MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et de l'Informatique

Département de Mathématiques et Informatique

Filière : Informatique

MEMOIRE DE FIN D'ETUDES

Pour l'Obtention du Diplôme de Master en Informatique

Option : **Ingénierie des Systèmes d'Information**

Présenté par :

BOUHADEF Asmaa

BENYOUCEF Kawther

THEME :

Extraction de motifs séquentiels : application à la fouille des séries temporelles

Soutenu le : 27 juin 2021

Devant le jury composé de :

KHELIFA Noureddine	MAA	Université de Mostaganem	Président
HAMAMI Dalila	MCB	Université de Mostaganem	Examinateur
BENAMEUR Abdelkader	MAA	Université de Mostaganem	Encadreur

Année Universitaire 2020-2021

Résumé

En data mining, l'extraction de motifs consiste à trouver les items qui apparaissent souvent ensemble dans les données. Lorsque ces données sont ordonnées selon une relation d'ordre, le temps par exemple, il s'agit d'extraction de motifs séquentiels. Une grande partie des données stockées par les entreprises, les administrations ou autres se prêtent à l'extraction de tels motifs. L'objectif de ce travail est d'explorer les techniques récentes utilisées dans ce domaine, notamment lorsqu'il s'agit de séries temporelles. Ces techniques trouvent leur application dans divers domaines, telles les habitudes de consommation, le comportement des internautes, ou encore la prévention des pannes.

Mots-clés :

Data mining, extraction de motifs séquentiels, séries temporelles.

Abstract

In data mining, pattern discovery consists of finding items that appear together in the data. When the data is ordered according to an ordering relation, time for example, it's sequentials patterns discovery. A big part of data hosted by enterprises, administrations, and others lends to the extraction of such patterns. The aim of this work is to explore recent techniques used in this field, especially when it's about time series. This techniques finds their applications in various fields, like consumption habits, the behavior of Internet users, or even the prevention of failures.

Keywords :

Data mining, sequentials patterns discovery, time series.

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	Expression Complète	
ECD	Extraction de Connaissances à partir de Données	
KDD	Knowledge Discovery in Databases	
PNR	Passenger Name Record	
FP-Tree	Frequent Pattern Tree	
FP-Growth	Frequent Pattern Growth	

Table des matières

Introduction générale

Contexte :

Le data mining, ou fouille de données en français, est un domaine en plein essor dans le monde de la recherche et de l'entreprise, et intéresse des secteurs nombreux et variés. En effet, le monde baigne dans un flot d'informations statistiques (résultats économiques, sondages, prévisions sur le climat, lapopulation, les ressources, etc.) dont il ne voit que l'écume sans en soupçonner les lames de fond. En amont de ces résultats, se trouve une production continue de données que l'on compare à un torrent ou de plus en plus à un déluge qu'il importe de maîtriser. L'extraction de motifs séquentiels dans les bases de données est l'un des problèmes lesplus intéressants du data mining, et qui apparaît fréquemment dans de nombreuses applications de la vie réelle.

D'un autre côté, et parmi les types de données les plus importants que les chercheurs de ce domaine se sont intéressés et ont consacré leur temps et leurs efforts à les comprendre et à les analyser, on trouve les données temporelles, bien connues sous le nom de séries temporelles ou celui de séries chronologiques. Il s'agit d'ensembles de valeurs numériques observées, relatives à un phénomène qui évolue dans le temps. Elles peuvent s'agir de données macro-économiques telles que le PIB d'un pays, l'inflation, ou les exportations, micro- économiques comme les ventes d'une entreprise ou le revenu d'un individu, financières comme le cours d'une action, politiques comme le nombre de votants ou de voix reçues par un candidat, ou enfin, démographiques comme la taille moyenne des habitants ou leurs âges.

Problématique :

Dans ce travail, nous nous intéressons plus précisément aux algorithmes d'extraction de *motifs séquentiels* (des motifs temporels) pour la découverte d'enchaînements fréquents dans les BDD avec des contraintes temporelles, et l'identification des événements d'individus afin de pouvoir suivre leurs comportements

séquentiels au cours du temps. Dans ce problème d'extraction de règles d'association séquentielles, l'extraction des connaissances se fait en recherchant des relations d'ordre, sous forme d'enchaînements appelés séquences entre *Items* (objets, attributs) ou ensembles d'*Items*. L'objectif est alors de trouver dans cette base de données, toutes les séquences d'*Items* apparaissant avec une certaine certitude selon une mesure d'intérêt choisie par l'utilisateur, par exemple, la fréquence. Il s'agit donc, de construire l'ensemble de tous ces motifs intéressants appelés *motifs séquentiels*.

Objectif :

L'objectif de ce projet est la compréhension du comportement des principaux algorithmes d'extraction de *motifs séquentiels* en expliquant et illustrant leur fonctionnement. Notre travail consiste dans un premier temps à étudier et à comprendre le fonctionnement des algorithmes d'extraction de *motifs séquentiels*, et dans un deuxième temps, à essayer d'évaluer et de comparer les performances de ces algorithmes en fonction de différents paramètres.

Organisation du rapport :

Le présent mémoire est organisé comme suit :

Dans le premier chapitre, nous allons porter notre attention sur le domaine de la fouille de données et définir la tâche de classification et son principe, et nous nous approfondirons dans la définition des séries temporelles et les domaines d'application.

Le deuxième chapitre sera consacré à la recherche des règles d'associations et le fonctionnement de l'algorithme *Apriori* en premier lieu, et *FP-Growth* en deuxième lieu. Le premier étant le pionnier du domaine, le deuxième l'un des plus utilisés.

Le troisième chapitre contiendra les détails de conception et d'implémentation de note outil, avec le déroulement d'un exemple illustratif. Il s'agira des détails de l'approche adoptée, et des explications de chacune de ses étapes.

Enfin, nous terminerons par une conclusion générale concernant tous le travail tout en exposant certaines perspectives.

Chapitre I :

Le data mining

1.1. Introduction

Ce premier chapitre définit le data mining, et décrit ses principales applications et ses apports aux différents domaines de la vie. Il situe en outre le data mining par rapport à la statistique, qui lui fournit bon nombre de ses méthodes et concepts théoriques, et par rapport à l'informatique, qui lui fournit sa matière première (les données), ses moyens de calcul et son vecteur de communication (la restitution des résultats) vers les autres applications informatiques et vers les utilisateurs.

1.2. Qu'est ce que le data mining ?

Le data mining est le processus de découverte significative de nouvelles corrélations, tendances et caractéristiques dans de grandes quantités de données stockées dans des entrepôts, en utilisant les technologies de reconnaissance de formes, des statistiques, et les techniques mathématiques. Le data mining peut aussi être défini comme l'analyse de données d'observation fixes afin de trouver des relations insoupçonnées et représenter ces données de façon originale et compréhensible pour le preneur de décision [5] [6] [3].

Il peut également être définit comme un « domaine interdisciplinaire qui utilise des techniques d'apprentissage automatique, de reconnaissance des formes, des bases de données, des statistiques et de visualisation pour l'extraction d'informations à partir de bases de données volumineuses ». Il sert à trouver des structures originales et des corrélations informelles entre les données. Il permet de mieux comprendre les liens entre des phénomènes en apparence distincts et d'anticiper des tendances encore peu discernables [2] [9].

1.3. Etapes du processus d'ECD

L'Extraction de Connaissances à partir des Données (ECD) se définit comme « l'acquisition de connaissances nouvelles, intelligibles et potentiellement utiles à partir de faits cachés au sein de grandes quantités de données » [5]. En fait, on cherche surtout à isoler des traits structuraux (patterns) qui soient valides, non triviaux, nouveaux, utilisables et si possible compréhensibles ou explicables.

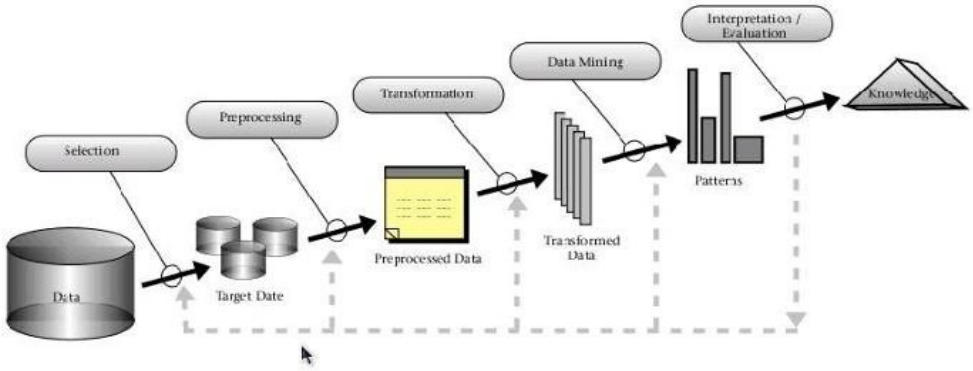


Figure 1 - *Les étapes d'ECD [5]*.

De la même façon que l'on a plusieurs définitions pour la fouille de données selon les domaines d'applications et la vision de chacun des domaines, les étapes du processus d'ECD (figure 1.1) varient d'un domaine à l'autre, mais tous ces domaines ont en commun ces étapes [5] :

- **la sélection des données :** la sélection d'attributs ou la sélection d'instances. Cette première phase est cruciale car du choix des descripteurs et de la connaissance précise va dépendre la mise au point des modèles de prédiction. L'information nécessaire à la construction d'un bon modèle de prévision peut être disponible dans les données mais un choix inapproprié de variables ou d'échantillons d'apprentissage peut faire échouer l'opération.
- **le prétraitement des données :** les données sélectionnées doivent être « préparées ». Avant tout, elles doivent être nettoyées puisqu'elles peuvent contenir plusieurs types d'anomalies : des données peuvent être omises à cause des erreurs de frappe ou à causes des erreurs dues au système lui-même, dans ce cas il faut remplacer ces données ou éliminer complètement leurs enregistrements. Des données peuvent être incohérentes, c.-à-d. qui sortent des intervalles permis, on doit les écarter où les normaliser.
- **la transformation des données :** parfois on est obligé à faire des transformations sur les données pour unifier leur poids ou la définition des structures optimales de représentation des données.
- **la fouille de données :** constitue véritablement le cœur du processus d'ECD, elle est souvent coûteuse et difficile à mettre en œuvre. Ici, il est fait le choix des techniques et algorithmes appropriés correspondants aux tâches à effectuer.

Il faudra faire des compromis selon les besoins dégagés et les caractéristiques communes des outils, car il n'existe pas de meilleure technique de fouille. A tout jeu de données et à tout problème correspond une ou plusieurs méthodes, le choix se fera en fonction de la tâche à résoudre, de la nature des données ou encore de l'environnement de l'entreprise. De plus, il est souhaitable de mettre en œuvre divers techniques afin de les comparer et d'en retenir une ou plusieurs combinées.

- **l'interprétation et l'évaluation des modèles :** généralement, l'objectif du data mining est d'aider à la prise de décision en fournissant des modèles compréhensibles aux utilisateurs. En effet, les utilisateurs ne demandent pas des pages de chiffres mais des interprétations des modèles obtenus. Les expériences montrent que les modèles simples sont plus compréhensibles mais moins précis, alors que ceux complexes sont plus précis mais difficiles a interpréter

1.4. Les tâches de la fouille de données

Globalement, les tâches de la fouille de données peuvent être divisées en tâches descriptives (description des propriétés des données) et tâches prédictives (déductions sur les données pour faire des prédictions) [5] [6] :

- **La caractérisation**

Par la caractérisation des données, on fait un résumé de toutes les caractéristiques générales des classes à étudier et on produit des règles de caractérisation.

- **La discrimination**

En comparant les caractéristiques générales de deux classes (classe cible et classe de contraste), la discrimination obtient des règles discriminatoires. Les techniques sont similaires avec celles de la caractérisation, sauf que la discrimination utilise en plus des mesures comparatives.

- **L'analyse d'associations**

Le but de l'analyse d'association est la découverte des règles d'association; basée sur la fréquence des items (le support) ou la probabilité conditionnée qu'un item apparait quand un autre apparait (la confiance), elle cherche à identifier les ensembles d'items fréquents. Notons également la recherche d'enchainements fréquents, qui utilise des principes similaires et qui est la base de l'extraction de motifs séquentiels.

- **La classification**

C'est la tâche la plus commune de fouille de données qui semble être une tâche humaine primordiale. Afin de comprendre notre vie, nous sommes constamment obligés à classer, catégoriser et évaluer. La classification consiste à étudier les caractéristiques d'un nouvel objet pour l'attribuer à une classe prédéfinie. Les objets à classifier sont généralement des enregistrements d'une base de données, la classification consiste à mettre à jours chaque enregistrement en déterminant la valeur d'un champ de classe.

Le fonctionnement de la classification se décompose en deux phases. La première étant la phase d'apprentissage. Dans cette phase, les approches de classification utilisent un jeu d'apprentissage dans le quel tous les objets sont déjà associés aux classes de références connues. L'algorithme de classification apprend du jeu d'apprentissage et construit un modèle. La seconde phase est la phase de classification proprement dite, dans laquelle le modèle appris est employé pour classifier de nouveaux objets.

- **La prédiction**

Très attractive pour les affaires (ou pour le domaine commercial), elle suppose de prédire des valeurs inconnues ou manquantes.

- **Le clustering**

Similaire à la classification, le but du clustering est d'organiser les données dans des classes; par contre, on n'a aucune information préalable et le clustering tout seul doit découvrir les classes. Le clustering est également nommé classification non-supervisée, parce que la classification n'est pas dictée par un modèle (des étiquettes). Le principe général de répartition des éléments à des classes repose sur une minimisation des distances intra-classe et sur une maximisation des distances interclasses.

- **L'analyse des anomalies**

Appelées aussi exceptions ou surprises, les anomalies représentent des déments qui ont un comportement différent du comportement général, ou autrement dit, des déviations de la tendance générale. Leurs significations sont multiples, variant du non-important (bruit) au très important (détection des fraudes).

