UNIVERSITE DU QUEBEC A TROIS RIVIERES

TRAVAIL DE MÉMOIRE

COMME EXIGENCE DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

EXPLORATION DES DONNEES DANS UN ENVIRONNEMENT SEQUENTIEL AU MOYEN DES REGLES D'ASSOCIATION

Par Abdoulahat Tandia

Septembre 2025

Résumé

À l'ère du Big Data, la capacité à extraire des informations significatives à partir d'ensembles de données vastes et complexes constitue un avantage stratégique pour les entreprises, les chercheurs et les décideurs. Parmi les nombreuses techniques utilisées en exploration de données, l'extraction de règles d'association se distingue comme l'une des méthodes les plus largement appliquées pour découvrir des relations entre les éléments dans les données transactionnelles [1]. Cependant, les règles d'association classiques présentent une limite importante : elles ne prennent généralement pas en compte l'ordre temporel ou la séquence des événements (Agrawal & Srikant, 1995). Cette lacune est particulièrement significative dans les domaines où la chronologie des actions est essentielle, comme le commerce en ligne, l'analyse des comportements utilisateurs ou les systèmes de recommandation [2].

Ce mémoire se concentre sur l'analyse des données séquentielles à l'aide des règles d'association séquentielles, qui intègrent la dimension temporelle dans l'extraction des motifs. L'identification de motifs séquentiels fréquents permet de mettre en lumière des relations cachées entre événements successifs, facilitant la prédiction des comportements d'achat, la recommandation de produits ou encore la détection de tendances comportementales (Tarus et al., 2018 ; Trivonanda et al., 2020). Contrairement aux règles d'association classiques, ces techniques modélisent explicitement les relations temporelles entre les éléments.

Ce travail vise à approfondir la compréhension de l'exploitation des données séquentielles à l'aide des règles d'association séquentielles. Nous présenterons les concepts fondamentaux de cette approche, ainsi que le principal algorithme d'extraction utilisé, notamment PrefixSpan et GSP.

En parallèle, nous discuterons des applications concrètes de ces techniques dans le secteur commercial, en particulier pour la prédiction des comportements clients, la recommandation de produits et la conception de stratégies marketing ciblées. Le mémoire abordera également les défis spécifiques liés à l'analyse de données séquentielles, ainsi que les perspectives de recherche futures.

Abstract

In the era of big data, the ability to extract meaningful insights from vast and complex datasets has become a crucial advantage for businesses, researchers, and decision-makers alike. Among the many techniques used in data mining, association rule mining stands out as one of the most widely applied methods for discovering relationships between items in transactional data. However, traditional association rules are often limited by the fact that they do not consider the temporal order or sequence of events. This limitation is especially significant in domains where the sequence of actions plays a crucial role, such as in e-commerce, user behavior analysis, and recommendation systems.

This thesis focuses on the exploration of sequential data through the use of sequential association rules, which take into account the temporal ordering of events within a sequence. By identifying frequent sequential patterns, it is possible to uncover hidden relationships between events that occur over time, such as predicting customer behavior, recommending products, or identifying purchasing patterns. Unlike classical association rules, which focus on the co-occurrence of items without any regard for their order, sequential association rules model the relationships between items as they appear in specific sequences.

The objective of this research is to provide a comprehensive understanding of how sequential data can be explored through the lens of sequential association rule mining. We will delve into the foundational concepts of sequential association rules and discuss various algorithms used to extract such patterns, including Prefix-projected Sequential Pattern Mining (PrefixSpan) and GSP. Furthermore, we will explore the practical applications of these techniques in online commerce, particularly for behavior prediction, product recommendation, and marketing strategies. The thesis will also address the challenges of working with sequential data and provide insights into future research directions in this field.

Remerciements

Je souhaite commencer ces remerciements par mes parents. Mon père, Mouhamadou Habiboula Tandia, sans qui rien de cette aventure, au Québec, à quelques 6300km de mon Sénégal natal, ne serait possible. Tu es pour moi, un modèle, un exemple, une véritable source d'inspiration.

Ma maman, une femme brave, battante et inspirante. Tes prières, encouragements et ton support m'ont accompagné tout au long mon parcours.

Je remercie également Monsieur Ismail Biskri, mon directeur de recherche, pour son accompagnement bienveillant, sa disponibilité constante et la qualité de ses conseils. Sa rigueur scientifique, son expertise mais aussi son soutien m'ont permis de structurer mes recherches et d'approfondir mes réflexions tout au long de ce travail.

Je tiens à adresser mes sincères remerciements à mes enseignants et intervenants, qui m'ont transmis leurs connaissances et qui m'ont donné les outils nécessaires pour mener à bien mes études.

Je suis également reconnaissant envers mes proches et mes amis, pour leur patience, leur compréhension et leur soutien infaillible tout au long de ces années d'études intense.

Enfin, je n'oublie pas les personnes ayant participé à mes recherches, notamment celles et ceux qui m'ont accordé leur temps pour des entretiens, des questionnaires ou des échanges d'idées. Leur collaboration a été essentielle à la réalisation de ce travail.

À toutes ces personnes, je dis un grand merci.

Table des matières

Liste des tableaux 8
Liste des figures 9
Chapitre I : Introduction 1
Chapitre II : Fondamentaux des règles d'association 6
1. Introduction aux règles d'association 6
2. Les différents types de règles d'association et leur domaine d'application 6
3. Terminologies fondamentales des règles d'association 7
3.1. Définition d'un item et d'un itemset 7
3.2 Exemple de jeu de données 7
3.3. Fréquence des items 7
3.4. Support 8
3.5. Confiance 8
3.6 Lift 8
3.7. Exemples de règles générées 9
4. Algorithmes classiques d'extraction de règles 9
4.1. L'algorithme Apriori 10
4.2. L'algorithme FP-Growth 11
5. Comparaison entre Apriori et FP-Growth 12
6. Limites des approches classiques 13
7. Vers l'extraction séquentielle 13
Chapitre III : Application des règles d'association aux données séquentielles 15
1. Définition et caractéristiques des données séquentielles 15
1.1 Définition 15
1.2 Exemple pratique d'achats 15
1.3 Caractéristiques clés des données séquentielles 15
2. Typologie des données séquentielles 17
3. Applications 17
4. Modélisation des données séquentielles 18
4.1. Représentation des données sous forme de séquences 18
4.2. Extraction des caractéristiques des séquences 19

4.3 Modèles adaptés aux données séquentielles 19
5. Règles d'association séquentielles 20
5.1. Particularité des règles séquentielles 20
5.2 Motifs séquentiels 20
5.3 Principaux types de motifs séquentiels 21
5.4 Applications des motifs séquentiels 21
6 Algorithmes pour l'exploration des données séquentielles 22
6.1 GSP (Generalized Sequential Pattern) 22
6.2 PrefixSpan (Prefix-projected Sequential Pattern Mining) 23
Chapitre IV : Méthodologie 27
1. Objectif de la méthodologie 27
2. Préparation des données 27
2.1 Description du jeu de données 27
2.2 Nettoyage des données 28
2.3 Transformation des variables 28
2.4 Modélisation séquentielle du client 28
2.5 Calculs de caractéristiques et analyse RFM 29
3. Clustering 30
3.1 Normalisation des données 30
3.2 Reduction de dimension 30
3.3. Détermination du nombre optimal de clusters (k) 30
3.4. Application des algorithmes de clustering 31
3.5 Évaluation des clusters 31
4. Analyse prédictive 33
4.1. Définition des fonctions d'analyse 33
4.2 Modèles testés 33
4.3 Méthodologie d'évaluation et d'optimisation 34
5. Choix des algorithmes d'exploration séquentielle 35
5.1 Algorithmes considérés 35
Chapitre V : Implémentation 37
1 Introduction 37
2 Interface utilisateur 37
3. Préparation des données 37

3.1 Chargement et conversion des types de données 38
3.2 Extraction des caractéristiques clients 38
3.3 Analyse RFM 41
3.4 Construction du profil client final et visualisation 43
Sauvegarde des Données Prétraitées 43
4. Clustering 44
4.1. Chargement des données et préparation 44
4.2. Standardisation et Réduction de Dimension (PCA) 46
4.3 Analyse des différents algorithmes de clustering 47
4.4 Conclusion sur le Clustering 57
5. Analyse Prédictive 58
5.1. Chargement et Prétraitement des Données 58
5.2. Définition d'Outils de Visualisation et d'Aide à l'Optimisation 59
5.3. Construction et évaluation des pipelines de modélisation 61
6. Comparaison Globale des Modèles et Choix de la Segmentation 72
Chapitre VI : Expérimentation 75
1. Nature de l'expérimentation 75
2. Description du dataset 76
2.1 Structure des données 76
2.2 Pertinence du dataset 76
Ce jeu de données présente plusieurs atouts majeurs pour une étude sur les règles d'association séquentielles : 76
3. Les différents paramètres 77
3.1. Paramètres liés à la préparation des données 77
3.2. Paramètres liés au clustering 77
3.3. Paramètres liés à la classification prédictive 77
3.4 Paramètres des algorithmes explorés GSP et PrefixSpan 78
4. Résultats et discussions 78
4.1 GSP 78
4.2 PrefixSpan 80
5. Synthèse des résultats et discussions 81
Chapitre VII : Conclusion 84

Liste des tableaux

Tableau 1Échantillon de transactions dans le commerce 7
Tableau 2 Règles d'association issues du jeu de transaction 9
Tableau 3 Représentation de la fréquence et du support des différents itemsets 10
Tableau 4 Représentation de la fréquence et du support des différents itemsets de taille 2 11
Tableau 5 Représentation de la fréquence et du support des différents itemsets de taille 3 11
Tableau 6 Comparaison entre Apriori et FP-Growth 13
Tableau 7 Exemple de transactions séquentielles 15
Tableau 8 Exemple de transactions séquentielles 17
Tableau 9 Applications des données séquentielles 18
Tableau 10 Exemple de séquences clients 20
Tableau 11 Domaine d'application des motifs séquentiels 22
Tableau 12 Ensemble de transactions séquentielles 22
Tableau 13 Comparaison des algorithmes GSP et PrefixSpan 24
Tableau 14 Structure des données brutes 28
Tableau 15 Resulats GSP PrefixSpan 28

Liste des figures

Figure 1 Visualisation de la variance expliquée après PCA 47
Figure 2 Courbe Elbow post PCA 48
Figure 3 Segmentation des clusters avec K-Means 49
Figure 4 Diagramme de Silhouette post PCA 51
Figure 5 Segmentation des clusters avec Fuzzy C-Means 53
Figure 6 Diagramme de Silhouette Fuzzy C-Means 54
Figure 7 Graphe des distances k-NN 55
Figure 8 Segmentation des clusters avec DBSCAN 56
Figure 9 Segmentation des clusters avec Birch 57
Figure 10 Learning Curve issue de SVC 63
Figure 11 Validation curve issue de SVC 64
Figure 12 Learning Curve issue de MPLClassifier 67
Figure 13 Learning Curve RFC 69
Figure 14 Learning Curve issue de DecisionTree 72

Chapitre I : INTRODUCTION

Chapitre I : Introduction

Au cours des dernières décennies, la production et la collecte de données ont connu une croissance exponentielle. La généralisation des systèmes d'information, le développement du commerce électronique, l'essor des réseaux sociaux et l'automatisation croissante des processus industriels ont conduit les entreprises et organisations à générer et stocker des volumes massifs de données qualifiés de Big Data. Ces données, à la fois hétérogènes, multidimensionnelles et dynamiques, ouvrent de nouvelles perspectives d'analyse, mais posent également des défis méthodologiques majeurs. Dans ce contexte, l'exploration de données (data mining) s'est imposée comme un champ incontournable de recherche et d'application, visant à extraire automatiquement des connaissances pertinentes à partir de vastes ensembles de données. L'enjeu est de transformer cette abondance d'informations brutes en éléments décisionnels exploitables, qu'il s'agisse de détecter des motifs récurrents, de mettre en évidence des tendances ou de révéler des structures latentes [1][2].

Parmi les différentes techniques développées dans ce domaine, l'extraction de règles d'association occupe une place particulière. Popularisée dès les années 1990 par Agrawal et Srikant avec l'algorithme Apriori [3][4], elle a été initialement conçue pour analyser des transactions commerciales et identifier des produits fréquemment coachetés. Ce paradigme repose sur la recherche de relations du type si un client achète un produit A, il a une forte probabilité d'acheter également un produit B. L'algorithme Apriori introduit alors les concepts de support et de confiance comme mesures fondamentales pour évaluer la pertinence des associations extraites. Au fil du temps, de nombreuses variantes et améliorations ont été proposées pour pallier les limites computationnelles d'Apriori, notamment FP-Growth [5], Eclat, ou encore AprioriTID. FP-Growth, en particulier, a permis de réduire considérablement le coût de génération des candidats grâce à une structure arborescente appelée FP-tree. Ces évolutions ont contribué à élargir les applications des règles d'association à des domaines variés tels que la bio-informatique, la cybersécurité, la médecine, la logistique ou le marketing prédictif [6][7].

Cependant, les approches classiques présentent une limite fondamentale : elles considèrent les transactions comme indépendantes et ne tiennent pas compte de l'ordre chronologique des événements. Or, dans de nombreux contextes réels, la dimension temporelle est essentielle à la compréhension des comportements. En marketing, par exemple, un client peut commencer par

acheter des produits de première nécessité avant de se tourner vers des articles complémentaires. En analyse comportementale, les utilisateurs suivent souvent des séquences d'actions typiques (installation d'une application, essai gratuit, achat d'une version payante). En médecine, la succession des symptômes ou des traitements permet d'identifier des schémas de progression de maladies.

Face à cette nécessité d'intégrer la notion de temps, est apparue une nouvelle branche du data mining : l'exploration de motifs séquentiels fréquents (Sequential Pattern Mining, SPM) [8][9]. Introduite par Agrawal et Srikant en 1995, cette approche vise à découvrir des séquences d'événements qui apparaissent fréquemment dans les données, tout en respectant l'ordre temporel. Les algorithmes tels que GSP (Generalized Sequential Patterns) [9] et PrefixSpan (Prefix-projected Sequential Pattern Mining) [10] ont marqué des avancées majeures dans ce domaine.

Le GSP repose sur une stratégie itérative de type Apriori-like, générant des séquences candidates de longueur croissante tout en éliminant celles qui ne satisfont pas le seuil minimal de support. Cependant, cette approche devient rapidement coûteuse pour de grandes bases de données. Pour y remédier, PrefixSpan, proposé par Pei et al. (2001), introduit une méthode basée sur la projection de préfixes, qui réduit le nombre de scans nécessaires et améliore l'efficacité du processus d'extraction. Depuis, plusieurs variantes ont vu le jour, telles que SPADE, SPAM, ou BIDE, optimisant soit la mémoire, soit la complexité temporelle.

Les applications des motifs séquentiels sont nombreuses : analyse du parcours client [11][12], détection de comportements frauduleux [13], analyse des séquences de navigation Web [14], ou encore étude des traitements médicaux et génétiques [15]. Dans le domaine du commerce de détail, l'exploitation de ces motifs permet de mieux comprendre les trajectoires d'achat et de construire des stratégies de recommandation plus fines, orientées sur le quand et le comment plutôt que sur le simple quoi.

Des travaux récents ont également cherché à intégrer des aspects complémentaires tels que la pondération temporelle [16], la prise en compte du contexte (saisonnalité, promotions, géolocalisation) [17], ou encore l'usage d'algorithmes hybrides combinant apprentissage automatique et exploration séquentielle [18][19]. Ces approches marquent la convergence entre data mining, machine learning et intelligence artificielle, offrant ainsi des modèles plus dynamiques et interprétables.

Ainsi, l'évolution des méthodes d'extraction de règles, des associations statiques vers les motifs séquentiels temporels, reflète un changement de paradigme majeur dans l'analyse des comportements complexes. L'enjeu actuel réside dans l'intégration de ces techniques au sein de chaînes analytiques complètes combinant segmentation, clustering, classification prédictive et extraction de motifs temporels.

La problématique au cœur de ce mémoire peut ainsi se formuler de la manière suivante : comment adapter et appliquer les méthodes d'extraction de règles d'association à des données séquentielles afin de mettre en évidence des régularités temporelles significatives dans les comportements d'achat des clients d'un supermarché ? Autrement dit, l'enjeu est de dépasser la simple détection de cooccurrences statiques pour découvrir des enchaînements d'événements susceptibles de révéler des régularités comportementales exploitables.

Pour répondre à cette problématique, l'objectif de ce travail est de concevoir et d'implémenter une méthodologie complète d'extraction de motifs séquentiels fréquents à partir de données transactionnelles enrichies d'une dimension temporelle. Cette méthodologie suit une chaîne analytique cohérente, allant du traitement des données brutes à la découverte de motifs temporels complexes.

Elle repose sur une série de notebooks Python développés spécifiquement pour ce projet, chacun correspondant à une étape clé.

- La première étape est consacrée à la préparation et à la transformation des données, incluant leur nettoyage et leur conversion en séquences temporelles exploitables.
- La seconde consiste en une phase de segmentation et de clustering destinée à identifier des profils homogènes de clients et à réduire l'hétérogénéité des comportements.
- Sur cette base, différents modèles de classification supervisée tels que SVC, Random Forest, MLP et Naive Bayes sont entraînés afin de prédire l'appartenance d'un client à un segment donné à partir d'indicateurs tels que les mesures RFM ou la diversité des produits achetés.
- Enfin, les algorithmes GSP et PrefixSpan [9][10] sont appliqués sur les séquences obtenues afin de détecter des régularités temporelles dans les parcours d'achat.

Au-delà de l'aspect technique, l'intérêt de cette étude est double. Sur le plan scientifique, elle illustre l'adaptation des règles d'association à un environnement séquentiel et souligne la complémentarité entre segmentation, classification et extraction séquentielle dans l'analyse des comportements clients. Sur le plan pratique, elle propose des résultats directement exploitables pour les entreprises, notamment dans le marketing et la gestion de la relation client. Les motifs séquentiels découverts peuvent contribuer à améliorer les systèmes de recommandation personnalisée, à affiner les stratégies de ciblage, à optimiser la gestion des stocks et à anticiper les besoins futurs des consommateurs. L'intégration de la dimension temporelle permet ainsi de dépasser une analyse statique des comportements pour se rapprocher d'une modélisation dynamique et prédictive des parcours d'achat.

Ce mémoire est structuré en sept chapitres principaux. Le premier étant l'introduction, le deuxième présente le cadre théorique et l'état de l'art, en introduisant les concepts fondamentaux de l'exploration de données, des règles d'association et de leurs variantes séquentielles. Le troisième est consacré à l'application de ces règles d'association. Le quatrième expose la méthodologie de notre travail, de la préparation des données à l'extraction des modèles en passant par le clustering. Le cinquième se focalise sur l'implémentation proprement dite. En sixième position vient maintenant le chapitre axé sur l'expérimentation. Enfin, nous consacrons le dernier chapitre à la conclusion.

Chapitre II : Fondamentaux des règles d'association

Chapitre II : Fondamentaux des règles d'association

1. Introduction aux règles d'association

Les règles d'association sont des outils d'exploration de données permettant de découvrir des relations fréquentes ou des motifs cachés entre des éléments au sein d'une base de données. Elles identifient des corrélations de la forme « si A se produit, alors B est susceptible de se produire », et sont mesurées par des indicateurs tels que le support, la confiance et le lift.

2. Les différents types de règles d'association et leur domaine d'application

Les règles d'association se déclinent en plusieurs variantes adaptées à différents contextes d'analyse :

- Règles d'association classiques : ce sont les plus courantes, introduites par Agrawal et al. [3][4]. Elles consistent à identifier des relations de co-occurrence entre des ensembles d'items dans de grandes bases transactionnelles. Leur principal domaine d'application est l'analyse des paniers d'achats, utilisée notamment dans le commerce de détail pour comprendre quels produits sont fréquemment achetés ensemble.

- Règles d'association maximales : elles visent à réduire la redondance en ne conservant que les ensembles d'items maximaux fréquents, c'est-à-dire ceux qui ne sont contenus dans aucun autre ensemble plus grand et fréquent. Cela permet de simplifier l'interprétation des résultats tout en préservant les motifs significatifs, notamment dans les systèmes de recommandation [1].

- Règles d'association séquentielles : contrairement aux règles classiques qui ignorent l'ordre, celles-ci tiennent compte de la dimension temporelle ou de l'ordre d'apparition des items [7][8][9]. Elles sont particulièrement adaptées à l'analyse des comportements clients dans des séquences d'achats, la détection de parcours utilisateurs sur le web ou encore l'étude des séquences médicales (par exemple, succession de traitements).

- Règles d'association floues : elles permettent d'intégrer des degrés d'appartenance ou d'incertitude en utilisant la logique floue. Ainsi, elles s'appliquent aux situations où les frontières entre catégories ne sont pas nettes, comme l'analyse de préférences clients subjectives ou la classification de profils de consommation [1][6].

- Règles d'association causales : au-delà de la simple corrélation, ces règles cherchent à identifier des relations de causalité entre variables. Bien qu'elles soient plus complexes à établir, elles trouvent leur application dans les systèmes d'aide à la décision, la bio-informatique ou encore l'évaluation de campagnes marketing [6].

Ces différentes familles de règles d'association élargissent considérablement le champ des applications possibles, allant du commerce électronique [10], à la santé, en passant par l'analyse comportementale, les réseaux sociaux, ou encore les systèmes prédictifs. Elles constituent ainsi un pilier méthodologique incontournable de l'exploration de données et de la fouille de motifs dans des environnements riches et complexes.

Pour comprendre les règles d'association, les terminologies clés suivant doivent être définis : le support, la confiance, la fréquence et le lift.

3. Terminologies fondamentales des règles d'association

3.1. Définition d'un item et d'un itemset

- Item : un produit ou élément individuel.
- Itemset : un ensemble de produits apparaissant ensemble dans une ou plusieurs transactions.

3.2 Exemple de jeu de données

Afin de permettre une compréhension efficiente de ces différentes terminologies citées précédemment, nous allons, à travers un exemple, définir un jeu de données hypothétique et identifier les éléments clés qui composent une règle d'association.

Exemple de jeu de données de transactions

Prenons en exemple ce tableau

Transaction ID	Produits Achetés
1	Pomme, Banane, Jus
2	Pomme, Yaourt
3	Pomme, Banane
4	Banane, Jus, Yaourt
5	Pomme, Banane, Jus, Yaourt

Tableau 1Échantillon de transactions dans le commerce

Dans cet exemple, chaque ligne représente une transaction où plusieurs produits ont été achetés ensemble. Nous allons maintenant identifier les terminologies à partir de cet ensemble de données.

3.3. Fréquence des items

La fréquence d'un item représente combien de fois cet item apparaît dans les transactions. Cela nous aide à comprendre l'importance de chaque produit dans le commerce.

