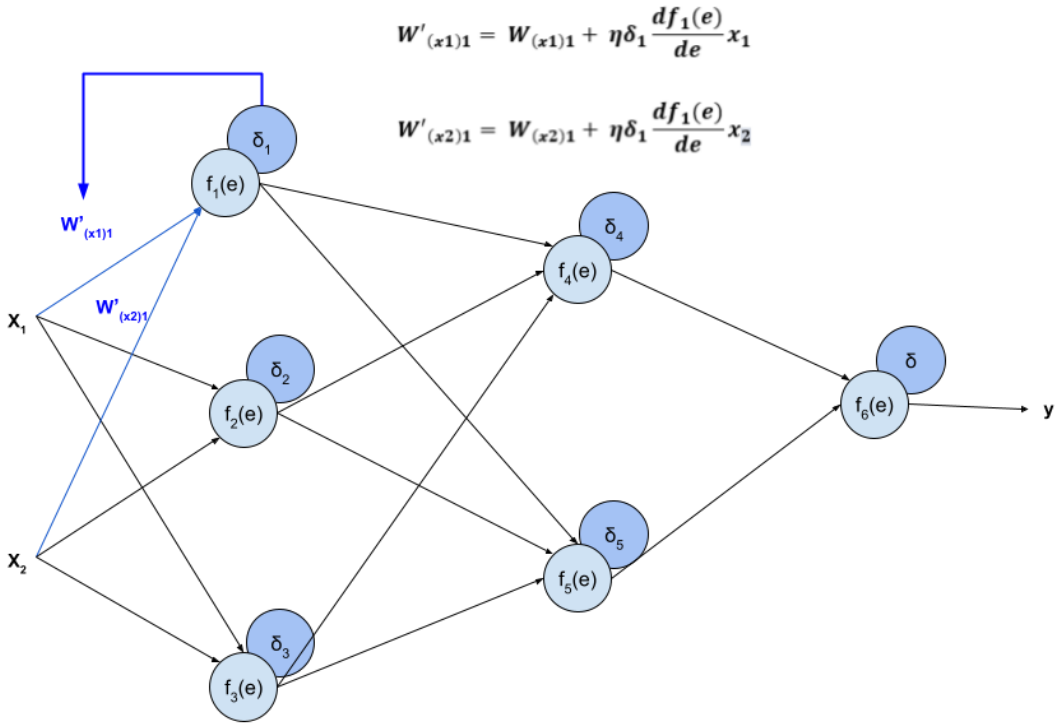

Figure 4.4: Mise à jour des poids et les biais par l'algorithme de descente de gradient.

Il s'agit d'une descente de gradient utilisant une technique efficace de calcul automatique des gradients : en seulement deux passages dans le réseau (un avant, un en arrière), l'algorithme de rétropropagation est capable de calculer le gradient de l'erreur du réseau en lien avec chaque paramètre individuel du modèle. Autrement dit,

il peut déterminer l'ajustement à appliquer à chaque poids de connexion et à chaque terme constant pour réduire l'erreur. Après avoir obtenu les gradients, il réalise simplement une étape de descente de gradient classique. L'intégralité du processus est répétée jusqu'à la convergence du réseau vers la solution.

4.3.2. Autoencodeurs

Les autoencodeurs sont des réseaux de neurones artificiels capables d'apprendre des représentations denses des données d'entrée, appelées représentations latentes ou codages, sans aucune supervision [30]. Ils sont capables de reconstruire une entrée à partir de la sortie. Ces codages ont généralement une dimensionnalité plus faible que les données d'entrée, d'où l'utilité des autoencodeurs dans la réduction de dimensionnalité, et certains autoencodeurs sont des modèles génératifs capables de produire aléatoirement de nouvelles données qui ressemblent énormément aux données d'entraînement.

Un autoencodeur simple ressemble à ceci :

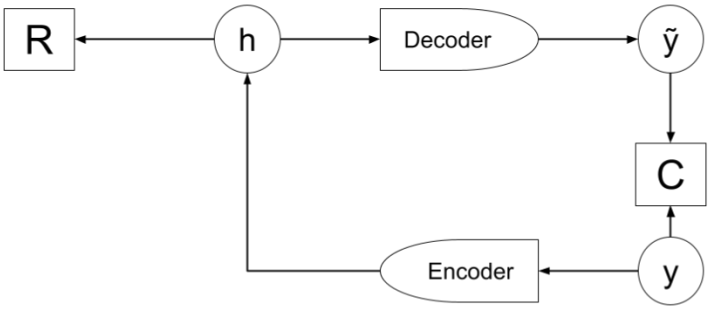

Figure 4.5: Architecture de l'autoencodeur

On a donc:

- y : est observé pendant l'apprentissage mais pas pendant le test ;
- h : est calculé à partir de l'entrée (cachée/interne) ;
- $\tilde{\mathrm{y}}$: est calculé à partir de la cachée (prédit y).
- R : la régularisation
- C : la reconstruction

Un autoencodeur possède généralement la même architecture qu'un perceptron multicouche , mais le nombre de neurones de la couche de sortie doit être égal au nombre d'entrées. Ici, **Figure 4.5**, on utilise un encodeur qui approxime la minimisation et pour z et fournit une représentation cachée h pour une y donnée [35].

$$\mathrm{h} = \mathrm{Encodeur}(\mathrm{y})$$

Ensuite, la représentation cachée est convertie en $\tilde{\mathrm{y}}$.

$$\tilde{\mathrm{y}} = Decodeur(h)$$

Pour résumer une forme très simple d'autoencodeur est la suivante:

- Tout d'abord, l'autoencodeur reçoit une entrée et l'applique à un état caché par le biais d'une transformation affine $\mathrm{h} = \mathrm{f}(\mathrm{W_h y} + \mathrm{b_h})$, où f est une fonction d'activation. Il s'agit de l'étape **encodeur**.
- Ensuite, $\tilde{\mathrm{y}} = \mathrm{g}(\mathrm{W_h y} + \mathrm{b_h})$ où g est une fonction d'activation. C'est l'étape du **décodeur**.

4.3.3. Autoencodeurs débruiteurs

Pour obliger un autoencodeur à apprendre des caractéristiques utiles, une approche consiste à rajouter du bruit sur ses entrées et à l'entrainer pour qu'il retrouve les entrées d'origine, sans le bruit. Cette idée date des années 1980 [36].

Le bruit peut être un bruit blanc gaussien ajouté aux entrées ou un blocage aléatoire des entrées ; comme dans une technique du dropout. Le processus est illustré dans la

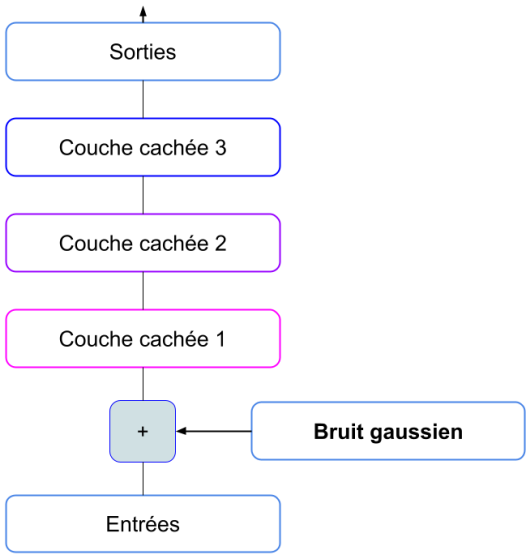

Figure 4.6

Figure 4.6: Autoencodeur débruteur avec un bruit gaussien

Bien que les autoencodeurs débruiteurs soient conçus pour gérer le bruit, les autoencodeurs standard peuvent ne pas bien fonctionner si les données d'entrée sont très bruyantes ou mal structurées.

4.3.4. Autoencodeurs variationnels

Une autre catégorie importante d'autoencodeurs a été proposée en 2014 par Diederik Kingma et Max Welling [37] et elle est rapidement devenue l'un des types d'autoencodeurs les plus populaires : les autoencodeurs variationnels (VAE). Ce sont des:

- Autoencodeurs probabilistes, ce qui signifie que leurs sorties sont partiellement déterminées par le hasard, même après l'entraînement ;

- Autoencodeurs génératifs, c'est-à-dire capables de générer de nouvelles instances qui semblent provenir du jeu d'entraînement.

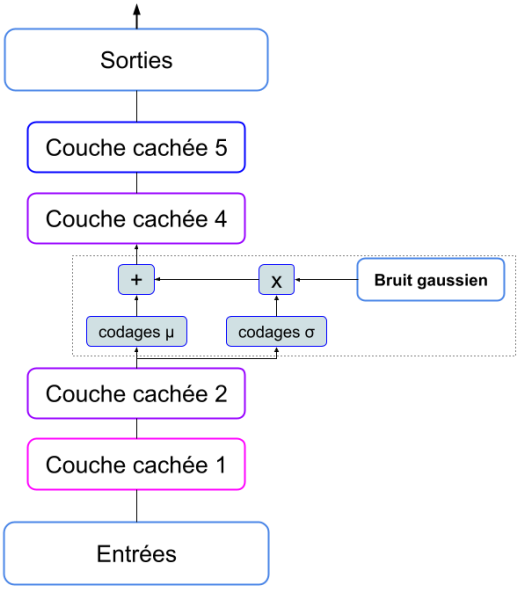

Figure 4.7: Autoencodeur variationnel

La **Figure 4.7** montre un autoencodeur variationnel. On peut reconnaître la structure de base de tous les autoencodeurs, avec un encodeur suivi d'un décodeur , mais on constate également un petit changement. Au lieu de produire directement un codage pour une entrée donnée, l'encodeur produit un codage moyen μ et un écart type σ. Le codage réel est ensuite échantillonné aléatoirement avec une loi normale de moyenne μ et d'écart type σ. Ensuite, le décodeur décode normalement le codage échantillonné. La partie droite de la figure montre une instance d'entraînement qui passe par cet autoencodeur. Tout d'abord, l'encodeur produit μ et σ, puis un codage est pris aléatoirement. Enfin, ce codage est décodé et la sortie finale ressemble à l'instance d'entraînement.

