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 :

BENSABER Yasmine et BENNAMA Fethi

THEME :

Etude des approches d'extraction et d'analyse des motifs dans les données de capteurs

Soutenu le :

Devant le jury composé de :

ABDALLAH BENSALLOUA C.	MCA	Université de Mostaganem	Président
HABIB ZAHMANI Mohammed	MCA	Université de Mostaganem	Examinateur
BENAMEUR Abdelkader	MAA	Université de Mostaganem	Encadreur

Année Universitaire 2021-2022

Résumé

L'analyse des données de capteurs a toujours attiré beaucoup d'attention, en particulier avec son utilisation pour la prise de décision. L'un des cas d'application les plus fréquents est la maintenance industrielle, où on cherche à trouver des motifs (séquences) qui se répètent dans les données issues des capteurs qui surveillent un équipement durant son fonctionnement. Les motifs ainsi trouvés, peuvent être utilisés pour prédire la défaillance de l'équipement, ou pour bien comprendre son comportement. La prise de la décision de procéder à des actions préventives ou correctives sera ainsi améliorée. Dans ce travail, nous allons étudier les solutions proposées pour extraire et analyser les motifs fréquents sur la base d'observations passées. À cette fin, nous ciblons l'étape de prétraitement des données, ainsi que celle de traitement central permettant l'extraction de motifs proprement dite.

Mots-clés:

Analyse des motifs, Data mining, Données de capteurs, Prédiction

Abstract

The analysis of sensor data has always attracted a lot of attention, especially with its use for decision making. One of the most frequent application cases is industrial maintenance, where one seeks to find repeating patterns (sequences) in the data from sensors that monitor equipment during its operation. The patterns thus found can be used to predict the failure of the equipment, or to better understand its behavior. Decision-making to proceed with preventive or corrective actions will thus be improved. In this work, we will study the proposed solutions to extract and analyze frequent patterns based on past observations. To this end, we target the data pre-processing step, as well as the central processing step allowing the actual pattern extraction.

Keywords:

Pattern analysis, Data mining, Sensor data, Prediction

ملخص

لطالما جذب تحليل بيانات المستشعرات الكثير من الاهتمام، لا سيما مع استخدامه في صنع القرار. واحدة من أكثر حالات التطبيق شيوعًا هي الصيانة الصناعي ة، حيث يسعى المرء إلى العثور على أنماط متكررة (تسلسلات) في البيانات من أجهزة الاستشعار التي تراقب المعدات أثناء تشغيلها. يمكن استخدام الأنماط التي تم العثور عليها للتنبؤ بفشل المعدات، أو لفهم سلوكها بشكل أفضل. وبالتالي، سيتم تحسين عملية صنع القرار للمضي قدمًا في الإجراءات الوقائية أو التصحيحية. في هذا العمل، سوف ندرس الحلول المقترحة لاستخراج وتحليل الأنماط المتكررة بناءً على الملاحظات السابقة. تحقيقًا لهذه الغاية، نستهدف خطوة المعالجة المسبقة للبيانات، بالإضافة إلى خطوة المعالجة المركزية التي تسمح باستخراج النمط الفعلي.

كلمات مفتاحية

تحليل الأنماط، التنقيب عن البيانات، بيانات الاستشعار، التنبؤ

Dédicaces

C'est avec grand respect et gratitude que je tiens à exprimer toute ma reconnaissance et ma sympathie et dédier ce travail modeste à trois personnes qui me sont chers, ma maman RIHANNA BELAYACHI , ma deuxième mère que j'aime KHADIDJA ZAAF « TITA » ainsi ma grand-mère adoré CHACHA BELAYACHI , les mots ne sauraient exprimer l'immense et profonde gratitude que je leur témoigne ici pour leurs précieux soutien, pour leurs patience, pour avoir crus en moi, pour leurs sourires réconfortants et pour leurs sacrifices qui m'ont permis d'atteindre cette étape dans ma vie et qu'ils m'ont jamais cessé de consentir pour mon instruction et mon bien être dieu me les gardes et les protèges.

-Mon cher papa MEJDOUB BENSABER.

- *Mon frère ABDALLAH, ma sœur MOUNIRA, ma meilleure amie CHAIMA KEMMINI ainsi mon ami MOHAMED BOUKHATEM pour leur patience et d'avoir tendu chaleureusement leurs bras.*
- *Toute ma famille et surtout mes grands-pères décédés BELAYACHI MOHAMED SGUIR et BENSABER ABDALLAH WELD MAKHLOUF.*

- *Tous ceux qui ont participé de près ou de loin à la réalisation de ce travail.*
- *Tous mes enseignants tout au long des cycles de mes études.*
- *Toute la promotion 2021/2022 de ISI.*

BENSABER Yasmine

Dédicaces

Au nom du dieu le clément et le miséricordieux louange à ALLAH le tout puissant.