- Pomme : apparaît dans les transactions 1, 2, 3, 5 → fréquence = 4
- Banane : apparaît dans les transactions 1, 3, 4, 5 → fréquence = 4
- Jus : apparaît dans les transactions 1, 4, 5 → fréquence = 3
- Yaourt : apparaît dans les transactions 2, 4, 5 → fréquence = 3

3.3.1 Fréquence des itemsets

- {Pomme, Banane} : apparaît dans les transactions 1, 3, 5 → fréquence = 3
- {Pomme, Jus} : apparaît dans les transactions 1, 5 → fréquence = 2
- {Banane, Jus} : apparaît dans les transactions 1, 4, 5 → fréquence = 3
- {Yaourt, Banane} : apparaît dans les transactions 4, 5 → fréquence = 2

3.4. Support

Le support d'un itemset mesure la proportion de transactions qui contiennent cet ensemble de produits. Il est calculé comme suit :

$$\text{Support}(X) = \frac{\text{Nombre de transactions contenant X}}{\text{Nombre total de transactions}}$$

* X étant l'itemset concerné

Par exemple, pour l'itemset {Pomme, Banane}, il apparaît dans 3 transactions sur 5, donc son support est :

$$\text{Support}(\{\text{Pomme, Banane}\}) = \frac{3}{5} = 0{,}6 \text{ ou } 60\%$$

3.5. Confiance

La confiance d'une règle mesure la probabilité qu'un produit B soit acheté, étant donné qu'un produit A ai été acheté. Une règle d'association a la forme suivante :

Si A alors B

La confiance est calculée comme suit :

$$\text{Confiance}(A \rightarrow B) = \frac{\text{Support}(A \cup B)}{\text{Support}(A)}$$

Par exemple, si nous avons la règle {Pomme} → {Banane}, nous pouvons calculer la confiance de la règle comme suit :

- Support ({Pomme, Banane}) = 3/5 = 0,6
- Support ({Pomme}) = 4/5 = 0,8

La confiance de la règle {Pomme} → {Banane} est donc :

$$\text{Confiance}(\{\text{Pomme}\} \rightarrow \{\text{Banane}\}) = \frac{\text{Support}(\text{Pomme} \cup \text{Banane})}{\text{Support}(\text{Pomme})} = \frac{0.6}{0.8} \quad 0{,}75 \text{ (ou } 75\%)$$

3.6 Lift

Le lift mesure la force de la relation entre les produits A et B, en comparant la confiance avec la probabilité d'occurrence de B indépendamment de A :

$$\text{Lift}(A \rightarrow B) = \frac{\text{confianceA} \rightarrow \text{B})}{\text{Support(B)}}$$

Par exemple, pour la règle {Pomme} →{Banane}, nous avons :

$$\text{Lift}(\{\text{Pomme}\} \rightarrow \{\text{Banane}\}) = \frac{0.75}{0.8} = 0{,}9375$$

Un lift supérieur à 1 suggère que A et B sont plus susceptibles de se produire ensemble que si leurs apparitions étaient indépendantes.

3.7. Exemples de règles générées

Sur la base de ces calculs, nous pourrions générer quelques règles d'association intéressantes :

Règles	Support	Confiance	Lift
{Pomme} → {Banane}	0,6	0,75	0,9375
{Banane} → {Pomme}	0,6	0,75	0,9375
{Jus} → {Pomme}	0,6	0,67	0,83
{Yaourt} → {Banane}	0,4	0,67	0,833

Tableau 2 Règles d'association issues du jeu de transaction

4. Algorithmes classiques d'extraction de règles

Les algorithmes classiques sont des outils puissants dans le domaine du data mining spécifiquement pour la découverte de règles d'association dans de grands ensembles de données transactionnelles. Leur utilité se manifeste principalement dans des contextes où l'on cherche à comprendre les relations implicites entre différents éléments (produits, comportements, actions, etc.) présents dans les données. Ils permettent donc aux entreprises de prendre des décisions plus éclairées, d'optimiser leurs processus, d'améliorer l'expérience client, et de développer des stratégies commerciales plus efficaces. Nous avons principalement les algorithmes Apriori et FP-Growth.

Application

Examinons les algorithmes Apriori et FP-Growth en utilisant l'exemple de transactions de commerce en ligne que nous avons déjà vu, pour comprendre comment ils fonctionnent pour générer des règles d'association.

Rappel de l'exemple de transactions

Nous prendrons comme exemple notre jeu de données du tableau 1

Commençons par l'algorithme Apriori puis nous passerons à l'algorithme FP-Growth.

4.1. L'algorithme Apriori

L'algorithme Apriori est un algorithme basé sur les propriétés des itemsets fréquents. Il génère des itemsets fréquents de taille croissante, en commençant par les itemsets de taille 1, puis en augmentant progressivement la taille de l'ensemble des produits (itemsets) à chaque itération. L'algorithme repose sur l'idée que tous les sous-ensembles d'un itemset fréquent sont également fréquents [4].

Étapes de l'algorithme Apriori

Étape 1 : Générer les itemsets de taille 1:

Nous comptons combien de fois chaque produit apparaît dans les transactions pour déterminer la fréquence.

Produit	Fréquence	Support (en %)
Pomme	4	80%
Banane	4	80%
Jus	3	60%
Yaourt	3	60%

Tableau 3 Représentation de la fréquence et du support des différents itemsets

Si nous appliquons un seuil de support minimum de 60% (c'est-à-dire 3 transactions sur 5), les produits Pomme, Banane, Jus, et Yaourt sont tous fréquents (ils satisfont le seuil de support).

Étape 2 : Générer les itemsets de taille 2 à partir des itemsets fréquents

Ensuite, nous générons des paires de produits et calculons leur support (combien de fois ces paires apparaissent ensemble dans les transactions).

Itemset (de taille 2)	Fréquence	Support (en %)
{Pomme, Banane}	3	60%
{Pomme, Jus}	2	40%
{Pomme, Yaourt}	2	40%
{Banane, Jus}	3	60%
{Banane, Yaourt}	2	40%

{Jus, Yaourt}	2	40%

Tableau 4 Représentation de la fréquence et du support des différents itemsets de taille 2

Les itemsets {Pomme, Banane} et {Banane, Jus} sont fréquents, car leur support est de 60% (ce qui est égal au seuil de 60%). Les autres paires (comme {Pomme, Jus} ou {Jus, Yaourt}) ne sont pas fréquentes car leur support est inférieur à 60%.

Étape 3 : Générer les itemsets de taille 3

Enfin, nous générons des itemsets de taille 3 à partir des itemsets fréquents précédemment trouvés et calculons leur support.

Itemset (de taille 3)	Fréquence	Support (en %)
{Pomme, Banane, Jus}	2	40%
{Banane, Jus, Yaourt}	2	40%

Tableau 5 Représentation de la fréquence et du support des différents itemsets de taille 3

Les itemsets de taille 3 {Pomme, Banane, Jus} et {Banane, Jus, Yaourt} ont un support de 40%, ce qui est inférieur au seuil de 60%, donc ils ne sont pas fréquents.

Conclusion de l'algorithme Apriori

À la fin, les itemsets fréquents sont :

- Itemsets de taille 1 : {Pomme}, {Banane}, {Jus}, {Yaourt}
- Itemsets de taille 2 : {Pomme, Banane}, {Banane, Jus}

Aucun itemset de taille 3 n'est fréquent car leur support est inférieur au seuil de 60%.

4.2. L’algorithme FP-Growth

L'algorithme FP-Growth (Frequent Pattern Growth) est une méthode plus efficace pour générer des itemsets fréquents. Il n'utilise pas une approche basée sur les candidats comme Apriori, mais construit une structure d'arbre, appelée arbre FP (Frequent Pattern Tree), pour extraire directement les itemsets fréquents.[5]

Étapes de l'algorithme FP-Growth

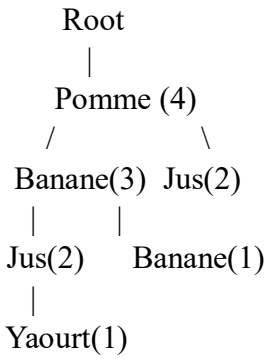
Étape 1 Construction de l'arbre FP

L'algorithme FP-Growth commence par construire un arbre FP qui représente les transactions de manière compacte. Il procède comme suit :

- Trier les produits dans chaque transaction par fréquence (produits les plus fréquents en premier).

- Construire l'arbre FP en parcourant les transactions et en insérant les éléments dans les chemins de l'arbre, en fusionnant les éléments partagés.

Par exemple, en ordonnant les produits par fréquence décroissante (Pomme, Banane, Jus, Yaourt), on obtient un arbre FP qui va organiser les transactions comme suit :



Étape 2. Extraction des itemsets fréquents

L'arbre FP permet ensuite de parcourir l'arbre pour extraire des itemsets fréquents en fonction des chemins qui apparaissent ensemble. Les chemins dans l'arbre FP montrent les itemsets qui sont fréquents.

Les itemsets fréquents extraits de l'arbre FP sont similaires à ceux obtenus avec Apriori :

- Itemsets de taille 1 : {Pomme}, {Banane}, {Jus}, {Yaourt}
- Itemsets de taille 2 : {Pomme, Banane}, {Banane, Jus}
- Itemsets de taille 3 : aucun itemset fréquent de taille 3 n'est trouvé.

5. Comparaison entre Apriori et FP-Growth

On peut voir la différence entre ces deux algorithmes dans ce tableau

Critère	Apriori	FP-Growth
Méthode	Génère des candidats et les vérifie	Crée un arbre FP et extrait directement les patterns fréquents
Efficacité	Moins efficace car nécessite plusieurs balayages de l'ensemble de données	Plus efficace car évite la génération de candidats
Mémoire	Peut nécessiter plus de mémoire pour stocker les candidats	Utilise un arbre compact pour réduire la mémoire

Adaptabilité aux grands ensembles de données	Moins performant avec de grands ensembles de données	Plus adapté aux grands ensembles de données
Exécution	Plus lent pour des bases de données vastes	Plus rapide en général avec de grands ensembles de données

Tableau 6 Comparaison entre Apriori et FP-Growth

On peut donc en déduire que l'algorithme Apriori est plus simple mais peut devenir lent avec des ensembles de données volumineux, car il génère de nombreux candidats. FP-Growth, quant à lui, est plus efficace pour des ensembles de données plus grands car il évite la génération de candidats en construisant un arbre FP compact, ce qui améliore les performances.

6. Limites des approches classiques

6.1. Problèmes de performance

- Génération excessive de candidats avec Apriori
- Complexité mémoire avec des bases très volumineuses

6.2. Sensibilité au seuil de support

- Risque d'éliminer des motifs rares mais utiles
- Difficile d'ajuster le seuil optimal

6.3. Ignorance de la dimension temporelle

- Aucune prise en compte de l'ordre d'apparition des items
- Impossible d'identifier des motifs séquentiels ou des dépendances temporelles

7. Vers l'extraction séquentielle

Les approches classiques d'extraction de règles d'association permettent d'identifier des corrélations entre des items, mais elles négligent la dimension temporelle. Or, dans de nombreux contextes réels, par exemple, le suivi des achats successifs des clients, l'évolution des habitudes de consommation, ou encore l'analyse de séquences d'événements, l'ordre dans lequel apparaissent les items est fondamental.

Ainsi, il ne suffit pas de savoir que deux produits sont souvent achetés ensemble, mais il est tout aussi pertinent d'identifier dans quel ordre ils sont achetés. Par exemple :

- Une règle classique dira : *« les clients qui achètent du pain achètent aussi du beurre »* ;
- Une règle séquentielle précisera : *« les clients achètent généralement du pain, puis du beurre lors de leur achat suivant »*.

C'est pourquoi, afin de mieux modéliser les comportements et d'exploiter l'aspect chronologique des données, il est nécessaire de s'intéresser aux règles séquentielles et aux algorithmes qui les exploitent. Cette approche constitue le cœur du chapitre suivant.

Chapitre III : Application des règles d’association aux données séquentielles

Chapitre III : Application des règles d'association aux données séquentielles

1. Définition et caractéristiques des données séquentielles

1.1 Définition

Les données séquentielles désignent des ensembles d'informations où l'ordre des éléments ou des événements est significatif. Ces données sont souvent collectées au fil du temps et représentent des interactions, des transactions ou des observations qui évoluent de manière chronologique. La séquence elle-même apporte des informations contextuelles essentielles pour l'analyse [7].

1.2 Exemple pratique d'achats

Imaginons un commerce en ligne qui vend des produits pour la maison, notamment des meubles, des appareils électroniques et des articles de décoration. Voici un ensemble de transactions de clients sur une période de deux semaines (du 01 au 15 octobre 2024) :

Client	Date	Articles achetés
C1	01-10-2024	Chaise, Table
C1	05-10-2024	Lampe, Tapis
C2	02-10-2024	Canapé, Table
C2	07-10-2024	Lampe, Table
C3	03-10-2024	Table, Chaise
C3	08-10-2024	Canapé, Lampe, Tapis
C4	04-10-2024	Table, Lampe
C4	10-10-2024	Canapé, Tapis, Chaise

Tableau 7 Exemple de transactions séquentielles

1.3 Caractéristiques clés des données séquentielles

Nous allons maintenant analyser les caractéristiques des données séquentielles à partir de cet exemple.

1.3.1 Ordre temporel

L'ordre temporel fait référence à la séquence dans laquelle les événements se produisent. Il permet de comprendre comment les clients évoluent dans leurs comportements d'achat au fil du temps.

Exemple dans notre commerce :
Prenons les achats de Client C1:
- Le 01-10-2024, C1 a acheté une chaise et une table.
- Le 05-10-2024, C1 a acheté une lampe et un tapis.

1.3.2 Temporalité

La temporalité fait référence à la durée ou aux intervalles de temps entre les événements. Cela peut indiquer des comportements d'achat spécifiques au fil du temps, comme des achats impulsifs ou des achats planifiés après un certain délai.

Exemple dans notre commerce :
- Client C1 : Un intervalle de 4 jours entre l'achat de la chaise et de la table (01-10-2024) et l'achat de la lampe et du tapis (05-10-2024).
- Client C3 : Un intervalle de 5 jours entre l'achat de la table et de la chaise (03-10-2024) et l'achat du canapé, lampe et tapis (08-10-2024).

1.3.3 Dépendances

Les dépendances désignent la relation entre différents événements dans une séquence. En d'autres termes, un événement passé influence la probabilité qu'un autre événement se produise par la suite.

Exemple dans notre commerce :

- Client C3 achète une table et une chaise (03-10-2024), puis un canapé, une lampe et un tapis (08-10-2024).

1.3.4 Variabilité

La variabilité se réfère à la diversité dans les séquences d'achats des clients. Les comportements d'achat peuvent être très différents d'un client à l'autre, tant en termes de produits achetés que de fréquence d'achats.

Exemple dans notre commerce :

- Client C1 effectue deux achats (le 01-10-2024 et le 05-10-2024), tandis que Client C2 effectue également deux achats, mais sur un intervalle plus long (02-10-2024 et 07-10-2024).

- Client C3 effectue deux achats aussi, mais avec des articles différents (meubles, puis canapé, lampe et tapis).

La variabilité des achats entre les clients montre que certains clients peuvent acheter des articles de manière rapprochée et dans des catégories similaires, tandis que d'autres peuvent acheter des articles très différents ou avec des intervalles de temps différents.

1.3.5 Récurrence

La récurrence se réfère au fait que certains événements ou comportements se répètent régulièrement au sein des séquences.

Exemple dans notre commerce :
- Client C2 a acheté une table et une lampe lors de deux achats différents (02-10-2024 et 07-10-2024).

Les récurrences des achats de produits comme la table et la lampe peuvent indiquer qu'un client a tendance à acheter des produits complémentaires dans le temps.

1.3.6 Longévité

La longévité se réfère à la durée pendant laquelle un client reste actif, c'est-à-dire combien de temps il continue à faire des achats. Cela peut être observé dans les séquences d'achats à long terme.

Exemple dans notre commerce :
- Client C1 a fait deux achats espacés de 4 jours.
- Client C4, en revanche, fait ses achats sur une période de 6 jours (le 04-10-2024 et le 10-10-2024).

La longévité des séquences peut donner des indices sur les clients qui restent actifs plus longtemps par rapport à ceux qui sont plus occasionnels.

On peut donc en déduire que les données séquentielles dans un commerce sont extrêmement riches et peuvent fournir des informations précieuses sur le comportement des clients.[8]

2. Typologie des données séquentielles

Plusieurs auteurs ont proposé des classifications détaillées des algorithmes de fouille séquentielle [12]

Type de données	Exemple
Séries chronologiques	Températures, prix journaliers
Données d'événements	Logs, transactions clients
Données d'interactions	Clics, navigation sur site
Données biologiques	Séquences ADN, expression génique

Tableau 8 Exemple de transactions séquentielles

3. Applications

Les données séquentielles jouent un rôle crucial dans de nombreux domaines, permettant d'extraire des informations précieuses grâce à leur structure temporelle. Comprendre leurs caractéristiques et les appliquer dans divers contextes offre d'importantes opportunités pour l'innovation et l'optimisation des processus.[9]

Domaine	Exemple et impact
Recommandation produits	Achat de PC → souris, sacoche → Augmente les ventes
Analyse financière	Transactions boursières → anticipation de tendance
Prévision de la demande	Ventes saisonnières → optimisation des stocks
Analyse comportementale	Visionnages sur Netflix → recommandations de séries

Tableau 9 Applications des données séquentielles

4. Modélisation des données séquentielles

Pour modéliser des données séquentielles, il est essentiel de prendre en compte la structure temporelle des données, l'ordre des événements et les relations entre les différentes séquences d'achats. Les données séquentielles peuvent être représentées à travers des séquences d'éléments (comme des produits achetés) et leur ordre dans le temps. Nous allons aborder la modélisation de ces données à travers différentes étapes en utilisant un exemple simple, et en expliquant les techniques et les structures utilisées pour cela.

Étapes de la modélisation des données séquentielles

- Représentation des données sous forme de séquences
- Extraction des caractéristiques des séquences
- Application de modèles adaptés à des données séquentielles
- Exploration des modèles de règles d'association séquentielle

4.1. Représentation des données sous forme de séquences

La première étape de la modélisation consiste à transformer les données brutes (transactions des clients) en séquences qui respectent l'ordre temporel des événements. Chaque séquence représente les achats effectués par un client à différents moments.

Exemple :

A partir du tableau 7 nous pouvons créer une séquence d'achats suivant :

- Client C1 : ` [Chaise, Table] -> [Lampe, Tapis] `
- Client C2 : ` [Canapé, Table] -> [Lampe, Table] `
- Client C3 : ` [Table, Chaise] -> [Canapé, Lampe, Tapis] `

- Client C4 : ` [Table, Lampe] -> [Canapé, Tapis, Chaise] `

Chaque séquence d'achat est ordonnée chronologiquement, ce qui permet de respecter le principe de l'ordre temporel.

4.2. Extraction des caractéristiques des séquences

Les caractéristiques des séquences incluent des informations telles que :

- Fréquence d'apparition des items : Combien de fois un produit a été acheté dans toutes les séquences (ex. : combien de fois les produits "table", "chaise", "canapé", etc. ont été achetés).
- Séquences communes : Quels sont les ensembles de produits qui apparaissent ensemble dans une séquence donnée. Par exemple, les clients qui achètent une table achètent aussi souvent une chaise dans les jours suivants.
- Intervalle de temps : La temporalité entre les événements. Par exemple, les achats peuvent être faits sur une période courte ou longue entre deux achats successifs.
- Ordre des achats : Certains clients peuvent acheter des produits dans un certain ordre, ce qui est important pour déduire des règles d'association séquentielle.

4.3 Modèles adaptés aux données séquentielles

Les modèles adaptés aux données séquentielles sont utilisés pour capturer les relations complexes entre les éléments au sein des séquences. Voici quelques modèles courants pour traiter les données séquentielles dans un contexte commercial :

a. Les règles d'association séquentielles

Les règles d'association séquentielles visent à découvrir des relations entre les produits achetés dans un ordre donné [10]. Par exemple, une règle pourrait être :

- Si un client achète une table, il est probable qu'il achète aussi une chaise dans un délai de 2 à 5 jours.

b. Modèles de séries temporelles

Un autre modèle adapté aux données séquentielles est l'utilisation de modèles de séries temporelles qui permettent d'analyser les comportements d'achat des clients en fonction du temps. Ces modèles incluent :

- Les modèles ARIMA (AutoRegressive Integrated Moving Average) : Iis peuvent être utilisés pour prédire les tendances des achats en fonction du temps, et détecter des motifs saisonniers.

- Les réseaux de neurones récurrents (RNN) : Ils sont particulièrement adaptés pour traiter des données séquentielles complexes, comme les séquences d'achats avec des comportements non linéaires.

c. Clustering temporel

Le clustering temporel peut être utilisé pour segmenter les clients en fonction de leurs séquences d'achats et de leurs comportements dans le temps. Par exemple, un algorithme de clustering comme K-Means ou DBSCAN peut être utilisé pour regrouper les clients ayant des comportements similaires.
Ces méthodes posent les bases de l'exploration séquentielle, qui sera détaillée dans la suite.

5. Règles d'association séquentielles

Dans la continuité de la modélisation, l'analyse porte désormais sur l'extraction de règles d'association temporelles.

5.1. Particularité des règles séquentielles

Les règles d'association séquentielles, contrairement aux règles classiques, tiennent compte de l'ordre chronologique dans lequel les items sont achetés.

- Forme générale : A → B, avec A apparaissant avant B dans la séquence.
- Exemple : *{table}* → *{lampe}*, signifie que l'achat d'une table précède fréquemment celui d'une lampe

5.2 Motifs séquentiels

Un motif séquentiel est une sous-séquence d'événements ou d'éléments qui apparaissent dans un ordre spécifique à travers plusieurs séquences de données. Le concept clé ici est l'ordre temporel, où l'objectif est de découvrir des modèles de comportement récurrents dans les séquences d'événements.

5.2.1 Exemple de motif séquentiel

Imaginons un commerce en ligne qui vend des meubles et des accessoires pour la maison. Voici un tableau des transactions réalisées par des clients sur une période :

Client	Séquence d'achats
C1	Chaise, Table, Lampe
C2	Table, Chaise, Lampe
C3	Chaise, Tapis, Table, Lampe
C4	Table, Lampe, Tapis

Tableau 10 Exemple de séquences clients

Dans cet exemple, les motifs suivants peuvent être extraits :

- Chaise → Table
  Ce motif signifie que, pour certains clients, après avoir acheté une chaise, ils achètent ensuite une table.

- Table → Lampe
  Ce motif signifie que, après l'achat d'une table, un client est souvent susceptible d'acheter une lampe.

Ces motifs séquentiels indiquent des relations temporelles entre les produits achetés, ce qui peut être utilisé pour recommander des produits ou optimiser les stratégies marketing.

5.3 Principaux types de motifs séquentiels

Les motifs séquentiels peuvent être classés en plusieurs catégories en fonction de leur complexité et de la nature de la séquence :

a. Motifs simples (premier ordre) : Il s'agit de produits ou d'événements qui apparaissent fréquemment dans les séquences, mais sans aucune relation temporelle définie.

- Exemple : Un produit comme Table peut être acheté fréquemment, indépendamment de l'ordre des achats.

b. Motifs séquentiels élémentaires : Ce sont des motifs d'une seule séquence, c'est-à-dire que les éléments apparaissent dans un ordre spécifique dans une séquence donnée, mais pas nécessairement dans d'autres.