D'un côté, on note que les VAE offrent une approche prometteuse pour la génération de données et la modélisation de distributions latentes complexes. Ils

permettent également une interpolation en douceur entre points de données et sont régularisés, ce qui les rend utiles pour la réduction de dimension et la génération d'images. Cependant, les VAE peuvent être plus complexes à entraîner que les autoencodeurs classiques, et ils ont tendance à produire des résultats légèrement flous dans la génération d'images en raison de leur nature probabiliste. De plus, l'interprétation des dimensions latentes peut être difficile.

4.4. Conclusion

En somme, nous avons exploré les domaines de l'apprentissage non supervisé et de l'apprentissage profond. L'apprentissage non supervisé a émergé comme une approche puissante pour extraire des informations utiles à partir de données non étiquetées, que ce soit par la réduction de dimension, la génération de données, ou encore la segmentation. L'apprentissage profond, d'autre part, a révolutionné le champ de l'apprentissage automatique en exploitant des architectures de réseaux neuronaux profonds pour des tâches variées. Cependant, les défis demeurent, notamment en termes de choix d'architecture, d'entraînement efficace, et d'interprétabilité. La combinaison de l'apprentissage non supervisé avec l'apprentissage profond ouvre de nouvelles perspectives, offrant ainsi un moyen de relever des défis complexes dans divers domaines.

Chapitre 5. Méthodologie

Initialement, notre travail visait à réaliser une étude comparative entre les méthodes d'extraction de règles d'association exhaustives et heuristiques. Cependant, en raison des contraintes d'implémentation pour la vérification des hypothèses sur les méthodes heuristiques d'extractions des règles, notre projet a évolué vers le développement d'un système de recommandations des produits d'une épicerie en ligne. Ainsi, dans ce chapitre, nous présentons les étapes nécessaires à la compréhension de la phase d'implémentation.

Nous examinerons les avantages des méthodes utilisées dans notre approche en les appliquant à notre jeu de données constitué des transactions effectuées par les utilisateurs d'épicerie en ligne d'Instacart.

5.1. Workflow

Un workflow, ou un flux de travaux, est une séquence de tâches effectuées dans un processus. La **Figure 5.1** illustre étape par étape le cheminement méthodologique suivi lors de la phase d'implémentation de notre projet.

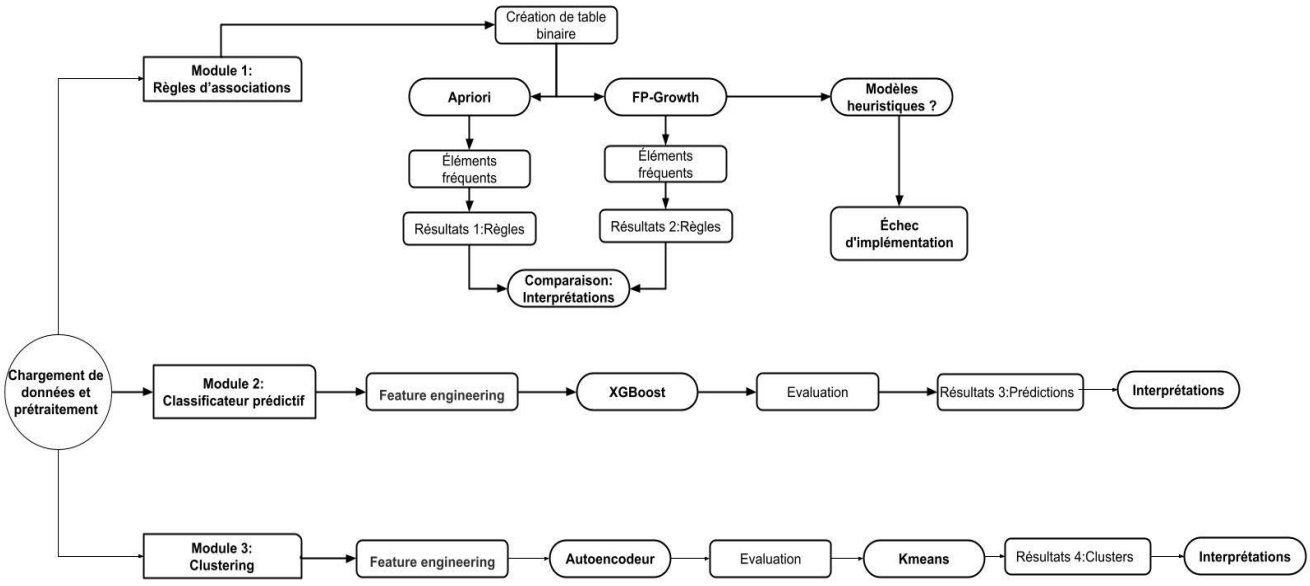

Figure 5.1: *Diagramme méthodologique*

5.1.1. Chargement des données et prétraitement

Nous entamons notre implémentation en chargeant les données, hébergées depuis le dépôt Kaggle, puis nous effectuons le prétraitement afin de mieux comprendre leur structure.

Le prétraitement fait référence à toutes les opérations effectuées sur cet ensemble de données avant son utilisation ou sa modélisation. Voici les techniques utilisées :

- **Nettoyage des données** : nous avons traité les valeurs aberrantes, des données manquantes.
- **Codage des variables** : Nous avons traité les types de données afin d'avoir les classes et les catégories exploitables.
- **Echantillonnage des données** : la technique d'échantillonnage a permis de gérer la taille des données pour être traitées, compte tenu des contraintes matérielles.
- **Séparation des données** : Avant de construire le modèle, nous avons séparé les données en ensembles d'entrainement, de validation et de test.

Ces techniques de prétraitement de nos données sont essentielles pour garantir la qualité et la robustesse des modèles d'apprentissage. Nous développons nos modèles selon trois modules : Extraction des règles d'associations, Entrainement du modèle prédictif et Clustering.

5.1.2. Module 1 : Extraction des règles d'associations

Extraire les règles d'associations nous permet de répondre à la question : quelles combinaisons de produits sont fréquemment achetées ensemble ?

Nous commençons par préparer la table des transactions des articles, ce sont les entrées des algorithmes Apriori et FP-Growth. Ensuite nous construisons une table

binaire à partir de ces transactions. Dans cette table binaire (voir **Figure 5.2**), les colonnes sont des identifiants d'articles et les lignes les identifiants des transactions.

```
truth_table.head()
```

	1	2	3	4	5	7	8	9	10	11	...	49677	49678	49679	49680	49681	49682	49683	49686	49687	49688
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	True	False	False	False
36	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
38	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
96	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
98	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

5 rows × 39123 columns

Figure 5.2: *Entête de la table binaire*

Ensuite, grâce à cette table, nous générons les ensembles d'itemsets fréquents à partir d'Apriori et FP-Growth. La recherche de règles à partir de méthodes heuristiques pourrait permettre de réduire la taille des règles en vue d'une visualisation, cependant, l'implémentation n'a pas été concluante.

5.1.3. Module 2 : Entrainement du modèle prédictif

Supposons que nous ne voulions pas formuler une recommandation basée sur les transactions de plusieurs utilisateurs, mais que nous ciblions spécifiquement un utilisateur pour lui recommander un produit en se basant sur ses commandes précédentes. Par conséquent, notre objectif est de prédire quels produits achetés précédemment (lors de commandes antérieures) seront inclus dans la prochaine commande de l'utilisateur. Nous entrainons le modèle XGBoost pour répondre à cet objectif. Nous effectuons un traitement de caractéristiques (feature engineering) pour créer des variables prédictives, puis créons un ensemble de données d'entraînement et de test. Ensuite, nous entraînons le modèle, évaluons sa performance (métriques F1-score et la courbe ROC) et l'utilisons pour effectuer des prédictions et interpréter les résultats. Le F1-score est une mesure de la précision d'un modèle de classification. La courbe ROC est un graphique qui représente la performance d'un modèle de classification à différents seuils de discrimination.

5.1.4. Module 3 : Clustering

Le clustering nous permet d'identifier des segments d'utilisateurs en fonction de leur comportement d'achat, en regroupant les utilisateurs similaires pour des offres promotionnelles spécifiques. Nous effectuons un traitement des caractéristiques pour obtenir 106 variables, puis nous entraînons un modèle d'encodage pour réduire la dimensionnalité. Nous évaluons sa performance, puis nous utilisons les données encodées pour modéliser avec l'algorithme k-means afin d'obtenir des clusters exploitables. Nous enchaînons ensuite avec l'évaluation des résultats, leur visualisation avec t-SNE, puis leur interprétation.