Je dédie ce modeste travail en signe de respect, reconnaissance et de remerciement :

A mes parents,

Sans leurs soutiens et leurs conseils, mes accomplissements n'auraient pas eu lieu, ils ont été derrière moi dans chacun de mes pas tout au long de ma vie, ma plus profonde gratitude leurs ai exprimé, aucun mot ne pourrait qualifier l'estime que je ne leur porte ni le bien qu'ils m'ont fait, apporter et donner.

A tous mes ami(e), frères et sœurs et tous ceux qui me sont chers.

- *Tous mes enseignants tout au long des cycles de mes études.*
- *Toute la promotion 2021/2022 de ISI.*

BENNAMA Fethi

Remerciements

Notre parfaite gratitude et nos remerciements à Allah le plus puissant qui nous a donné la force, le courage et la volonté pour mener à bien ce modeste travail.

C'est avec une profonde reconnaissance et considération particulière que je remercie mon encadrant Monsieur ***Abdelkader BENAMEUR*** *pour son soutien, ses conseils judicieux et sa grande bienveillance durant l'élaboration de cet ouvrage.*

J'exprime aussi ma reconnaissance à tous les membres du juryd'avoir accepté de lire, de présider et examiner ce modeste travail et d'apporter les critiques nécessaires à la mise en forme de ceprojet.

Mes vifs remerciements pour l'ensemble des enseignants du département de l'informatique qui ont contribué à notre formation d'ingénierie des systèmes d'information.

Je tiens aussi à remercier mon binôme Fethi BENNAMA pour sa bienveillance et son sérieux durant la réalisation du mémoire.

Enfin, à tous ceux qui nous ont aidés de près ou de loin pour laréalisation de ce projet de fin d'étude, qu'ils trouvent ici, l'expression de nos sincères remerciements.

BENSABER Yasmine

Remerciements

En tout premier lieu, je remercie le bon Dieu, tout puissant, de m'avoir donné la force pour survivre, ainsi que l'audace pour dépasser toutes les difficultés.

Jetender à exprimer toute ma reconnaissance à mon Directeur de mémoire Monsieur ***Abdelkader BENAMEUR****. Je le remercie de m'avoir encadré, orienté et aidé. J'ai profité pendant longtemps du savoir et du savoir-faire dont j'ai pu bénéficier au cours de nombreuses discussions. J'aimerais aussi le remercier pour l'autonomie qu'il m'a accordé, et ses précieux conseils qui m'ont permis de mener à bien ce travail.*

J'exprime toute ma reconnaissance à mes jurys de la faculté des sciences exactes et de l'informatique de l'université d'Abd El Hamid Ibn Badis, trouvent ici l'expression de mes vifs

remerciements pour avoir bien voulu juger ce travail.

J'adresse mes sincères remerciements à tous les intervenants et toutes les personnes qui par leurs paroles, leurs contributions dans ce travail, leurs écrits, leurs conseils et leurs critiques ont guidé mesréflexions et ont accepté de répondre à mes questions

durant mes recherches. Je ne peux achever ce projet, sans exprimer mes sincères gratitudes à

tous les professeurs de notre faculté, pour leur dévouement et leur assistance tout au long de

ma formation.

Je remercie mes très chers parents, qui ont toujours été là pour moi,

« Vous avez tout sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous m'avez données un magnifique modèle de labeur et de persévérance. Je suis redevables d'une éducation dont me suis fières ».

Enfin, Je remercie très spécialement ma famille pour leur encouragement.

BENNAMA Fethi

Liste des figures

Liste des tableaux

Liste des abréviations

ECG : Electrocardiogramme

WH : Capteur de hauteur d'eau

RP : Capteur de précipitation d'eau

WF : Capteur de débit de sortie d'eau

RA : Règles d'association

MLP : Réseaux de neurones multicouches

KNN : Nearest Neighbor Classification

STA : Modèle approximatif

BIP : Behavior, Interaction, Priority

LTL : Logique temporelle linéaire

SAX : Symbolic Aggregate approXimation

MDL : La longueur de description minimale

Table des matières

Introduction générale

La capacité de faire des prédictions sur des événements futurs est au cœur d'une grande partie de la science ; il n'est donc pas surprenant que la prédiction ait suscité un grand intérêt dans la communauté d'extraction de connaissances au cours de la dernière décennie. En générale, l'extraction de connaissances désigne l'analyse de données depuis différentes perspectives et le fait de transformer ces données en informations utiles, en établissant des relations entre les données ou en repérant des motifs. La plupart des travaux antérieurs ont tenté de prédire l'avenir sur la base de l'actuelle valeur d'un phénomène. Cela peut également avoir une grande utilité dans l'étude des séries temporelles.