- **L'analyse de l'évolution et de la déviation**

Ceci s'applique aux données temporelles. L'analyse de l'évolution cherche à repérer des tendances et consiste à caractériser, comparer, classifier les données ou à grouper les

données dans des clusters. L'analyse de la déviation mesure les différences entre les valeurs attendues et celles obtenues et s'interroge sur les causes des déviations.

1.5. Applications de la fouille de données

Il existe un nombre croissant d'applications réussies dans un large éventail de domaines aussidivers que [2] [9] :

- Analyse de l'imagerie satellite ;
- Bio-informatique ;
- Détection de fraude par carte de crédit ;
- Enquête criminelle ;
- Gestion de la relation client ;
- Prédiction de charge électrique ;
- Prévisions financières ;
- Diagnostic médical ;
- Prédiction de la part d'audience de la télévision ;
- Conception de produits ;
- Evaluation immobilière ;
- Marketing ciblé ;
- Synthèse de texte ;
- Optimisation de la centrale thermique ;
- Prévisions météorologiques ;
- Analyse du panier de marché ;

1.6. Classification des méthodes de data mining

Les principales méthodes de data mining et d'analyse de données se répartissent en deux grandes familles : les méthodes descriptives et les méthodes prédictives. Dans les méthodes descriptives, qui réduisent, résument, synthétisent les données, il n'y a pas de variable à expliquer, à prédire : pas de variable privilégiée. Dans les méthodes prédictives, qui expliquent les données, il y a une variable cible, une variable à expliquer : variable privilégiée [2] [9].

Nous affinons cette classification dans le Tableau suivant, ou apparaissent grisées les méthodes ressortissant à la statistique et l'analyse des données traditionnelles.

Type	Famille	Sous-famille	Algorithme
méthodes descriptives	modèles géométriques	analyse factorielle (projection et visualisation dans un espace de dimension inférieure)	analyse en composantes principales ACP (variables continues)
			analyse factorielle des correspondances AFC (variables qualitatives et binaires)
			analyse des correspondances multiples ACM (variables qualitatives et binaires)
		analyse typologique (regroupements dans tout l'espace en classes homogènes)	méthodes de partitionnement (centres mobiles, *k-means*, nuées dynamiques, *k-medoids*...)
			méthodes hiérarchiques (ascendantes, descendantes)
		analyse typologique + réduction de dimension	classification neuronale (réseaux de Kohonen)
	modèles combinatoires		classification par agrégation des similarités (variables qualitatives)
	modèles à base de règles logiques	détection de liens	recherche d'associations
			recherche de séquences similaires
méthodes prédictives	modèles à base de règles logiques	arbres de décision	arbres de décision (variable à expliquer continue ou qualitative)
	modèles à base de fonctions mathématiques	réseaux de neurones	réseaux à apprentissage supervisé (perceptron, réseau à fonction radiale de base...)
		modèles paramétriques ou semi-paramétriques	régression linéaire, ANOVA, MANOVA, ANCOVA, MANCOVA, modèle linéaire général GLM, régression PLS (variable à expliquer continue)
			analyse discriminante de Fisher, régression logistique, régression logistique PLS (variable à expliquer qualitative)
			modèle log-linéaire (variable à expliquer = comptage = nombre d'individus prenant une combinaison donnée de modalités de variables qualitatives)
			modèle linéaire généralisé GLZ, modèle additif généralisé GAM (variable à expliquer continue, discrète, comptage ou qualitative)
	prédiction sans modèle	analyse probabiliste	*k*-plus proches voisins (*k*-NN)

Tableau 1 - *Classification des méthodes de data mining*

1.6.1. Comparatif des méthodes

Nous récapitulons dans le Tableau 2 les avantages et inconvénients des méthodes les plus courantes d'exploration de données au vu des trois qualités essentielles attendues : l'absence d'hypothèses les plus restrictives ; la capacité de traiter toujours exhaustivement les données en un temps raisonnable, la possibilité de manier des données lacunaires et de types hétérogènes, numériques ou non.

Techniques	Absence d'hypothèses sur le problème à résoudre	Traitement exhaustif des bases de données	Traitement des données hétérogènes ou lacunaires
Classification			
méthode des centres mobiles et ses variantes	non (nombre de classes et centres initiaux fixés)	oui	variables numériques et sans valeurs manquantes
classification hiérarchique	oui, mais les classes au niveau *n* sont déterminées par ceux au niveau *n*–1	non (algorithme non linéaire), on ne peut traiter plus de quelques milliers d'observations	oui (possibilité de traiter des variables non numériques avec une distance *ad hoc*)
classification neuronale (Kohonen)	non (nombre de classes fixé)	oui	les variables $\notin [0,1]$ doivent être transformées
classification par agrégation des similarités	oui	dépend de l'implémentation	variables qualitatives
Classement et prédiction			
arbres de décision	comme la classification hiérarchique (sorte « d'arbre à l'envers »)	non (mais moins vite limité que la classification hiérarchique)	certains arbres comme CHAID doivent discrétiser les variables continues
réseaux de neurones perceptrons	oui (mais il faut fixer le nombre de neurones cachés)	non (pas d'apprentissage sur plusieurs centaines de variables)	les variables $\notin [0,1]$ doivent être transformées
réseaux à fonction radiale de base	comme les perceptrons	oui	les variables $\notin [0,1]$ doivent être transformées
analyse discriminante	non (hypothèses sur les lois conditionnelles X_i/Y)	oui	variables numériques et sans valeurs manquantes
analyse discriminante sur coordonnées factorielles d'une ACM (méthode DISQUAL)	oui (permet de s'affranchir largement des hypothèses sur les lois conditionnelles X_i/Y)	oui	oui (les valeurs manquantes sont traitées comme des valeurs à part entière)
régression linéaire	non (linéarité en x de $E(Y\|X=x)$ + hypothèses sur les résidus)	oui	variables numériques et sans valeurs manquantes
régression logistique, modèle linéaire généralisé	non (linéarité en x de $g(E(Y\|X=x))$ + non séparation complète)	oui (sur une machine assez puissante, si le nombre d'observations est très grand)	oui (découper en classes les variables continues avec des valeurs manquantes)
Associations			
détection d'associations	oui	dépend du paramétrage	oui
séquences similaires	oui	oui (idem)	oui

Tableau 2 - *Comparatif des méthodes de data mining*

1.6.2. Exemple 1 : les réseaux de neurones

Il est difficile de parler du data mining sans parler des réseaux de neurones, qui sont à la base à la fois de certaines techniques descriptives et de certaines techniques prédictives. Ils se sont largement étendus grâce à leur puissance de modélisation (ils peuvent approcher n'importe quelle fonction suffisamment régulière). Cependant, leur utilisation est parfois freinée par les difficultés qu'elle présente : l'aspect de la « boîte noire », la délicatesse des réglages à effectuer, la puissance informatique requise et surtout les sur-risques une solution globale non optimale.

Un réseau de neurones (ou réseau neuronal) a une architecture calquée sur celle du cerveau, organisé en neurones et synapses, et se présente comme un ensemble de nœuds (ou neurones formels, ou unités) d'un premier niveau, appelé couche d'entrée, et chaque modalité d'une variable qualitative correspondant également à un nœud de la couche d'entrée. Le cas échéant, lorsque le réseau est utilisé dans une technique prédictive, il y a une ou plusieurs variables à expliquer, elles correspondent alors chacune à un nœud (ou plusieurs) de la couche de sortie.

Entre la couche d'entrée et la couche de sortie sont parfois connectés des nœuds appartenant à un niveau intermédiaire : la couche cachée. Un nœud reçoit des valeurs en entrée et renvoie 0 à n valeurs en sortie. Une fonction de combinaison calcule une première valeur à partir des nœuds connectés en entrée et du poids des connexions. Ainsi, dans les réseaux les plus courants, les perceptrons, il s'agit de la somme pondérée des valeurs des nœuds en entrée. Enfin, pour déterminer une valeur en sortie, une seconde fonction, appelée fonction de transfert (ou d'activation), est appliquée à cette valeur. Les nœuds de la couche d'entrée sont triviaux, dans la mesure où ils ne combinent rien, et ne font que transmettre la valeur de la variable qui leur correspond.

Un nœud de perceptron se présente donc comme suit :

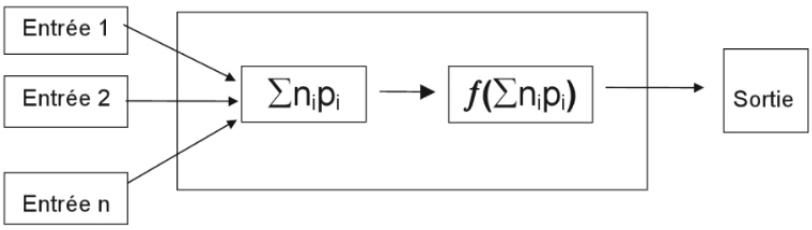


Figure 2 - *Nœud d'un réseau de neurones*

1.6.3. Exemple 2 : les arbres de décision

La technique des arbres de décision est l'une des plus intuitives et des plus populaires du data mining, d'autant plus qu'elle fournit des règles explicites de classement et supporte bien les données hétérogènes, manquantes et les effets non linéaires. Pour les applications pertinentes du marketing, actuellement la seule grande concurrente des arbres de décision est la régression logistique. Remarquons que les arbres de décision sont à la frontière entre les méthodes prédictives et descriptives, puisque leur classement s'opère en segmentant la population à laquelle ils s'appliquent.

Principe d'un arbre de décision :

La technique d'arbre de décision est employée en classement pour détecter des critères permettant de répartir les individus d'une population en n classes (souvent n=2) prédéfinies. On commence par choisir la variable qui, par ses modalités, sépare le mieux les individus de chaque classe, de façon à avoir des sous-populations, que l'on appelle nœuds, contenant chacune la plus forte proportion possible d'individus d'une seule classe, puis on réitère la même opération sur chaque nouveau nœud obtenu jusqu'à ce que la séparation des individus ne soit plus possible ou plus souhaitable (au vu de certains critères dépendants du type d'arbre).

Par construction, les nœuds terminaux (les feuilles) sont tous majoritairement constitués d'individus d'une seule classe. Un individu est affecté à une feuille, et donc à une certaine classe avec une assez forte probabilité, quand il satisfait l'ensemble des règles permettant d'arriver à cette feuille. L'ensemble des règles de toutes les feuilles constitue le modèle de classement (Figure 3).

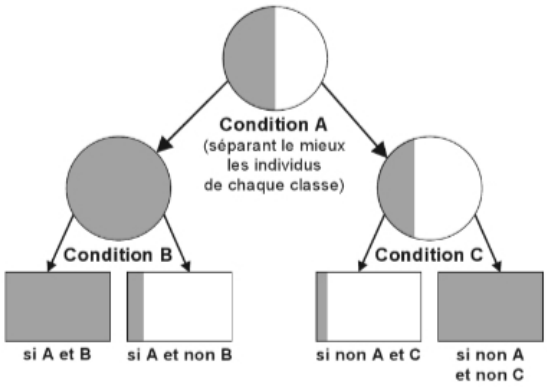


Figure 3 – *Exemple d'un arbre de décision*

1.6.4. Exemple 3 : la méthode des centres mobiles

Cette méthode connaît des variantes comme la méthode des k- moyens et la méthode des nuées dynamiques, a le déroulement suivant:

- à l'étape 1 : on choisit *k* individus comme centres initiaux des classes (on tire au sort, ou l'on prend les premiers, ou l'on en prend *1* sur *n/k*).
- à l'étape 2 : on calcule les distances entre chaque individu et chaque centre *c*, de l'étape précédente, et on affecte chaque individu au centre le plus proche, ce qui définit *k* classes.

- à l'étape 3 : on remplace les *k* centres *c*; par les centres des *k* classes définies à l'étape 2 (ces centres ne sont pas forcément des individus de la population).
- à l'étape 4 : on regarde si les centres sont restés suffisamment stables (en comparant leur déplacement aux distances entre centres initiaux) ou si un nombre fixé d'itérations a été atteint :
    - si oui, on arrête (en général, après au moins une dizaine d'itérations).
    - si non, on retourne à l'étape 2.

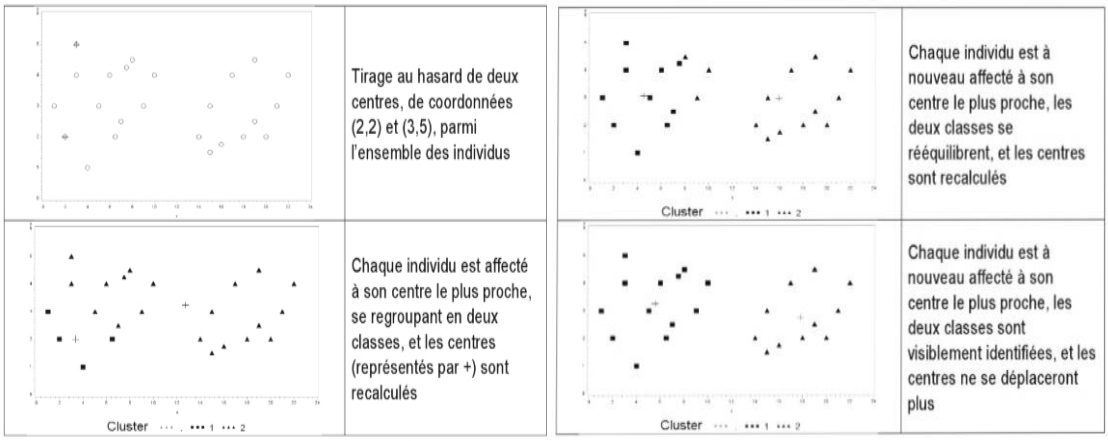


Figure 4 : *La méthode des centres mobiles*

1.7. Les séries temporelles

Les séries temporelles sont une manière structurée de représenter des données qui ont une dimension temporelle. Une série temporelle est une liste de dates, dont chacune est associée à une valeur (un nombre). Visuellement, il s'agit d'une courbe qui évolue au fil du temps. Ce sont des donnés mesurées à des intervalles de temps régulier. On peut les voir comme des suites d'observations répétées d'un même phénomène à des dates différentes (la température moyenne journalière en un lieu donné, la consommation moyenne en électricité chaque mois, le prix du baril de pétrole chaque jour etc.).