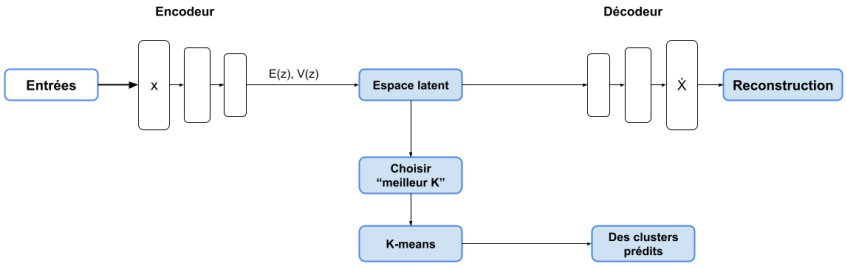

Figure 5.3: *Diagramme du modèle hybride: autoencodeur - kmeans*

5.1.5. Conclusion

En somme, nous avons utilisé trois approches pour analyser le panier de consommation. Tout d'abord, nous avons extrait les règles d'association en utilisant Apriori et FP-Growth, puis nous avons mené une étude comparative entre les deux méthodes. Ensuite, nous sommes passés à l'entraînement d'un modèle prédictif, car les règles étaient insuffisantes pour obtenir un système ciblé spécifiquement sur un utilisateur, tandis que le jeu de données permettait d'atteindre cet objectif. C'est ainsi que nous avons utilisé un modèle ensembliste avec la variante XGBoost. Enfin, nous avons combiné les modèles autoencodeur et k-means dans une approche hybride autoencodeur-kmeans pour le clustering profond, afin d'obtenir des clusters exploitables.

Chapitre 6. Implémentation

Dans ce chapitre nous mettons en avant les moyens et la méthodologie utilisés pour la mise en œuvre de ce projet de recherche.

6.1. Introduction

Nous présentons entre autres les outils, les technologies et les langages de programmation utilisés pour développer ce projet. Nous rappelons que la problématique de ce projet est de développer un système de recommandation de produits en faisant l'analyse des paniers de consommation. Nous effectuerons une comparaison de différentes méthodes de modélisation utilisées sur notre jeu de données.

6.2. Environnement de développement

Nous avons implémenté différentes méthodes, à savoir les algorithmes d'extraction de règles d'association, les méthodes d'ensemble, et nous avons terminé par la combinaison d'autoencodeurs avec k-means pour extraire les caractéristiques et réaliser le clustering.

6.2.1. Langage de programmation

Comme langage de programmation, nous avons utilisé **Python** pour implémenter les différents algorithmes. Python est un langage de haut niveau, interprété et polyvalent. Créé par Guido van Rossum et publié pour la première fois en 1991 [38], Python offre une syntaxe claire et concise, ce qui le rend particulièrement adapté aux débutants en programmation. Il prend en charge plusieurs paradigmes de programmation, notamment la programmation impérative, orientée objet et fonctionnelle, offrant ainsi une grande flexibilité. Python est également doté d'une vaste bibliothèque standard qui facilite le développement de divers types d'applications, de scripts simples aux applications d'apprentissage automatique complexe.

6.2.2. Librairies

Nous avons fait usage de plusieurs librairies pour atteindre nos objectifs, il s'agit de : MLxtend, Matplotlib, Zipfile, Pandas, Scikit-learn , XGBoost, TensorFlow, Keras.

- **MLxtend** : La bibliothèque MLxtend (extensions Machine Learning) possède de nombreuses fonctions intéressantes pour les tâches quotidiennes d'analyse de données et d'apprentissage automatique. Elle propose des fonctionnalités telles que des méthodes d'ensemble avancées, des outils pour l'extraction de caractéristiques, la visualisation des données, ainsi que des utilitaires pour simplifier le processus de création, d'évaluation et de comparaison de modèles. Nous l'avons utilisé pour la génération des règles d'associations.
- **Matplotlib**[2] est une bibliothèque de visualisation en Python très populaire et puissante, permettant de créer une grande variété de graphiques de manière flexible et personnalisable. Elle offre une interface pour produire des graphiques en 2D et 3D, des histogrammes, des diagrammes en barres, des nuages de points et bien plus encore.
- **Zipfile** : est un module intégré à Python qui permet de créer, manipuler et extraire des fichiers au format ZIP.
- **Pandas** est un puissant outil en Python utilisé pour la manipulation et l'analyse de données structurées. Elle offre des structures de données flexibles, notamment les DataFrames, permettant de traiter et d'analyser facilement des ensembles de données tabulaires.
- **Scikit-learn** : cette librairie propose une vaste gamme d'outils et d'algorithmes d'apprentissage supervisé et non supervisé, ainsi que des utilitaires pour la préparation et la sélection de caractéristiques, l'évaluation de modèles et la validation croisée. Scikit-learn offre une interface cohérente et intuitive pour créer, entraîner et déployer des modèles, ce qui en fait un choix privilégié pour les projets d'apprentissage

[2] https://matplotlib.org/stable/index.html

automatique, en mettant l'accent sur la simplicité, la performance et la facilité d'utilisation. Nous l'avons utilisé pour construire nos modèles prédictifs d'ensemble.

- **XGBoost** est une bibliothèque d'apprentissage automatique populaire qui se concentre sur les arbres de décision et les méthodes de boosting. Elle offre des implémentations optimisées d'algorithmes de boosting en gradient, améliorant les performances des modèles de classification et de régression.
- **TensorFlow** est une bibliothèque d'apprentissage automatique et de calcul numérique développée par Google. Tensorflow propose une infrastructure robuste pour la construction et le déploiement de réseaux de neurones, avec une grande variété d'outils pour l'entraînement, l'optimisation et la gestion des modèles. Elle est largement utilisée dans la recherche en intelligence artificielle, l'apprentissage profond.
- **Keras** est une interface haut niveau pour la création de réseaux de neurones en Python. Intégrée à des bibliothèques comme TensorFlow, Keras simplifie la création de réseaux neuronaux en fournissant une API simple et cohérente pour la conception.

6.2.3. Environnement d'implémentation

Afin d'extraire les règles d'association et de mener une étude comparative efficace, nous avons opté pour l'utilisation de l'environnement de développement Azure Machine Learning ainsi que de l'environnement Anaconda en local. Anaconda est une plateforme qui facilite le lancement des projets d'intelligence artificielle, offrant une suite complète d'outils et de bibliothèques populaires pour l'analyse de données, l'apprentissage automatique et d'autres tâches liées à l'IA.

Nous avons fait le choix d'utiliser une machine virtuelle (VM), lancer dans l'environnement Azure, afin de réaliser nos expériences avec les algorithmes d'extractions de règles d'association. Ci-dessous (**Figure 6.1**) les caractéristiques du VM.

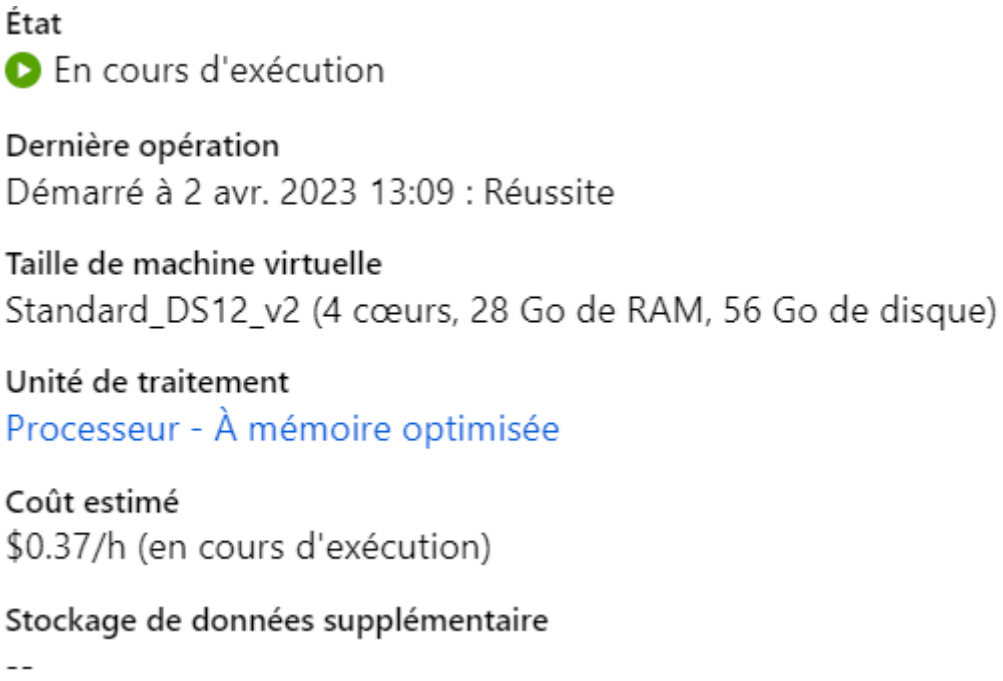

Figure 6.1: *Caractéristiques de la machine virtuelle*

La machine virtuelle possède **un CPU à 4 cœurs**, **28 Giga-octets de RAM**, et un disque SSD de 56 Giga-octets. Démarrer sur un système d'exploitation **Linux UBUNTU.**

Pour l'expérimentation des autres algorithmes, nous avons opéré le choix d'utiliser une machine avec un système Windows 11, un processeur AMD Ryzen 7 3750H avec Radeon Vega Mobile Gfx, 2300 MHz, 4 cœurs, une RAM de 32 Giga-octets et un processeur graphique NVIDIA GeForce GTX 1650 avec 8 Giga-octets de mémoire graphique.

6.3. Les données

Dans ce travail, nous avons choisi d'utiliser un jeu de données de l'ensemble de données d'achats d'épicerie en ligne d'Instacart[3] publié en 2017 [39] et disponible sur kaggle[4]. Cet ensemble de données anonymisées contient un échantillon de plus de 3

[3] https://www.instacart.ca/
[4] https://www.kaggle.com/c/instacart-market-basket-analysis

millions de commandes d'épicerie provenant de plus de 200 000 utilisateurs d'Instacart et reparties en 5 tables (*orders, products, aisles, departments et order_products*). Pour chaque utilisateur, il est fourni entre 4 et 100 de ses commandes, avec l'ordre des produits achetés dans chaque commande. La semaine et l'heure à laquelle la commande a été passée, ainsi qu'une mesure relative du temps entre les commandes.