Prenant l'exemple de l'analyse du comportement des capteurs qui est l'un des défis importants en raison de l'utilisation croissante de ces capteurs pour prendre une décision dans de nombreuses applications de décision et promouvoir l'automatisation dans les systèmes d'information en fournissant des services efficaces et intelligents.

Les capteurs sont des dispositifs qui permettent de mesurer les informations d'état au fil du temps et de surveiller les composants physiques en générant des données chronologiques. Par exemple, un capteur environnemental mesurera la température ambiante en continu, tandis qu'un électrocardiogramme (ECG) mesurera les paramètres du rythme cardiaque d'un sujet. Ces données ont généralement des dépendances implicites intégrées aux valeurs reçues. Par exemple, les valeurs adjacentes enregistrées par un capteur de température seront généralement variées progressivement dans le temps, et ce facteur doit être explicitement utilisé dans l'exploration de données à traiter. La nature de la dépendance temporelle peut varier considérablement selon l'application.

Nous nous intéressons à l'étude de cas d'applications dans le domaine de la maintenance industrielle où on cherche à trouver des motifs (séquences) qui se répètent dans les données issues de capteurs qui surveillent un équipement physique durant son

fonctionnement, afin de bien comprendre son comportement et détecter toute panne qui peut influencer le bon fonctionnement de l'ensemble du système. La prise de la décision de procéder à des actions préventives ou correctives sera ainsi améliorée. À cette fin, nous ciblons l'étape de prétraitement des données, ainsi que celle de traitement central permettant l'extraction de motifs proprement dite.

Organisation de travail

Afin d'aborder tous ces aspects, nous allons étudier les solutions proposées pour extraire et analyser les motifs fréquents sur la base d'observations passées en ciblant l'étape de prétraitement des données ainsi que celle de traitement central permettant l'extraction de motifs proprement dite. Nous organisons ce mémoire, en plus d'une introduction générale et une conclusion générale, en quatre chapitres :

Dans le premier chapitre, nous allons se positionner dans un domaine qui inclut notre problématique, il s'agit du data mining où nous présentons un aperçu de son processus, ses tâches et technique de bases en terminons avec les différents algorithmes existants.

Pour le deuxième chapitre, nous allons faire une projection plus spécifique d'un type de donnée de data mining en relation avec notre problématique, il s'agit des séries temporelles où nous étudierons l'aspect algorithmique et méthodique proposé par des chercheurs comme solution pour extraire et analyser les motifs fréquents sur la base d'observations passées.

Dans le troisième chapitre, nous allons nous étaler sur notre approche concernant l'extraction et l'analyse des motifs à partir des séries temporelles. A travers un cas d'application réel, l'approche sera mise en évidence dans son intégralité.

Le quatrième chapitre décrit la mise en œuvre de la solution logicielle qui mettra en action l'approche proposée. Cette description sera accompagnée d'une série d'illustrations.

Des perspectives futures du travail clôtureront ce rapport dans la conclusion générale.

Chapitre 1

Data mining : un survol

1.1 Introduction

L'analyse des données de capteurs a toujours attiré beaucoup d'attention en particulier avec son utilisation pour la prise de décision. Au cours de ce chapitre, nous allons se positionner sur un domaine qui inclut notre problématique, il s'agit du data mining où nous allons étudier les différents outils théoriques en se basant sur l'aspect algorithmique et méthodique.

1.2 Définition du data mining

La fouille de données est l'étude de la collecte, du nettoyage, du traitement, de l'analyse et de l'obtention d'informations utiles aperçues à partir des données. Il existe une grande variation en termes de domaines, de problèmes, d'applications, de formulations et représentations de données rencontrées dans des applications réelles. Par conséquent, « data mining » est un terme générique qui est utilisé pour décrire ces différents aspects de traitement de l'information (1) (2).

1.3 Domaines d'applications

Data mining est une discipline jeune avec des applications vastes et diverses. De nombreux outils ont été développés pour des applications spécifiques à un domaine. Cela inclut la finance, le commerce de détail, les télécommunications, la détection et la prévention des intrusions, data mining pour les données biologiques et pour les applications scientifiques (figure 1).