On représente habituellement une série temporelle (x_t) tel que $\{1<t<T\}$ (t : le numéro de l'observation) à l'aide d'un graphique avec en abscisse les dates et en ordonnée les valeurs observées [8] [1]. L'étude des séries temporelles correspond à l'analyse statistique d'observation régulièrement espacée dans le temps. Cette étude est appliquée de nos jours dans des domaines aussi variées que l'astronomie, météorologie, médecine ou économie. L'objectif de l'étude de ces séries est de décrire, modéliser,

expliquer puis prévoir les phénomènes dans le futur. Si la série est stable autour de sa moyenne donc elle est stationnaire, sinon non stationnaire, et si un phénomène se reproduit à des périodes régulières, donc ce phénomène est saisonnier.

L'analyse des séries temporelles a beaucoup évolué dans le domaine des statistiques et peut être utilisée, par exemple, pour prévoir l'offre et la demande futures d'un service ou d'une marchandise. L'analyse de ces séries repose sur le suivi du phénomène (ou variable) sur une période donnée (plusieurs années, par exemple), puis prédit l'avenir en fonction de ces observations et de la croissance de celles-ci.

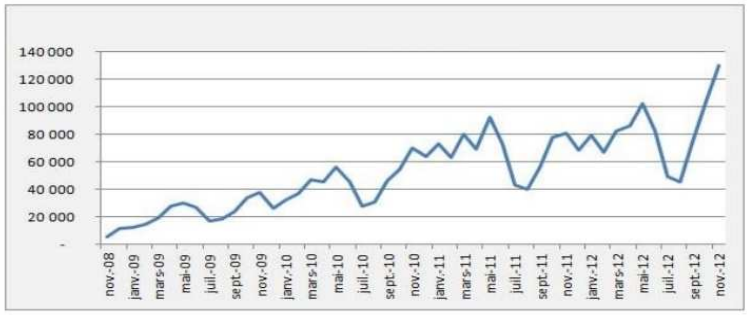


Figure 5 – *Exemple de série temporelle*

1.7.1. Domaines d'application

- **Finance et économétrie :** évolution des indices boursiers, des prix, des données économiques des entreprises, des ventes et achats de biens, des productions agricoles ou industrielles [2].
- **Médecine et Biologie :** suivie des évolutions des pathologies, analyse d'électro-encéphalogrammes et d'électrocardiogrammes, suivie de maladies cancéreuses.
- **Traitement de données :** mesures successives de position ou de direction d'un objet mobile (trajectographie) [2].

On trouve des exemples de séries temporelles dans de très nombreux domaines. La liste précédente n'est qu'un échantillon. Par contre, l'analyse de séries temporelles est utilisée pour de nombreuses applications telles que :

- Prévisions économiques.
- Prévisions de vente.
- Analyse budgétaire.
- Analyse du marché boursier.

- Projections de rendement.
- Processus et contrôle de qualité.
- Études d'inventaire.
- Projections de charge de travail.
- Études d'utilité.
- Analyse du recensement.
- Etc.

1.8. Conclusion

Dans ce chapitre, nous avons présenté dans un premier temps l'importance du data mining et son rôle dans différents domaines ainsi que ses objectifs d'extraction de connaissances à partir des données. Ensuite, nous avons présenté les différentes méthodes de fouille de données. Enfin, nous avons introduit les séries temporelles à travers plusieurs exemples applicatifs. Le prochain chapitre introduira l'extraction des motifs séquentiels en particulier.

Chapitre II : La recherche des motifs séquentiels

2.1. Introduction

L'extraction des règles d'association est une technique descriptive, l'une des plus populaires du data mining, surtout dans des secteurs comme le web mining, où elle permet d'analyser les pages parcourues par un internaute, et dans la grande distribution où elle permet d'analyser les produits simultanément achetés par un client. Ceci explique les surnoms parfois donnés à cette technique : analyse du panier de la ménagère « *market basket analysés* ». Elle ne pose pas les mêmes difficultés théoriques que les techniques de classification et de classement, les difficultés étant plutôt de traiter efficacement d'énormes volumes de données (parfois des millions de tickets de caisse) et d'y distinguer des associations nouvelles et intéressantes au sein d'une énorme majorité d'associations inintéressantes ou déjà connues [6].

2.2. Principe d'extraction des règles d'association

Rechercher des règles d'association consiste à rechercher les règles du type : « Si pour un individu, la variable $A = X_A$, la variable $B = X_B$, etc., alors, dans 80 % des cas, la variable $Z = X_z$, cette configuration se rencontrant pour 20 % des individus. ». Autrement dit, on recherche les valeurs conjointes les plus fréquentes d'un ensemble de variables d'une base [2] [9]. À noter que certaines recherches récentes portent sur les règles « négatives » dans lesquelles on s'intéresse aux produits non achetés. La valeur de 80 % est appelée indice de confiance et la valeur de 20 % est appelée indice de support de la règle $\{A = X_A, B = X_B,..\} = \{Z = X_z\}$ [6].

La première partie de la règle est appelée « antécédent » ou « condition », la seconde « conséquent » ou « résultat », les expressions de la forme $\{A = X_A\}$ sont appelées « *items* ». Dans une règle, un *item* n'est jamais à la fois dans la condition et le résultat. Une règleest donc une expression de la forme : *Si Condition Alors Résultat*.

2.3. L'algorithme *Apriori*

L'algorithme *Apriori* [6] a montré comment des règles d'association pouvaient être générées dansun délai réaliste, du moins pour des bases de données relativement petites. Depuis lors, beaucoup d'efforts ont été consacrés à la recherche d'améliorations sur l'algorithme de base pour permettre le traitement de bases de données de plus en plus volumineuses.

L'algorithme repose sur le résultat très important suivant [3] [10] :

Théorème 1 :

Si un ensemble d'items est pris en charge, tous ses sous-ensembles (non vides) sont également pris en charge. Preuve, la suppression d'un ou plusieurs item d'un ensemble d'items ne peut pas réduire et augmentera souvent le nombre de transactions auxquelles il correspond. Par conséquent, la prise en charge d'un sous-ensemble d'un ensemble d'items doit être au moins aussi élevée que celle de l'ensemble d'items d'origine. Il s'ensuit que tout sous- ensemble (non vide) d'un ensemble d'items pris en charge doit également être pris en charge.

Théorème 2 :

Si $L_k = \emptyset$ (l'ensemble vide) alors L_{k+1}, L_{k+2} etc. doivent également être vides. Preuve, si des ensembles d'items pris en charge de cardinalité $k+1$ ou plus existent, ils auront des sous- ensembles de cardinalité k et il découle du théorème 1 que tous ceux-ci doivent être pris en charge. Cependant, nous savons qu'il n'y a pas d'itemsets pris en charge de cardinalité k car L_k est vide. Par conséquent, il n'y a pas de sous-ensembles pris en charge de cardinalité $k+1$ ou plus et L_{k+1}, L_{k+2} etc. doivent tous être vides.

```
Create L_1 = set of supported itemsets of cardinality one
Set k to 2
while (L_{k-1} ≠ ∅) {
    Create C_k from L_{k-1}
    Prune all the itemsets in C_k that are not
        supported, to create L_k
    Increase k by 1
}
The set of all supported itemsets with at least two members is L_2 ∪ ··· ∪ L_{k-2}
```

Figure 6 - *L'algorithme Apriori.*

Nous avons besoin d'une méthode pour passer successivement de chaque itemset L_{k-1} au prochain L_k. Nous pouvons le faire en deux étapes. Nous utilisons d'abord L_{k-1} pour former un ensemble candidat C_k contenant des itemsets de cardinalité k. C_k doit être construit de telle manière qu'il est certain d'inclure tous les jeux d'items pris en charge de cardinalité k mais peut contenir d'autres jeux d'items qui ne sont pas pris en charge. Ensuite, nous devons générer L_k en tant que sous-ensemble de C_k.

Nous pouvons généralement rejeter certains des membres de C_k comme membres possibles de L_k en inspectant les membres de L_{k-1}. Le reste doit être comparé aux transactions de la base de données pour établir leurs valeurs de prise en charge. Seuls les ensembles avec un support supérieur ou égal à *minsup* sont copiés de C_k vers L_k.

Pour démarrer le processus, nous construisons C_1, l'ensemble de tous les itemsets comprenant un seul item, puis faisons un passage dans la base de données en comptant le nombre de transactions qui correspondent à chacun de ces itemsets. La division de chacun de ces décomptes par le nombre de transactions dans la base de données donne la valeur de *support* de chaque ensemble d'éléments à un seul élément. Nous rejetons tous ceux avec *support* < *minsup* pour donner L_1. Le processus impliqué peut être représenté schématiquement à la figure 7, en continuant jusqu'à ce que L_k soit vide [10].

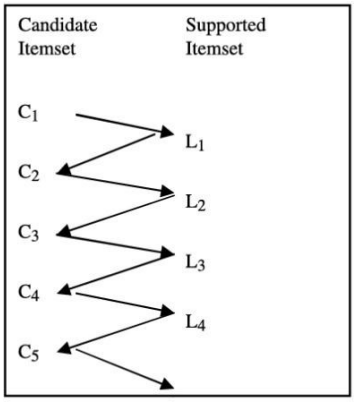


Figure 7 - *Schéma illustrant l'algorithme Apriori*

2.4. Arbres à motifs fréquents

2.4.1. L'algorithme FP-Growth

On suppose que nous disposons d'une base de données de transactions, chacune comprenant un certain nombre d'éléments, tels que *le lait, le poisson, le fromage, les œufs, le lait, la viande, le fromage au beurre, la crème, le pain*. Chaque enregistrement correspond à un achat dans un supermarché. Une collection d'articles, tels que *{poisson, viande, crème}* est appelée transaction telle que les achats d'une personne dans un ensemble d'articles. Le nombre de support (ou simplement le nombre) d'un ensemble d'éléments est le nombre de fois que les éléments se produisent ensemble dans une transaction, éventuellement avec d'autres éléments [10].

Le support d'un *itemset* est défini comme la valeur du nombre de support divisé par le nombre de transactions dans la base de données. L'objectif est de trouver des règles d'association liant les articles dans les achats entre eux, par ex : *œufs, lait → pain, fromage, viande*. Ce qui signifie que les transactions contenant des *œufs* et du *lait* comprennent généralement aussi le *pain*, le *fromage* et la *viande*.

Nous faisons cela en deux étapes [10] :

1. Trouver des *itemsets* tels que *{oeufs, lait, pain}* avec une valeur de support suffisammentélevée (définie par l'utilisateur).
2. Pour chacun de ces *itemsets*, extraire une ou plusieurs règles d'association, tous les éléments de l'ensemble d'éléments apparaissant à gauche ou à droite.

Propriété de fermeture vers le bas des itemsets :

Si un *itemsets* est fréquent, tout sous-ensemble (non vide) de celui-ci est également fréquent. Ceci est généralement utilisé sous une forme différente : si un *itemsets* est peu fréquent, tout sur-ensemble de celui-ci doit également être rare. Par exemple, si *{a, b, c, d}* est peu fréquent, alors *{a, b, c, d, e, f}* doit également être peu fréquent. Si ces derniers étaient fréquents, alors *{a, b, c, d}* en tant que sous-ensemble de celui-ci doit également être fréquent, mais nous savons que ce n'est pas le cas. La signification pratique de ce résultat est que les seuls *itemsets* avec, disons, 6 éléments qui valent la peine d'être pris en compte sont ceux qui sont créés à partir d'un *itemset* fréquent avec 5 éléments en ajoutant un item supplémentaire.

Fonctionnement de l'algorithme :

L'algorithme *FP-Growth* comporte deux étapes. Tout d'abord, la base de données de transactions est traitée pour produire une structure de données appelée *FP-tree* (*Frequent Pattern Tree*) qui capture l'essence de la base de données en ce qui concerne l'extraction des ensembles d'éléments fréquents. Ensuite, l'arbre *FP-tree* est traité récursivement, en construisant une séquence d'arbres réduits connus sous le nom d'arbres *FP conditionnels*. La base n'est traitée qu'à la première de ces étapes et n'est analysée que deux fois. Quant à la méthode alternative pratiquement imaginable, la base de données devrait être analysée au moins une fois, réduire le nombre de scans à seulement deux est une caractéristique très précieuse de cet algorithme.

Importance d'algorithme FP-Growth :

On prétend que *FP-Growth* a un ordre de grandeur plus rapide qu'*Apriori*. Naturellement, cela dépend d'un certain nombre de facteurs, par exemple, si l'arbre FP peut être représenté d'une manière suffisamment compacte pour tenir dans la mémoire principale. Comme pratiquement tous les algorithmes de ce domaine, il existe un certain nombre de variantes d'*Apriori* et de *FP- Growth* qui visent à les rendre moins coûteux en mémoire ou en calcul. Dans les sections suivantes, l'algorithme *FP-Growth* est décrit et illustré par une série de figures montrant l'arbre FP correspondant à un exemple de base de transactions, suivi d'une séquence d'arbres FP conditionnels à partir de laquelle il est simple d'extraire les ensembles d'éléments fréquents.