Nous utilisons donc cet ensemble de données dans le cadre de notre projet pour expérimenter les modèles permettant de prédire les produits qu'un utilisateur achètera à nouveau, essaiera pour la première fois ou ajoutera au panier ensuite au cours d'une session.

Par exemple la **Figure 6.2** présente un aperçu des deux premières commandes pour *user_id* = 1.

user id	order id	order number	order dow	order hour of day	add to cart order	product id	product name
1	2539329	1	2	8	1	196	Soda
1	2539329	1	2	8	2	14084	Organic Unsweetened Vanilla Almond Milk
1	2539329	1	2	8	3	12427	Original Beef Jerky
1	2539329	1	2	8	4	26088	Aged White Cheddar Popcorn
1	2539329	1	2	8	5	26405	XL Pick-A-Size Paper Towel Rolls
1	2398795	2	3	7	1	196	Soda
1	2398795	2	3	7	2	10258	Pistachios
1	2398795	2	3	7	3	12427	Original Beef Jerky
1	2398795	2	3	7	4	13176	Bag of Organic Bananas
1	2398795	2	3	7	5	26088	Aged White Cheddar Popcorn
1	2398795	2	3	7	6	13032	Cinnamon Toast Crunch

Figure 6.2: *Aperçu du jeu de données*

Description des tables avec les variables :

orders (commandes avec 3,4 millions de lignes, 206 000 utilisateurs) :

- order_id: identifiant d'une commande
- user_id: identifiant d'un utilisateur

- eval_set: jeux d’evaluation
- order_number: le numéro d'ordre de cet utilisateur
- order_dow: le jour de la semaine où la commande a été passée
- order_hour_of_day: l'heure du jour où la commande a été passée
- days_since_prior: jours depuis la dernière commande, maximum à 30

products (50 000 lignes):

- product_id: identifiant d’un produit
- product_name: nom du produit
- aisle_id: clé entrangere pour lier au table aisle
- department_id: clé lié au table department

aisles (134 lignes):

- aisle_id: identifiant d’une rangée
- aisle: nom d’une rangee

deptartments (21 lignes):

- department_id: identifiant du département
- department: le nom du département

order_products__SET (30 millions de lignes):

- order_id: clé de table orders
- product_id: clé de la table produit
- add_to_cart_order: l'ordre dans lequel chaque produit a été ajouté au panier
- reordered: 1 si ce produit a été commandé par cet utilisateur dans le passé, 0 sinon

6.4. Interfaces

Nous avons utilisé principalement 2 environnements pour le développement du projet.

- **Microsoft Azure**

Le module 1 (voir **Figure 6.3**) a été développé en utilisant le service Machine Learning Studio de Microsoft Azure, en créant une machine virtuelle avec des caractéristiques permettant de réaliser la comparaison entre les performances des deux algorithmes Apriori et FP-Growth.

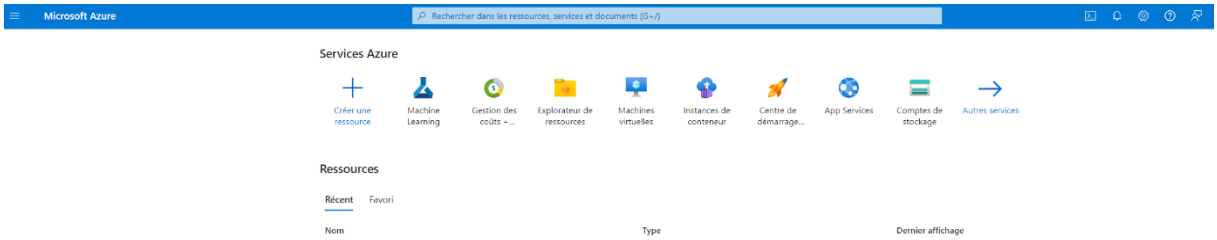

***Figure 6.3**: Interface de Microsoft Azure Student*

- **Anaconda, Jupyter Notebook**

Jupyter Notebook (voir **Figure 6.4**) est un environnement de développement interactif qui permet d'écrire et d'exécuter du code dans un navigateur web, et de combiner du code, du texte descriptif, des équations mathématiques et des visualisations dans un même document appelé "notebook".

Les modules 2 et 3 (voir **Figure 6.5** et **Figure 6.6**) ont été développés avec Jupyter Notebook. Nous importons les données dans notre notebook et les analysons avec Python.

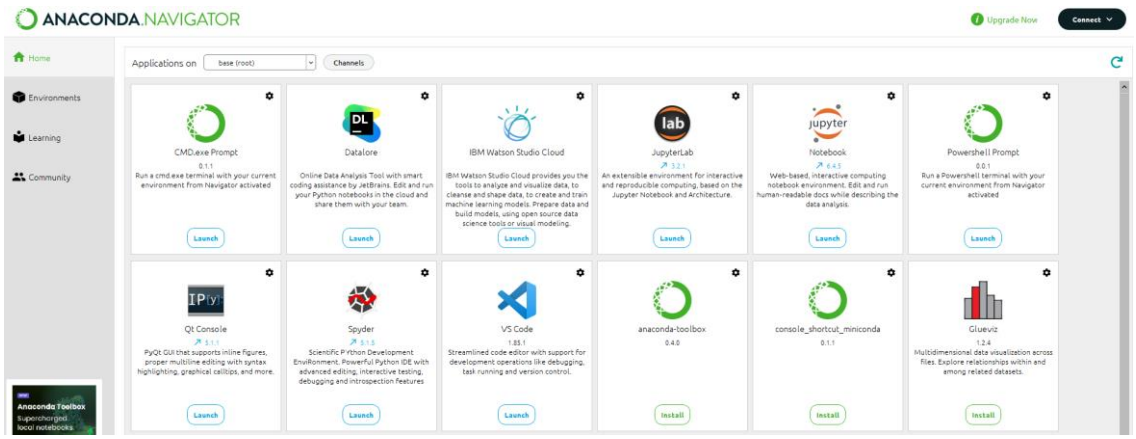

Figure 6.4: *Interface d'application Anaconda*

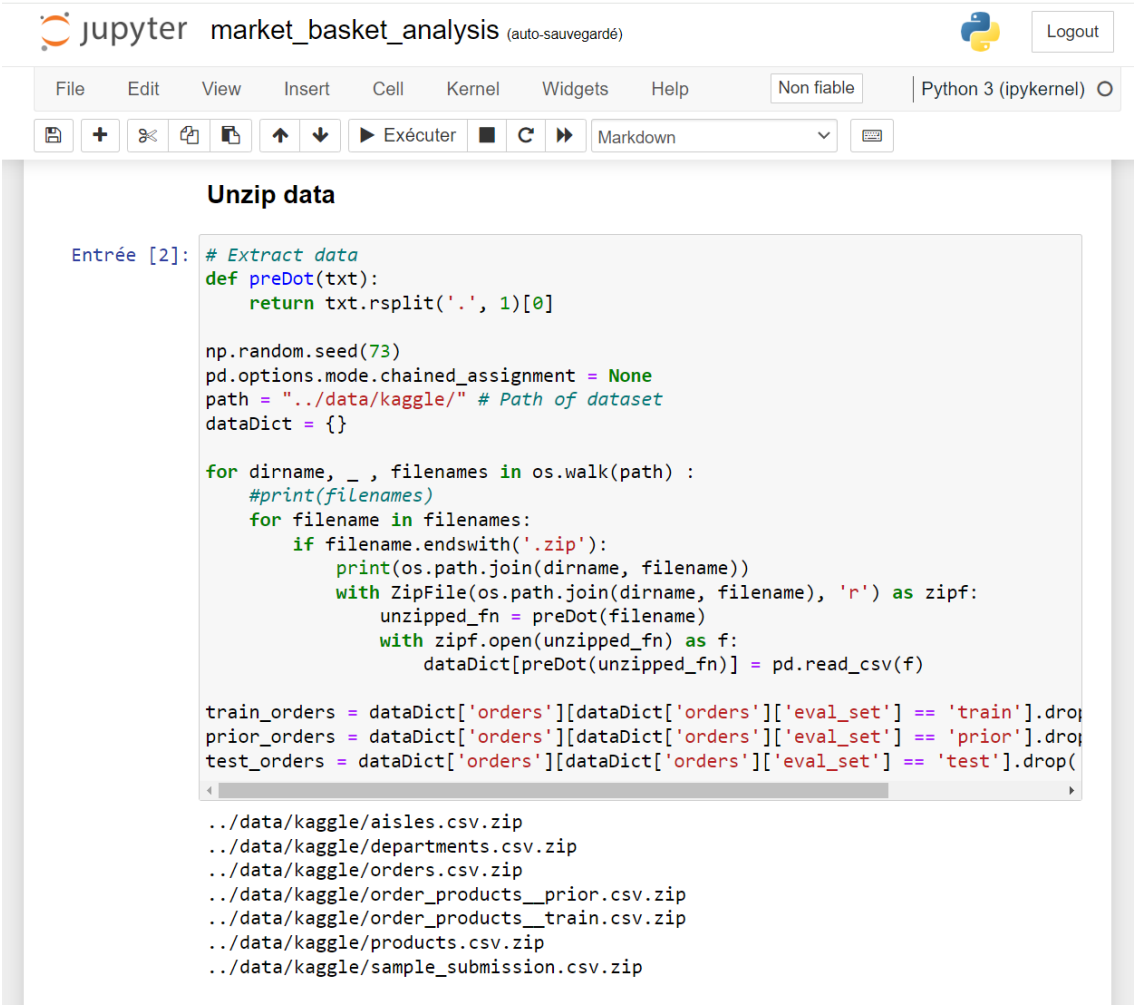

Figure 6.5: *Chargement des données via Jupyter Notebook*

Data preprocessing

```
orders = dataDict['orders']
order_products_train = dataDict['order_products__train']
order_products_prior = dataDict['order_products__prior']
products = dataDict['products']
aisles = dataDict['aisles']
departements = dataDict['departments']
```

```
# Reshape
aisles['aisle'] = aisles['aisle'].astype('category')
departements['department'] = departements['department'].astype('category')
orders['eval_set'] = orders['eval_set'].astype('category')
products['product_name'] = products['product_name'].astype('category')
```

```
orders.head()
```