- Exemple : Chaise → Table dans une transaction.

c. Motifs séquentiels longs : Il s'agit de motifs qui apparaissent dans plusieurs séquences et contiennent plusieurs éléments en ordre. Par exemple, une séquence complète d'achats dans un commerce en ligne.

- Exemple : Chaise → Table → Lampe peut être un motif récurrent où le client achète une chaise, puis une table, puis une lampe.

d. Motifs avec intervalle : Dans certaines situations, les éléments d'un motif peuvent apparaître avec un certain délai temporel entre eux. Par exemple, un client peut acheter une chaise, puis attendre plusieurs jours avant d'acheter une table.

- Exemple : Chaise → Table après une semaine.

5.4 Applications des motifs séquentiels

Les motifs séquentiels peuvent être utilisés pour diverses applications pratiques dans le contexte du commerce en ligne :

Application	Exemples pratiques
Recommandation produits	Achat de chaise → recommandation d'une lampe
Marketing ciblé	Promotion personnalisée basée sur le comportement passé
Optimisation des stocks	Anticiper les ventes futures (ex. lampe après table)
Prévision d'achats	Anticiper les achats à venir via motifs historiques
CRM (fidélisation)	Identifier les parcours clients récurrents pour adapter la communication

Tableau 11 Domaine d'application des motifs séquentiels

6 Algorithmes pour l'exploration des données séquentielles

L'exploration des données séquentielles implique la découverte de motifs séquentiels fréquents dans des séries d'événements ou de transactions. Pour cela, plusieurs algorithmes ont été développés. Nous allons explorer deux des algorithmes les plus populaires dans ce domaine : GSP (Generalized Sequential Pattern et PrefixSpan (Prefix-projected Sequential Pattern Mining). Pour chaque algorithme, nous présenterons un exemple simple pour illustrer leur fonctionnement.

6.1 GSP (Generalized Sequential Pattern)

GSP est un algorithme qui cherche des motifs séquentiels fréquents dans une base de données de séquences. Il se base sur une approche de génération et test, où on génère des candidats de motifs séquentiels, puis on les teste pour voir s'ils sont fréquents.

Exemple simple de GSP :

Supposons que nous avons un ensemble de transactions sous forme de séquences d'achats de produits par des clients dans un commerce en ligne.

Client	Séquence d'achats
C1	{A}, {B}, {C}
C2	{A}, {C}
C3	{A}, {B}, {C}
C4	{B}, {C}

Tableau 12 Ensemble de transactions séquentielles

Nous cherchons des motifs séquentiels fréquents dans ces transactions avec un seuil minimum de support de 2.

Étapes de GSP :

- Génération des candidats : Au départ, on génère des motifs de longueur 1 (des éléments individuels) : `{A}`, `{B}`, `{C}`.

- Filtrage des candidats fréquents : En comptant la fréquence de chaque motif, on trouve que les motifs fréquents sont `{A}`, `{B}`, et `{C}`, chacun ayant un support de 3, 3, et 4 respectivement.

- Génération de motifs plus longs : Ensuite, on génère des motifs de longueur 2 à partir des motifs fréquents de longueur 1, tels que ` {A, B} `, `{A, C}`, `{B, C}`.

- Filtrage des candidats fréquents : On compte à nouveau les occurrences de chaque motif de longueur 2 dans les séquences et on garde ceux dont le support est supérieur ou égal au seuil (2 dans ce cas).

- Le motif ` {A, B} ` apparaît dans 2 séquences (C1 et C3), donc il est fréquent.
- Le motif ` {A, C} ` apparaît dans 2 séquences (C1 et C3), donc il est aussi fréquent.
- Le motif ` {B, C}` apparaît dans 3 séquences (C1, C3, C4), donc il est également fréquent.

- Arrêt de l'algorithme : Aucune génération de motifs de longueur supérieure n'est possible, car aucun autre motif fréquent n'existe.

Résultats :
Motifs séquentiels fréquents :
- `{A}`, `{B}`, `{C}`
- ` {A, B} `, `{A, C}`, `{B, C}`

6.2 PrefixSpan (Prefix-projected Sequential Pattern Mining)

PrefixSpan est un algorithme qui, contrairement à GSP et SPADE, projette les séquences sur des préfixes afin de réduire l'espace de recherche et améliorer l'efficacité de l'extraction des motifs séquentiels. Plutôt que de générer des candidats de motifs, PrefixSpan décompose le problème en sous-problèmes plus petits en projetant chaque séquence par son préfixe [13].

Exemple simple de PrefixSpan :

Prenons à nouveau les données du tableau 12

Le seuil minimum de support est 2.

Étapes de PrefixSpan :

- Projections des séquences : L'algorithme commence par analyser les séquences en les projetant sur leurs préfixes. Par exemple, si l'on prend le préfixe `{A}`, l'algorithme extrait toutes les séquences contenant `{A}`, puis les projette sur le suffixe (la partie restante après le préfixe).

- Préfixe `{A}` :
- Projections : `{B}, {C}` (pour C1 et C3)

- Extraction des motifs fréquents à partir des préfixes projetés : L'algorithme extrait ensuite les motifs fréquents dans les projections. Par exemple, dans les projections de `{A}`, on peut extraire le motif `{B}`, `{C}`, et `{B, C}`.

- Répétition pour des préfixes plus longs : Ce processus est répété pour chaque préfixe, générant des motifs séquentiels plus longs jusqu'à ce que les projections ne contiennent plus de motifs fréquents.

- Filtrage des motifs fréquents : À chaque étape, les motifs sont filtrés en fonction du support.

Résultats :
Motifs séquentiels fréquents :
`{A}`, `{B}`, `{C}`
` {A, B}`, `{A, C}`, `{B, C}`

PrefixSpan est particulièrement efficace lorsqu'il s'agit de traiter de grandes bases de données de séquences, car il ne génère pas de candidats et se concentre uniquement sur les préfixes des séquences.

Comparaison des algorithmes

Algorithme	Méthode principale	Avantages	Inconvénients
GSP	Génération et test de candidats	Facile à comprendre et à implémenter	Peut-être inefficace pour de grandes bases de données
PrefixSpan	Projections sur les préfixes	Très efficace, ne génère pas de candidats	Peut être difficile à comprendre pour les débutants

Tableau 13 Comparaison des algorithmes GSP et PrefixSpan

Chacun des deux algorithmes, GSP et PrefixSpan, présente des avantages en fonction du contexte et de la taille des données.

Ce chapitre a présenté un panorama complet allant de la définition et la structuration des données séquentielles à leur exploration par les règles d'association séquentielles. Les caractéristiques

(ordre, temporalité, dépendances, variabilité, récurrence, longévité) posent les fondations de l'analyse.

Les motifs séquentiels et les règles temporelles apportent une vision fine des comportements clients, tandis que les algorithmes GSP et PrefixSpan offrent des outils concrets pour extraire ces régularités.[16]

Ensemble, ces éléments constituent le cœur de l'approche séquentielle adoptée dans ce mémoire, ouvrant la voie à des applications pratiques en recommandation, marketing, prévision et fidélisation.

Passons à présent à la méthodologie utilisée pour notre travail de mémoire.

Chapitre IV : Méthodologie

Chapitre IV : Méthodologie

1. Objectif de la méthodologie

Ce chapitre expose la démarche méthodologique adoptée pour analyser les comportements d'achat à partir de données séquentielles.
L'approche repose sur un workflow structuré en quatre grandes étapes :

- Préparation des données : nettoyage, transformation et modélisation des enregistrements bruts.
- Segmentation et clustering : regroupement des clients selon la similarité de leurs comportements.
- Analyse prédictive : anticipation des comportements futurs à partir de l'historique.
- Expériementation : mise en évidence de règles temporelles récurrentes.

On peut le schématiser comme suit :

Données brutes (transactions)

↓

Préparation des données

↓

Segmentation et clustering

↓

Analyse prédictive

↓

Expérimentation

2. Préparation des données

La première étape vise à rendre les données exploitables pour les traitements ultérieurs. Sans préparation, les analyses risquent d'être biaisées ou inexploitables.

2.1 Description du jeu de données

A titre d'illustration, nous utiliserons ce jeu de données comprenant notamment les champs suivants :

Attribut	Description
T_ID	Identifiant unique de la transaction
P_ID	Code unique du produit

Attribut	Description
P_Label	Libellé/nom du produit
Q	Nombre d'unités achetées
Date	Date et heure de la transaction
PU	Montant unitaire de l'article
C_ID	Identifiant client

Tableau 14 Structure des données brutes

2.2 Nettoyage des données

Avant toute analyse, un nettoyage des données a été effectué pour garantir la qualité des résultats. Les étapes suivantes ont été réalisées :

- Suppression des transactions sans identifiant client (C_ID manquant)

- Suppression des produits avec quantité ou prix négatifs ou nuls

- Uniformisation des types de données (dates, chaînes de caractères)

Formellement, on note :

Données nettoyées = $\{t \in \mathcal{T} \mid C\_ID(t) \neq \emptyset, Q(t) > 0, PU(t) > 0 \}$

2.3 Transformation des variables

Les données brutes doivent être enrichies pour faciliter l'analyse.

- Conversion des dates en champs séparés (Date seule, Heure seule).
- Création d'une variable Montant total (TotSale) :

  TotSale =Q× PU

Cela nous permet d'améliore la richesse descriptive des données et permet de capturer des indicateurs économiques pertinents.

Un filtre a également été appliqué pour ne garder que les transactions à valeur positive.

2.4 Modélisation séquentielle du client

L'objectif ici est de modéliser les comportements d'achat des clients comme des séquences de paniers d'achat, où chaque panier représente un regroupement cohérent d'achats dans une période temporelle donnée.

Soit un client ci, sa séquence d'achats est modélisée par :

S(ci)= {P1, P2,…,Pk}

Où chaque panier Pj correspond à un ensemble d'articles achetés dans une session ou un mois donné.

Formellement :

Pj={(p1, q1),(p2,q2),…,(pm, qm) }

Avec :

- pj : l'identifiant de l'article
- qj : la quantité achetée

  Cette représentation séquentielle permet de conserver l'ordre chronologique des achats et constitue une base adéquate pour l'expérimentation

Chaque panier est identifié par le triplet (C_ID, Date, T_ID).

2.5 Calculs de caractéristiques et analyse RFM

Dans le but de segmenter les comportements des clients et de dresser leurs profils de façon détaillés, plusieurs opérations ont été réalisées. Cela part du calcul des caractéristiques à l'analyse RFM.
Ces métriques serviront de base pour des analyses plus poussées dans le clustering

2.5.1 Extraction des caractéristiques (I, Iu, Imax, E, Savg)

Il s'agit de :

- I (quantité totale achetée) : identifie les clients les plus actifs en termes de volume d'achat
- Imax (quantité maximale achetée en une session) : indique les clients ayant des pics d'achat importants
- Iu (produits uniques) : montre la diversité des achats d'un client.
- E (entropie de Shannon) : mesure la régularité ou la variabilité des achats
- Savg (prix moyen par session) : identifie les clients qui dépensent plus ou moins en moyenne, un indicateur de leur valeur financière

2.5.2 Analyse RFM (Recency, Frequency, Monetary)

Il s'agit ici:

- Récence (R) : dernière date d'achat, qui permet d'identifier les clients actifs ou inactifs.
- Fréquence (F) : nombre de transactions, qui mesure la fidélité des clients.
- Valeur monétaire (M) : dépenses totales du client, qui identifie les clients les plus lucratifs .

En plus de cela l'attribution de quartiles (R_quartile, F_quartile, M_quartile) et du score global RFM permettent de classer les clients en segments comme suit:

- Meilleurs clients (111).
- Clients fidèles (F_quartile = 1).
- Gros dépensiers (M_quartile = 1).
- Clients perdus ou inactifs (411, 444).

A titre d'illustration, prenons le Tableau 15 Exemple de transactions séquentielles. Le calcul de caractéristiques et l'analyse RFM nous permet d'obtenir les résultats suivants :

Client	R	F	M	R_quartile	F_quartile	M_quartile
C1	5	2	4	1	1	1
C2	3	2	4	2	2	2
C3	2	2	5	3	3	3
C4	0	2	5	4	4	4

3. Clustering

L'étape de clustering a pour but de segmenter les clients selon la similarité de leurs comportements d'achat.

3.1 Normalisation des données

La normalisation est essentielle pour garantir que toutes les variables contribuent de manière égal e à l'analyse. L'objectif sera donc de standardiser les données pour que chaque variable ait une moyenne de 0 et un écart-type = 1

3.2 Reduction de dimension

Pour faciliter la visualisation et éviter le sur-apprentissage nous utiliserons l'analyse de composante principale (ACP). Elle nous permet aussi de réduire la complexité des données.

3.3. Détermination du nombre optimal de clusters (k)

Le choix du nombre de clusters k est crucial pour assurer une segmentation pertinente et robuste. Deux méthodes dont choisi. Il s'agit de elow et sihhouette. Ces deux méthodes permettent d'équilibrer la cohérence interne et la séparation entre les groupes.

- Méthode du coude (elbow method) : analyse de l'inertie intra-cluster.

- Score de silhouette : mesure de la cohérence interne de chaque cluster.

3.4. Application des algorithmes de clustering

Dans le cadre du clustering, plusieurs algorithmes ont été choisi. Le choix de ces méthodes repose sur leur complémentarité :

- K-Means : c'est une méthode rapide et efficace pour partitionner des données en groupes homogènes. Elle suppose des clusters sphériques et bien séparés, ce qui est compatible avec des données normalisées et réduites. Le jeu de données est segmenté en k groupes

Formule de distance pour K-Means :

$$Distance(x, \mu_j) = \sqrt{\sum_{i=1}^{n} (x_i - \mu_{ji})^2}$$

où :

x_i est la i-ème composante du vecteur de caractéristiques du client,

μ_{ji} est la i-ème composante du centroïde du cluster j,

n est le nombre de dimensions (caractéristiques extraites, telles que le montant total dépensé, fréquence d'achat, diversité de produits, etc.).

- Fuzzy C-Means : c'est une version floue du K-Means, chaque individu est affecté à chaque cluster avec un degré d'appartenance. Certains clients peuvent avoir un comportement "intermédiaire". Le clustering flou permet donc de capter cette nuance et d'offrir une segmentation plus souple.

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) : utile pour détecter des groupes de formes arbitraires, notamment en présence de bruit. Contrairement à K-Means, DBSCAN peut identifier des clusters de forme arbitraire et est robuste aux valeurs aberrantes. Il est utile quand on soupçonne l'existence de groupes de densité variable.

- BIRCH : c'est un algorithme hiérarchique optimisé pour les grandes bases de données. Il construit une structure compacte (arbre CF).

3.5 Évaluation des clusters

Afin de garantir la validité des clusters obtenus, deux méthodes d'évaluation ont été mobilisées. Celles-ci permettent de mesurer la compacité, la séparation et la cohérence des regroupements produits par les algorithmes. Il s'agit du Silhouette Score et de la méthode du coude

3.5.1 Silhouette Score

Le silhouette score permet de mesurer, pour chaque observation, la qualité de son appartenance à un cluster. Il combine deux critères :

- la cohésion intra-cluster : proximité d'un point avec les autres éléments de son propre cluster.
- la séparation inter-cluster : éloignement d'un point par rapport aux points des clusters voisins.

$$s(i) = \frac{b(i)-a(i)}{max(a(i),b(i))}$$

a(i): la distance moyenne entre l'observation iii et les autres points de son propre cluster (cohésion intra-cluster),

b(i) : la distance moyenne entre l'observation iii et les points du cluster le plus proche auquel elle n'appartient pas (séparation inter-cluster).

Les valeurs de s(i) varient entre :

- +1 : l'observation est bien regroupée ;
- 0 : elle est proche de la frontière entre deux clusters ;
- −1 : elle est probablement mal assignée.

Le score moyen sur l'ensemble du jeu de données indique la qualité globale du clustering.

3.5.2. Méthode du coude (Elbow Method)

La méthode du coude repose sur l'analyse de l'inertie intra-classe, c'est-à-dire la somme des distances carrées entre les points et leur centroïde au sein d'un même cluster. Pour différentes valeurs de KKK (nombre de clusters), cette inertie diminue naturellement. Cependant, à partir d'un certain seuil, la diminution devient marginale : ce point d'inflexion est appelé le coude et correspond souvent au nombre optimal de clusters.

$$\text{Inertie} = \sum_{j=1}^{k} \sum_{x \in Cj} \|x - \mu_j\|^2$$

où Cj est l'ensemble des points du cluster j, et μj son centroïde.

Une inertie faible indique que les points sont fortement regroupés autour de leur centre, ce qui est souhaitable. Cependant, l'inertie tend à diminuer avec un nombre croissant de clusters, d'où la nécessité de coupler cette mesure avec la silhouette score pour éviter le sur-apprentissage.

Justification de l'approche

L'utilisation conjointe de plusieurs algorithmes permet de comparer les structures détectées et de sélectionner l'approche la plus adaptée aux données, valider la robustesse des clusters à travers la similitude ou non des résultats des différentes algorithmes traité

4. Analyse prédictive

L'analyse prédictive représente une étape cruciale dans l'exploitation des données, visant à anticiper les comportements futurs à partir de données historiques. Dans le cadre de cette étude, nous nous focalisons sur la prédiction des segments clients (*high*, *medium*, *low*) identifiés lors de la phase de clustering, en utilisant leurs comportements passés comme variables explicatives. Cette démarche s'inscrit dans une logique d'optimisation marketing, où la capacité à anticiper les profils clients offre un avantage compétitif significatif.

Plusieurs modèles de machine learning avec validation croisée, recherche d'hyperparamètres, et évaluation sur jeu de test ont été comparé. Nous détaillerons le procédé dans ce qui suit.

4.1. Définition des fonctions d'analyse

Différentes fonctions graphiques sont définies pour diagnostiquer le surapprentissage (overfitting) et le sous-apprentissage (underfitting) mais aussi comparer visuellement les modèles pour choisir le plus robuste:

- plot_learning_curve: pour visualiser les performances du modele d'apprentissage
- plot_validation_curve: pour évaluer l'impact d'un hyperparamètre sur les scores
- print_pretty_cv_results() : pour afficher les résultats de la validation croisée de manière lisible.

4.2 Modèles testés

Pour garantir une modélisation robuste et adaptée à notre problématique de segmentation client, nous avons opté pour une approche comparative systématique de plusieurs algorithmes de classification.

Notre sélection couvre les principales familles d'algorithmes de machine learning, offrant ainsi une vision complète des approches possibles :

Famille	Modèle	Points forts
Modèles linéaires	SVM	Gestion des frontières complexes
Neuronaux	MLP	Apprentissage non linéaire
Ensemblistes	Random Forest	Robustesse, interprétabilité
Arbres décision	Decision Tree	Transparence, rapidité

Cette approche comparative systématique nous permet de démontrer la rigueur scientifique de notre démarche, d'éviter le biais méthodologique qu'entraînerait l'utilisation d'un seul algorithme potentiellement inadapté à la structure des données, de justifier objectivement le choix du modèle optimal, de mieux comprendre la structure sous-jacente de nos données et de garantir la reproductibilité des résultats obtenus.

4.3 Méthodologie d'évaluation et d'optimisation

4.3.1 Protocole d'évaluation des Modèles

Pour garantir la robustesse de nos résultats, nous avons mis en place une méthodologie d'évaluation rigoureuse reposant sur :

- la validation croisée (5-folds)

Afin de renforcer la robustesse de nos résultats, nous avons recours à une validation croisée à 5 plis (*5-fold cross-validation*). Cette technique consiste à diviser aléatoirement l'ensemble des données en cinq sous-ensembles de taille égale. À chaque itération, quatre sous-ensembles sont utilisés pour l'entraînement du modèle, tandis que le cinquième sert à l'évaluation. Ce processus est répété cinq fois, en faisant varier à chaque fois le sous-ensemble utilisé pour le test. La performance finale est alors estimée par la moyenne des résultats obtenus sur les cinq plis.

Cette méthode permet de limiter les risques de surapprentissage (*overfitting*), en s'assurant que le modèle ne s'adapte pas trop étroitement aux données d'entraînement, et évalue sa capacité à généraliser à des données non vues.

- les métriques d'évaluation

Tout d'abord, l'accuracy mesure le taux global de classification correcte sur l'ensemble des données testées. Elle fournit une vision d'ensemble de la performance, mais peut être trompeuse en cas de classes déséquilibrées. Pour une analyse plus fine, la matrice de confusion est utilisée : elle détaille le nombre de bonnes classifications et d'erreurs par classe, permettant notamment d'identifier les confusions entre certains segments spécifiques. Enfin, le rapport de classification regroupe trois indicateurs essentiels : la précision, qui évalue la capacité du modèle à éviter les fausses classifications dans une classe donnée ; le rappel, qui mesure sa capacité à identifier tous les éléments pertinents d'une classe ; et le F1-score, qui constitue une moyenne harmonique entre précision et rappel, offrant ainsi un compromis équilibré entre ces deux aspects.

- Optimisation des hyperparamètres

Il a été réalisé selon une stratégie hybride combinant deux approches complémentaires, en fonction de la complexité des algorithmes. Pour les modèles dits « simples » tels que le SVM ou le k-NN, nous avons eu recours à GridSearchCV, une méthode de recherche exhaustive qui teste systématiquement l'ensemble des combinaisons prédéfinies d'hyperparamètres. Cette approche garantit l'identification de la meilleure configuration dans l'espace exploré, au prix toutefois d'un coût computationnel élevé.

En revanche, pour les modèles plus complexes comme Random Forest ou les réseaux de neurones (MLP), nous avons privilégié RandomizedSearchCV. Cette méthode repose sur un échantillonnage aléatoire d'un nombre fixé de combinaisons, permettant une exploration plus rapide et une réduction significative du temps de calcul, tout en conservant une bonne capacité d'optimisation.

Cette approche différenciée présente plusieurs avantages : elle permet de trouver des configurations optimales sans avoir à explorer de manière exhaustive tout l'espace des hyperparamètres, elle maintient un équilibre entre la qualité des résultats et les ressources mobilisées, et elle adapte l'effort d'optimisation à la complexité intrinsèque de chaque modèle. Dans l'ensemble, cette méthodologie garantit à la fois la robustesse des modèles construits, l'efficience du processus d'optimisation, et la disponibilité d'indicateurs fiables pour orienter le choix final des hyperparamètres.

5.1 Algorithmes considérés

Deux algorithmes bien établis ont été étudiés pour répondre à cette problématique :

GSP (Generalized Sequential Pattern) et PrefixSpan (Prefix-Projected Sequential Pattern Mining). Nous les évaluerons dans notre expérimentation afin de déterminer celui offrant les meilleurs résultats.