Construction de l'arbre FP (FP-Tree) :

Pour illustrer le processus, nous utilisons les données de transaction ci-dessous. Il n'y a que cinq transactions conservées dans la base, chaque élément étant représenté par une seule lettre :

N°	Transactions
1	*f, a, c, d, g, i, m, p*
2	*a, b, c, f, l, m, o*
3	*b, f, h, j, o*
4	*b, c, k, s, p*
5	*a, f, c, e, l, p, m, n*

La première étape consiste à parcourir la base de transactions pour compter le nombre d'occurrences de chaque élément, qui est le même que le nombre de support de l'ensemble d'éléments correspondant à un seul élément. Le résultat est le suivant :

f, c	4
a, m, p, b	3
l, o	2
d, g, i, h, j, k, s, e,n	1

Comme la quantité de données est si petite, nous utiliserons la valeur hautement irréaliste: *minsupportcount* = 3. Il n'y a que six éléments pour lesquels l'ensemble d'éléments correspondant à un seul élément a un nombre de support de *minsupportcount* ou plus. Dans l'ordre décroissant du nombre de supports, ils sont : *f, c, a, b, m et p*.

Nous effectuons ensuite le deuxième et dernier scan de la base de données des transactions. Au fur et à mesure que chaque transaction est lue, tous les éléments qui ne sont pas dans les éléments ordonnés sont supprimés et les éléments restants sont triés par ordre décroissant (c'est-à-dire l'ordre des éléments dans les éléments ordonnés) avant d'être passés au processus de construction de l'arbre FP. Cela donne le même effet que si les données de transaction étaient à l'origine les cinq transactions :

f, c, a, m, p
f, c, a, b, m
f, b
c, b, p
f, c, a, m, p

Initialisation :

Nous pouvons représenter schématiquement l'état initial de l'arbre *FP* par un seul nœud, représentant la racine. Nous allons également représenter l'arbre évolutif par le contenu de quatre tableaux: deux tableaux bidimensionnels nœuds et fils, avec un index numérique qui correspondra à la numérotation des nœuds dans l'arbre (zéro indique le nœud racine). Notons que *child* peut avoir un nombre indéfini de colonnes, mais seuls les deux premiers sont nécessaires pour cet exemple.

index	*item name*	*count*	*linkto*	*parent*
0	*root*			

nodes array

child1	*child2*

child array

index	*startlink*	*endlink*
f		
c		
a		
b		
m		
p		

link arrays

Tableau 3 - *Tableau correspondant a Initial Form de FP-tree : Root Node Only*.

Traitement de la transaction 1 : *f, c, a, m, p* :

(Item *f*) : Comme il s'agit du premier élément de la transaction, nous considérons que le « nœud actuel » est le nœud racine. Dans ce cas, il n'a pas de nœud descendant avec le nom d'élément *f*, donc un nouveau nœud pour l'élément *f* est ajouté numéroté 1, avec son nœud parent numéroté 0 (indiquant le nœud racine) dans la figure 8. Notons qu'un élément avec le nom*f* et le nombre de support 1 est indiqué par *f/1* dans la figure.

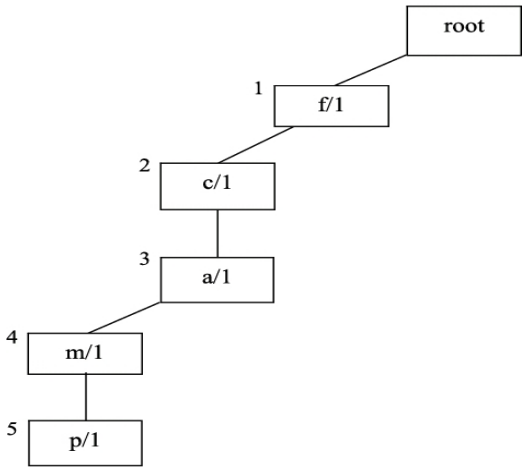


Figure 8 - *FP-tree après le traitement de la transaction 1.*

(Item *c*) : Le nœud actuel est maintenant le nœud 1, qui n'a pas de nœud descendant avec le nom d'élément *c*, donc un nouveau nœud est ajouté numéroté 2, pour l'élément *c* avec son nœud parent numéroté 1. Ainsi de suite pour les autres Item *a, m, p.*

index	*item name*	*count*	*linkto*	*parent*
0	*root*			
1	*f*	1		0
2	*c*	1		1
3	*a*	1		2
4	*m*	1		3
5	*p*	1		4

nodes array

child1	*child2*
1	
2	
3	
4	
5	

child array

index	*startlink*	*endlink*
f	1	1
c	2	2
a	3	3
b		
m	4	4
p	5	5

link arrays

Tableau 4 - *Tableaux correspondant à FP-tree après le traitement de la transaction 1.*

Traitement de la transaction 2 : *f, c, a, b, m*

(Items *f, c* et *a*) : Il existe déjà une chaîne de nœuds de la racine aux nœuds *f*, *c* et *a* tour à tour, donc aucun changement n'est nécessaire sauf pour augmenter le nombre de nœuds 1,2 et 3et les lignes correspondantes de nœuds de tableau par un, donnant :

index	item name	count	linkto	parent
0	root			
1	f	2		0
2	c	2		1
3	a	2		2
4	m	1		3
5	p	1		4

nodes array

child1	child2
1	
2	
3	
4	
5	

child array

index	startlink	endlink
f	1	1
c	2	2
a	3	3
b		
m	4	4
p	5	5

link arrays

Tableau 5 - *Tableaux correspondant à FP-tree après le traitement des trois premiers éléments dela transaction 2.*

(Item *b*) : Il n'y a pas de descendant du nœud actuel (le dernier nœud accédé), c'est-à-dire le nœud 3, qui porte le nom d'élément *b*, donc un nouveau nœud numéroté 6 est ajouté pour l'élément *b* avec son nœud parent numéroté 3.

Recherche des ensembles d'éléments fréquents à partir de l'arborescence FP

Après avoir construit l'arbre FP, nous pouvons maintenant l'analyser pour extraire tous les *itemsets* fréquents pour la base de données de transaction.

L'algorithme génère d'abord une séquence d'ensembles à deux éléments en étendant l'ensemble d'éléments {*p*} en ajoutant un élément à la position la plus à gauche. Il le fait pour tous les éléments qui sont au-dessus de *p* dans le tableau *orderedItems* à tour de rôle. Ainsi, les ensembles d'éléments à deux éléments {*m, p*}, {*b, p*}, {*a, p*}, {*c, p*} et {*f, p*} sont examinés tour à tour. Si l'un d'entre eux est fréquent, son arbre FP conditionnel est construit et une séquence d'ensembles d'éléments à trois éléments est générée en étendant l'ensemble d'éléments à deux éléments en ajoutant un élément dans la position la plus à gauche. Le processus se poursuit de cette manière jusqu'à ce que toute la structure arborescente ait été examinée. À chaque étape où l'ensemble d'éléments actuel est développé en ajoutant un élément supplémentaire dans la position la plus à gauche, seule les éléments du tableau *orderedItems* au-dessus de celui qui se trouvait précédemment à l'extrême gauche sont pris en compte.

Pour rechercher les *Itemsets* terminé par Item *p* nous devons vérifier si des ensembles d'éléments à deux éléments formés en ajoutant un élément supplémentaire à l'ensemble d'éléments {*p*} sont également fréquents. Pour ce faire, nous construisons d'abord un arbre FP conditionnel pour l'ensemble d'éléments {*p*}. Il s'agit d'une version réduite de l'arbre FP d'origine qui contient uniquement les branches qui commencent à la racine et se terminent aux deux nœuds étiquetés *p*, mais avec les nœuds renumérotés et souvent avec des *supports* différents.

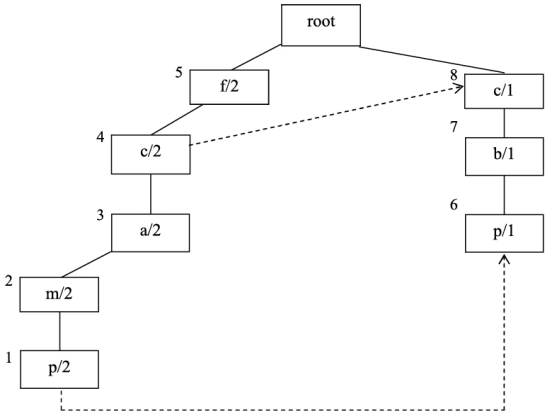


Figure 9 - *Arborescence FP conditionnelle pour {p} - version finale*

La meilleure approche pour construire l'arbre FP conditionnel pour *{p}* est de construire l'arbre de bas en haut, branche par branche, en utilisant les nombres des *p* nœuds. Chaque nouveau nœud entré dans l'arborescence « hérite » du nombre de supports du nœud *p* en bas de la branche. Cela donne la version finale de l'arbre FP conditionnel pour l'ensemble d'éléments *{p}* illustré à la figure suivante :

index	*item name*	*count*	*linkto*	*parent*
1	*p*	2	6	2
2	*m*	2		3
3	*a*	2		4
4	*c*	2	8	5
5	*f*	2		
6	*p*	1		7
7	*b*	1		8
8	*c*	1		

nodes2 array

oldindex
5
4
3
2
1
11
10
9

oldindex

index	*startlink2*	*lastlink*
p	1	6
m	2	2
a	3	3
c	4	8
f	5	5
b	7	7

link arrays

Tableau 6 - *Tableaux correspondant à l'arborescence FP conditionnelle pour {p}*

2.5. Conclusion

Dans ce chapitre, nous nous sommes focalisé particulièrement sur les règles d'associations et son principe ensuite nous avons présenté l'algorithme *Apriori* qui est très influent et, qui a montré comment des règles d'association pouvaient être générées dans un délai réaliste, du moins pour des bases de données relativement petites. Ensuite on a décrit le processus de création de l'arbre FP et d'en extraire les ensembles d'éléments fréquents.

Chapitre 3 : Mise en œuvre de l'approche et implémentation

3.1. Introduction

Pour atteindre l'objectif de classification des séries temporelles, nous avons choisis à chaque composante de cette classification une méthode et nous l'avons implémenté notre choix sur la méthode *SAX* et nous l'avons choisir pour implémenté les motifs séquentiel des données l'algorithme *PrefixSpan* . Les détails de ces choix feront l'objet de ce chapitre.

3.2. Que voulons-nous faire des séries temporelles ?

3.2.1. Clustering

Les algorithmes de clustering examinent les données pour trouver des groupes d'éléments similaires. Par exemple, une compagnie d'assurance peut regrouper ses clients en fonction du revenu, de l'âge, des types de police souscrite ou de l'expérience des sinistres antérieurs. Dans une application de diagnostic de défaut, les défauts électriques peuvent être regroupés en fonction des valeurs de certaines variables clés.

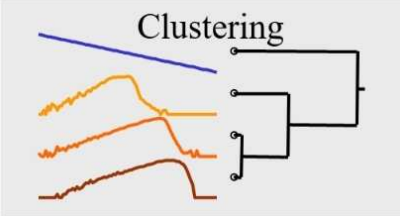


Figure 10 - *Le clustering des séries temporelles*

3.2.2. Classification

La classification est l'opération statistique qui consiste à regrouper des objets (individus ou variables) en un nombre limité de groupes, les classes (ou segments, ou clusters), qui ont deux propriétés. D'une part, ils ne sont pas prédéfinis par l'analyste mais découverts au cours de l'opération, contrairement aux classes du classement. D'autre part, les classes de la classification regroupent les objets ayant des caractéristiques similaires et séparent les objets ayant des caractéristiques différentes (homogénéité interne et hétérogénéité externe), ce qui peut être mesuré par des critères telle l'inertie interclasse. Comme le classement, la classification consiste à répartir les objets en groupes. Toutefois, cette répartition n'est pas effectuée en fonction d'un critère prédéfini, et ne vise pas à rassembler les objets dotés de la même valeur pour ce critère.

Autrement dit, on ne sait pas à l'avance la classe à laquelle chaque objet appartient, contrairement au classement. Même le nombre de classes n'est pas toujours fixé à l'avance. Cela vient de ce qu'il n'y a pas de variable à expliquer: la classification est descriptive et non prédictive. Elle est beaucoup utilisée en marketing, médecine, sciences humaines, etc. En marketing, on lui donne souvent le nom de segmentation ou typologie ou analyse typologique. En médecine, on parle de nosologie. En biologie et zoologie, on parle de taxinomie ou taxonomie. Les anglo-saxons parlent de clustering.

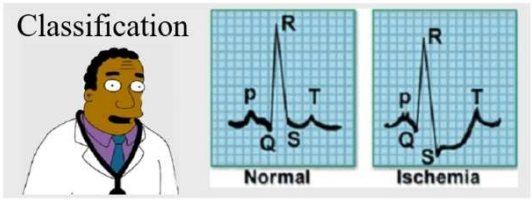


Figure 11 – *La classification des séries temporelles*

3.2.3. Motif Discovery

La découverte de motifs dans les données de séries temporelles a reçu une attention particulière dans la communauté d'exploration de données depuis sa création, principalement parce que la découverte de motifs est significative et a plus de chances de réussir lorsque les données sont volumineuses. Les algorithmes de découverte de motifs traitent généralement de trois aspects : la définition des motifs, le prétraitement basé sur le domaine et, enfin, les étapes algorithmiques.

Les définitions typiques des motifs signifient la similitude ou le support des motifs. Les domaines imposent des exigences de prétraitement à la recherche de motifs significatifs tels que l'alignement, l'interpolation et la transformation des données. Les algorithmes de découverte de motifs varient en fonction de l'évaluation exacte ou approximative de la définition. De plus, les algorithmes nécessitent des représentations différentes (*approche symbolique des agrégats* (*SAX*), *DFT*, etc.) et des mesures de similarité (corrélation, distance de déformation temporelle dynamique (*DTW*), etc.).