	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
0	2539329	1	prior	1	2	8	NaN
1	2398795	1	prior	2	3	7	15.0
2	473747	1	prior	3	3	12	21.0
3	2254736	1	prior	4	4	7	29.0
4	431534	1	prior	5	4	15	28.0

Figure 6.6: *Interface des sorties*

Jupyter Notebook est largement utilisé dans le domaine de la science des données, de l'apprentissage automatique, de l'enseignement et de la recherche en raison de sa capacité à fournir une expérience de codage interactive et à créer des documents facilement partageables et reproductibles.

6.5. Fonctionnement du système développé

Nous avons développé notre système en utilisant des notebooks Jupyter. La description pour lancer chaque module se présente comme suit :

- **Module 1**

On crée un environnement sur la plateforme Cloud Azure Machine Learning, avec une machine virtuelle ayant les caractéristiques citées dans la partie environnement d'exécution, installer les librairies nécessaires ou les dépendances. Puis importer le notebook et exécuter le premier module de règles d'association.

- **Module 2 et 3**

Les deux autres modules peuvent être exécutés sur une machine disposant de la distribution Anaconda avec les dépendances. Nous avons opéré le choix d'une machine avec un système d'exploitation Windows 11, un processeur AMD Ryzen 7, 4 cœurs, une RAM de 32 Giga-octets et un processeur graphique NVIDIA GeForce GTX 1650 avec 8 Giga-octets de mémoire graphique.

6.6. Conclusion

Dans ce chapitre dédié à l'implémentation des algorithmes de construction de règles d'association, des méthodes d'ensemble et des réseaux de neurones, nous avons exploré les technologies utilisées pour développer notre système. Nous avons mis en avant l'importance d'une compréhension approfondie des données en amont et du choix de modèles adaptés à chaque problème.

Chapitre 7. Expérimentations et discussions

Dans ce chapitre, nous expérimentons notre système sur des données réelles d'Instacart en menant des interprétations et des discussions sur les résultats obtenus.

7.1. Introduction

Ici, nous présentons les résultats de notre recherche sur le développement d'un système de recommandation en faisant l'analyse du panier de consommation.

Nous récupérons les données qui sont constituées des commandes d'achats, puis nous utilisons diverses techniques afin de préparer les données pour l'entrainement des divers modèles. Nous présentons les interprétations et discussions de ces résultats.

7.2. Résultats des expérimentations et interprétations

Le système est développé en trois(3) phases. La première approche consiste à utiliser les règles d'association, la deuxième consiste à développer un modèle de prédiction en utilisant XGBoost, et la troisième phase à entrainer un réseau de neurones de type autoencodeur et combiner à k-means pour avoir des clusters des utilisateurs exploitables.

7.2.1. Première approche : Règles d'association

Notre approche basée sur les règles d'association, se déroule en deux phases, la première phase consiste à construire des règles avec l'algorithme Apriori et la deuxième à construire des règles avec l'algorithme FP-Growth, et terminer avec une étude comparative.

7.2.1.1. Résultats d'Apriori et interprétations

Rappelons que notre but est de recommander des articles susceptibles d'intéresser l'utilisateur en analysant la composition de son panier d'achats.

La **Figure 7.1** nous indique le nombre d'itemset fréquents en fonction du support.

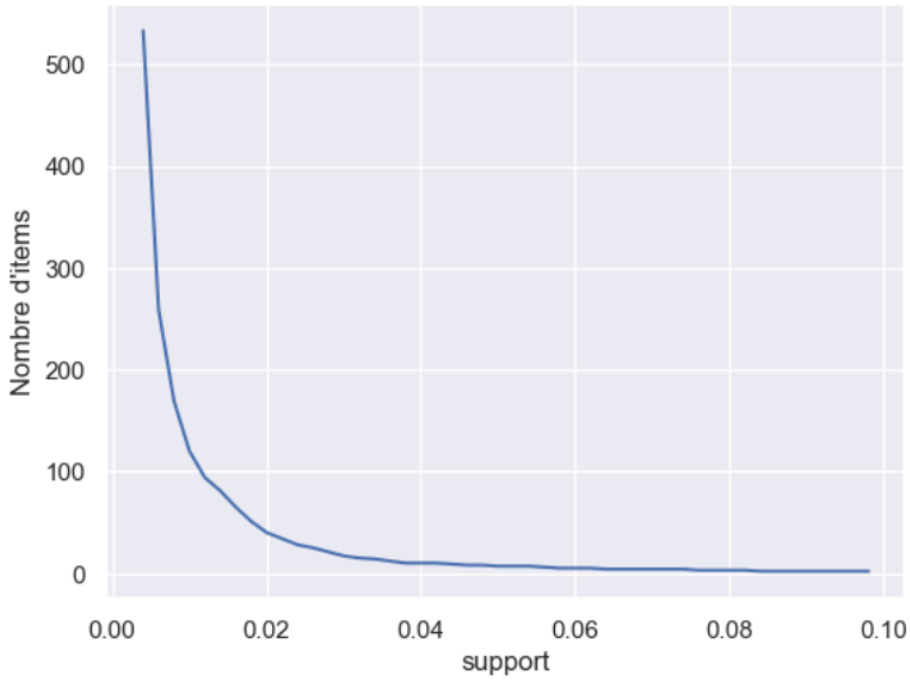

Figure 7.1: *Apriori: nombre d'éléments fréquents en fonction du support*

D'après la **Figure 7.2** (La première colonne, **support**, représente le support ; La deuxième colonne, **itemsets**, représente l'ID d'item ou d'itemsets fréquents ; Puis la dernière colonne représente le nombre d'éléments), avec un support de 0.008 (0.8%) , nous obtenons 169 ensembles d'éléments fréquents.

Out[28]:

	support	itemsets	length
0	0.011485	(196)	1
1	0.009260	(260)	1
2	0.009733	(432)	1
3	0.008887	(2295)	1
4	0.009893	(3957)	1
...	...	...	...
164	0.009382	(45066, 24852)	2
165	0.016447	(47626, 24852)	2
166	0.016889	(24852, 47766)	2
167	0.012156	(26209, 47626)	2
168	0.010281	(47626, 47766)	2

169 rows × 3 columns

Figure 7.2: *Itemsets fréquents avec support 0.8%*

Si nous fixons un support minimum à 0.001, nous construisons **60 règles d'association**, dont les 10 règles les plus intéressantes (une règle est plus intéressante lorsque sa confiance plus élevée) sont :

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
(28204)	(24852)	0.024823	0.142719	0.009222	0.371508	2.603072	0.005679	1.364028	0.631515
(45066)	(24852)	0.027064	0.142719	0.009382	0.346663	2.428991	0.005519	1.312157	0.604671
(47209)	(13176)	0.055583	0.117980	0.018444	0.331825	2.812560	0.011886	1.320044	0.682381
(27966)	(13176)	0.042268	0.117980	0.013566	0.320952	2.720400	0.008579	1.298907	0.660318
(5876)	(13176)	0.026713	0.117980	0.008132	0.304422	2.580293	0.004980	1.268040	0.629257
(27966)	(21137)	0.042268	0.083028	0.012728	0.301118	3.626710	0.009218	1.312056	0.756233
(16797)	(24852)	0.049494	0.142719	0.014847	0.299969	2.101819	0.007783	1.224633	0.551518
(47766)	(24852)	0.056467	0.142719	0.016889	0.299096	2.095698	0.008830	1.223107	0.554122
(4920)	(24852)	0.030935	0.142719	0.008856	0.286277	2.005883	0.004441	1.201141	0.517475
(4605)	(24852)	0.028672	0.142719	0.008163	0.284689	1.994754	0.004071	1.198473	0.513405

Figure 7.3: *10 règles plus intéressantes avec Apriori (min_sup=0.001)*

Par exemple, on remarque que l'association 28204 (Organic Fuji Apple) → 24852 (Banana) se produit souvent avec une confiance de 37.15 %. Les utilisateurs achètent souvent les articles *Organic Fuji Apple et Banana* ensemble.