Chapitre V : Implémentation

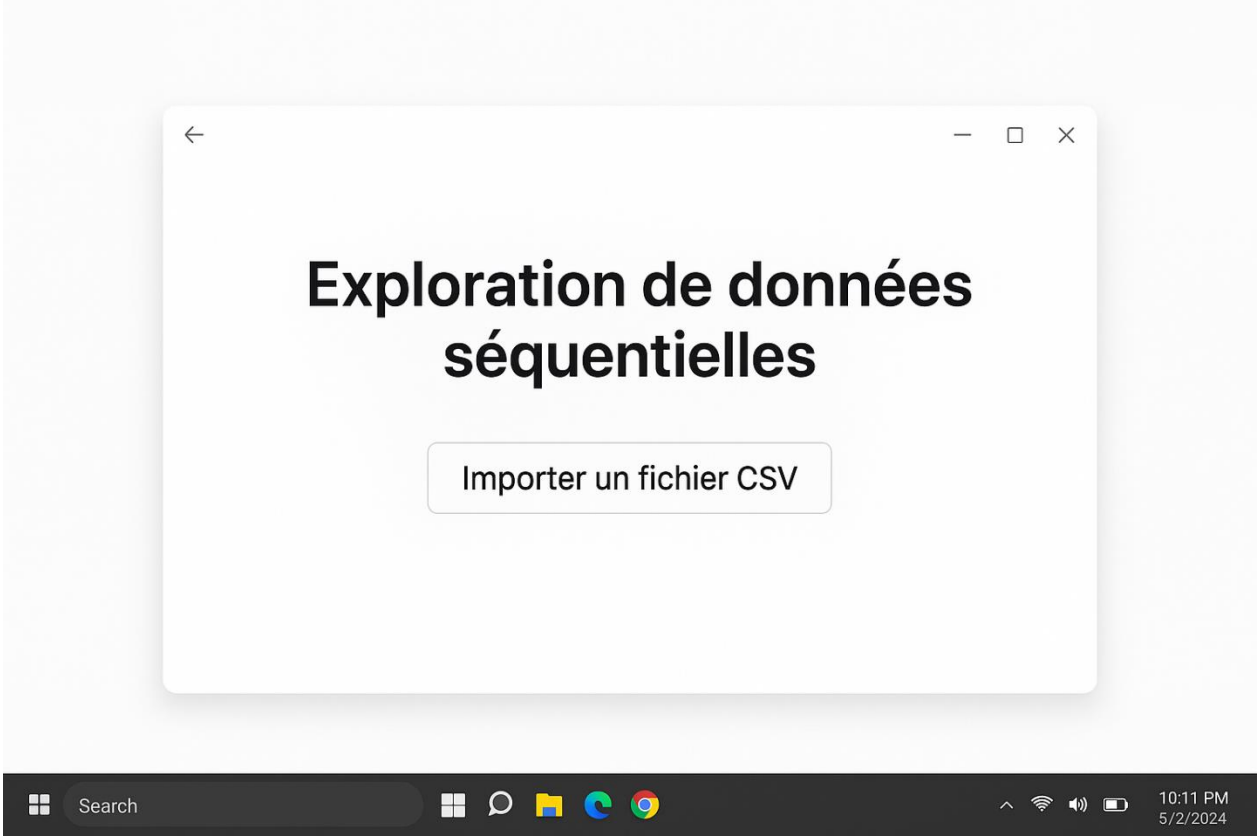
Chapitre V : Implémentation

1 Introduction

Ce chapitre présente la mise en œuvre concrète de la méthodologie exposée précédemment. L'objectif est de traduire les concepts théoriques en solutions algorithmiques à l'aide de Python, afin de valider leur faisabilité sur des données réelles issues d'un environnement transactionnel. Chaque étape du pipeline est détaillée : préparation des données, extraction de caractéristiques, segmentation (RFM et clustering), analyse prédictive, et expérimentation. Les performances sont évaluées à chaque étape.

2 Interface utilisateur

Présentons d'abord l'interface utilisateur



3. Préparation des données

La première étape de l'implémentation consiste à préparer les données de manière à garantir leur qualité et leur compatibilité avec les traitements analytiques ultérieurs. Pour cela, nous avons utilisé les bibliothèques Python standards telles que pandas, numpy, et datetime. Le jeu de données principal utilisé dans cette étude est :

- dbase.csv : base de données contenant les profils clients et leurs historiques de transactions.

3.1 Chargement et conversion des types de données

Nous commençons par l'importation des bibliothèques nécessaires ainsi que le chargement du fichier CSV contenant les transactions clients.

```python
df = pd.read_csv(config.CSV_PATH, sep="\t", index_col=0)

print(f"📥 Données chargées depuis {config.CSV_PATH}")
print(df.head())
```

Python

Un renommage de nos colonnes a été effectué afin de faciliter notre code

```python
# Renommage des colonnes
df.rename(columns={
    'InvoiceNo': 'BasketID',
    'StockCode': 'ProdID',
    'Description': 'ProdDesc',
    'Quantity': 'Qta',
    'InvoiceDate': 'BasketDate',
    'UnitPrice': 'Sale',
    'CustomerID': 'CustomerID',
    'Country': 'Country',
    'TotSale': 'TotSale'
}, inplace=True)

# Vérifier les colonnes après renommage
print("✅ Colonnes après renommage :", df.columns.tolist())
print(df.head())
```

✓ 0.0s

Toujours dans le but de faciliter l'analyse et la manipulation des données, nous avons affiché puis converti certains types de variables :

- BasketDate a été transformé en format datetime.
- BasketID et CustomerID ont été convertis en type object pour une meilleure gestion des identifiants.

```python
df = df.astype({'BasketDate': 'datetime64[ns]',
                'BasketID': 'object',
                'CustomerID': 'object'})
```

✓ 0.0s Python

3.2 Extraction des caractéristiques clients

Plusieurs indicateurs sont extraits afin de caractériser le comportement d'achat de chaque client. Ces indicateurs sont globaux et issus de regroupements par l'identifiants du client CostumerID.

3.2.1 Quantité totale achetée (I)

Nous groupons les données par CustomerID et calculons la somme des quantités (colonne "Qta") achetées par chaque client.

```
# Calculate various features
I = df.groupby('CustomerID', as_index=False)['Qta'].sum()
I.columns = ['CustomerID', 'I']
I
```

✓ 0.1s Python

3.2.2 Nombre d'articles uniques achetés (Iu)

En exploitant la colonne ProdID, nous calculons le nombre d'articles uniques achetés par client.

```
Iu = df.groupby('CustomerID')['ProdID'].nunique().reset_index()
Iu.columns = ['CustomerID', 'Iu']
Iu
```

✓ 0.4s

3.2.3 Maximum d'articles achetés lors d'une session (Imax)

Pour calculer la quantité maximale achetée lors d'un panier (basket), nous groupons d'abord par CustomerID et BasketID puis prenons la somme de "Qta" sur chaque panier. Ensuite, nous regroupons par CustomerID pour extraire le maximum de ces sommes.

```
#The maximum number of items purchased by a customer during a shopping session:
Imax = df.groupby(['CustomerID', 'BasketID'], as_index=False)['Qta'].sum()[['CustomerID', 'Qta']].grou
Imax.columns = ['CustomerID', 'Imax']
Imax
```

✓ 0.1s Python

3.2.4 Entropie des ventes (E)

L'entropie de Shannon, appliquée aux ventes (TotSale). On regroupe par client et on applique une fonction lambda qui :

- Arrondit les valeurs,
- Calcule les effectifs des ventes,
- Puis calcule l'entropie.

```
E = df.groupby('CustomerID')['TotSale'].apply(lambda x : entropy(x.round().value_counts(), base=2)).reset_index()
E.columns = ['CustomerID', 'E']
E
```

✓ 2.5s Pytho

3.2.5 Dépense moyenne par session (Savg)

Pour déterminer la dépense moyenne par session, les données sont regroupées par CustomerID et BasketID sur la colonne TotSale, puis la moyenne de ces sommes est calculée pour chaque client.

```
#The average price spent by a customer during a shopping session:
Savg = df.groupby(['CustomerID', 'BasketID'], as_index=False)['TotSale'].sum()[['CustomerID', 'TotSale']].groupby('CustomerID', as_index=False)['TotSale'].mean()
Savg.columns = ['CustomerID', 'Savg']
Savg
```

✓ 0.2s

	CustomerID	Savg
0	12346.0	0.000000
1	12347.0	615.714286
2	12348.0	449.310000
3	12349.0	1757.550000
4	12350.0	334.400000
...	...	...
4367	18280.0	180.600000
4368	18281.0	80.820000
4369	18282.0	58.866667
4370	18283.0	130.930000
4371	18287.0	612.426667

3.2.6 Fusion des caractéristiques et création du profil client

Les différents indicateurs calculés sont ensuite fusionnés pour constituer un profil global pour chaque client.

```
# merge features
features = I.merge(Iu, on='CustomerID').merge(Imax, on='CustomerID').merge(E, on='CustomerID').merge(Savg, on='CustomerID').set_index('CustomerID')

features
```

✓ 0.0s

CustomerID	I	Iu	Imax	E	Savg
12346.0	0	1	74215	1.000000	0.000000
12347.0	2458	103	676	4.310157	615.714286
12348.0	2341	22	1254	3.498731	449.310000
12349.0	631	73	631	3.954218	1757.550000
12350.0	197	17	197	2.675698	334.400000
...	...	...	...	...	...
18280.0	45	10	45	1.846439	180.600000
18281.0	54	7	54	1.842371	80.820000
18282.0	98	12	75	3.026987	58.866667
18283.0	1397	263	251	2.594064	130.930000
18287.0	1586	59	990	4.198218	612.426667

4372 rows × 5 columns

Ce DataFrame, nommé features, contient les variables I, Iu, Imax, E et Savg. Il servira de base pour la suite des analyses (RFM, normalisation, clustering, etc.).

3.3 Analyse RFM

Nous avons également appliqué l'analyse RFM (Récence, Fréquence, Montant) pour distinguer les comportements clients. La construction de la table RFM se fait via un groupby et une agrégation spécifique :

```
#RFM Analysis  (Recency, Frequency, Monetary)
rfm = df.groupby('CustomerID').agg({'BasketDate': lambda date: (df['BasketDate'].max() - date.max()).days,
                                    'BasketID': lambda basket: basket.nunique(),
                                    'TotSale': lambda sale: sale.sum()})
rfm.columns = ['R', 'F', 'M']
rfm['BasketDate'] = df.groupby('CustomerID')['BasketDate'].max()
rfm
```

✓ 7.5s

CustomerID	R	F	M	BasketDate
12346.0	325	2	0.00	2011-01-18 10:17:00
12347.0	1	7	4310.00	2011-12-07 15:52:00
12348.0	74	4	1797.24	2011-09-25 13:13:00
12349.0	18	1	1757.55	2011-11-21 09:51:00
12350.0	309	1	334.40	2011-02-02 16:01:00
...	...	...	...	...
18280.0	277	1	180.60	2011-03-07 09:52:00
18281.0	180	1	80.82	2011-06-12 10:53:00
18282.0	7	3	176.60	2011-12-02 11:43:00
18283.0	3	16	2094.88	2011-12-06 12:02:00
18287.0	42	3	1837.28	2011-10-28 09:29:00

Cette table est fondamentale pour segmenter les clients en fonction de leur historique d'achat. Pour affiner la segmentation, nous calculons les quartiles des variables R, F et M :

```
quantiles = rfm.quantile(q=[0.25, 0.5, 0.75])
quantiles
```

✓ 0.0s

Ces valeurs de quartiles permettront d'attribuer des scores selon le rang du client (le mieux positionné recevant le score 1).

Deux fonctions sont définies pour attribuer des scores :

```python
# Arguments (x=value, p=recency, d=quartiles)
def RScore(x, p, d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.50]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4

# Arguments (x=value, p=frequency, monetary, d=quartiles)
def FMScore(x, p, d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.50]:
        return 3
    elif x <= d[p][0.75]:
        return 2
    else:
        return 1
```

✓ 0.0s

On applique ensuite ces fonctions aux colonnes R, F, M :

```python
#  RFM segmentation
rfm['R_quartile'] = rfm['R'].apply(RScore, args=('R', quantiles))
rfm['F_quartile'] = rfm['F'].apply(FMScore, args=('F', quantiles))
rfm['M_quartile'] = rfm['M'].apply(FMScore, args=('M', quantiles))
rfm
```

✓ 0.2s Python

Après avoir attribué les scores, nous avons créé une segmentation basée sur la combinaison des trois valeurs (RFM). Cela permet de distinguer plusieurs profils et le nombre de clients qui y est:

```python
print('Meilleurs clients:', len(rfm[rfm['RFM'] == '111']))
print('Clients fideles :', len(rfm[rfm['F_quartile'] == 1]))
print('Gros Acheteurs:', len(rfm[rfm['M_quartile'] == 1]))
print('Clients en risque:', len(rfm[rfm['RFM'] == '311']))
print('Clients perdus:', len(rfm[rfm['RFM'] == '411']))
print('Clients perdus à faible valeur:', len(rfm[rfm['RFM'] == '444']))
```

✓ 0.0s Python

Voici la répartition des segments :

- Meilleurs Clients : 498
- Clients Fidèles : 1087
- Gros Acheteurs : 1093
- Clients en risque : 91
- Clients perdus : 14
- Clients perdus à faible valeur : 403

3.4 Construction du profil client final et visualisation

Le profil client final est constitué en fusionnant le DataFrame features avec la table RFM :

```
customer_profile = features.merge(rfm[['R', 'F', 'M', 'BasketDate']], on='CustomerID')
customer_profile
```

✓ 0.0s

CustomerID	I	Iu	Imax	E	Savg	R	F	M
12346.0	0	1	74215	1.000000	0.000000	325	2	0.00
12347.0	2458	103	676	4.310157	615.714286	1	7	4310.00
12348.0	2341	22	1254	3.498731	449.310000	74	4	1797.24
12349.0	631	73	631	3.954218	1757.550000	18	1	1757.55
12350.0	197	17	197	2.675698	334.400000	309	1	334.40
...	...	...	...	...	...	...	...	...
18280.0	45	10	45	1.846439	180.600000	277	1	180.60
18281.0	54	7	54	1.842371	80.820000	180	1	80.82
18282.0	98	12	75	3.026987	58.866667	7	3	176.60
18283.0	1397	263	251	2.594064	130.930000	3	16	2094.88
18287.0	1586	59	990	4.198218	612.426667	42	3	1837.28

Sauvegarde des Données Prétraitées

Après avoir appliqué l'ensemble des opérations de transformation et de normalisation nous avons obtenu le DataFrame log_customer_profile. Ce DataFrame contient les caractéristiques normalisées qui constitueront la base optimale pour les analyses et modèles suivants.

Pour assurer la reproductibilité de nos analyses et faciliter l'intégration ultérieure de ces données dans le pipeline de traitement, nous avons exporté log_customer_profile dans un fichier CSV. Le code correspondant est le suivant :

```
# Sauvegarde
log_customer_profile.to_csv(config.PROCESSED_CSV_PATH, sep='\t', index=False)
print(f"💾 Données sauvegardées dans {config.PROCESSED_CSV_PATH}")
print(log_customer_profile.to_csv)
```

✓ 0.1s

```
                                    CustomerID         I        Iu      Imax         E      Savg    R  \
CustomerID
12346.0       12346.0      -inf  0.000000  4.870492  1.000000      -inf  325
12347.0       12347.0  3.390582  2.012837  2.829947  4.310157  2.789379    1
12348.0       12348.0  3.369401  1.342423  3.098298  3.498731  2.652546   74
12349.0       12349.0  2.800029  1.863323  2.800029  3.954218  3.244908   18
12350.0       12350.0  2.294466  1.230449  2.294466  2.675698  2.524266  309
...               ...       ...       ...       ...       ...       ...  ...
18280.0       18280.0  1.653213  1.000000  1.653213  1.846439  2.256718  277
18281.0       18281.0  1.732394  0.845098  1.732394  1.842371  1.907519  180
18282.0       18282.0  1.991226  1.079181  1.875061  3.026987  1.769869    7
18283.0       18283.0  3.145196  2.419956  2.399674  2.594064  2.117039    3
18287.0       18287.0  3.200303  1.770852  2.995635  4.198218  2.787054   42

                  F         M          BasketDate
CustomerID
12346.0     0.301030      -inf 2011-01-18 10:17:00
12347.0     0.845098  3.634477 2011-12-07 15:52:00
12348.0     0.602060  3.254606 2011-09-25 13:13:00
12349.0     0.000000  3.244908 2011-11-21 09:51:00
12350.0     0.000000  2.524266 2011-02-02 16:01:00
...              ...       ...                 ...
...
18283.0     1.204120  3.321159 2011-12-06 12:02:00
18287.0     0.477121  3.264175 2011-10-28 09:29:00
```

4. Clustering

L'objectif est de partitionner de maniére approfondie les données. Pour ce faire, nous commençons par charger le profil client prétraité, nous procédons à une standardisation et une réduction dimensionnelle, puis nous appliquons divers algorithmes de clustering. Chacun de ces algorithmes (Kmeans, Fuzzy C-means, DBSCAN, Birch) est analysé puis visualisé.

4.1. Chargement des données et préparation

La première étape consiste à importer le fichier contenant les profils clients prétraités.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, MiniBatchKMeans
from sklearn.metrics import silhouette_score, silhouette_samples
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import cm
from sklearn.metrics.pairwise import euclidean_distances
import config
```

```
# Charger le profil client
customer_profile = pd.read_csv(config.PROCESSED_CSV_PATH, sep='\t', index_col=0)
print(f"📥 Données chargées depuis {config.PROCESSED_CSV_PATH}")
customer_profile
```

	I	Iu	Imax	E	Savg	R	F	M
CustomerID								
12346.0	-inf	0.000000	4.870492	1.000000	-inf	325	0.301030	-inf
12347.0	3.390582	2.012837	2.829947	4.310157	2.789379	1	0.845098	3.634477
12348.0	3.369401	1.342423	3.098298	3.498731	2.652546	74	0.602060	3.254606
12349.0	2.800029	1.863323	2.800029	3.954218	3.244908	18	0.000000	3.244908
12350.0	2.294466	1.230449	2.294466	2.675698	2.524266	309	0.000000	2.524266
...	...	...	...	...	...	...	...	...
18280.0	1.653213	1.000000	1.653213	1.846439	2.256718	277	0.000000	2.256718
18281.0	1.732394	0.845098	1.732394	1.842371	1.907519	180	0.000000	1.907519
18282.0	1.991226	1.079181	1.875061	3.026987	1.769869	7	0.477121	2.246991
18283.0	3.145196	2.419956	2.399674	2.594064	2.117039	3	1.204120	3.321159
18287.0	3.200303	1.770852	2.995635	4.198218	2.787054	42	0.477121	3.264175

Nous constatons la présence de valeur inf, nous procédons donc à un nettoyage préventif pour s'assurer que toutes les valeurs sont valides. Nous allons d'abord les remplacerons par Nan. On va ensuite supprimer les valeurs manquantes créées

```
# Remplacez les valeurs infinies par NaN
customer_profile_pca.replace([np.inf, -np.inf], np.nan, inplace=True)
```

```
# Supprimer les lignes contenant des NaN
customer_profile_pca.dropna(inplace=True)
```

4.2. Standardisation et Réduction de Dimension (PCA)

4.2.1. Standardisation

Cette étape garantit que chaque variable contribue de manière égale à la distance que nous utiliserons dans nos algorithmes de clustering

```
scaler = StandardScaler()
std_customer_profile = scaler.fit_transform(customer_profile_pca)
```

4.2.1 PCA et visualisation de la variance expliquée

```
pca = PCA()
pca.fit(std_customer_profile)
plt.step(range(1, customer_profile.shape[1] + 1), pca.explained_variance_ratio_.cumsum(),
         where='mid', label='Cumulative Explained Variance')
plt.bar(range(1, customer_profile.shape[1] + 1), pca.explained_variance_ratio_,
        alpha=0.4, color='g', label='Individual Explained Variance')

plt.ylabel('Explained Variance')
plt.xlabel('Principal Components')
plt.legend(loc='center right');
```

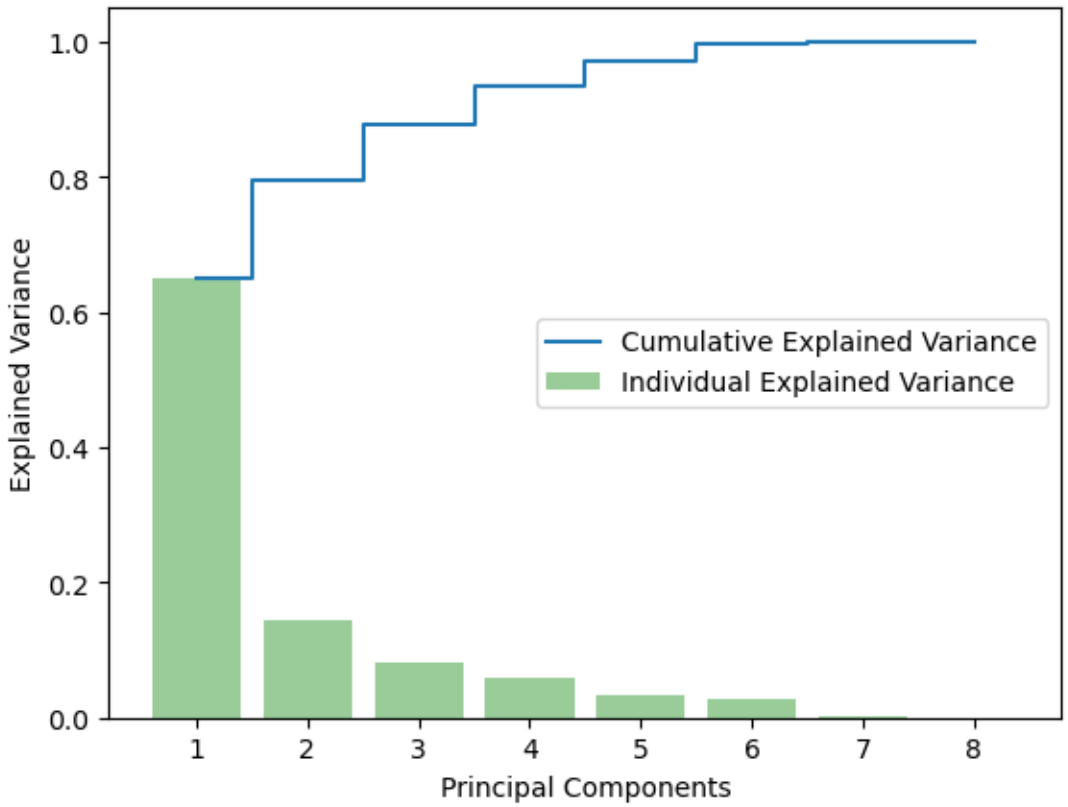


Figure 1 Visualisation de la variance expliquée après PCA

Le graphique obtenu montre la variance cumulée expliquée par les composantes principales. Ici, il apparaît qu'en conservant deux (2) composantes, environ 80 % de la variance est préservée. Cette réduction facilite la visualisation et la modélisation.

4.3 Analyse des différents algorithmes de clustering

4.3.1 Méthode classique : K-Means

Le K-Means est l'algorithme de partitionnement le plus utilisé. Il minimise les distances intra-clusters et tente de regrouper les données autour de centres prédéfinis.

a. Détermination du Nombre Optimal de Clusters

Avant d'appliquer K-Means, nous évaluons le nombre de clusters optimal avec la méthode elbow. Le but est d'identifier le nombre de clusters optimal en calculant l'inertie (somme des distances au carré entre les points et les centres de clusters) pour différents nombres de clusters.