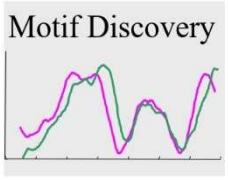


Figure 12 - *Motif Discovery*

3.2.4. Rule Discovery

Nous considérons le problème de la découverte de règles à partir de séries temporelles à valeurs réelles pour la prédiction basée sur des règles. Sans perte de généralité, le problème de la découverte de règles est exemplaire en utilisant deux séries temporelles T_A et T_B, depuis la recherche de règles de plus de deux séries temporelles qui peuvent être traitées par paires. Ça peut également s'appliquer à la situation d'une seule série en laissant $T_A = T_B$.

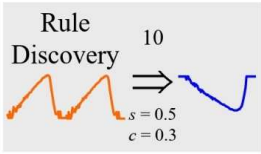


Figure 13 - *Rule Discovery*

3.2.5. Query by content

Etant donné une série temporelle de requête Q et une mesure de similitude / disimilarité $D(Q, C)$, il s'agit de trouver la série temporelle la plus similaire dans la base de données DB .

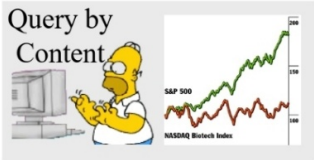
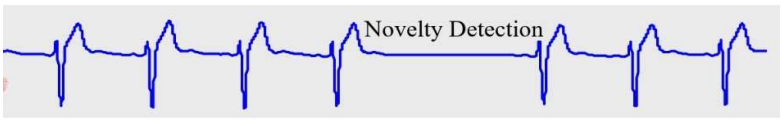


Figure 14 - *Query by Content*

3.2.6. Novelty Detection

La détection de nouveauté (ou d'anomalie) fait référence à l'identification automatique de modèles nouveaux ou anormaux intégrés dans de grandes quantités de données « normales ». Lorsqu'il s'agit de séries temporelles (transformées en vecteurs), cela signifie des événements anormaux intégrés parmi de nombreux points de séries chronologiques normaux.



Figue 15 - *Novelty Detection*

3.3. Les mesures de similarité et de di-similarité

La classification des séries temporelles se base en grande partie sur la mesure de distance. Il existe différentes mesures qui peuvent être appliquées pour mesurer la distance entre les séries temporelles. Certaines sont proposées pour des représentations spécifiques des séries temporelles, par exemple «*MINDIST*» qui est compatible avec *SAX*, et certains d'entre eux fonctionnent indépendamment des méthodes de représentation, ou sont compatibles avec les séries de données brutes.

Dans la classification des séries temporelles, la distance est évaluée approximativement. En particulier, afin de comparer des séries avec des intervalles d'échantillonnage irréguliers et avec des longueurs différentes, il est d'une grande importance de déterminer la similarité entre les séries. Il existe différentes mesures de distances présentées pour déterminer la similarité entre les séries temporelles, comme par exemple la distance euclidienne. Une des façons les plus simples pour calculer la distance entre deux séries est de les considérer comme des séries uni-variées, puis calculer la mesure de la distance à travers tous les points de temps.

3.3.1. Trouver des séries similaires dans le temps

Parce que cette similarité est sur chaque point du temps, les distances établies sur la corrélation ou la distance euclidienne sont nécessaires pour cet objectif. Cependant, parce que ce genre de mesure de distance est coûteux sur les séries brutes, le calcul est effectué sur les séries transformées, comme la transformée de *Fourier*, les *ondelettes* ou *PAA*. La segmentation des séries temporelles qui sont corrélées est classée comme basée sur la similarité dans le temps.

3.3.2. Métrique de distance euclidienne

L'une des mesures de similarité les plus simples pour les séries temporelles est la mesure de distance euclidienne. Supposons que les deux séquences aient la même longueur *n*, nous pouvons voir chaque séquence comme un point dans l'espace euclidien à *n* dimensions, et définir la dissemblance entre les séquences *C* et *Q* et *D (C, Q)*.

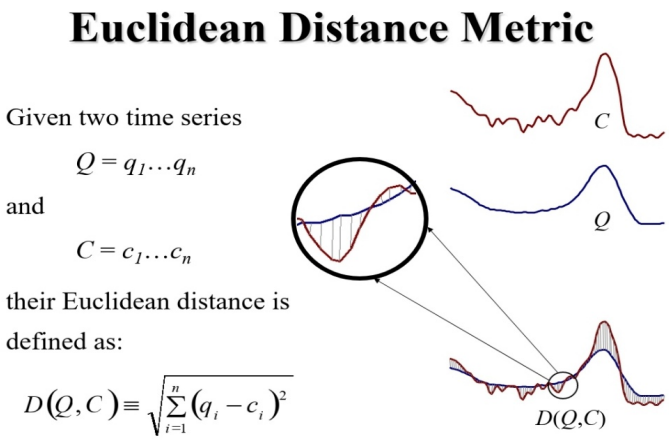


Figure 16 - *La distance euclidienne.*

3.4. Représentation des séries temporelles

Elle peut être divisée en deux parties :

3.4.1. Les méthodes adaptatives aux données

Les méthodes de représentation adaptatives aux donnés opèrent sur toutes les séries de l'ensemble de données et tentent de minimiser l'erreur globale de reconstruction en utilisant une longueur arbitraire pour les segments (non-égaux). Cette technique a été appliquée dans différentes approches telles que l'interpolation polynômiale par morceaux (*PPI*), la régression polynômiale par morceaux (*PPR*), approximation linéaire par morceaux (*PLA*), approximation constante par morceaux (*PCA*), approximation constante par morceaux adaptative (*APCA*), décomposition de valeur singulière (*SVD*), langage naturel (*NL*), langage naturel symbolique (*NLG*), approximation par agrégation symbolique (*SAX*).

3.4.2. Les méthodes non adaptatives aux données

Les approches non adaptatives aux données sont des représentations qui sont appropriées à la segmentation de séries de taille fixe (de même longueur), et la comparaison entre les représentations de plusieurs séries est simple. Les méthodes dans cette catégorie sont les ondelettes : *HAAR*, *DAUBECHIES*, *Coeiflets*, *Symlets*, transformé en ondelettes discrète (*DWT*), polynômes spectrales de Chebyshev, *DFT* spectrale, Mappages hasardeux, approximation par agrégation par morceaux (*PAA*) et PLA indexable (*IPLA*).

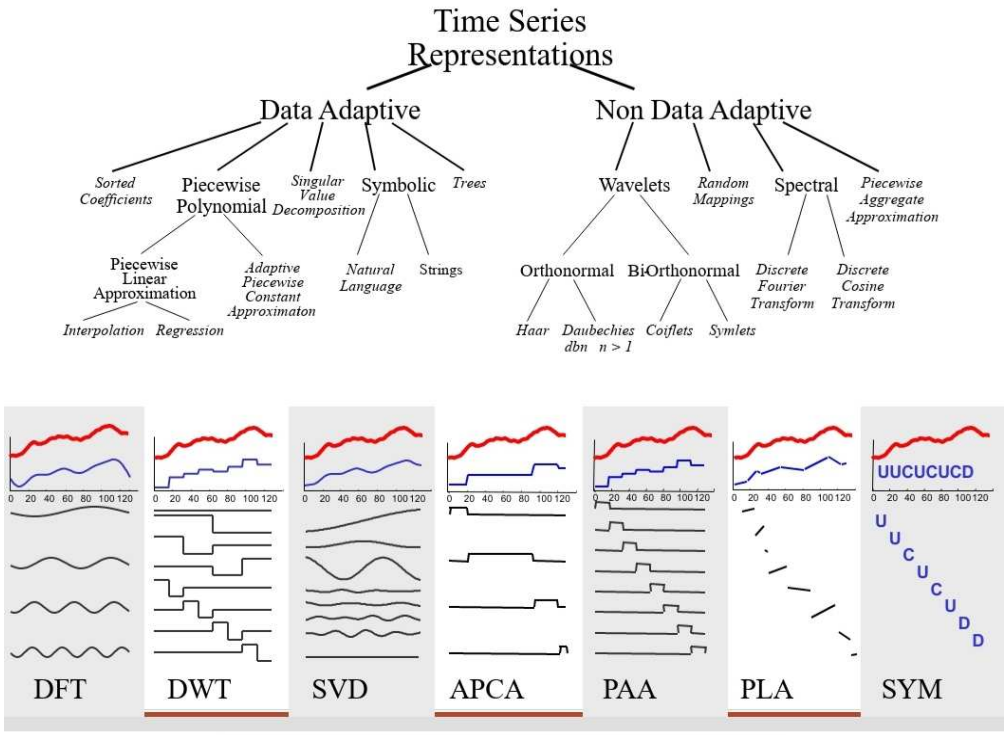


Figure 17 - *Méthodes de représentation des séries temporelles.*

3.5. Les étapes de l'approche proposée

3.5.1. La normalisation des valeurs des séries temporelles

La normalisation permet de rendre une variable statistique centrée réduite. En probabilités et statistiques, une variable centrée réduite est une variable aléatoire dont on a modifié les valeurs afin de fixer sa moyenne et sa variance. Centrer une variable consiste à soustraire son espérance à chacune de ses valeurs initiales, soit retrancher à chaque donnée la moyenne. Elle constitue simplement en un changement d'origine, qui place la moyenne de la distribution au point 0 de l'axe des abscisses. Réduire une variable consiste à diviser toutes ses valeurs par son écart type.

Elle passe par les étapes suivantes :

- On calcule la moyenne générale pour chaque série et l'écart type qui se présentent sous les formules suivantes :

La formule de la moyenne :

$$\bar{x} = \frac{x1 + x2 + \ldots + xn}{n} = \frac{1}{n}\sum_{i=1}^{n}(x_i)$$

Avec :

x : La moyenne,

$x_1, x_2 \ldots x_n$: Les valeurs de la série.

n : le nombre des valeurs de la série.

La formule de l'écarte type :

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Avec :

σ : Le symbole de l'écart type.

x : La moyenne,

x_i : Les valeurs de la série.

- Ensuite, chaque valeur de la série est substituée de la moyenne et divisée par l'écart type :

La formule finale de la normalisation :

$$x' = \frac{xi - \bar{x}}{\sigma}$$

- Une nouvelle série est alors constituée, dont les valeurs se trouvent dans l'intervalle [-1, +1].

3.5.2. Algorithme PAA (*Piecewise Aggregate Approximation*)

Réduction de la dimension :

On note une séries temporelle $X = x_1, \ldots, x_n$ et l'ensemble des séries temporelles qui constituent la base de donnée $Y = \{y_1, \ldots . y_k\}$. Sans perte de généralité, on suppose que chaque séquence dans *Y* est de longueur de *n* unités.

Soit *N* la dimension de l'espace transformé. Une série temporelle *X* de longueur *n* est représenté dans l'espace *N* par un vecteur : $X' = x'_1, \ldots, x'_N$

Par exemple, une série composée de huit (*n*) points est projeté en deux dimensions (*N*). La série est divisée en deux (*N*) trames et la moyenne de chaque trame est calculée. Un vecteur de ces moyens devient les données de la représentation réduite. Les spécialistes de ce domaine ont proposé indépendamment *PAA*, où, dans chaque séquence les

données sont divisées en *k* segments à longueur égale et la valeur moyenne de chaque segment est utilisée en tant que coordonnées d'un vecteur caractéristique à *k* dimensions.

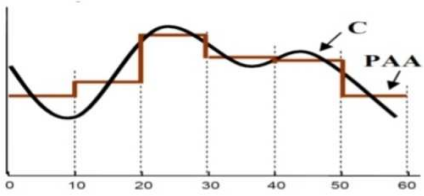


Figure 18 - *Une série temporelles C représentée par PAA*

3.5.3. La méthode SAX (*Symbolic Aggregate Approximation*)

SAX est une représentation symbolique pour les séries temporelles qui permet la réduction de la dimensionnalité et l'indexation avec une mesure de distance inférieure. Dans les tâches de fouille de données classiques comme le clustering, la classification, l'indexation, etc., *SAX* est aussi une bonne représentation par rapport à d'autres aussi bien connues telles que *Discrete Wavelet Transform (DWT)* et *Transformation de Fourier discrète (DFT)*, tout en dépensant moins d'espace de stockage.

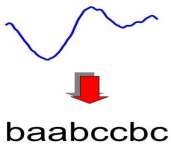


Figure 19 - *Principe de SAX*

Comment obtenir la représentation en SAX :

Commençons par convertir la série temporelle en représentation *PAA*, puis convertissons les résultats de *PAA* en symboles. Cela prend un temps linéaire :

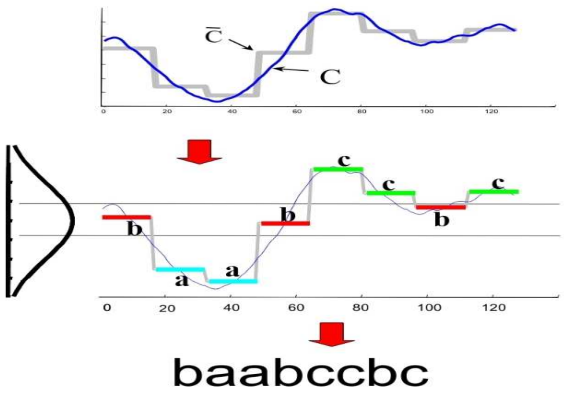


Figure 20 - *Une série temporelles C représentée par SAX*

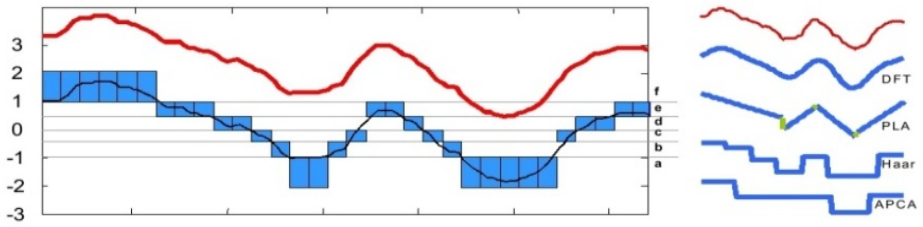


Figure 21 - *Comparaison de SAX avec les autres méthodes*

Pour $k = 32$, une série temporelle brute de longueur 128 est transformée en le mot « ***fffffffeeeddcbaabceedcbaaaaacddee*** ». Nous pouvons utiliser plus de symboles pour représenter la série puisque chaque symbole nécessite moins de bits que les nombres réels (*float*, *double*).