A partir de ces 60 règles, nous avons construits nos recommandations ci-dessous (Les paniers ***Figure 7.4*** respectivement les recommandations ***Figure 7.5***).

```
                                                    basket
10       (Pistachios, Zero Calorie Cola, Organic Half &...  \
25       (Shelled Pistachios, Organic Roasted Turkey Br...
74       (Antioxidant Infusions Beverage Malawi Mango, ...
78       (Organic Green Onions, Organic Cilantro, Nutri...
200      (Ultra Soft Bathroom Tissue Double Rolls, Sele...
...                                                    ...
3420786  (Organic Cilantro, No Salt Added Black Beans, ...
3420862  (Organic Navel Orange, Organic Baby Romaine Le...
3420924  (Organic Red Delicious Apple, Organic Lori's L...
3420933  (Cara Cara Orange, YoKids Squeezers Organic Lo...
3421082  (Dark Chocolate Mint Snacking Chocolate, Phish...
```

Figure 7.4: *Panier de consommation*

```
                                          recommendations
10                                (Bag of Organic Bananas)
25       (Organic Fuji Apple, Limes, Large Lemon, Seedl...
74                                               (Banana)
78       (Large Lemon, Organic Hass Avocado, Organic St...
200                                              (Banana)
...                                                    ...
3420786  (Organic Fuji Apple, Honeycrisp Apple, Large L...
3420862  (Organic Whole Milk, Large Lemon, Organic Cucu...
3420924  (Organic Whole Milk, Large Lemon, Organic Hass...
3420933  (Organic Fuji Apple, Limes, Honeycrisp Apple, ...
3421082  (Organic Fuji Apple, Limes, Honeycrisp Apple, ...
```

Figure 7.5*: Recommandations des produits*

Le système de recommandations fonctionne bien, par exemple pour un utilisateur ayant ajouté les produits ***Pistachios, Zero calorie cola,Organic half***, le système lui recommande ***Bag of Organic Bananas.*** En interrogeant le système de prédire le produit à ajouter au panier constitué uniquement de ***Organic Fuji Apple***, nous obtenons bien ***Banana.***

7.2.1.2. Résultats de FP-Growth et Interprétations

Avec FP-Growth, nous avons obtenu 60 règles , dont les 10 premières les plus intéressantes sont :

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
(47209, 21137, 27966)	(13176)	0.002904	0.117980	0.001738	0.598425	5.072272	0.001395	2.196403	0.805188
(47209, 21137, 30391)	(13176)	0.001951	0.117980	0.001067	0.546875	4.635331	0.000837	1.946528	0.785799
(39928, 47209)	(13176)	0.002652	0.117980	0.001448	0.545977	4.627719	0.001135	1.942678	0.785996
(27966, 8174)	(13176)	0.002126	0.117980	0.001151	0.541219	4.587387	0.000900	1.922529	0.783677
(4605, 16797)	(24852)	0.002134	0.142719	0.001143	0.535714	3.753633	0.000839	1.846452	0.735160
(47209, 22035)	(13176)	0.002180	0.117980	0.001158	0.531469	4.504745	0.000901	1.882521	0.779711
(47209, 8174)	(13176)	0.002828	0.117980	0.001494	0.528302	4.477905	0.001160	1.869883	0.778884
(47209, 27966)	(13176)	0.007766	0.117980	0.004047	0.521099	4.416854	0.003131	1.841760	0.779649
(47209, 22825)	(13176)	0.002683	0.117980	0.001387	0.517045	4.382495	0.001071	1.826301	0.773896
(47209, 35951)	(13176)	0.002431	0.117980	0.001250	0.514107	4.357585	0.000963	1.815255	0.772393

Figure 7.6: *Règles extraites avec FP-Growth*

Nous remarquons que la règle la plus intéressante a une **confiance de 59,84 %**, ce qui est nettement meilleur que le résultat obtenu avec Apriori.

7.2.1.3. Comparaison

En exécutant sur une machine avec les caractéristiques de ***Figure 6.1*** et **Min_sup = 0.0001** à l'entrée.

	Apriori	FP-Growth
Test de mémoire	Erreur (MemoryError)	Réussi
Nombre d'itemsets fréquents	NaN	136 394

Tableau 7.1: *Tableau comparative d'Apriori et FP-Growth*

Nous remarquons que FP-Growth est bien une amélioration d'Apriori en optimisant la taille de mémoire utilisée.

7.2.2. Deuxième approche : Méthodes ensemblistes

Les règles d'association nous ont permis de recommander des produits en se basant sur les transactions de multiples utilisateurs, comment ferions-nous pour cibler spécifiquement un utilisateur et lui recommander des produits sur la base de son

habitude de consommation ? Nous avons fait appel à l'algorithme XGBoost pour répondre à cette question. En conséquence, nous voulons prédire quels produits achetés précédemment figureront dans la prochaine commande d'un utilisateur.

Après l'entrainement de XGBoost (avec paramètres : objective='binary:logistic', booster='gbtree',eval_metric='logloss' , max_depth=5, colsample_bytree=0.4, subsample=0.75) sur nos données (80% de données d'entrainement et 20% de test); nous obtenons des performances suivantes :

Report :

	precision	recall	f1-score	support
0.0	0.92	0.99	0.95	1528774
1.0	0.63	0.17	0.26	166159
accuracy			0.91	1694933
macro avg	0.77	0.58	0.61	1694933
weighted avg	0.89	0.91	0.88	1694933

Figure 7.7: *Performances XGBoost*

Nous avons bien deux classes, c'est-à-dire, la classe 0 correspondant au produit non acheté à nouveau et 1 si le produit a été acheté de nouveau. Le score de précision est de **91 %** c'est-à-dire que notre modèle prédit correctement dans **91 %** des cas. Notons que l'ensemble de données n'étant pas symétrique (support de classe 1 diffère du support de classe 0), il est préférable de choisir le F1_score (le F1-score permet d'avoir une vue d'ensemble de la performance du modèle qui prend en compte à la fois les faux positifs et les faux négatifs). C'est une très bonne performance et est acceptable car le F1 score vaut **61 %**. La courbe ROC (Receiver operating characteristic) de notre modèle donne:

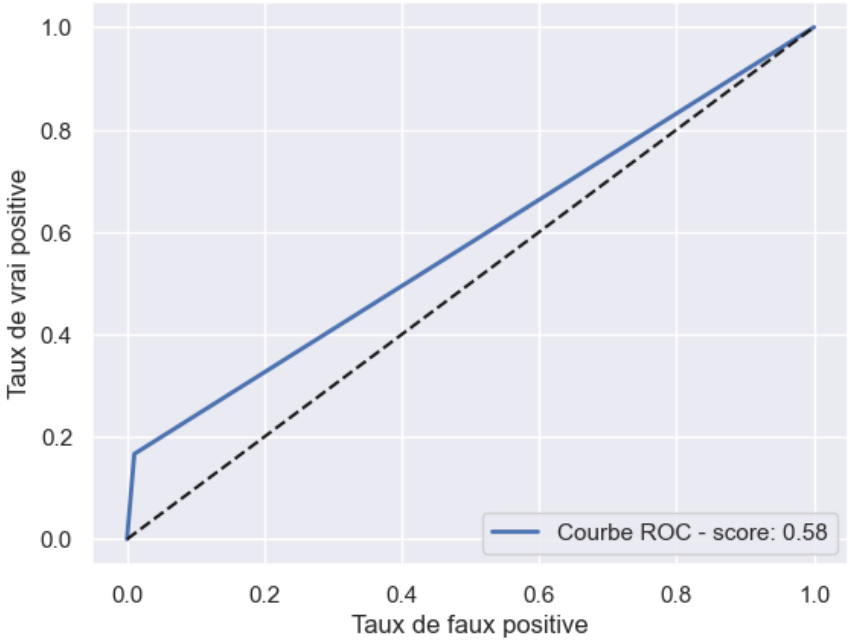

Figure 7.8: *Courbe ROC du modèle XGBoost*

Cette courbe ROC (voir **Figure 7.8**) permet de montrer les performances globales de notre modèle de prédiction, elle représente le taux de vrais positifs et le taux de faux positifs. Le score AUC (zone sous la courbe ROC) est de **0.58 ,** un seuil acceptable pour les prédictions, car notre jeu de données n'est pas symétrique.

Ainsi, avec le modèle construit, nous pouvons prédire les produits pouvant être achetés pour les paniers les plus récents.

Out[114]:

	order_id	products
0	2774568	21903 39190 47766
1	1416320	5134 21137 21903
2	1735923	17008
3	1980631	6184 9387 13575 13914 22362 41400 46061
4	3202221	4793 9637 24852

Figure 7.9: *Prédictions pour les derniers paniers*

Nous recommandons donc l'ensemble { 21903, 39190, 47766} pour le panier 2774568, ainsi de suite pour 75 000 derniers paniers.

7.2.3. Troisième approche : Réseaux de neurones pour le partitionnement

Les données relatives aux clients constituent la base de la segmentation du marché. Nous nous sommes servis de l'autoencodeur débruiteur pour coder les caractéristiques des produits afin de mieux comprendre les utilisateurs d'Instacart. Nous avons fait ce choix afin de réduire 36 variables (les variables obtenues après une ingénierie de caractéristiques) dont nous disposons sur les utilisateurs. Avec autant de caractéristiques, une approche de réseaux de neurones semble être la solution idéale pour extraire de ces 36 variables, les caractéristiques essentielles pour segmenter la base clientèle.