```
elbow = dict()
for k in range(2, 15):
    kmeans = KMeans(init='k-means++', n_clusters=k)
    clusters = kmeans.fit_predict(pca_customer_profile)
    elbow[k] = kmeans.inertia_

plt.title('Elbow Method for Optimal k')
plt.ylabel('Inertia')
plt.xlabel('k');
```

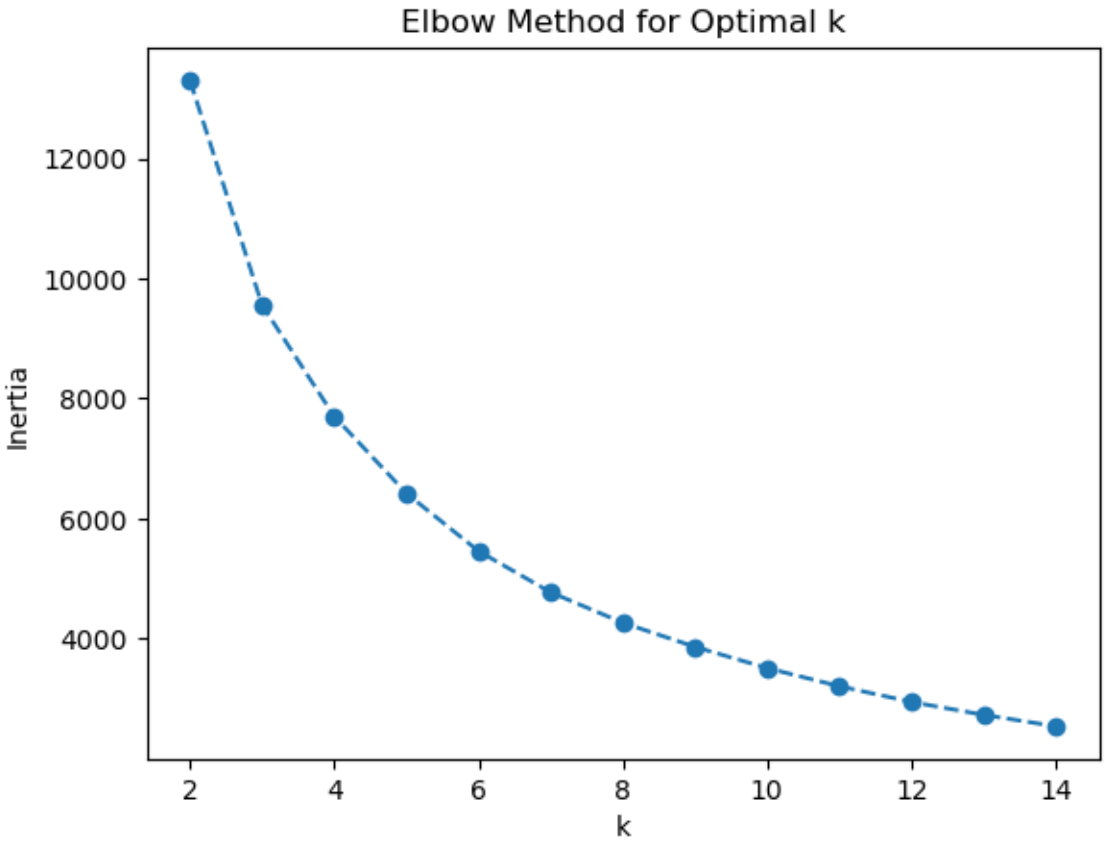


Figure 2 Courbe Elbow post PCA

La courbe Elbow permet d'identifier le point d'inflexion où l'ajout de clusters n'améliore plus significativement l'inertie. Ici, un coude apparait vers k = 3, ce qui est ensuite testé.

b. Application du K-Means

```
kmeans = KMeans(init='k-means++', n_clusters=3)
clusters_customers = kmeans.fit_predict(pca_customer_profile)
```

L'algorithme répartit les clients en trois clusters. Les résultats sont ensuite visualisés pour analyser la qualité de la segmentation.

c. Visualisation des clusters et des centroïdes

Plusieurs types de visualisations ont été générés afin d'évaluer la qualité et la cohérence des clusters. Chacune de ces figures apporte un éclairage complémentaire sur la structure des regroupements, leur cohésion et leur séparation. Vous trouverez ci-dessous une description détaillée des principales figures obtenues et des interprétations associées.

- Scatter Plot avec Centroids (K-Means)

```
plt.scatter(*pca_customer_profile.T, c=clusters_customers, s=20, alpha=0.7)
plt.xlabel('pca_1')
plt.ylabel('pca_2')
# select cluster centers
centroids = kmeans.cluster_centers_
plt.scatter(*centroids.T, c='black', s=70, alpha=0.5);
```

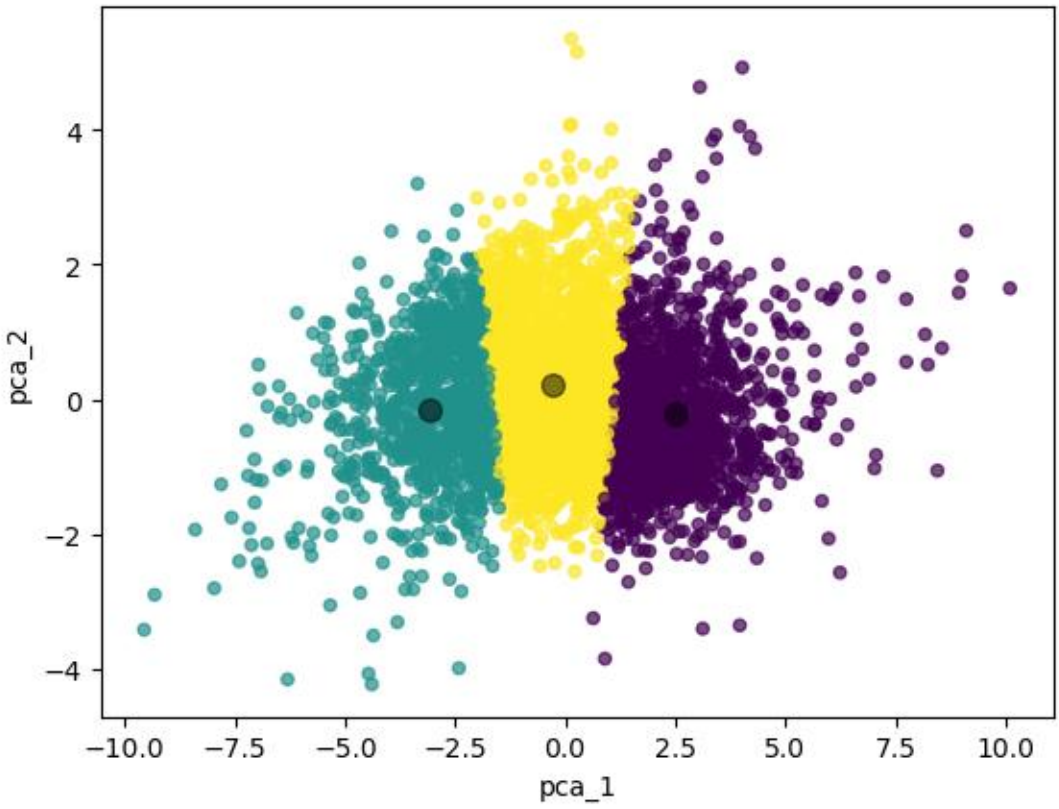


Figure 3 Segmentation des clusters avec K-Means

Cette figure projette les clients dans l'espace réduit à 2 dimensions via PCA, où chaque point est coloré en fonction du cluster assigné par K-Means. Les centroïdes (affichés en noir) représentent les points centraux calculés par l'algorithme et illustrent la moyenne des coordonnées de chaque cluster.

On voit ici une séparation graphique nette des couleurs qui Indique que les clusters sont relativement bien formés et que le K-Means a réussi à distinguer des groupes distincts de clients. Les centroïdes sont placés au cœur de chaque regroupement, ce qui confirme un bon fonctionnement du modèle. On observe que les clusters ont des frontières relativement nettes, mais il peut y avoir des zones de transition entre groupes (entre le cluster jaune et le vert/violet), ce qui est normal dans des données réelles. Les groupes semblent de tailles équilibrées, sans cluster trop petit ou trop grand. Cela peut indiquer une bonne segmentation client.

Diagramme de Silhouette

Évaluer la qualité de la segmentation en mesurant le score de silhouette pour chaque cluster.

```
plot_silhouette_score(pca_customer_profile, clusters_customers)
```

Cluster 0 avg silhouette: 0.3921852234406703

Cluster 1 avg silhouette: 0.3656779045999496

Cluster 2 avg silhouette: 0.33268890198858564

Total avg silhouette: 0.358953417525498

-

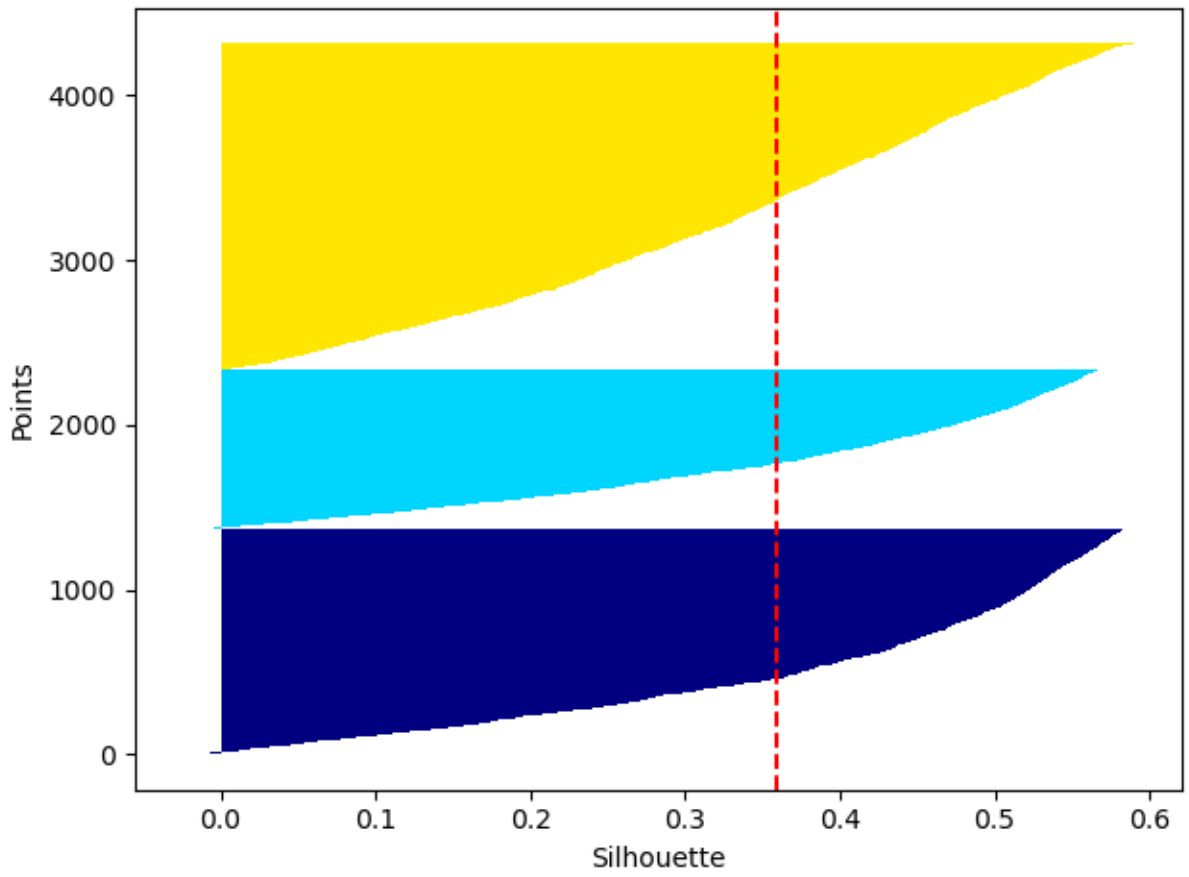


Figure 4 Diagramme de Silhouette post PCA

Ce diagramme en barres horizontales est une représentation graphique des scores de silhouette pour chaque point de l'ensemble de données, groupés par cluster. On peut voir que :

Le Cluster 0 (jaune) avec une Silhouette Moyenne de 0.392 a une Meilleure cohésion et séparation

Le Cluster 1 (bleu clair) avec une Silhouette Moyenne de 0.366 a une Cohésion correcte, mais certains points pourraient être à la frontière.

Cluster 2 (bleu foncé) avec une Silhouette Moyenne de 0.333 a une Cohésion un peu plus faible, plus de dispersion interne.

La ligne rouge verticale représente le score de silhouette moyen global, ici ≈ 0.359, ce qui signifie que dans l'ensemble, les clusters sont modérément bien définis, mais ils révèlent une séparation suffisante pour distinguer trois groupes distincts. L'équilibre entre les clusters et leur interprétabilité en font une option intéressante.

```
customer_profile_kmeans = customer_profile_pca.copy(deep=True)
customer_profile_kmeans['cluster'] = clusters_customers
customer_profile_kmeans['cluster'] = customer_profile_kmeans['cluster'].map({0: 'high', 1: 'low', 2: 'medium'})
customer_profile_kmeans
```

	I	Iu	Imax	E	Savg	R	F	M	cluster
CustomerID									
12347.0	3.390582	2.012837	2.829947	4.310157	2.789379	1	0.845098	3.634477	low
12348.0	3.369401	1.342423	3.098298	3.498731	2.652546	74	0.602060	3.254606	low
12349.0	2.800029	1.863323	2.800029	3.954218	3.244908	18	0.000000	3.244908	low
12350.0	2.294466	1.230449	2.294466	2.675698	2.524266	309	0.000000	2.524266	high
12352.0	2.672098	1.770852	2.103804	3.865416	2.147651	35	1.041393	3.189044	medium
...	...	...	...	...	...	...	...	...	...
18280.0	1.653213	1.000000	1.653213	1.846439	2.256718	277	0.000000	2.256718	high
18281.0	1.732394	0.845098	1.732394	1.842371	1.907519	180	0.000000	1.907519	high
18282.0	1.991226	1.079181	1.875061	3.026987	1.769869	7	0.477121	2.246991	high
18283.0	3.145196	2.419956	2.399674	2.594064	2.117039	3	1.204120	3.321159	low
18287.0	3.200303	1.770852	2.995635	4.198218	2.787054	42	0.477121	3.264175	low

4.3.2 Méthodes Alternatives de Clustering

Afin de tester différentes approches, plusieurs autres algorithmes ont également été implémentés.

a. Fuzzy C-Means

Contrairement à K-Means, le Fuzzy C-Means n'assigne pas un client à un seul cluster de manière stricte, mais calcule pour chaque client un degré d'appartenance à chaque cluster.

```
plt.scatter(*pca_customer_profile.T, c=clusters_customers, s=20, alpha=0.7)
plt.xlabel('pca_1')
plt.ylabel('pca_2')
# select cluster centers
centroids = fcmeans.centers
plt.scatter(*centroids.T, c='black', s=70, alpha=0.5);
```

Python

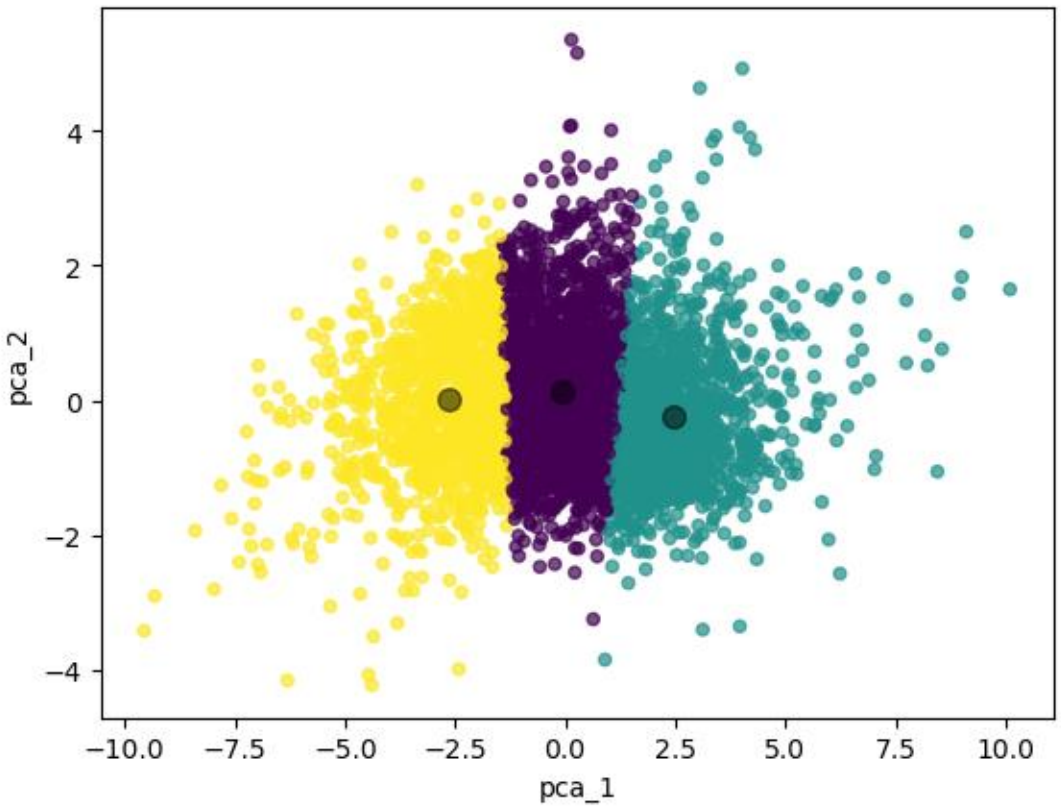


Figure 5 Segmentation des clusters avec Fuzzy C-Means

Cette projection PCA confirme que trois clusters offrent une segmentation naturelle selon la dimension principale PC1, avec un segment très homogène (jaune), un segment intermédiaire (violet) et un segment plus hétérogène (vert).

```python
fcmeans = FCM(n_clusters=3)
fcmeans.fit(pca_customer_profile)
clusters_customers = fcmeans.u.argmax(axis=1)
plot_silhouette_score(pca_customer_profile, clusters_customers)
```

Python

Cluster 0 avg silhouette: 0.34693638406840666

Cluster 1 avg silhouette: 0.37355414763105427

Cluster 2 avg silhouette: 0.33373190665148983

Total avg silhouette: 0.351427645211127

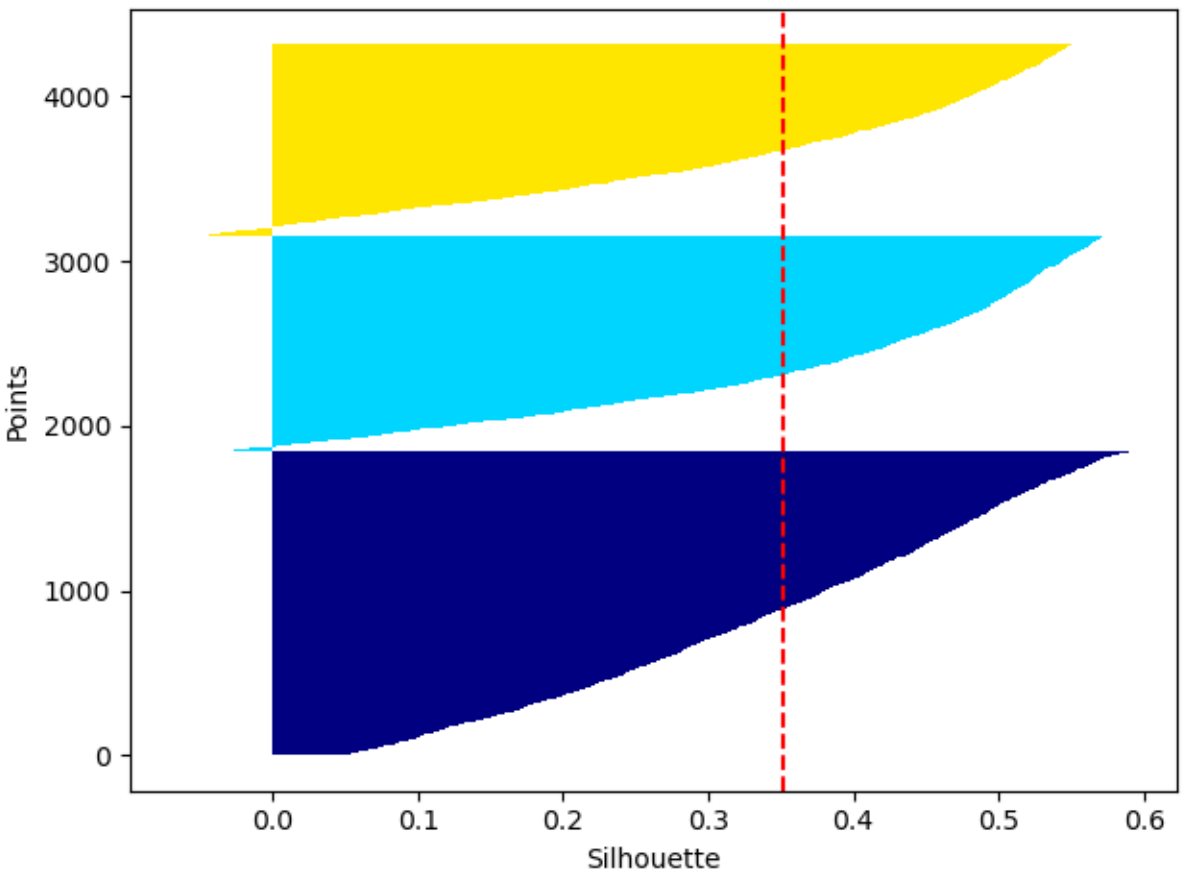


Figure 6 Diagramme de Silhouette Fuzzy C-Means

La flexibilité de l'assignation floue permet de capturer des zones de transition entre clusters. Cependant, le score de silhouette global 0.351427 est légèrement inférieur à celui de K-Means, ce qui suggère une compacité moindre, et une difficulté à définir des frontières nettes.

b. DBSCAN – Clustering Basé sur la Densité

DBSCAN fonctionne en identifiant des zones à forte densité et est capable de détecter des outliers (bruit). Avant, une recherche du paramètre eps (epsilon) est réalisée via une méthode des k-plus proches voisins.