.

Le choix des symboles de SAX :

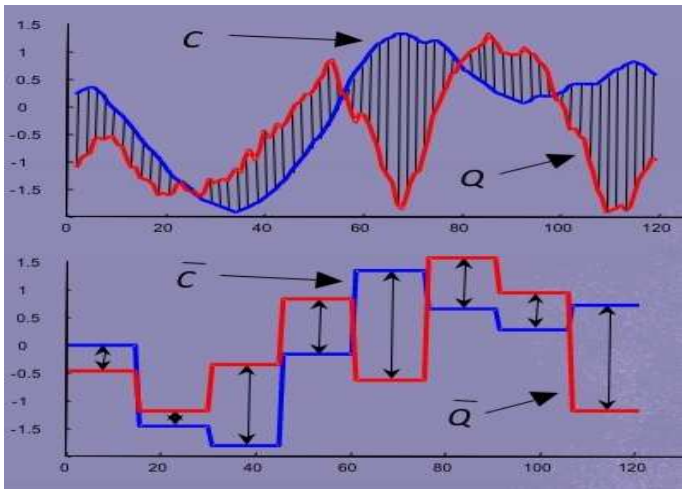
Dans un objectif de réduction de la dimensionnalité, il est important de définir des unités temporelles permettant de regrouper les points des séries temporelles. Le tableau suivant contient les points de coupures des intervalles de valeurs obtenues par *PAA* et à représenter par le même symbole dans *SAX* :

a	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
β 1	−0, 43	−0, 67	−0, 84	−0, 97	−1, 07	−1, 15	−1, 22	1, 28	−1, 34	−1, 38	−1, 43	−1, 47	−1, 5	−1, 53	−1, 56	−1, 59	−1, 62	−1, 64
β 2	0, 43	0	−0, 25	−0, 43	−0, 57	−0, 67	−0, 76	−0, 84	−0, 91	−0, 97	−1, 02	−1, 07	−1, 11	−1, 15	−1, 19	−1, 22	−1, 25	−1, 28
β 3		0, 67	0, 25	0	−0, 18	−0, 32	−0, 43	−0, 52	−0, 6	−0, 67	−0, 74	−0, 79	−0, 84	−0, 89	−0, 93	−0, 97	−1	−1, 04
β 4			0, 84	0, 43	0, 18	0	−0, 14	−0, 25	−0, 35	−0, 43	−0, 5	−0, 57	−0, 62	−0, 67	−0, 72	−0, 76	−0, 8	−0, 84
β 5				0, 97	0, 57	0, 32	0, 14	0	−0, 11	−0, 21	−0, 29	−0, 37	−0, 43	−0, 49	−0, 54	−0, 59	−0, 63	−0, 67
β 6					1, 07	0, 67	0, 43	0, 25	0, 11	0	−0, 1	−0, 18	−0, 25	−0, 32	−0, 38	−0, 43	−0, 48	−0, 52
β 7						1, 15	0, 76	0, 52	0, 35	0, 21	0, 1	0	−0, 08	−0, 16	−0, 22	−0, 28	−0, 34	−0, 39
β 8							1, 22	0, 84	0, 6	0, 43	0, 29	0, 18	0, 08	0	−0, 07	−0, 14	−0, 2	−0, 25
β 9								1, 28	0, 91	0, 67	0, 5	0, 37	0, 25	0, 16	0, 07	0	−0, 07	−0, 13
β 10									1, 34	0, 97	0, 74	0, 57	0, 43	0, 32	0, 22	0, 14	0, 07	0
β 11										1, 38	1, 02	0, 79	0, 62	0, 49	0, 38	0, 28	0, 2	0, 13
β 12											1, 43	1, 07	0, 84	0, 67	0, 54	0, 43	0, 34	0, 25
β 13												1, 47	1, 11	0, 89	0, 72	0, 59	0, 48	0, 39
β 14													1, 5	1, 15	0, 93	0, 76	0, 63	0, 52
β 15														1, 53	1, 19	0, 97	0, 8	0, 67
β 16															1, 56	1, 22	1	0, 84
β 17																1, 59	1, 25	1, 04
β 18																	1, 62	1, 28
β 19																		1, 64

Tableau 7 - *Les points de coupures des intervalles*

Si l'on dispose des ressources nécessaires, il peut donc être intéressant de construire des représentations symboliques adaptives. De nombreuses représentations symboliques peuvent être envisagées, non seulement en fonction des données à analyser, mais aussi en fonction des tâches d'analyses à effectuer.

La distance euclidienne :



$$DR(\bar{Q},\bar{C}) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} (\bar{q}_i - \bar{c}_i)^2}$$

$\hat{C}$ = baabccbc

$\hat{Q}$ = babcacca

$$MINDIST(\hat{Q},\hat{C}) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} (dist(\hat{q}_i,\hat{c}_i))^2}$$

Figure 22 - *La distance PAA limite inférieure de la distance euclidienne*

3.5.4. Calcul de la distance

La prochaine phase de segmentation est de calculer la distance entre les séries pour mesurer la similarité. Et dans ce vaste domaine, il existe beaucoup de méthodes mais seulement pour le calcul numérique. Dans notre travail nous nous intéressons à la distance entre symboles, ce qui nous pousse à utiliser une table prédéfinie dont la suivante est un bref exemple :

	A	B	C	D
A	0	0.25	0.75	0.5
B	0.25	0	0.65	0.45
C	0.75	0.65	0	0.21
D	0.5	0.45	0.25	0

Tableaux 8 - *Exemple de distances entre quatre symboles*

3.5.5. Exemple d'application

1	554.36	16	553.94
2	553.75	17	553.80
3	554.26	18	553.20
4	554.45	19	554.18
5	554.00	20	554.81
6	554.67	21	554.08
7	554.34	22	553.63
8	553.85	23	553.98
9	553.69	24	553.49
10	553.59	25	554.00
11	554.54	26	554.11
12	554.52	27	554.07
13	553.44	28	554.68
14	553.48	29	554.25
15	554.64	30	554.37

Tableaux 9 - *Echantillon représentant une série temporelle*

En ce moment, on rend cette série centrée réduite en appliquant les différentes étapes de la normalisation (calcul de la moyenne et de l'écart type) :

La moyenne

$$\bar{X} = \frac{554.36+553.75+554.26+\cdots\ldots\ldots\ldots\ldots\ldots+554.25+554.37}{30} = \frac{17176.28}{30} = 572.54.$$

L'écart type :

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\sigma = \sqrt{\frac{1}{30}}\,(554.36\text{-}572.54) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(554.37\text{-}572.54) = \sqrt{341.223}.$$

$$\text{donc } \sqrt{341.223} = 18.472.$$

Après, on doit remplacer chaque valeur (échantillon) par la valeur normalisée, ce qui veut dire qu'on doit appliquer sur chaque valeur la formule :

$$x' = \frac{xi - \bar{x}}{\sigma}.$$

$$x' = \frac{554.36 - 572.54}{18.472} = -0.984.$$

Donc la séries deviennent normalisées, le tableau suivant présente la série normalisé :

1	-0.984	**16**	-0.906
2	-0.917	**17**	-0.014
3	-0.989	**18**	-0.046
4	-0.979	**19**	-0.993
5	-0.903	**20**	-0.959
6	-0.967	**21**	-0.999
7	-0.985	**22**	-0.023
8	-0.911	**23**	-0.904
9	-0.920	**24**	-0.931
10	-0.925	**25**	-0.903
11	-0.974	**26**	-0.997
12	-0.975	**27**	-0.999
13	-0.933	**28**	-0.966
14	-0.931	**29**	-0.990
15	-0.960	**30**	-0.983

Tableau 10 - *La série après normalisation*

Ensuite, on applique l'algorithme *PAA* avec un k = 6, c.à.d. qu'on va deviser notre série qui contiens 30 valeurs par 5 segments dont chacun contient 6 valeurs. Après on

$$\text{Exemple : } \bar{X} = \frac{-0.984+(-1.017)+(-0.989)+(-0.979)+(-1.003)+(-0.967)}{6} = \frac{-5.939}{6} = -0.9893$$

calcule pour chaque segment sa moyenne, et on le remplace par celle-ci :

1	-0.989
2	-0.998
3	-0.915
4	-0.901
5	-0.989

On continue ce calcul jusqu'à que la série devienne de taille réduite.

Tableau 11 - *La série après application de PAA*

La dimension de la série qui contient 30 valeurs a été diminuée jusqu'à 5 valeurs, ce qui nous montre l'intérêt essentielle de l'approche *PAA*.

Passons maintenant à la différence entre *PAA* et SAX qui est de convertir nos résultats numériques qu'on a obtenus par l'approche *PAA* aux symboles comme montré précédemment (chaque intervalle de moyenne est remplacé par un symbole). Notre

1	A
2	B
3	B
4	D
5	C

série après application de l'algorithme *SAX* devient :

Tableau 12 - *La série après application de SAX***.**

3.6. Extraction de motifs par l'algorithme *PrefixSpan*

PrefixSpan (Exploration de modèles séquentiels projetés par préfixe) est un algorithme très connu pour l'exploration des données séquentielles. Il extrait les modèles séquentiels par une méthode de croissance de motifs basée sur la projection. L'idée de base derrière cette méthode est que, plutôt que de projeter des bases de données de séquences en évaluant les occurrences fréquentes de sous-séquences, la projection est faite sur des préfixes fréquents. Cela permet de réduire le temps de traitement, ce qui augmente finalement l'efficacité de l'algorithme.

L'exploration des motifs séquentiels démarre à partir du préfixe de longueur 1, recherche la base de données de projection correspondante pour obtenir la séquence fréquente correspondant au préfixe de longueur 1, puis extraie récursivement la séquence fréquente correspondant au préfixe de longueur 2, et ainsi de suite , jusqu'à ce que, récursivement, creuse dans une extraction de préfixe plus longue.

Entrée : ensemble de données de séquence S et seuil de support α

Sortie : tous les ensembles de séquences fréquentes qui répondent aux exigences de prise en charge

1) Trouver tous les préfixes d'une longueur de 1 et la base de données de projection correspondante.

2) Compter le préfixe de longueur 1 et supprimer l'item correspondant au préfixe dont le degré de support est inférieur au seuil α de l'ensemble de données S, et obtenir toutes les séquences d'item 1 fréquentes, i = 1.

3) L'extraction récursive pour chaque préfixe dont la longueur est i répond aux exigences de prise en charge :

   a) Trouver la base de données de projection correspondant au préfixe. Si la base de données de projection est vide, elle revient de manière récursive.

   b) Compter le nombre de supports pour chaque élément dans la base de données de projection. Renvoie récursivement si le nombre de supports pour tous les éléments est inférieur au seuil α.

   c) Combiner les éléments individuels qui satisfont au nombre de supports avec le préfixe actuel pour obtenir plusieurs nouveaux préfixes.

   d) Soit i = i + 1, le préfixe est le préfixe après fusion des éléments, et effectuer récursivement l'étape 3.

Exemple d'implémentation de l'algorithme PrefixSpan :

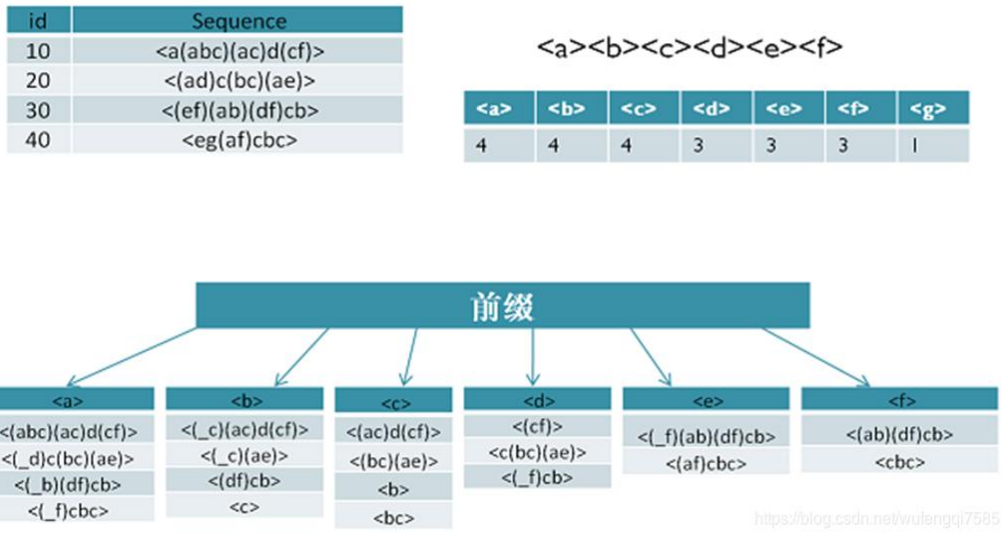


Figure 23 - *Exemple d'implémentation de l'algorithme PrefixSpan*

Le préfixe de longueur 1 inclut ***<a>, <b>, <c>, <d>, <e>, <f>, <g>***. Nous devons rechercher récursivement les six préfixes pour trouver la séquence fréquente

correspondant à chacun préfixe. Comme le montre la figure ci-dessous, le suffixe correspondant à chaque préfixe est également marqué. Étant donné que ***g*** n'apparaît que dans la séquence 4, le nombre de supports n'est que de 1, il est donc impossible de continuer à extraire. Notre séquence fréquente de longueur 1 est ***<a>, <b>, <c>, <d>, <e>, <f>***. Supprimer **g** de toutes les séquences, c'est-à-dire que le 4ème enregistrement devient ***<e (af) cbc>***.