➢ **Résultats d'Autoencodeur débruiteur**

```
Model: "autoencoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder (Sequential)        (None, 8)                 728

 decoder (Sequential)        (None, 36)                756

=================================================================
Total params: 1484 (5.80 KB)
Trainable params: 1484 (5.80 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 7.10: *Paramètres du modèle*

Notre modèle autoencodeur débruiteur a 1484 paramètres, soit un encodeur charger d'encoder 36 caractéristiques en 8 caractéristiques (voir **Figure 7.10** et **Figure 7.11**).

	dim1	dim2	dim3	dim4	dim5	dim6	dim7	dim8
10589486	0.523440	0.343620	0.322294	0.455695	0.037547	0.208765	0.433902	0.422876
8128189	0.590128	0.372753	0.180763	0.402194	0.616123	0.571708	0.538587	0.279390
11763256	0.413725	0.646168	0.272439	0.494418	0.508439	0.258994	0.628783	0.238827
11247203	0.248935	0.688258	0.302435	0.560342	0.598162	0.265603	0.664242	0.249060
6993533	0.556190	0.356060	0.232649	0.583987	0.588782	0.328175	0.526882	0.370653

Figure 7.11*: Aperçu des dimensions réduites*

Apres l'entrainement sur 5 époques, nous avons obtenu les performances représentées à la **Figure 7.12**.

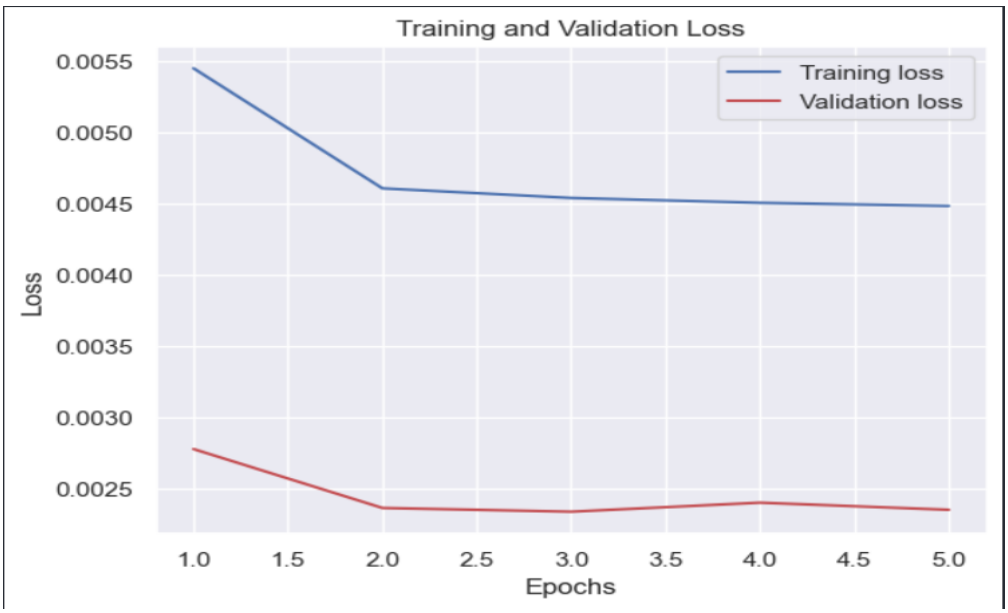

Figure 7.12: *Performances d'autoencodeur*

Nous observons une diminution progressive de la perte sur l'ensemble d'entraînement et de validation. Il y a une stabilisation du taux d'erreur entre les époques 2 et 5, avec une perte de 0,0046 sur les données d'entraînement et de 0,0024 sur les données de validation. Notre modèle a bien appris à généraliser.

➢ Partitionnement

Pour compromis de ressource en termes de mémoire de la machine, nous avons fait le choix d'échantillonner l'ensemble de nos données en 10 000 lignes représentatives pour le clustering.

La représentation avec t-sne nous donne une idée approximative du nombre de cluster à avoir (voir ***Figure 7.13***).

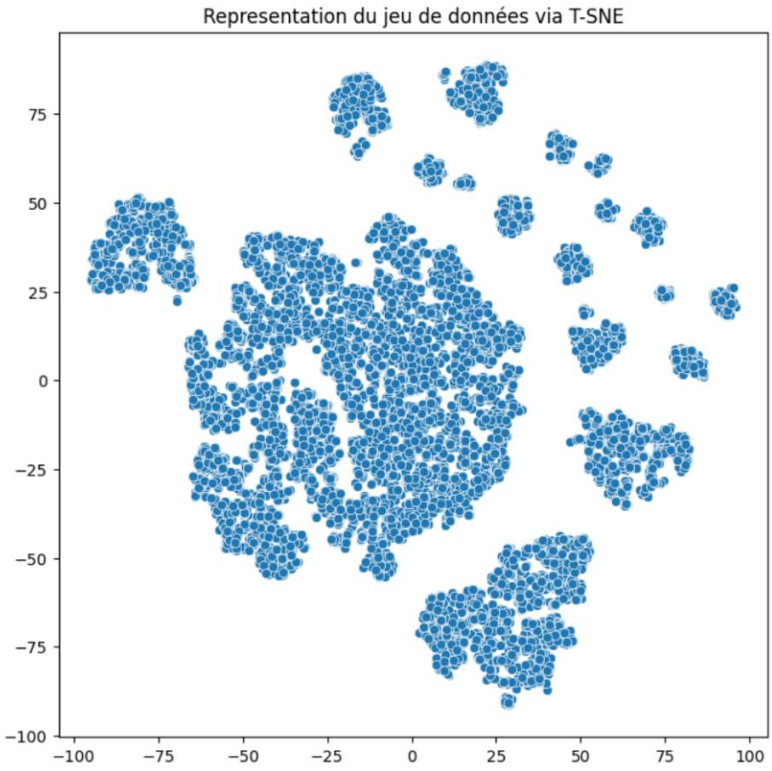

Figure 7.13: *Représentation via t-sne*

Après 3 itérations sur le modèle K-means, la méthode du coude donne (voir ***Figure 7.14***) :

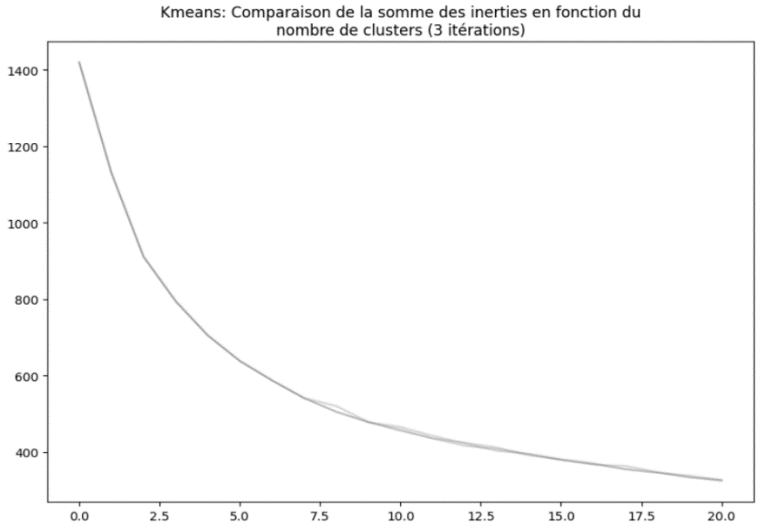

Figure 7.14*: Méthode du coude*

Nous déduisons le nombre optimal de cluster entre 2.5 – 7.5.

Le score de Davies Bouldin est minimal pour **4 clusters** avec le plus petit score de **1.172 (**voir ***Figure 7.15*****).** Nous choisissons donc de partitionner l'ensemble des utilisateurs d'Instacart en 4 groupes homogènes.

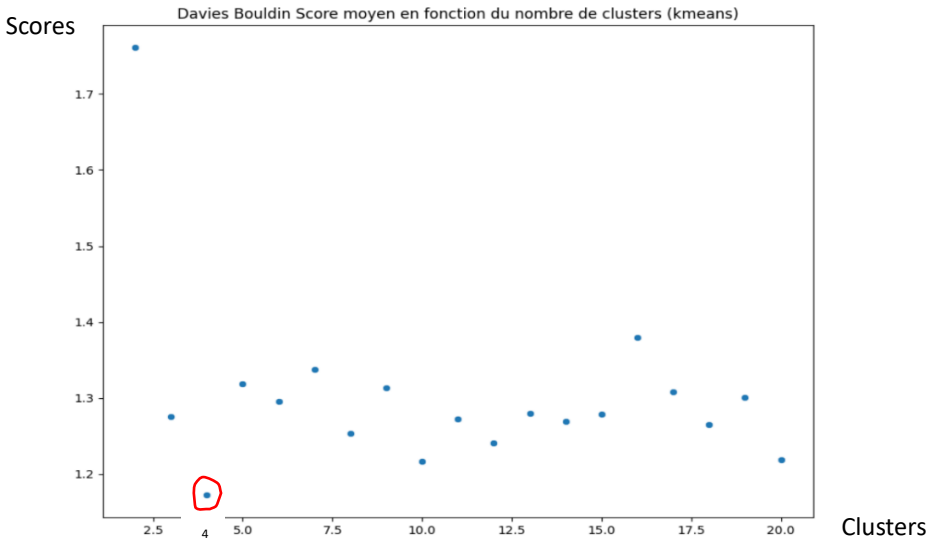

Figure 7.15: *Score de Davies Bouldin*

Et nos 4 clusters représentés via t-sne sont(Voir ***Figure 7.16***) :

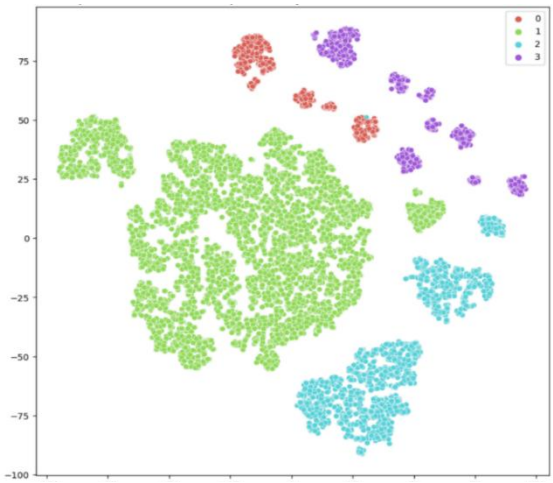

Figure 7.16:*Représentation t-SNE de la séparation du jeu de données via k-means (4) clusters*

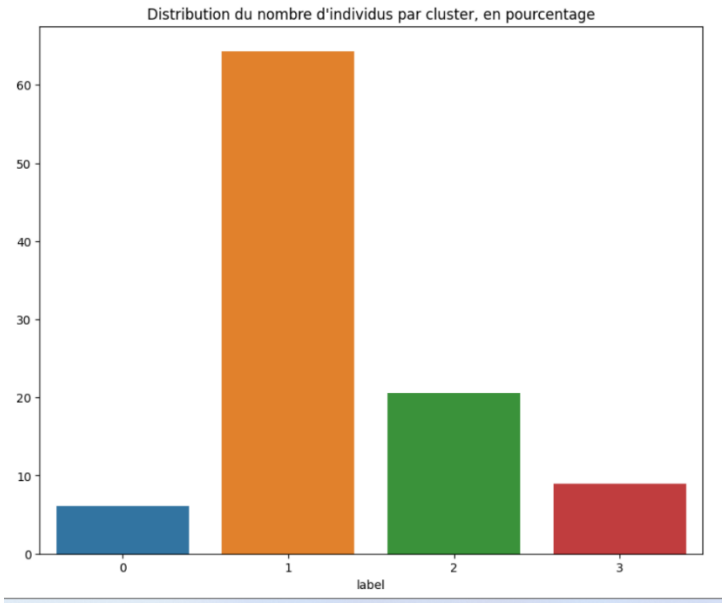

***Figure 7.17** : Distribution du nombre d'individus*

Dans la **Figure 7.17,** nous remarquons que plus de 60 % d'individus appartiennent au cluster 1, environ 20 % appartiennent au cluster 2, ensuite 10 % au cluster 3 et moins de 10 % au cluster 0.

7.3. Conclusion

Nous avons obtenu des résultats satisfaisants en utilisant les règles d'associations pour extraire des relations utiles, puis construire un modèle prédictif performant avec un apprentissage d'ensemble, enfin, nous avons réalisé un partitionnement avancé grâce à un réseau de neurones combiné à un algorithme de clustering.

Chapitre 8. Conclusion

Dans le cadre de ce travail, nous nous sommes intéressés à l'analyse du panier de consommation pour le développement de système de recommandation des produits d'une plateforme d'achat en ligne. Notre objectif consistait à extraire des règles d'associations, trouver un meilleur modèle de prédiction et réaliser un clustering efficace des utilisateurs.

Avant de résoudre notre problématique, nous avons présenté les différentes méthodes d'apprentissage automatique les plus connues qui existent dans la littérature et permettrait d'atteindre notre objectif. Nous avons rappelé les concepts de base des règles d'association, présenter l'algorithme Apriori , FP-Growth ainsi que les méthodes heuristiques. Ensuite nous avons présenté les méthodes d'ensembles telles que les arbres de décision, la forêt aléatoire avec ses variantes. Enfin, nous avons aussi présenté les réseaux de neurones et les méthodes de partitionnement.

A travers les expérimentations qui ont été effectués dans le cadre de ce mémoire, nous avons obtenu des résultats significatifs :

i. L'extraction des règles d'association à partir des données d'utilisateurs et leurs commandes, nous a permis de trouver des ensembles de produits que les utilisateurs pourraient acheter à nouveau ou ajouter à leur dernier panier. L'extraction de ces règles dépend fortement du choix de seuils de support et de confiance.

ii. Nous avons entraîné et obtenu un modèle prédictif performant avec l'algorithme XGBoost. Notre modèle atteint un F1-score de 61 %, ce qui est considéré comme acceptable. Nous pouvons donc l'utiliser pour prédire le comportement futur des utilisateurs individuellement.