```
knn = NearestNeighbors(n_neighbors=20)
nbrs = knn.fit(pca_customer_profile)
distances, indices = nbrs.kneighbors(pca_customer_profile)
distances = np.sort(distances, axis=0)[:, 1]
plt.plot(distances);
```

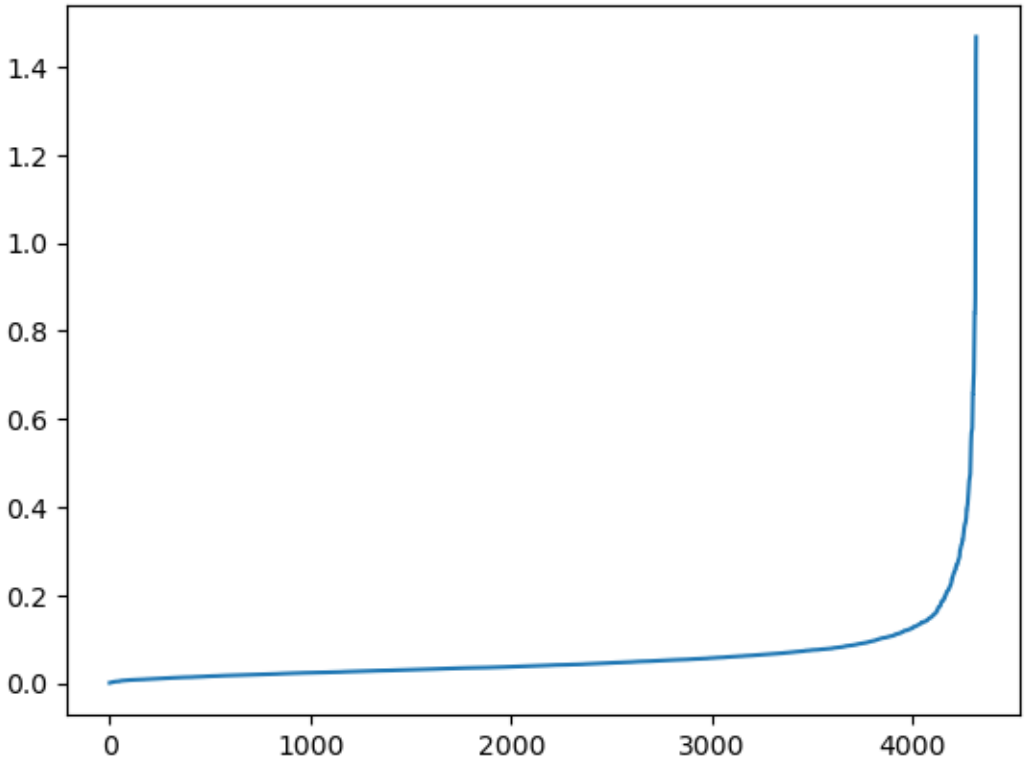


Figure 7 Graphe des distances k-NN

Le graphe des distances k-NN aide à estimer la valeur optimale pour eps en identifiant un "coude" dans la courbe.

Ensuite, DBSCAN est appliqué :

```
dbscan = DBSCAN(eps=0.25, min_samples=20)
clusters_customers = dbscan.fit_predict(pca_customer_profile)
plt.xlabel('pca_1')
plt.ylabel('pca_2')
plt.scatter(*pca_customer_profile.T, c=clusters_customers, s=20, alpha=0.7);
```

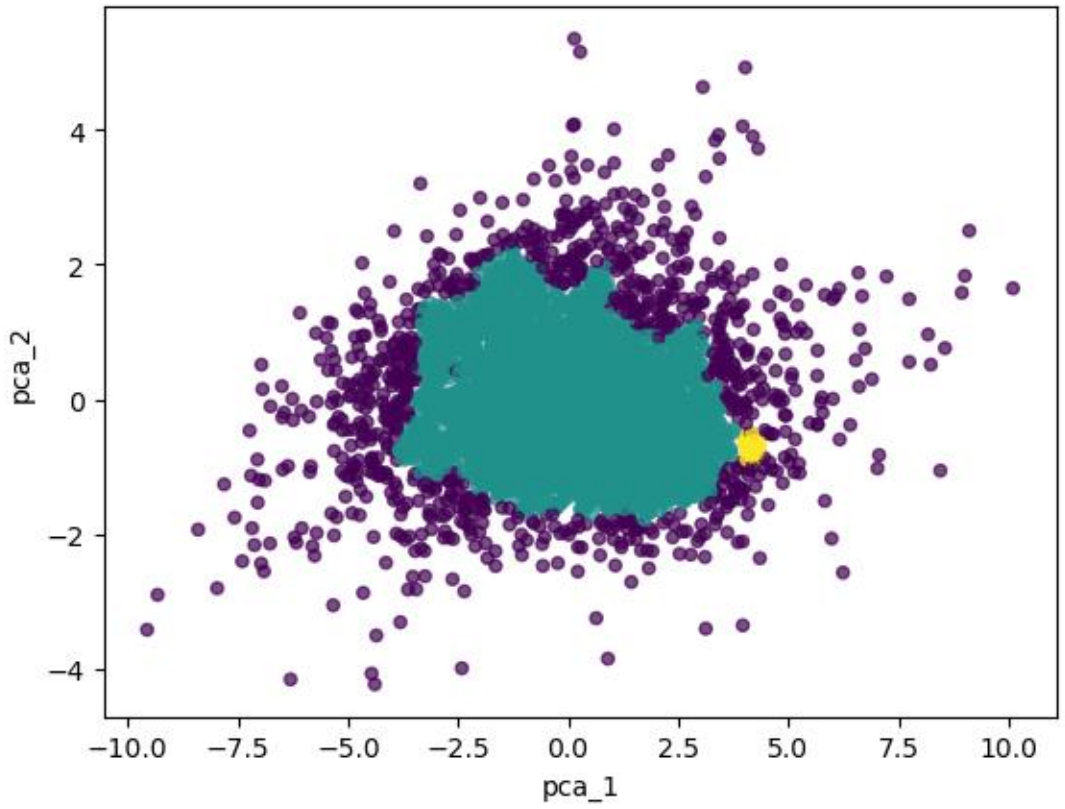


Figure 8 Segmentation des clusters avec DBSCAN

Ce graphique représente la répartition des clients projetés dans l'espace à deux dimensions (via PCA) et colorés en fonction des clusters détectés par DBSCAN. La couleur de chaque point correspond à l'étiquette de cluster attribuée par DBSCAN.

L'application de DBSCAN avec eps=0.25 et min_samples=20 a permis de découvrir un cluster principal cohérent(vert), un micro-cluster distinct (jaune) ainsi qu'un ensemble non négligeable de clients isolés(violet). Cela confirme la capacité de cet algorithme à révéler des structures latentes dans les données sans hypothèse préalable sur le nombre de segments, tout en mettant en évidence les comportements atypiques. Bien que prometteur pour la détection d'anomalies, ce type de segmentation offre une granularité différente qui n'est pas toujours idéale pour alimenter des modèles prédictifs nécessitant des classes bien définies

c. Birch (Balanced Iterative Reducing and Clustering using Hierarchies)

Enfin, l'algorithme Birch est appliqué. Cet algorithme est particulièrement adapté pour des grands ensembles de données en compressant les informations avant de les regrouper.

```
birch = Birch(n_clusters=3)
clusters_customers = birch.fit_predict(pca_customer_profile)
plt.xlabel('pca_1')
plt.ylabel('pca_2')
plt.scatter(*pca_customer_profile.T, c=clusters_customers, s=20, alpha=0.7);
```

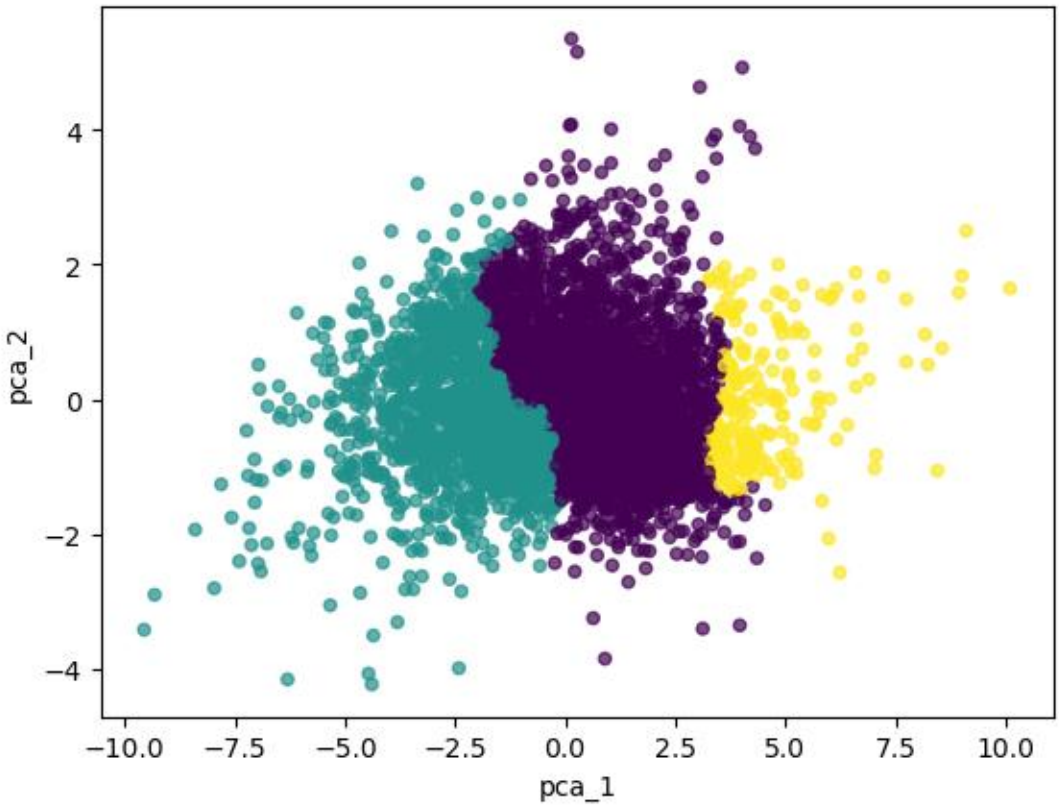


Figure 9 Segmentation des clusters avec Birch

On note une séparation nette entre trois groupes distincts, particulièrement visibles le long de la première composante principale (pca_1). Sur la droite du graphique, on observe un cluster moyennement compact, représenté en jaune, rassemblant des clients ayant des comportements très homogènes et concentrés autour de valeurs positives de pca_1. À l'opposé, le cluster étendu situé à gauche, en vert, montre une dispersion beaucoup plus importante, traduisant une plus grande hétérogénéité comportementale au sein de ce segment. Enfin, au centre de l'espace factoriel, on distingue un cluster intermédiaire (violet), qui constitue le noyau principal des données. Il regroupe des clients au profil moyen ou standard, ne présentant ni des comportements extrêmes ni une structure fortement différenciée. Cette répartition renforce l'idée que la première composante principale joue un rôle déterminant dans la segmentation, en capturant des différences comportementales majeures entre les clients.,

4.4 Conclusion sur le Clustering

L'ensemble des approches explorées a permis d'aboutir à une segmentation client robuste et exploitable. Bien que plusieurs algorithmes aient été testés, notamment BIRCH, Fuzzy C-Means

et DBSCAN, le choix final s'est porté sur K-Means, pour des raisons à la fois méthodologiques, pratiques et stratégiques.

Sa simplicité algorithmique et son interprétabilité visuelle en font un excellent outil pour des analyses marketing opérationnelles. Chaque cluster est représenté par un centroïde clair, qui synthétise les comportements types des clients, facilitant à la fois l'analyse descriptive et la prise de décision.

En termes de qualité de segmentation, K-Means a permis d'identifier trois groupes relativement bien définis dans l'espace réduit par ACP. Même si le score de silhouette moyen (≈ 0,359) témoigne d'un certain chevauchement entre les segments, la répartition équilibrée et homogène des clusters permet une exploitation fiable dans un contexte prédictif. Cette nuance entre cohésion interne et séparation externe est acceptable dans le cadre d'une application orientée vers la modélisation comportementale.

Enfin, la rapidité d'exécution et la reproductibilité de K-Means en font un choix pragmatique pour des analyses à itérations multiples. Dans un contexte où les pipelines sont appelés à être ajustés ou réexécutés fréquemment, cette stabilité constitue un atout majeur.

En résumé, le processus de clustering a suivi une méthodologie rigoureuse : nettoyage et normalisation des données, réduction de dimension par ACP, puis évaluation comparative de plusieurs algorithmes. Le choix de K-Means avec k = 3 est justifié non seulement par la qualité des segments obtenus, mais aussi par sa parfaite intégration dans l'architecture de modélisation prédictive.

5. Analyse Prédictive

Ce chapitre décrit la mise en œuvre de la phase d'analyse prédictive. L'objectif est de tirer parti de la segmentation issue de l'algorithme KMeans (fichier customer_profile_kmeans.csv) pour entraîner des modèles capables de prédire à quel segment appartient un nouveau client. L'approche repose sur l'utilisation de pipelines de classification intégrant à la fois le prétraitement des données, l'optimisation des hyperparamètres et l'évaluation des performances.

5.1. Chargement et Prétraitement des Données

5.1.1 Importation et Chargement du Jeu de Données

Pour l'analyse prédictive, le fichier utilisé est celui issu de la segmentation réalisée par K-Means.

```
df_customer_profile = pd.read_csv('E:/Memoire/data/customer_profile_kmeans.csv', sep='\t')

print(df_customer_profile.head())
```
✓ 0.0s Python

```
   CustomerID         I        Iu      Imax         E      Savg    R  \
0     12347.0  3.390582  2.012837  2.829947  4.310157  2.789379    1
1     12348.0  3.369401  1.342423  3.098298  3.498731  2.652546   74
2     12349.0  2.800029  1.863323  2.800029  3.954218  3.244908   18
3     12350.0  2.294466  1.230449  2.294466  2.675698  2.524266  309
4     12352.0  2.672098  1.770852  2.103804  3.865416  2.147651   35

          F         M cluster
0  0.845098  3.634477     low
1  0.602060  3.254606     low
2  0.000000  3.244908     low
3  0.000000  2.524266    high
4  1.041393  3.189044  medium
```

Nous procédons par la suite à la conversion en NumPy. C'est une étape qui permettra optimiser les opérations numériques.

```
customer_profile = df_customer_profile.to_numpy()
```
✓ 0.0s Python

5.1.2 Division du Jeu de Données

Les caractéristiques (X) et la variable cible (y) sont ensuite séparées. Le jeu de données est ensuite divisé en ensembles d'entraînement et de test (avec stratification pour conserver la répartition des classes).

```
# Suppose que la dernière colonne correspond à la variable cible
X, y = customer_profile[:, :-1], customer_profile[:, -1]

# Split train/test : 75% train et 25% test, en conservant la distribution des cibles
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, stratify=y)
```
✓ 0.0s Python

Cette séparation garantit que l'estimation de la performance (accuracy, etc.) sera réalisée sur des données extérieures à l'entraînement, préservant ainsi l'objectivité de l'évaluation.

5.2. Définition d'Outils de Visualisation et d'Aide à l'Optimisation

Afin d'évaluer et de suivre l'entraînement de nos modèles, plusieurs fonctions utilitaires sont définies :

5.2.1 Fonction plot_learning_curve

Cette fonction trace la courbe d'apprentissage du modèle passé dans le pipeline. On calcule ici, pour différents sous-échantillons du jeu d'entraînement, la performance en score sur l'entraînement et sur la validation croisée.

```python
def plot_learning_curve(pipeline, X, y, scorer='accuracy', cv=5, train_sizes=np.linspace(.1, 1.0, 5),
                        shuffle=False, random_state=None):

    train_sizes, train_scores, test_scores = learning_curve(pipeline, X, y, train_sizes=train_sizes, cv=cv,
                                                            scoring=scorer, n_jobs=-1, shuffle=shuffle,
                                                            random_state=random_state)

    mean_train_score = np.mean(train_scores, axis=1)
    std_train_score = np.std(train_scores, axis=1)
    mean_test_score = np.mean(test_scores, axis=1)
    std_test_score = np.std(test_scores, axis=1)

    plt.title(str(pipeline['estimator'].__class__.__name__) + ' learning curve')
    plt.xlabel('training set size')
    plt.ylabel('score')

    plt.plot(train_sizes, mean_train_score, label='train score', color='navy', marker='.', lw=2)
    plt.fill_between(train_sizes, mean_train_score + std_train_score,
                     mean_train_score - std_train_score, color='navy', alpha=0.2)
    plt.plot(train_sizes, mean_test_score, label='cross-validation score', color='darkorange', marker='.', lw=2)
    plt.fill_between(train_sizes, mean_test_score + std_test_score,
                     mean_test_score - std_test_score, color='darkorange', alpha=0.2)

    plt.legend(loc='lower right').get_frame().set_facecolor('white')
```

Cette courbe aide à détecter les problèmes de sur-ajustement (high training score, faible validation) ou de sous-ajustement. Elle fournit ainsi une validation graphique de la capacité du modèle à généraliser.

5.2.2 Fonction plot_validation_curve

Cette fonction permet d'observer l'évolution du score en fonction d'un hyperparamètre donné à partir d'une validation croisée.

```python
def plot_validation_curve(pipeline, X, y, param_grid, param_name, scorer='accuracy', cv=5):

    param_range = param_grid[param_name]
    train_scores, test_scores = validation_curve(pipeline, X, y, param_name=param_name, param_range=param_range,
                                                 cv=cv, scoring=scorer, n_jobs=-1)

    mean_train_score = np.mean(train_scores, axis=1)
    std_train_score = np.std(train_scores, axis=1)
    mean_test_score = np.mean(test_scores, axis=1)
    std_test_score = np.std(test_scores, axis=1)

    plt.title(str(pipeline['estimator'].__class__.__name__) + ' validation curve')
    plt.xlabel(param_name)
    plt.ylabel('score')

    plt.plot(param_range, mean_train_score, label='training score', color='navy', marker='.', lw=2)
    plt.fill_between(param_range, mean_train_score - std_train_score,
                     mean_train_score + std_train_score, alpha=0.2, color='navy')
    plt.plot(param_range, mean_test_score, label='cross-validation score', color='darkorange', marker='.', lw=2)
    plt.fill_between(param_range, mean_test_score - std_test_score,
                     mean_test_score + std_test_score, alpha=0.2, color='darkorange')

    plt.legend(loc='lower right').get_frame().set_facecolor('white')
```

Elle permet de mesurer l'impact direct de l'hyperparamètre (le coefficient C) sur la performance.

5.2.3 Fonction print_pretty_cv_results

Cette fonction formate et affiche les résultats de la validation croisée issus des recherches d'hyperparamètres.

```
def print_pretty_cv_results(grid):
    df = pd.DataFrame(grid.cv_results_)[['params', 'mean_test_score', 'rank_test_score']].sort_values(by='rank_test_score')
    df['params'] = df['params'].apply(lambda param_dict: {k.replace('estimator__', ''): v for k, v in param_dict.items()})
    df.rename(columns={'mean_test_score': 'mean_val_score',
                       'rank_test_score': 'rank_val_score'}, inplace=True)

    def highlight_equal(s, value, column):
        is_max = pd.Series(data=False, index=s.index)
        is_max[column] = s.loc[column] == value
        return ['background-color: lightyellow' if is_max.any() else '' for v in is_max]

    return df.head(15).reset_index(drop=True).style.apply(highlight_equal, value=1, column=['rank_val_score'], axis=1)
```

✓ 0.0s

Ces fonctions sont réutilisées pour tous les modèles afin de visualiser leur apprentissage et de comparer les performances lors de la recherche des meilleurs hyperparamètres.

5.3. Construction et évaluation des pipelines de modélisation

Le notebook teste plusieurs classificateurs. Pour chacun, un pipeline est défini incluant une normalisation par StandardScaler puis l'estimateur en question. La recherche d'hyperparamètres est réalisée via GridSearchCV ou RandomizedSearchCV, puis les performances sont évaluées sur l'ensemble de test avec diverses métriques.

5.3.1 Support Vector Classifier (SVC)

Implémentation du Pipeline SVC et Optimisation

On commence par mettre en place et entraîne le modèle de classification SVM (SVC) à l'aide d'un pipeline de traitement automatisé combiné à une recherche d'hyperparamètres.

```python
pipeline = Pipeline([('scaler', StandardScaler()),
                     ('estimator', SVC())])

tuned_parameters = {'estimator__C': (1, 10, 100, 1000)}

svc_grid = GridSearchCV(pipeline,
                        param_grid=tuned_parameters,
                        scoring='accuracy',
                        cv=5,
                        n_jobs=-1,
                        refit=True,
                        return_train_score=True,
                        verbose=True).fit(X_train, y_train)
```

✓ 0.7s Python

Ainsi on automatise le choix du meilleur modèle SVM, en testant différentes valeurs de C, et en utilisant la validation croisée pour garantir des résultats fiables et généralisables.

a. Visualisation des performances

Learning Curve et Validation Curve : deux fonctions sont utilisées pour diagnostiquer les modèles :

- Courbes d'apprentissage : pour détecter le surapprentissage ou le sous-apprentissage.
- Courbes de validation : pour analyser l'effet d'un hyperparamètre donné.

```python
plot_learning_curve(svc_grid.best_estimator_, X, y)

#plt.savefig('../tex/img/classification/svm_lc.png')
```

✓ 1.2s Python

Figure 10 Learning Curve issue de SVC

Légende

- Axe des abscisses (x) : Taille croissante du jeu d'entraînement, exprimée en nombre d'échantillons utilisés pour l'apprentissage.
- Axe des ordonnées (y) : Score de performance, ici mesuré par l'accuracy.
- Courbe bleu foncé (« train score ») : Précision du modèle sur les données d'entraînement.
- Courbe orange (« cross-validation score ») : Précision moyenne obtenue par validation croisée à chaque taille d'échantillon.
- Zone ombrée orange : Intervalle de variation des scores de validation croisée, correspondant à ± un écart-type.

Analyse des tendances observées

- Stabilité de l'apprentissage sur le jeu d'entraînement
Le score d'apprentissage reste très élevé (proche de 1.0) quelle que soit la taille du jeu de données. Cela indique que le modèle SVC est très flexible et parvient à parfaitement ajuster les données d'entraînement, sans difficulté.

- Amélioration rapide de la validation croisée
Le score de validation commence autour de 0.35 avec un petit nombre d'échantillons, puis

augmente rapidement. À partir de 2000 exemples, il dépasse 0.90 et atteint une valeur stable avoisinant 0.98, illustrant ainsi la capacité de généralisation remarquable du modèle avec un volume de données suffisant.

- Réduction progressive de l'écart entre apprentissage et validation
L'écart entre les deux courbes (train/test) diminue au fur et à mesure que la taille de l'échantillon augmente. Cette convergence progressive des courbes suggère une atténuation du surapprentissage initial et un meilleur équilibre entre complexité du modèle et capacité de généralisation.

Ce graphique valide le bon fonctionnement du SVC. Il montre que trop de complexité (C élevé) mène à un surajustement. Il justifie le choix de C = 100 comme valeur optimale, car c'est là que le modèle maximise la performance tout en conservant une bonne généralisation.