Maintenant, nous commençons à extraire des séquences fréquentes, en commençant par un préfixe de longueur 1. Ici, nous utilisons ***d*** comme exemple pour extraire récursivement, et les autres nœuds sont les mêmes que ***d***.

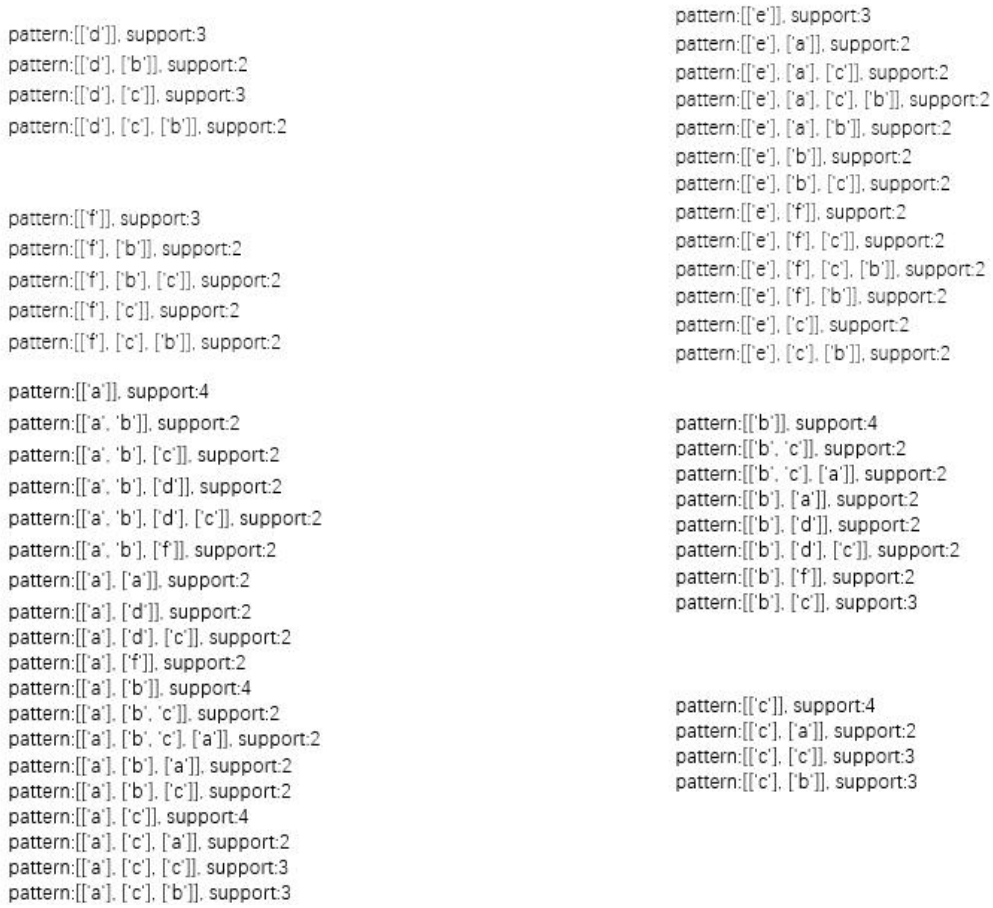
id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

Tout d'abord, nous trouvons la projection du préfixe (suffixe) dans toutes les séquences par le préfixe ***d*** : <(cf)> <c(bc)(ae)> <(_f)cb>. Comptons toutes les occurrences de suffixes de ***d***, ce qui donne ***{a:1, b:2, c:3, d:0, e:1, f:1, _f:1}.*** Notons que ***f*** et ***_f*** sont différents, le premier est un ensemble d'éléments différent du préfixe ***d*** et le dernier est le même ensemble d'éléments avec le préfixe ***d <(df)>***.

Nous obtenons >= 2 séquence fréquente pour et correspond au suffixe ***<_c(ae)>***, qui compte ***{a:1, _c:1, e:1}.*** Le support n'est pas à la hauteur, donc ce minage est terminé. Revenons ensuite pour compter les éléments :

Écrivons la séquence de projection préfixée par *<_f>*, *<(bc)(ae)>*, puis nous comptons le support du suffixe → ***{b:2, a:1, c:1, e:1, _f:1}*** seul ***b*** satisfait le seuil de support. Nous obtenons donc trois séquences fréquentes préfixées par ***dc***. Nous continuons à penser récursivement à des séquences fréquentes de préfixes. Le degré de support du suffixe correspondant n'est pas à la hauteur. Cette exploitation minière est terminée. Le résultat est le suivant.

modèle :[['d']], support :3 modèle :[['d'], ['b']], support :2 modèle :[['d'], ['c']], support : 3 modèle :[['d'], ['c'], ['b']], support :2



Figue 24 - *Résultats de l'algorithme PrefixSpan*

3.7. Conception et implémentation

3.7.1. Représentation en fichiers CSV

Tout d'abord, nous notons que nos données seront prises de fichiers *CSV*. Un fichier *CSV* est un simple fichier texte dans lequel les valeurs sont séparées par une virgule, ce qui permet de sauvegarder les données dans un format de tableur. Chaque ligne comporte le même nombre de valeur (des champs) et parfois ces valeurs sont entourées de guillemets anglais (“ ”). À la place des virgules, les données peuvent également être séparées par des points-virgules, par une tabulation ou par le symbole « | » et, là encore, peuvent être entre guillemets.

3.7.2. Langage python

Python est un langage de programmation multi-paradigme. Il favorise la programmation impérative structurée, et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions, il est ainsi similaire à Perl, Ruby, Scheme et Smalltalk.

Le langage Python est placé sous une licence libre proche de la licence *BSD* et fonctionne sur la plupart des plates-formes, des supercalculateurs aux ordinateurs centraux, de *Windows* à *Unix* en passant par *Linux* et *Mac OS*, avec *Java* ou encore *.NET*. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple. Il est également apprécié par les pédagogues qui y trouvent un langage où la syntaxe, clairement séparée des mécanismes de bas niveau, permet une initiation plus aisée aux concepts de base de la programmation.

Python est un langage conçu pour produire du code de qualité, portable et facile à intégrer : grâce à sa syntaxe claire, cohérente et concise, Python permet aux développeurs de produire du code de qualité, lisible et maintenable. Écrire du code Python est un exercice agréable, même en respectant les conventions de codage. Fourni dès le départ avec des modules de tests, Python est un langage agile. Le terme agile est issu de la méthodologie de programmation agile très proche de la programmation itérative. Cette méthodologie, introduit entre autres des principes de tests continus du code. Même si elle n'est pas imposée, Python permet la programmation orientée objet. Tous les mécanismes objet essentiels sont implémentés et toutes les données manipulées sont des instances de classes, comme pour SmallTalk ou Ruby

3.7.3. PyCharm

PyCharm est IDE utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec *Django*. Développé par l'entreprise tchèque *JetBrains*, c'est un logiciel multiplateforme qui fonctionne sous *Windows*, *Mac OS X* et *Linux*. Il est décliné en édition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous licence *Apache*.

3.7.4. Implémentation sur notre machine

En premier pas nous allons choisir le fichier qui contient les données qu'on va traiter, après on va les charger dans notre interface comme montré sur la Figure 24 :

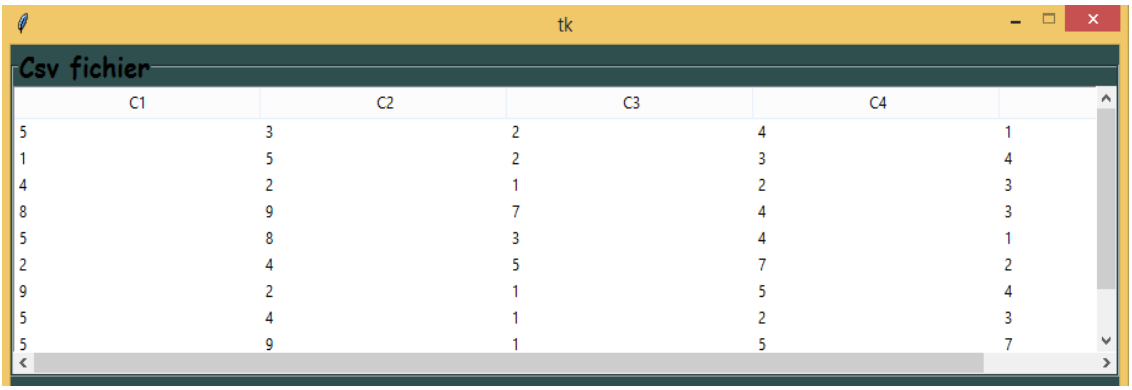
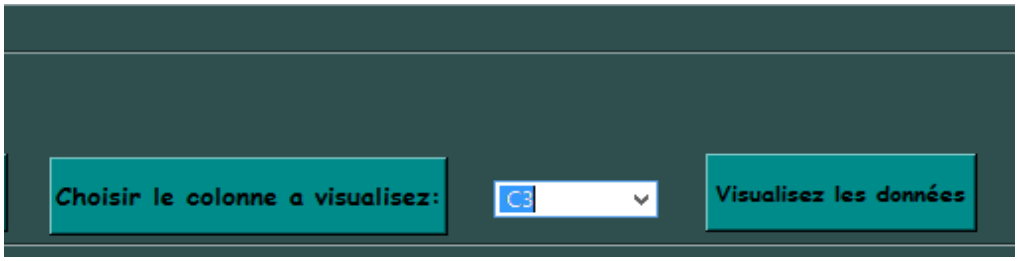


Figure 25 - *Chargement des données sur notre application*

Après le chargement des données, on va appliquer les différentes étapes que nous avons déjà montrées. Il faut d'abord sélectionner la colonne qui contient les données :



Figue 26 - *Choisir la colonne à visualiser*

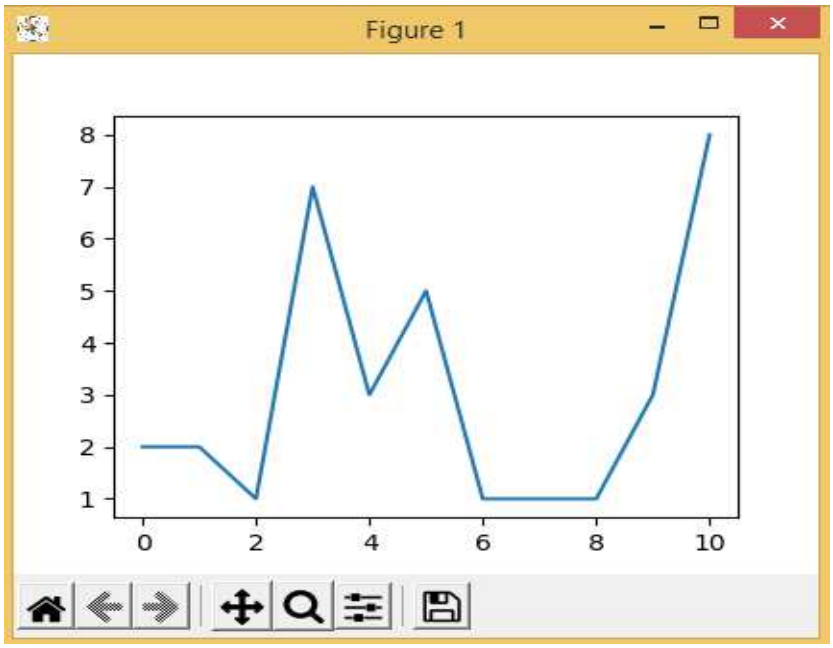


Figure 27 - *Visualisation des données par colonnes*

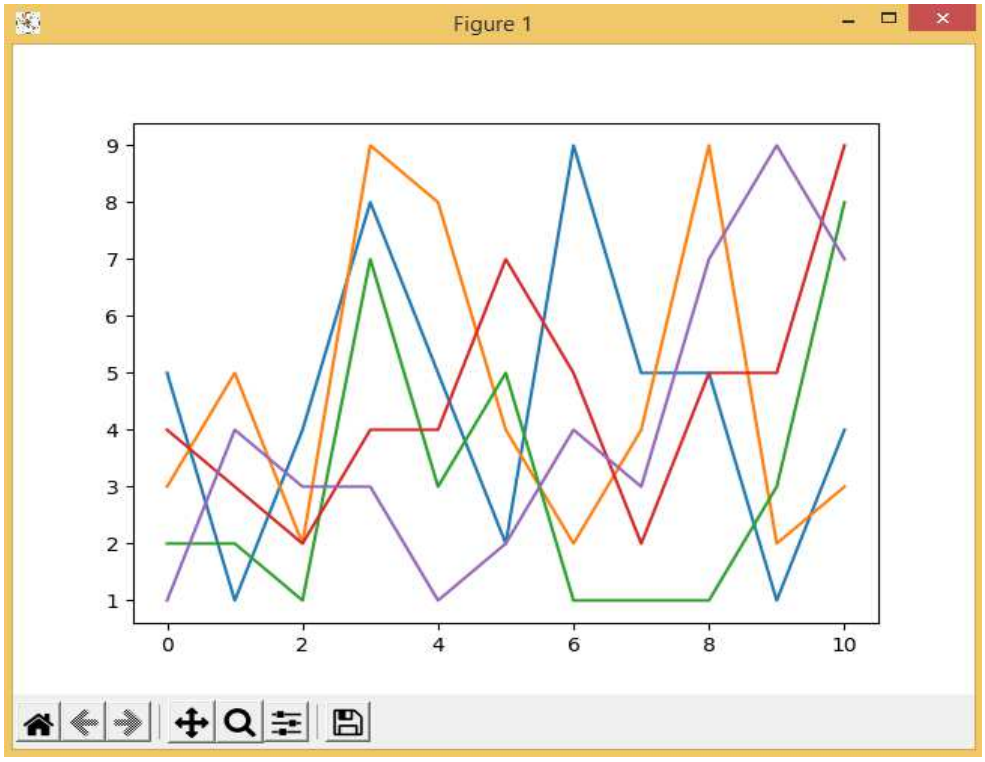


Figure 28 - *Visualisation de toutes les données chargées sur notre interface*

Dans cette étape nous allons choisir le nombre d'intervalles pour pouvoir appliquer la méthode *PAA* et méthode *SAX* :

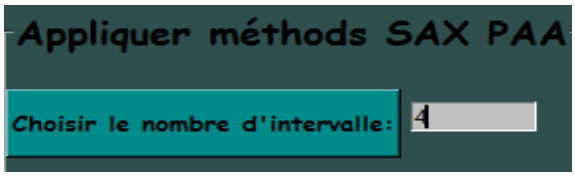


Figure 29 - *Choisir le nombre d'intervalles*

La dimension de la série a été diminuée jusqu'à 4 valeurs, ce qui nous montre l'intérêt essentielle de l'approche *PAA*.