iii. Enfin, nous avons partitionné les utilisateurs en 4 clusters en utilisant l'autoencodeur débruiteur pour extraire des caractéristiques, puis combiné à l'algorithme K-means, et avons visualisé les groupes avec t-SNE.

En dépit des méthodes avancées significatives présentées dans cette étude, plusieurs axes de recherche restent à explorer en raison de l'avancée majeure dans l'apprentissage automatique. De futurs travaux pourraient se concentrer sur la segmentation dynamique, pour identifier des groupes de clients en évolution car les préférences évoluent avec le temps. De plus , les modèles génératifs peuvent être utilisés pour générer des paniers de consommation synthétiques, créant ainsi des scénarios hypothétiques basés sur les tendances de consommation observées. En fin, en utilisant des modèles génératifs conditionnels, il est possible de personnaliser la génération en fonction des caractéristiques spécifiques des clients.

Bibliographie

1. Lavigne, F. *Artificially Intelligent - analyse du panier*. MSDN Magazine, 2018.
2. Kaushik, S. *How Is Machine Learning Evolving In 2024*. MobileAppDaily, 2024.
3. Ouedraogo, P.A., *Méthode d'élagage des règles d'association et estimation de la perte d'information dans les données médicales*, in *mathématiques et informatique appliquées*. Université du Québec à Trois-Rivières. p. 109. 2021.
4. Agrawal, R. and R. Srikant. *Fast algorithms for mining association rules*. in *Proc. 20th int. conf. very large data bases, VLDB*. Santiago, 1994.
5. Nikulski, J. *The Ultimate Guide to AdaBoost, random forests and XGBoost*. Towardsdatascience, 2020.
6. Mohssen, G.S. and C. Tjortjis, *A survey on association rules mining using heuristics.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, Santiago 1994: p. 1307.
7. Mohammed, Z., *Association Analysis: Basic Concepts and Algorithms*. Cambridge University Press. 2014.
8. Ali, L., *sélection des mots clés basée sur la classification et l'extraction des règles d'association*, in *mathématiques et informatique appliquées*. Université du Québec à Trois-Rivières. p. 139. 2017.
9. Jiawei, H., P. Jian, and Y. Yiwen, *Mining frequent patterns without candidate generation.* ACM sigmod record: p. 1-12. 2000.
10. Desyatnikov, R. *Frequent Pattern (FP) Growth Algorithm In Data Mining*. SoftwareTestingHelp, 2023.
11. Gagandeep, K. and A. Shruti, *A survey of genetic algorithm for association rule mining.* International Journal of Computer Applications, 2013.
12. Nabila, N., *Une approche d'optimisation par essaim de particules pour la recherche en mémoire de cas*, in *informatique cognitive*. Université du Québec à Montréal. p. 220. 2013.
13. Dervis, K., *Artificial bee colony algorithm.* scholarpedia, 2010.
14. Nagesh, K. and J. Reddy, *Ant Colony Optimization for Multi-Purpose Reservoir Operation.* Water Resources Management, 2006.
15. Chaouche, Y. *Explorez vos données avec des algorithmes non supervisés*. Openclassrooms, 2022.
16. Chloé-Agathe, A., *Introduction au Machine Learning 2e édition*. Dunod. 2022.
17. Aurélien, G., *Machine Learning avec Scikit-Learn: Mise en oeuvre et cas concrets*. Dunod. 2019.
18. Kapoor, R. *Apprentissage supervisé et non supervisé*. Alteryx; 2023.
19. Mezghani, N., *Arbres de décision.* Teluq, 2019.
20. François, D. and G. Rémi, *Apprentissage à partir d'exemples.* Notes de cours, 2000.
21. Leo, B., *Random forests.* Machine learning: p. 5-32, 2001.

22. Guillaume, M., *Machine Learning TD-Méthodes Ensemblistes.* Notes de cours, 2022.
23. Pall, G., B. Jon, and S. Johannes, *Random forests for land cover classification.* Pattern recognition letters, 2006: p. 294-300.
24. Yoav, F., S. Robert, and A. Naoki, *A short introduction to boosting.* Journal-Japanese Society For Artificial Intelligence, 1999: p. 771-780.
25. Chaouche, Y. *Modélisez vos données avec les méthodes ensemblistes*. Openclassrooms, 2022.
26. Mesfioui, M., *"Notes de cours sur l'acp".* Notes de cours, 2022.
27. Lloyd, S., *Least squares quantization in PCM.* IEEE Transactions on Information Theory: p. 129-137, 1982.
28. Belaidi, N. *Algorithme t-SNE : comment ça fonctionne*. Blent, 2022.
29. Pradesh, U. *Davies-Bouldin Index*. GeeksforGeeks, 2023.
30. Aurélien, G., *Deep Learning avec Keras et TensorFlow: Mise en œuvre et cas concrets.* Dunod. 2020.
31. Frank, R., *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan books Washington, DC,. 1962.
32. Donald, H., *The organization of behavior: A neuropsychological theory.* Psychology press. 2005.
33. Bento, C. *Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis*. Towardsdatascience, 2021.
34. Hinton, G., R. David, and W. Ronald, *Learning representations by back-propagating errors.* nature: p. 533-536, 1986.
35. Le Cun, Y. and C. Alfredo, *NYU Deep Learning.* Spring2021, 2021.
36. Le Cun, Y. and F. Fogelman-Soulié, *Modèles connexionnistes de l'apprentissage.* Intellectica: p. 114-143, 1987.
37. Diederik, K. and W. Max, *Auto-encoding variational bayes.* arXiv:1312.6114, 2013.
38. Van Rossum, G. and F. Drake, *Python*. 2009.
39. Stanley, J. *3 Million Instacart Orders, Open Sourced*. 2017.