```
plot_validation_curve(svc_grid.best_estimator_, X, y, param_grid=tuned_parameters, param_name='estimator__C')
```
✓ 1.4s Python

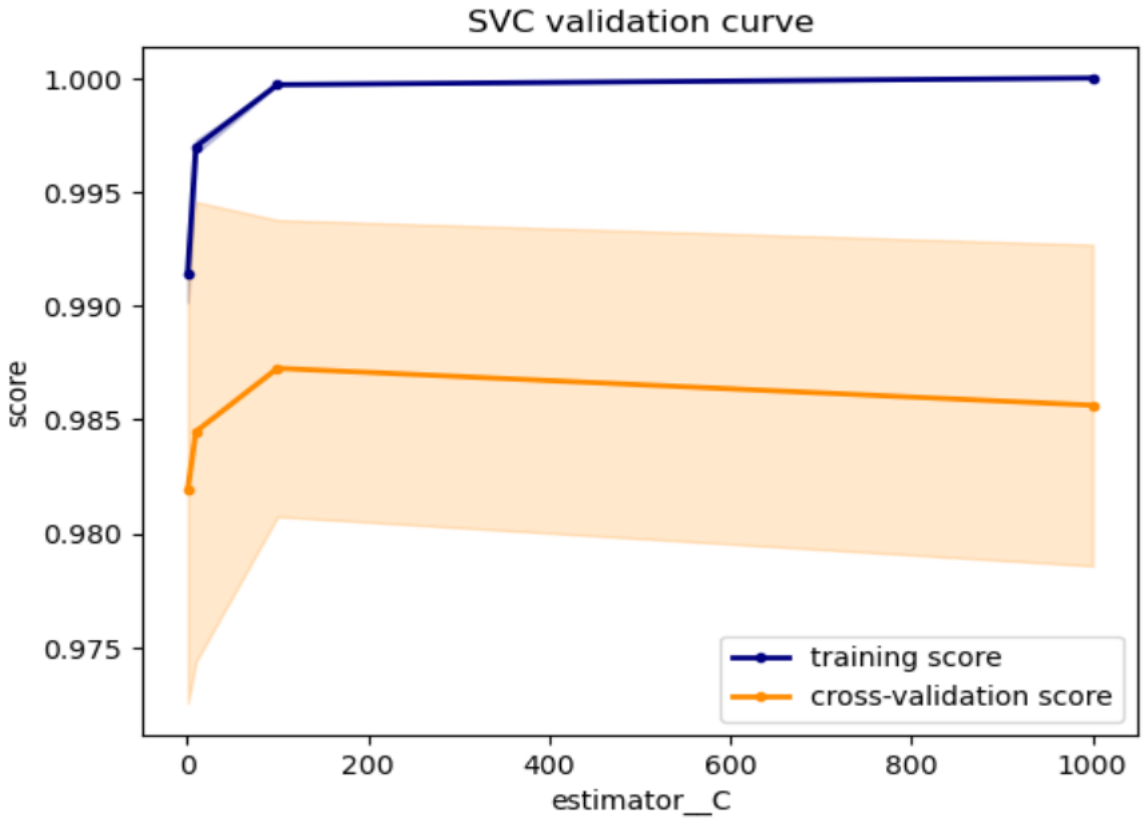


Figure 11 Validation curve issue de SVC

Légende

- Axe des abscisses (x) : différentes valeurs testées pour C (1, 10, 100, 1000).
- Axe des ordonnées (y) : Score de performance, ici mesuré par l'accuracy.
- Courbe bleu foncé (« train score ») : score moyen sur le jeu d'entraînement.

- Courbe orange (« cross-validation score ») : score moyen en validation croisée (5-fold).
- Zone ombrée orange : Intervalle de variation des scores de validation croisée, correspondant à ± un écart-type.

Analyse des tendances observées

- Training score (bleu) proche de 1,0 quel que soit C : le modèle s'ajuste très bien aux données d'entraînement même pour de faibles valeurs de C.

- Validation score (orange) augmente avec C :

• Pour C = 1, le score CV est le plus bas (environ 0,98).

• En augmentant C (10, 100), la performance CV monte jusqu'à environ 0,986–0,987.

• Au-delà (C = 1000), la courbe se stabilise, témoignant d'un plateau.

En résumé, cette validation curve montre que régler C entre 10 et 100 optimise la performance sur des données non vues, en garantissant un bon équilibre entre biais et variance.

Test Set :

```
y_pred = svc_grid.best_estimator_.predict(X_test)
accuracy_score(y_pred, y_test)
```

✓ 0.0s Python

Après entraînement du modèle SVC avec validation croisée, le meilleur estimateur a été évalué sur un jeu de test indépendant. Le score obtenu est de 0.9861 soit une précision de 98,61 %, ce qui démontre une excellente capacité du modèle à généraliser. Ce résultat valide à la fois la qualité des clusters identifiés en amont (en tant que cible) et la cohérence des descripteurs utilisés pour prédire l'appartenance des clients à ces groupes.

5.3.2 Réseaux de neurones multicouches (MLP)

Après l'entraînement du modèle SVC, nous avons exploré l'approche des réseaux de neurones multicouches (MLP) dans le but de prédire avec précision le segment auquel appartient un client (low, medium ou high)

Elle configure et entraîne un réseau de neurones artificiels (MLPClassifier) à l'aide d'un pipeline de traitement automatisé et d'une recherche systématique d'hyperparamètres

```python
pipeline = Pipeline([('scaler', StandardScaler()),
                     ('estimator', MLPClassifier())])

tuned_parameters = {'estimator__hidden_layer_sizes': ((50,), (100,), (150,)),
                    'estimator__activation': ('logistic', 'tanh'),
                    'estimator__solver': ('lbfgs',)}

mlp_grid = GridSearchCV(pipeline,
                        param_grid=tuned_parameters,
                        scoring='accuracy',
                        cv=5,
                        n_jobs=-1,
                        refit=True,
                        return_train_score=True,
                        verbose=True).fit(X_train, y_train)
```

✓ 7.4s Python

Il combine prétraitement, entraînement et validation dans un pipeline reproductible et rigoureux.

Visualisation des performances

```python
plot_learning_curve(mlp_grid.best_estimator_, X, y)

#plt.savefig('../tex/img/classification/nn_lc.png')
```

✓ 3.0s Python

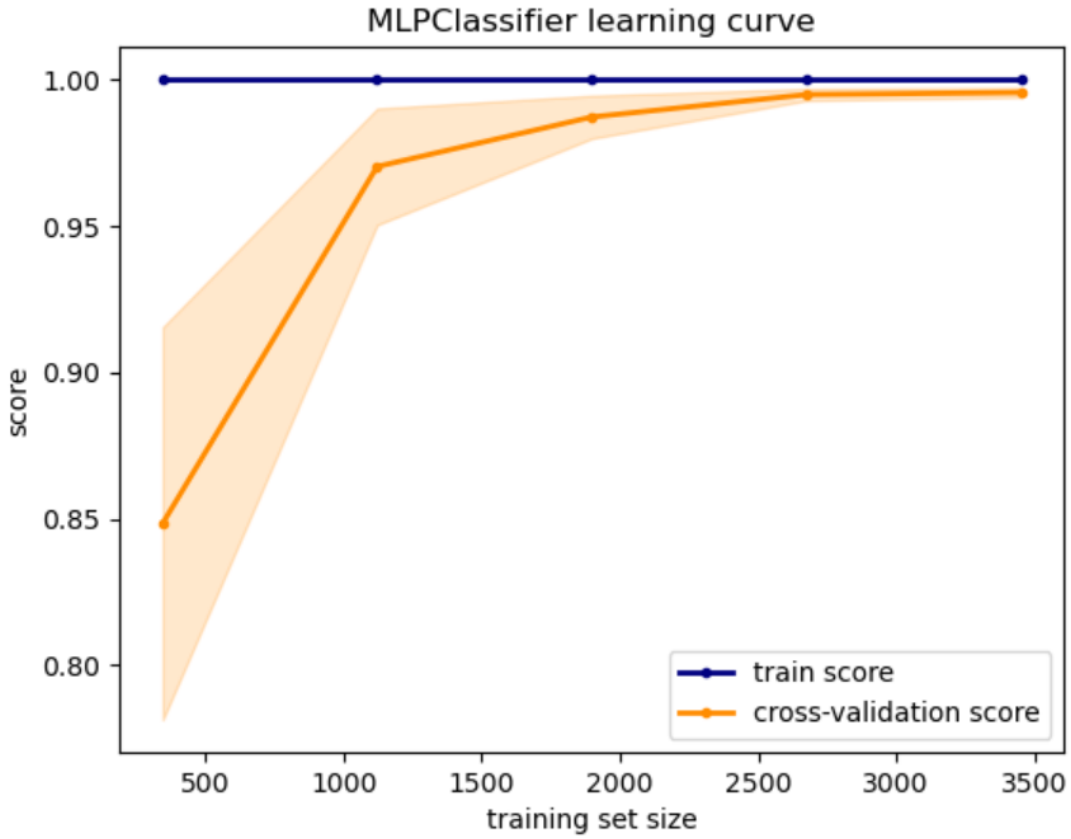


Figure 12 Learning Curve issue de MPLClassifier

Légende

- Axe des abscisses (x) : taille de l'échantillon d'apprentissage (nombre d'exemples utilisés pour entraîner le modèle).
- Axe des ordonnées (y) : score de précision (accuracy) mesuré à la fois sur les données d'apprentissage (courbe bleue) et sur les données de validation croisée (courbe orange).
- Les zones colorées représentent les intervalles de confiance (écarts types) autour des scores moyens.

Analyse des tendances observées :

- Score d'apprentissage (train score)
Le score obtenu sur le jeu d'apprentissage reste constamment proche de 1.00, quelle que soit la taille de l'échantillon. Ce comportement témoigne de la capacité du réseau de neurones à mémoriser parfaitement les données d'entraînement, même avec un petit nombre d'exemples. Bien que cela puisse être le signe d'un possible surapprentissage, cette hypothèse doit être croisée avec le comportement de la courbe de validation.

- Score de validation (cross-validation score)
Le score de validation progresse régulièrement à mesure que la taille du jeu d'entraînement

augmente. Il démarre aux alentours de 0.85 pour de petits jeux de données et atteint près de 0.99 sur les ensembles plus larges. Cette montée en puissance traduit une excellente capacité de généralisation du modèle, renforcée par la convergence visible entre les courbes d'apprentissage et de validation.

- Stabilité du modèle (zone d'incertitude)
L'intervalle de confiance autour du score de validation est initialement large, ce qui est typique lorsque l'on dispose de peu d'exemples. Toutefois, il se resserre nettement à mesure que la taille des données augmente, traduisant une meilleure stabilité du modèle et une réduction des fluctuations entre les différentes validations croisées.

Les résultats issus de la courbe d'apprentissage du MLPClassifier sont très encourageants. Le modèle démontre une précision élevée et constante, tant sur les données d'entraînement que de validation, sans signal évident de surapprentissage ou de sous-apprentissage. Par ailleurs, son comportement s'améliore nettement avec l'ajout de données, ce qui en fait un candidat robuste pour la prédiction sur de nouveaux clients. Ces observations confirment l'intérêt du MLPClassifier dans le contexte de segmentation prédictive étudié dans ce mémoire.

Test Set :

```
y_pred = mlp_grid.best_estimator_.predict(X_test)
accuracy_score(y_pred, y_test)
```

✓ 0.0s Python

Le modèle de réseau de neurones (MLPClassifier) a obtenu un score de précision de 0,9925, soit 99,26 %, sur l'échantillon de test. Ce résultat témoigne de sa capacité à prédire avec une grande fiabilité le segment d'un client (high, medium ou low) sur des données totalement inédites. Cette excellente performance s'explique par l'adéquation du modèle à la structure des données comportementales, sa faculté à généraliser sans surapprentissage, ainsi que par la stabilité observée sur les courbes d'apprentissage, où les scores d'entraînement et de validation étaient très proches. L'ensemble de ces éléments confirme la robustesse et l'efficacité du MLPClassifier dans le cadre de cette analyse prédictive.

5.3.3 Random Forest

L'algorithme Random Forest a été exploré pour sa robustesse et sa capacité à modéliser des interactions complexes entre variables.

Ce bloc de code met en place un pipeline de modélisation avec Random Forest, combiné à une optimisation aléatoire des hyperparamètres via RandomizedSearchCV. Ce bloc de code permet de construire, optimiser et entraîner un modèle Random Forest performant et robuste, en testant intelligemment un ensemble d'hyperparamètres. Le pipeline assure un traitement cohérent des données, tandis que RandomizedSearchCV trouve les paramètres les plus efficaces sans tester toutes les combinaisons possibles.

```
pipeline = Pipeline([('scaler', StandardScaler()),
                     ('estimator', RandomForestClassifier(random_state=42))])

tuned_parameters = {'estimator__n_estimators': (100, 500, 1000),
                    'estimator__bootstrap': (True, False),
                    'estimator__max_depth': (10, 30, 60),
                    'estimator__min_samples_split': (2, 5),
                    'estimator__min_samples_leaf': (1, 2),
                    'estimator__max_features': ('sqrt', 'log2')}

random_search = RandomizedSearchCV(
    pipeline,
    param_distributions=tuned_parameters,
    n_iter=20,
    scoring='accuracy',
    cv=3,
    n_jobs=-1,
    verbose=True,
    random_state=42
)
```

Visualisation des performances

```
plot_learning_curve(random_search.best_estimator_, X, y)
```

✓ 3.8s Python

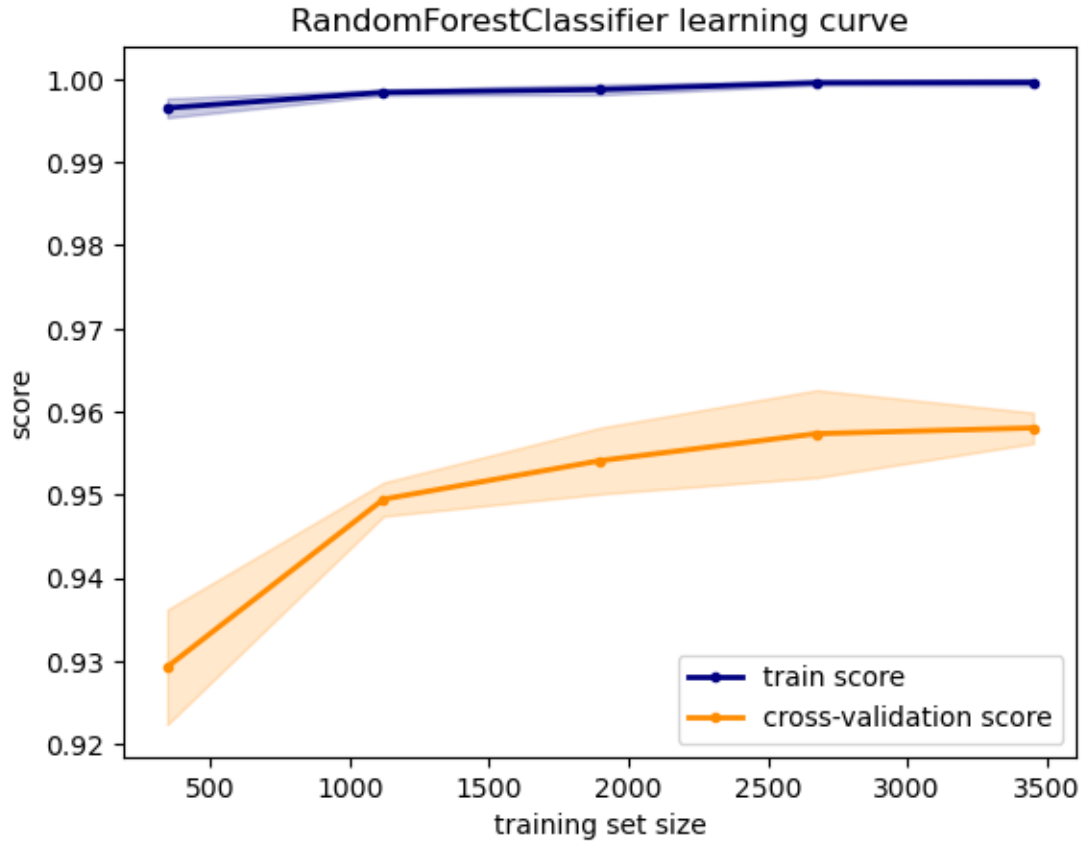


Figure 13 Learning Curve RFC

Légende

- Courbe bleue : performances du modèle sur les données d'entraînement (train score) ;
- Courbe orange : performances du modèle sur les données de validation (cross-validation score).

Observations

- Score d'entraînement élevé : Le modèle atteint un score proche de 1.0 (100 %) sur l'ensemble d'entraînement, ce qui est typique des forêts aléatoires qui s'ajustent très bien aux données.

- Score de validation croissant : Le cross-validation score augmente progressivement avec la taille des données d'apprentissage, pour atteindre environ 0.96 (96 %), ce qui montre une bonne capacité de généralisation.

- Zone d'ombre (bande orange) : L'intervalle de confiance autour du score de validation est relativement étroit, surtout pour les grands ensembles de données. Cela montre une stabilité du modèle avec des résultats cohérents lors de la validation croisée.

Analyse des tendances observées :

Le modèle présente une capacité d'apprentissage efficace, marquée par une amélioration continue de ses performances à mesure que la taille du jeu d'entraînement augmente. Bien que l'écart entre les courbes d'entraînement et de validation reste relativement constant, cela suggère uniquement un léger surapprentissage, demeurant maîtrisé et sans conséquence notable sur la capacité de généralisation. De manière globale, l'algorithme affiche un bon compromis biais-variance, ce qui témoigne d'un apprentissage stable et cohérent, apte à produire des prédictions fiables sur de nouvelles données.

La Random Forest se révèle robuste, précise et bien calibrée pour notre tâche de classification des profils clients. Sa performance élevée sur le set de test (comme mesurée par l'accuracy) confirme les résultats de cette courbe d'apprentissage.

Test Set:

```
y_pred = random_search.best_estimator_.predict(X_test)
accuracy = accuracy_score(y_pred, y_test)
```

Accuracy: 0.9639

Le modèle Random Forest a atteint une précision de 96,39 %, ce qui signifie que près de 96,4 % des clients ont été correctement classés dans leur segment respectif (high, medium ou low). Ce

niveau de performance confirme que le modèle généralise efficacement sur des données inédites, attestant ainsi de sa robustesse et de sa fiabilité pour la tâche de classification des profils clients.

5.3.4 Decision Tree

L'algorithme Decision Tree a été mis en œuvre afin de proposer un modèle à la fois rapide à entraîner et facilement interprétable. Particulièrement apprécié pour sa lisibilité, notamment à travers sa représentation graphique, l'arbre de décision permet de capturer des relations non linéaires entre les variables explicatives. Son implémentation s'appuie sur un pipeline complet intégrant le prétraitement des données, l'apprentissage du modèle, ainsi qu'une optimisation aléatoire des hyperparamètres via la méthode RandomizedSearchCV, permettant d'améliorer les performances tout en limitant le coût computationnel.

```
pipeline = Pipeline([('scaler', StandardScaler()),
                     ('estimator', DecisionTreeClassifier())])



tuned_parameters = {'estimator__splitter': ('best', 'random'),
                    'estimator__max_depth': (10, 30, 60),
                    'estimator__min_samples_split': (2, 5, 10),
                    'estimator__min_samples_leaf': (1, 2, 4),
                    'estimator__max_features': ('auto', 5),
                    'estimator__criterion': ('gini', 'entropy')}



dt_random_search = RandomizedSearchCV(pipeline,
                      param_distributions=tuned_parameters,
                      n_iter=20,
                      scoring='accuracy',
                      cv=3,
                      n_jobs=-1,
                      random_state=42,
                      verbose=True)
```

Visualisation des performances

```
plot_learning_curve(dt_random_search.best_estimator_, X, y)
```

✓ 0.6s Python

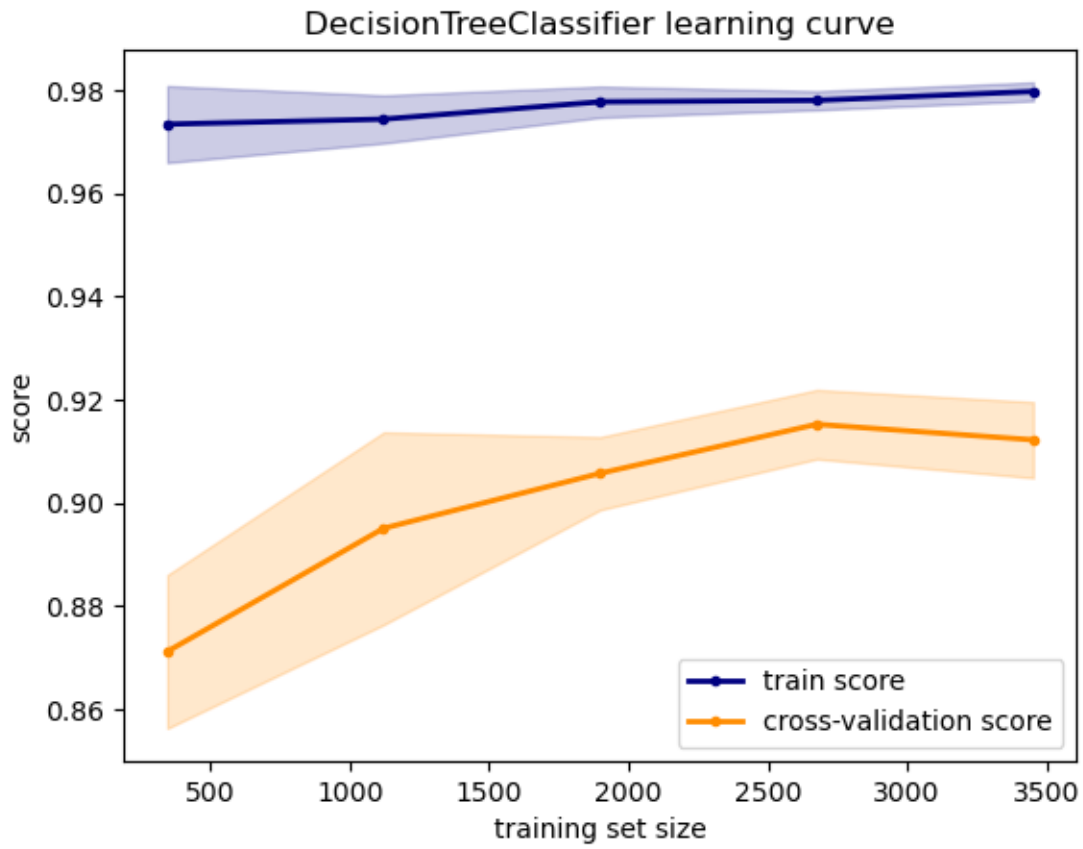


Figure 14 Learning Curve issue de DecisionTree

Test Set:

```
y_pred = dt_random_search.best_estimator_.predict(X_test)
accuracy_score(y_pred, y_test)
```

✓ 0.0s Python

Accuracy 0.9138

Le modèle Decision Tree a permis de classer correctement 91,38 % des clients du jeu de test dans leur segment respectif (high, medium ou low), ce qui constitue une performance tout à fait satisfaisante. Ce taux de précision confirme que l'algorithme a su extraire des règles simples mais pertinentes à partir des attributs comportementaux calculés (tels que les variables RFM, I, Iu, etc.), lui permettant de séparer efficacement les différentes classes. Malgré sa simplicité, l'arbre de décision démontre ainsi une bonne capacité à modéliser les comportements clients de manière interprétable et exploitable.

6. Comparaison Globale des Modèles et Choix de la Segmentation

Les performances observées sur le jeu de test sont les suivantes:

- MLPClassifier (réseau de neurones) : 99,26 % d'accuracy

- SVC (Support Vector Classifier) : 98,61 % d'accuracy
- K-Nearest Neighbors (KNN) : 96,76 % d'accuracy
- Random Forest : 96,39 % d'accuracy
- Naïve Bayes : 96 % d'accuracy
- Decision Tree : 91,38 % d'accuracy

Tous les modèles testés affichent des performances élevées, avec une nette supériorité des classificateurs MLP et SVC, qui atteignent des taux de précision proches ou supérieurs à 98 %. Cependant, le choix du modèle prédictif ne peut être dissocié du type de segmentation sur lequel repose la classification. Dans cette étude, le fichier utilisé pour l'apprentissage supervisé est customer_profile_kmeans.csv, ce qui signifie que la variable cible (le segment client) a été produite par l'algorithme K-Means.

Le choix de cette segmentation se justifie par plusieurs raisons. D'une part, K-Means produit des clusters bien séparés, comme le montrent les visualisations obtenues par ACP (scatter plots), les centroïdes représentatifs de chaque groupe, ainsi que les diagrammes de silhouette. Cette stabilité structurelle facilite la construction de modèles prédictifs efficaces. D'autre part, la méthode K-Means est rapide, reproductible et facilement intégrable dans un pipeline complet de traitement des données, ce qui en fait un choix stratégique cohérent pour des tâches de modélisation itérative.

Les résultats obtenus avec les différents classificateurs, tous entraînés sur cette même base, valident ce choix méthodologique : la majorité des modèles dépasse les 96 % de précision, confirmant la robustesse de la segmentation en amont. Par conséquent, le modèle prédictif retenu peut s'appuyer en toute confiance sur cette base pour générer des prédictions fiables sur de nouveaux clients.

Les modèles prédictifs développés au cours de cette étude peuvent désormais être mobilisés dans une perspective opérationnelle, notamment pour anticiper les comportements futurs des clients, orienter les décisions commerciales, et optimiser les actions marketing en fonction du profil de chaque segment. Grâce à la robustesse des performances obtenues, ces outils prédictifs constituent une base solide pour le déploiement de stratégies personnalisées, en réponse aux besoins spécifiques identifiés au sein de chaque groupe client.

Cette étape marque ainsi la transition vers une analyse plus fine du comportement d'achat, à travers l'étude des motifs séquentiels présents dans les transactions. Le chapitre suivant s'intéressera donc à l'expérimentation, afin de révéler des régularités temporelles dans les parcours d'achat, enrichissant ainsi la compréhension dynamique des comportements clients.

Chapitre VI : Expérimentation

Chapitre VI : Expérimentation

1. Nature de l'expérimentation

L'expérimentation menée dans le cadre de ce mémoire vise à mettre en pratique la méthodologie théorique élaborée précédemment afin de répondre à la problématique centrale.

Cette expérimentation adopte une approche dynamique et temporelle. Chaque transaction est replacée dans son contexte chronologique, de manière à reconstruire les séquences d'achats des clients au fil du temps. Cela permet de dégager des enchaînements caractéristiques tels que :

- Un client qui achète régulièrement des articles de décoration finit souvent par se tourner vers du mobilier,
- L'achat d'un produit festif est fréquemment suivi de l'acquisition d'accessoires complémentaires,
- Certains paniers initiaux comportent toujours des articles de première nécessité avant d'évoluer vers des achats plus spécifiques.

La nature de l'expérimentation ne se limite donc pas à prédire une classe ou une catégorie, mais cherche à découvrir des motifs récurrents reflétant les comportements clients. En ce sens, elle combine plusieurs dimensions méthodologiques :

- Préparation séquentielle des données : transformation des transactions en séquences structurées par client et par période (mois, trimestre). Cette étape est essentielle pour conserver la dimension temporelle, sans laquelle les régularités séquentielles ne pourraient être détectées.
- Application d'algorithmes spécialisés : GSP et PrefixSpan permettent d'extraire efficacement des séquences fréquentes à partir de bases volumineuses
- Segmentation et filtrage des séquences : certains clients présentant très peu de transactions ou des comportements atypiques sont écartés ou regroupés, afin de garantir que les motifs extraits représentent de véritables régularités globales et non des cas isolés.
- Analyse et interprétation métier : au-delà de l'aspect algorithmique, les motifs détectés sont replacés dans une perspective de prise de décision. L'expérimentation cherche ainsi à illustrer comment les résultats obtenus peuvent être utilisés pour

- Affiner les recommandations personnalisées, optimiser la gestion des stocks ou encore concevoir des campagnes marketing adaptées aux temporalités d'achat.

2. Description du dataset

L'expérimentation repose sur l'utilisation du jeu de données « Online Retail », publié sur Kaggle[i] et issu du référentiel UCI Machine Learning Repository. Ce dataset regroupe les transactions effectuées entre le 1er décembre 2010 et le 9 septembre 2011 par les clients d'un site de commerce en ligne basé au Royaume-Uni. Il s'agit d'un jeu de données réel et largement utilisé dans la littérature scientifique sur l'exploration de données transactionnelles.

2.1 Structure des données

Le dataset est fourni au format CSV (encodage UTF-8) et contient plusieurs colonnes principales :

- InvoiceNo : identifiant unique de la transaction (chaque facture regroupe un panier d'articles).
- StockCode : code unique attribué à chaque produit.
- Description : désignation textuelle de l'article.
- Quantity : quantité achetée pour un produit donné.
- InvoiceDate : date et heure de la transaction, indispensable pour la construction des séquences temporelles.
- UnitPrice : prix unitaire du produit.
- CustomerID : identifiant du client (permettant de regrouper les transactions par acheteur).
- Country : pays du client, la majorité étant situés au Royaume-Uni.

2.2 Pertinence du dataset

Ce jeu de données présente plusieurs atouts majeurs pour une étude sur les règles d'association séquentielles :

- Volumétrie : il couvre près de 500 000 transactions, ce qui permet d'évaluer la robustesse des algorithmes sur un volume représentatif.
- Diversité des produits : les articles appartiennent à des catégories variées (décoration, cuisine, fêtes, etc.), ce qui rend possible l'observation de comportements d'achat hétérogènes.

  [i]https://www.kaggle.com/datasets/carrie1/ecommerce-data?resource=download

- Dimension temporelle riche : grâce aux dates de transaction, il est possible de reconstituer les séquences chronologiques d'achats par client.

- Cas d'usage réaliste : les problématiques de recommandation, d'optimisation des stocks et de marketing ciblé sont directement applicables au commerce en ligne.

L'implémentation a été réalisée en Python 3.10 sous Visual Studio Code, en s'appuyant principalement sur les bibliothèques pandas, numpy, scikit-learn, mlxtend, seaborn, matplotlib.

3. Les différents paramètres

Il est important de rappeler et d'évaluer l'ensemble des paramètres manipulés lors du processus méthodologique.

3.1. Paramètres liés à la préparation des données

- Fenêtre temporelle (BasketDate) : la prise en compte de la dimension temporelle (date d'achat) a permis de conserver l'évolution des comportements clients dans le temps.
- Variables RFM (Récence, Fréquence, Montant) : ces paramètres sont essentiels pour caractériser les habitudes d'achat. Leur normalisation a été nécessaire afin d'éviter la domination d'une variable sur les autres.
- Standardisation : application d'un StandardScaler pour centrer et réduire les données, garantissant une comparaison homogène entre variables de nature différente.

3.2. Paramètres liés au clustering

- Nombre de clusters (k) : paramètre principal de KMeans, déterminé à l'aide de la méthode du coude (*Elbow method*) et du coefficient de silhouette.
- Méthodes alternatives de clustering :
    - Fuzzy C-Means : paramètre de degré de flou (m) et nombre de clusters k.
    - DBSCAN : paramètres eps (rayon de voisinage) et min_samples (nombre minimum de points).
    - Agglomerative Clustering : choix du linkage (ward, single) et du nombre de clusters.

Ces méthodes ont permis de comparer les performances et la stabilité des partitions avant de retenir KMeans.

3.3. Paramètres liés à la classification prédictive

Au-delà des paramètres liés au clustering, une étape de validation supervisée a mobilisé d'autres paramètres :

- SVM : choix du noyau (linéaire, RBF), paramètre de régularisation C, paramètre γ.
- KNN : nombre de voisins k, type de distance (euclidienne, Manhattan).
- Random Forest : nombre d'arbres (n_estimators), profondeur maximale.

- MLP (réseau de neurones) : nombre de couches cachées, fonction d'activation, nombre d'itérations.

Ces paramètres ont servi à renforcer la validation et la comparaison des approches. On cherche ici à anticiper les comportements futurs à partir de données historiques.

3.4 Paramètres des algorithmes explorés GSP et PrefixSpan

a.1 Paramètres de structuration des données :

- Les séquences sont construites par CustomerID, chaque client étant associé à une séquence ordonnée de produits (*ProdID*) et des dates correspondantes (*BasketDate*).
- L'ordre chronologique est garanti par le tri df.sort_values(by=['CustomerID', 'BasketDate']).
  Cela intègre la temporalité directement dans la structuration des séquences.

a.2 Paramètres liés au comptage des candidats :

- Génération initiale des candidats : à l'itération k=1, chaque produit distinct observé dans les séquences devient un candidat.
- Extension des candidats : à chaque itération suivante (k=2,3,…), de nouveaux candidats sont créés par combinaison des motifs fréquents précédents.
  C'est un paramètre implicite lié à la profondeur de l'algorithme GSP.

a.3 Paramètres de filtrage des motifs fréquents :

- Support minimal (min_support) : seuil de fréquence utilisé pour retenir les motifs fréquents
- Contrainte temporelle (max_time_diff) : limite maximale du nombre de jours entre le premier et le dernier élément d'un motif pour qu'il soit valide.

a.4 Paramètres de validation et de robustesse :

- Les dates sont systématiquement converties en format datetime pour assurer une comparaison fiable.
- Un contrôle est prévu pour détecter et signaler les séquences contenant des dates manquantes ou mal converties.

4. Résultats et discussions

4.1 GSP

L'exécution de l'algorithme GSP a permis d'identifier des motifs séquentiels fréquents de différentes tailles. Les résultats sont structurés selon la taille des motifs (valeur de *k*), ce qui permet d'analyser progressivement la complexité des associations.

a. Motifs fréquents de taille 1 (k = 1)

Les motifs de taille 1 correspondent à des produits individuels apparaissant de manière répétée dans les séquences. Parmi eux, on observe notamment :

- 85123A, 85099B, 21733, 22697, mais aussi des codes comme M, 22423, 22427.

Ces codes de produits font références a des articles bien identifié dans la base de données. 85099B par exemple correspond à JUMBO BAG RED RETROSPOT.

Interprétation :

- Ces produits représentent des articles phares du catalogue, c'est-à-dire des biens achetés par un grand nombre de clients.
- Leur fréquence en fait des produits stratégiques dans une perspective marketing, car ils peuvent servir de produits d'appel, bénéficier de promotions ciblées, ou être utilisés comme base pour construire des offres combinées.

b. Motifs fréquents de taille 2 (k = 2)

Les motifs de taille 2 révèlent des associations entre deux produits au sein des séquences. Parmi les motifs identifiés :

- ('21733', '85123A') : association directe entre deux produits phares.
- ('84380', '84378') : relation de coachât forte entre deux articles probablement complémentaires.
- ('22697', '22699') : produits achetés de manière conjointe.
- ('22469', '22427') et ('22469', '22470') : un même produit se retrouve associé à plusieurs autres, montrant son rôle de produit pivot dans les paniers.

Interprétation :

- Ces paires traduisent des comportements de co-achat régulier, qui constituent une base solide pour des stratégies de ventes croisées.
- La présence répétée d'associations récurrentes valide l'idée de proposer des packs promotionnels ou des recommandations de type « les clients qui ont acheté A achètent aussi B ».

c. Motifs fréquents de taille 3 (k = 3)

Un seul motif de longueur 3 a été identifié :

- ('22423', '22697', '22699')

Interprétation :

- Ce motif traduit un enchaînement ordonné de trois produits, ce qui reflète un comportement d'achat plus structuré.
- La séquence montre qu'un client ayant acheté 22423 est susceptible de compléter son panier avec 22697, puis avec 22699.

- Cela ouvre la voie à des applications en marketing prédictif : une fois que le client achète les deux premiers articles, il est pertinent de lui recommander le troisième de manière proactive.

Discussion globale

Les résultats obtenus montrent une progression intéressante :

- k = 1 met en évidence les produits dominants du catalogue.
- k = 2 révèle des associations fortes entre produits, utiles pour la vente croisée et les recommandations.
- k = 3, bien que moins riche, démontre la possibilité d'identifier des séquences ordonnées, offrant des opportunités pour la prédiction d'achats.

Ces résultats sont particulièrement intéressants pour envisager des stratégies marketing différenciées : promotion des best-sellers, vente croisée sur les associations fréquentes, et recommandations personnalisées basées sur des séquences récurrentes.

4.2 PrefixSpan

L'application de l'algorithme PrefixSpan sur les séquences de transactions issues de notre base de données a permis d'extraire différents motifs séquentiels fréquents :

- Support: 4, Motif: [('85099B',)]
- Support: 2, Motif: [('37449',)]
- Support: 8, Motif: [('M',)]
- Support: 3, Motif: [('85123A', '21733')]
- Support: 2, Motif: [('85123A', '21733'), ('21733', '85123A')]
- Support: 3, Motif: [('85123A',)]

Les motifs affichés sont composés des codes produits qui correspondent à des articles précis dans la base transactionnelle. Par exemple '85123A' correspond au produit WHITE HANGING HEART T-LIGHT HOLDER dans la base de données

Le support, en valeur absolu, indique combien de séquences contiennent ce motif. Par exemple le support 4 du motif '85099B' indique que le produit '85099B' apparaît dans 8 séquences clients.

Parmi les résultats observés, on retrouve notamment :

- Des motifs unitaires tels que ('85099B'), ('37449'), ou encore ('M'), correspondant à des produits isolés achetés de manière répétée par plusieurs clients.
- Des associations intra-panier comme ('85123A', '21733'), indiquant que certains articles sont régulièrement achetés ensemble dans la même transaction.
- Des séquences ordonnées, par exemple [('85123A', '21733'), ('21733', '85123A')], traduisant une récurrence dans les achats : des clients ayant acheté un couple de produits tendent à les racheter par la suite, parfois dans un ordre différent.

Interprétation des résultats

- Produits dominants
  Certains articles apparaissent de manière répétée avec un support non négligeable (ex. 85099B, 85123A). Cela reflète des produits phares du catalogue, qui constituent une base intéressante pour des actions de marketing ciblées telles que des promotions récurrentes ou des programmes de fidélisation.

- Complémentarité des achats
  L'apparition de motifs multi-produits (ex. ('85123A', '21733')) met en évidence des relations de co-achat. Ces informations sont stratégiques pour le cross-selling : proposer à un client un produit complémentaire à celui qu'il achète régulièrement.

- Habitudes de réachat
  Les séquences ordonnées, bien que rares, traduisent un comportement cyclique : certains clients reproduisent leurs achats sur plusieurs périodes. Ces motifs permettent d'anticiper les besoins et d'envisager des recommandations personnalisées ou des rappels marketing (ex. e-mails ciblés).

Discussion

Bien que les motifs extraits apportent des indications intéressantes, plusieurs limites apparaissent :

- Support faible : beaucoup de motifs ne concernent que 2 ou 3 séquences sur l'ensemble de la base. Cela réduit leur portée statistique et leur intérêt pratique pour des stratégies marketing généralisées.

- Absence de seuil de confiance : contrairement à GSP ou aux règles d'association classiques, PrefixSpan ne calcule pas la confiance d'une règle. Les motifs identifiés sont fréquents, mais leur pouvoir prédictif reste incertain.

5. Synthèse des résultats et discussions

L'application des algorithmes PrefixSpan et GSP a permis de dégager des motifs séquentiels fréquents de différentes natures. Le tableau suivant synthétise les principaux résultats obtenus pour chaque méthode.

Algorithme	Type de motifs trouvés	Exemples de motifs fréquents	Support / Fréquence observée
PrefixSpan	Produits unitaires (k=1)	('85099B'), ('37449'), ('M'), ('85123A')	Support faible (2 à 8 séquences)
	Paires d'articles (intra-panier)	('85123A', '21733')	Support = 3
	Séquences ordonnées (inter-panier)	[('85123A', '21733'), ('21733', '85123A')]	Support = 2
GSP	Produits unitaires (k=1)	85123A, 85099B, 21733, 22697, 22423…	Nombreux produits fréquents

	Paires séquentielles (k=2)	(21733 → 85123A), (84380 → 84378), (22697 → 22699)	Plusieurs paires récurrentes
	Séquences complexes (k=3)	(22423 → 22697 → 22699)	1 séquence trouvée

Tableau 15 : Comparaison des résultats

A la lumière de nos résultats, on note que les deux méthodes convergent sur certains résultats, notamment l'identification de produits phares (85123A, 85099B) et de relations d'achat fréquentes (21733 – 85123A). Toutefois, leurs apports diffèrent : PrefixSpan se concentre sur des motifs simples mais à faible support, ce qui le rend pertinent pour une exploration initiale des séquences, tandis que GSP propose une structuration progressive des résultats par taille de motif (k=1, 2, 3), fournissant une vision plus riche et directement exploitable pour la conception de stratégies marketing complètes (produits d'appel, ventes croisées, recommandations séquentielles). Nous retiendrons l'algorithme GSP.

Les résultats obtenus grâce à l'algorithme GSP confirment la pertinence et la richesse des informations extraites à partir des transactions clients. L'identification de produits fréquents, tels que 85123A, 85099B, 21733, ainsi que des paires et séquences récurrentes, reflète fidèlement les comportements d'achat des clients. Ces motifs permettent de visualiser des tendances claires dans les séquences de consommation et de comprendre la progression des achats dans le temps, ce qui constitue un élément central pour l'analyse séquentielle visée par ce mémoire.

L'extraction de motifs de différentes tailles (unitaires, paires séquentielles et séquences complexes) offre une vision hiérarchisée et structurée du comportement client, facilitant l'interprétation et l'exploitation pratique des résultats. Les motifs identifiés présentent un support suffisant pour être considérés comme fiables et significatifs, et permettent de tirer des conclusions utiles pour l'optimisation des ventes, le marketing ciblé ou les recommandations séquentielles.

Globalement, les résultats sont satisfaisants. Ils répondent aux objectifs de l'étude en fournissant des informations exploitables sur les produits phares, les associations récurrentes et les séquences d'achat. La richesse et la clarté des motifs extraits montrent que l'algorithme GSP est efficace dans le contexte de notre base de données et permet de produire des insights directement applicables pour l'analyse du comportement client et la prise de décision marketing.

Chapitre VII : CONCLUSION

Chapitre VII : Conclusion

Dans un contexte où les données clients constituent un levier stratégique majeur pour les entreprises, ce mémoire s'est attaché à explorer des approches avancées d'analyse comportementale à partir de données transactionnelles réelles. En combinant des techniques de préparation et de structuration des données, de segmentation, de classification prédictive et d'extraction de motifs séquentiels, l'objectif était de construire une vision à la fois statique et dynamique des comportements d'achat.

La première étape du travail a porté sur la structuration des données, en s'appuyant sur des indicateurs classiques tels que les variables RFM (Récence, Fréquence, Montant) et des métriques complémentaires, comme l'entropie ou la diversité d'achat. Sur cette base, une segmentation client a été réalisée à l'aide du K-Means, choisi pour sa simplicité, son interprétabilité et sa compatibilité avec les étapes suivantes de l'analyse.

La phase suivante a consisté à développer des modèles de classification supervisée afin de prédire l'appartenance des clients à des segments identifiés. Plusieurs modèles ont été évalués (SVC, MLP, Random Forest, k-NN), avec des performances très satisfaisantes, certains dépassant 98 % de précision, démontrant la pertinence et la puissance discriminante des variables sélectionnées ainsi que la robustesse du pipeline analytique.

Enfin, l'analyse séquentielle a permis d'aller au-delà d'une lecture statique des clients, en explorant la dimension temporelle de leurs comportements. L'algorithme GSP a permis d'extraire des motifs séquentiels significatifs, allant de produits unitaires à des séquences complexes, révélant des comportements récurrents et des patterns d'achat exploitables pour des stratégies de recommandation personnalisée, de fidélisation ou de promotion ciblée.

Au terme de ce travail, plusieurs enseignements peuvent être dégagés : d'une part, l'analyse des transactions clients peut générer une valeur opérationnelle concrète si elle combine segmentation, prédiction et séquentialité ; d'autre part, la prise en compte de la temporalité des comportements permet de mieux anticiper les logiques d'achat et de concevoir des stratégies marketing plus pertinentes.

Ce mémoire établit ainsi les fondements d'une exploitation intelligente des données transactionnelles, conciliant rigueur analytique, efficacité algorithmique et potentiel stratégique, et offre des perspectives pour enrichir les analyses futures grâce à l'intégration de nouvelles dimensions comportementales ou contextuelles.

Bibliographie

1. Han, J., Pei, J., & Kamber, M. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann.
2. Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., … & Zhou, Z.-H. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems, 14*(1), 1–37
3. Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *ACM SIGMOD Conference on Management of Data*, 207–216.
4. Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Data Bases*, 487–499.
5. Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *ACM SIGMOD Conference on Management of Data*, 1–12.
6. Shmueli, G., Bruce, P. C., Gedeck, P., & Patel, N. R. (2020). *Data Mining for Business Analytics: Concepts, Techniques, and Applications in Python* (3rd ed.). Wiley.
7. Zaki, M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning, 42*(1–2), 31–60.
8. Mooney, C. H., & Roddick, J. F. (2013). Sequential pattern mining – Approaches and algorithms. *ACM Computing Surveys, 45*(2), 1–39.
9. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. C. (2001). PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. *Proceedings of the 17th International Conference on Data Engineering*, 215–224.
10. Chen, Y.-L., Tang, K., Shen, R.-J., & Hu, Y.-H. (2005). Market basket analysis in a multiple store environment. *Decision Support Systems, 40*(2), 339–354.
11. Fournier-Viger, P., Lin, J. C.-W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., & Lam, H. T. (2017). The SPMF open-source data mining library: Mining sequential patterns, frequent itemsets, association rules and more.
12. Mabroukeh, N. R., & Ezeife, C. I. (2010). A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys*, 43(1), 1–41., car cette revue
13. Yin, H., Cui, B., Li, J., Chen, L., & Yao, J. (2012). Mining sequential patterns efficiently by prefix-projected pattern growth. *Information Systems*, 37(6), 583–595.
14. Ding, W., Lo, D., Han, J., & Khoo, S.-C. (2009). Efficient mining of frequent patterns from uncertain data. *IEEE Transactions on Knowledge and Data Engineering*, 21(12), 1697–1711.

15. Zheng, Y., Zhang, L., Xie, X., & Ma, W.-Y. (2009). Mining interesting locations and travel sequences from GPS trajectories. *Proceedings of the 18th International Conference on World Wide Web (WWW)*, 791–800.

16. Moens, S., Aksehirli, E., & Goethals, B. (2013). Frequent itemset mining for big data. *2013 IEEE International Conference on Big Data*, 111–118.
Zaki, M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1–2), 31–60.

17. Zaki, M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1–2), 31–60.
18. Lin, J. C.-W., Fournier-Viger, P., & Hong, T.-P. (2014). Mining time-interval sequential patterns in large databases. *Applied Intelligence*, 41(1), 144–160.
19. Ayres, J., Flannick, J., Gehrke, J., & Yiu, T. (2002). Sequential Pattern Mining Using a Bitmap Representation. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 429–435.