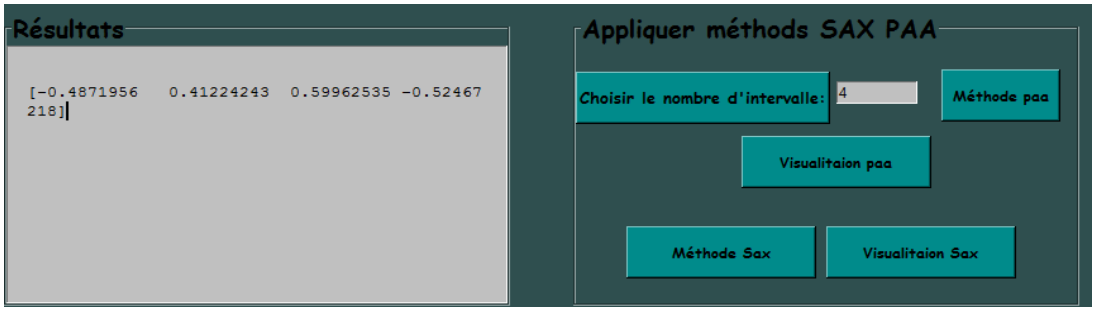


Figure 30 - *Résultat de l'application de PAA*

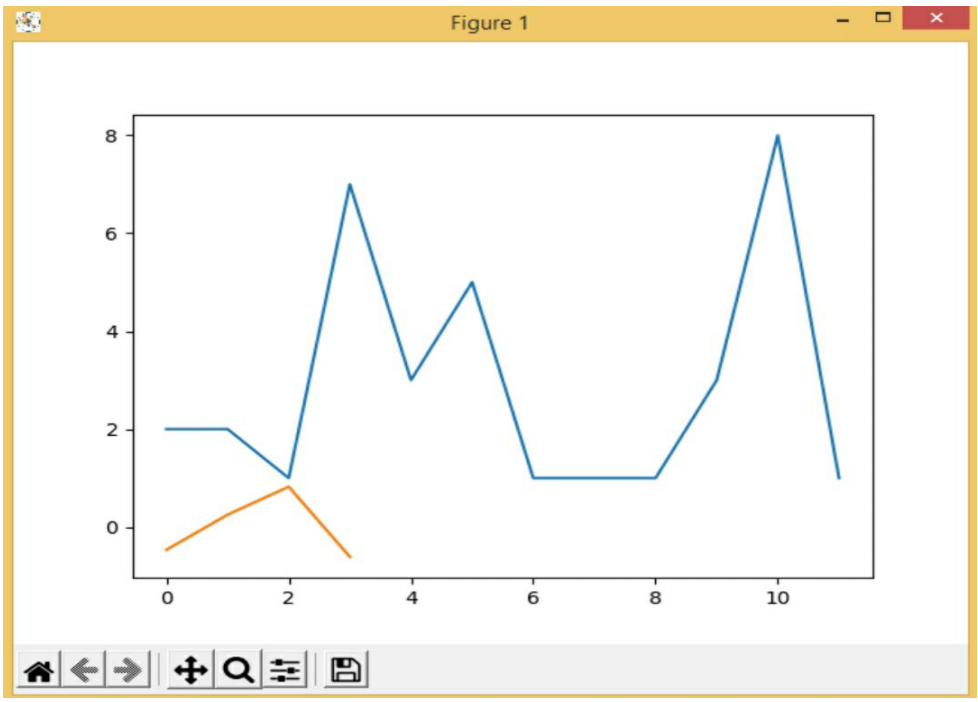


Figure 31 - *Visualisation des données après application de la méthode PAA*

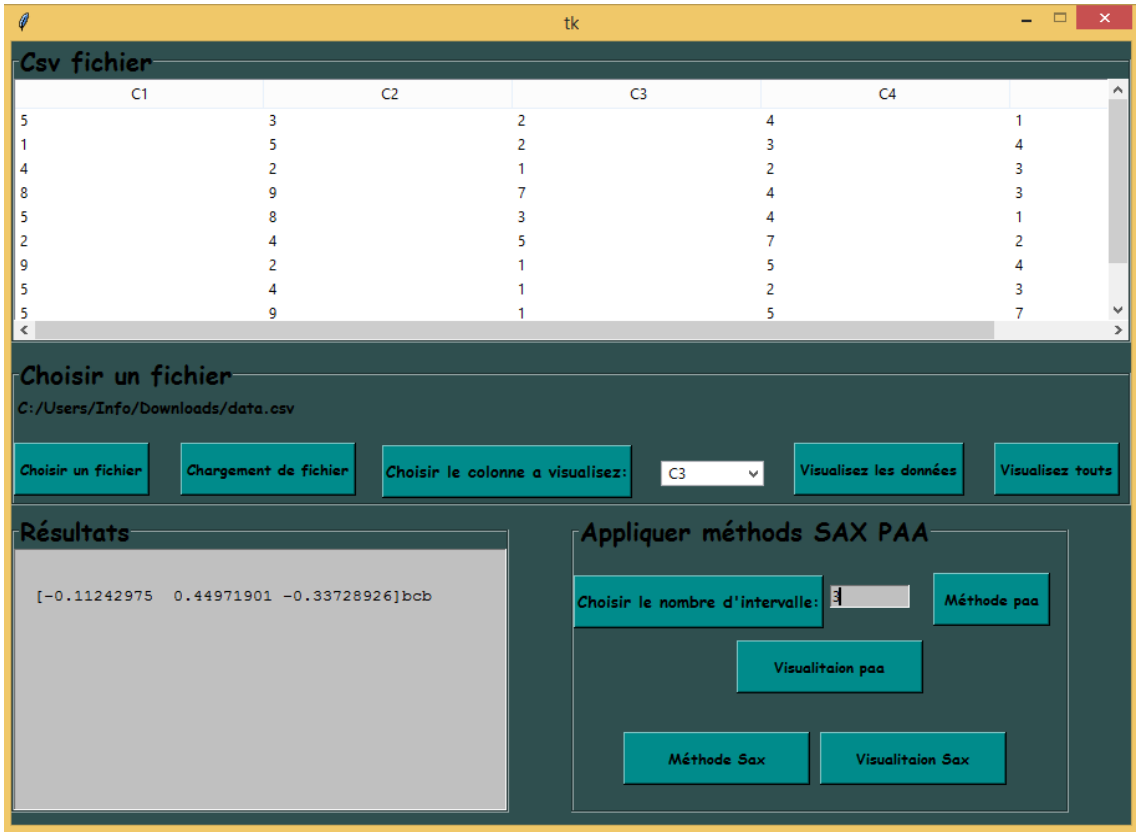


Figure 32 - *Résultat de l'application de SAX*

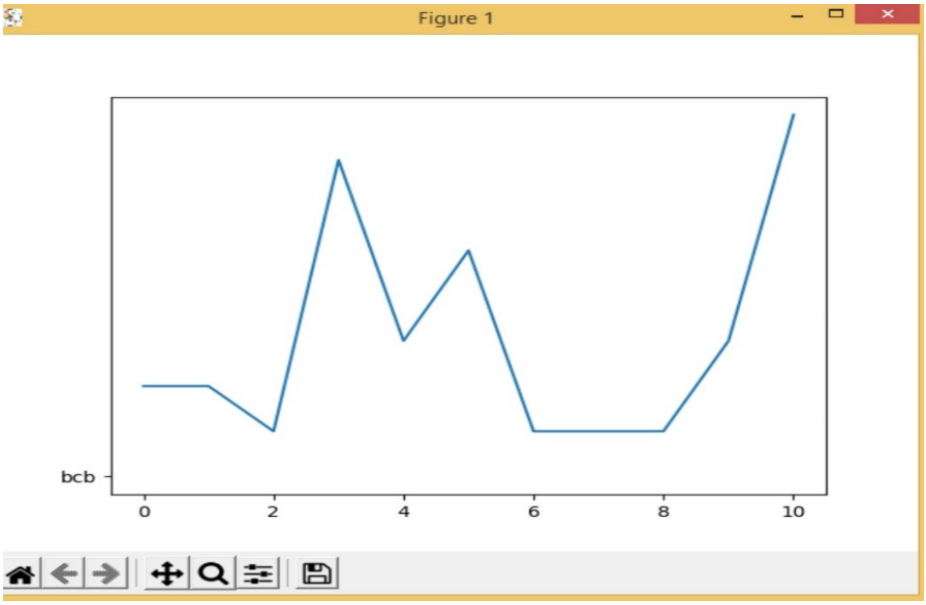


Figure 33 - *Visualisation des données après application de la méthode SAX*

Enfin l'algorithme *PrefixSpan* va projeter des bases de données de séquences en évaluant les occurrences fréquentes de sous-séquences, la projection est faite sur des préfixes fréquents.

3.8. Conclusion

Dans ce chapitre, nous avons détaillé l'approche que nous avons adopté pour l'analyse des séries temporelles en utilisant les motifs séquentiels. En fait, pour pouvoir appliquer les techniques d'extraction des motifs séquentiels sur des séries temporelles, il fallait convertir ces dernières en une représentation symbolique. Le choix s'est porté sur la méthode *SAX*, l'une des méthodes les plus performantes dans ce domaine. La suite était d'appliquer l'algorithme *PrefixSpan* pour extraire des motifs à partir de la nouvelle représentation. Les motifs extraits représentent en effet des séquences qui se répètent souvent, et sont obtenues par l'opération inverse.

Conclusion générale

Dans ce travail, nous nous sommes intéressés à l'analyse des séries temporelles, en étudiant l'applicabilité des méthodes d'extraction de motifs fréquents sur cette problématique. Pour ce fait, nous nous sommes positionnés d'abord dans le domaine du data mining, le premier chapitre lui a été alors consacré. Nous avons donné quelques unes de ses définitions, et présenté le processus plus général d'extraction de connaissances à partir des données. Nous avons aussi présenté les tâches de data mining et ses domaines d'application. Dans ce chapitre, nous avons aussi survolé les séries temporelles, et donné quelques unes de leurs applications.

Dans le deuxième chapitre, nous sommes passés à une tâche particulière du data mining, à savoir l'extraction des règles d'association. Dans ce contexte, nous avons introduit cette problématique, et présenté l'algorithme fondateur et célèbre *Apriori*. L'idée était que cet algorithme constitue la base pour la plupart de ses successeurs. Un accent particulier a été mis ensuite sur l'algorithme *FP-Growth*, l'un des algorithmes les plus utilisés aussi dans ce domaine. L'idée était de bien comprendre ce domaine.

Dans le troisième chapitre, nous nous sommes mis à expliquer l'approche que nous avons adoptée dans notre travail. Après l'introduction de quelques concepts clés, nous avons introduit la démarche suivie. Le début était avec les méthodes de représentation *PAA* et *SAX*, que nous avons détaillé et illustré par des exemples. La suite était avec l'algorithme *PrefixSpan* que nous avons choisit pour l'extraction des motifs.

Les résultats ont montré l'applicabilité de cette approche, et la possibilité de l'utiliser dans des tâches bien concrètes d'analyse de séries temporelles. Ces dans ce sen que s'inscrit les perspectives futures de ce travail, à savoir aller au bout d'une tâche d'analyse de séries temporelle, comme la classification ou le clustering.

.

Bibliographie

[1] Aghabozorgi A. S., Shirkhorshidi A.S, Wah T. Y., *Time-series clustering –A decade review*, Information Systems, 53 (2015), 16-38.

[2] Berry, M.-J.-R., Linoff, G.-S. : *Data Mining Techniques : For Marketing, Sales, andCustomer Relationship Management*. Second Edition, 2004.

[3] Bramer, M. : *Principles of Data Mining*. Fourth Edition. Springer-Verlag, 2020.

[4] Chanda, A.-K., Saha, S., Nishi, M.-A., Samiullah, M., Ahmed, C.-F. : *An efficient approach to mine flexible periodic patterns in time series databases*. Engineering Applications of Artificial Intelligence, 44 (2015), 46–63.

[5] Fayyad U., P-Shapiro G., Smyth P. : *From Data Mining to Knowledge Discovery in Databases* ; American Association for Artificial Intelligence. 0738-4602-1996.

[6] Han, J., Kamber, M. : *Data mining concepts and techniques*. Second Edition. Diane Cerra San Francisco.

[7] Lefébure, R., Venturi, G. : *Le Data Mining*. Editions Eyrolles. 2001.

[8] Liao T. W., *Clustering of time series data –a Survey* ; Pattern Recognition 38 (2005) 1857-1874.

[9] Tufféry, S. : *Data mining et statistique décisionnelle*. Editions Technip, 2007.

Webographie

[1] https://www.talend.com/fr/resources/data-mining-techniques

[2] upgrad.com/blog/data-mining-techniques

[3] https://towardsdatascience.com/a-brief-introduction-to-time-series-classification-algorithms-7b4284d31b97

[4] https://www.researchgate.net/publication/312529649_Efficient_Learning_of_Timeseries_Shapelets

[5] https://gaz.wiki/wiki/fr/PyCharm

[6] https://info.blaisepascal.fr/langages/python