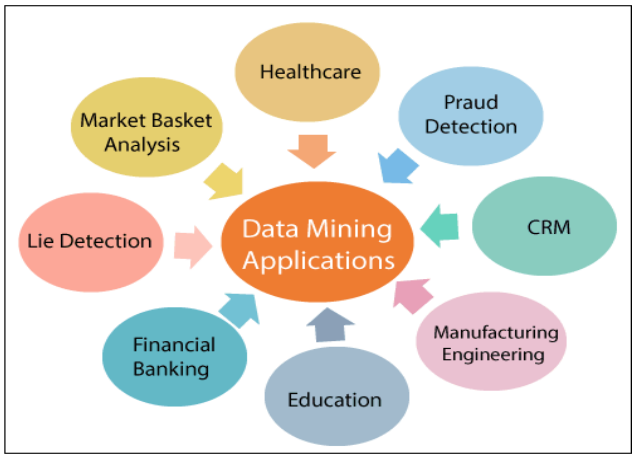


Figure 1 : *Applications du Data mining.*

1.3.1 Marketing et ventes

Il s'agit de domaines dans lesquels les entreprises possèdent d'énormes volumes de données enregistrées avec précision et les prédictions elles-mêmes sont le principal intérêt. Prenant l'exemple des compagnies de téléphonie mobile qui luttent contre le désabonnement en détectant des modèles de comportement qui pourrait bénéficier de nouveaux services, puis faire la publicité de ces services pour fidéliser leur clientèle. Incitations fournies spécifiquement pour conserver les clients peuvent être coûteux, et data mining réussie leur permet de cibler précisément les clients où ils sont susceptibles de rapporter le maximum d'avantages (figure 2) (1) (2) (3).

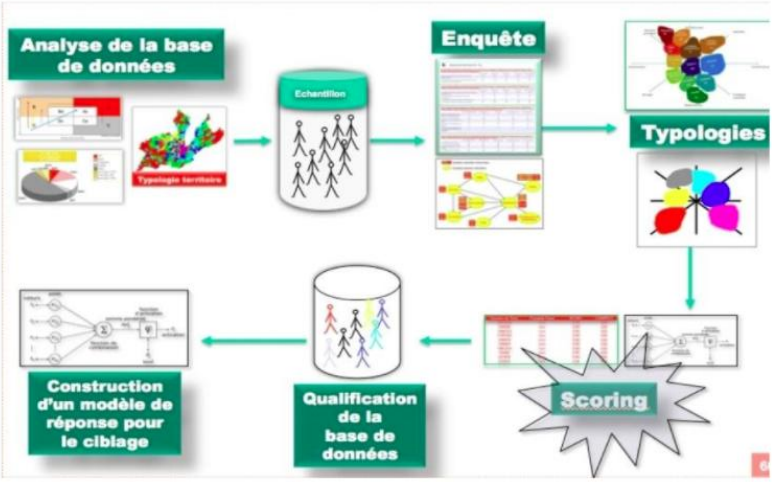


Figure 2 : *Etudes du marché.*

1.3.2 Diagnostic médical

Considérons un ensemble de séries temporelles des lectures ECG qui sont recueillies auprès de différents patients cardiaques (figure 3). Il est souhaitable de déterminer la série anormale à partir de cet ensemble. Cette application peut être associée à différents problèmes, selon la nature des données d'entrée disponibles.

Par exemple, considérons le cas où aucun exemple précédent de séries ECG anormales n'est disponible. Dans de tels cas, le problème peut être mappé à la valeur aberrante du problème de détection. Une série temporelle qui diffère considérablement des autres séries, les données peuvent être considérées comme une valeur aberrante. Cependant, la méthodologie de la solution change considérablement si des exemples précédents de séries normales et anormales sont disponibles. Dans de tels cas, le problème correspond à un problème de classification sur des données de séries temporelles (2).

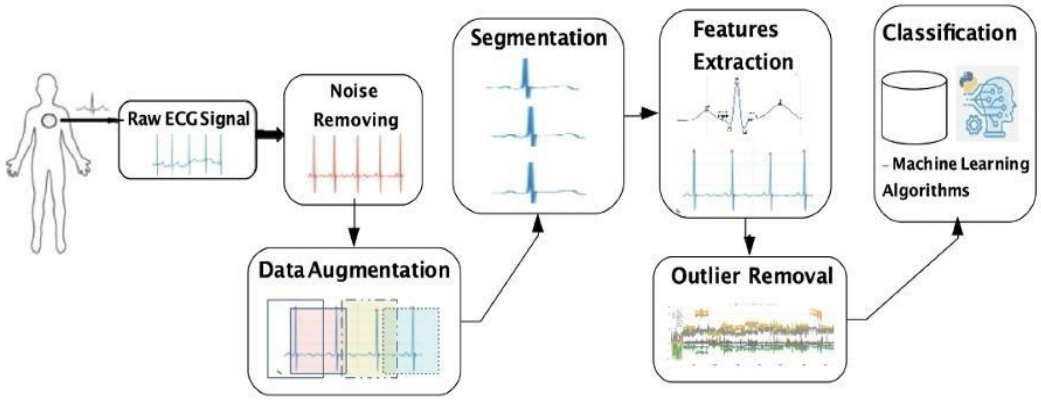


Figure 3 : *Lectures ECG de patients cardiaques* (2).

1.3.3 Secteur industriel

Les entreprises spécialisées dans le domaine industriel des barrages cherchent à automatiser et contrôler les lectures de capteurs sans fil qui mesurent les apports d'eau tel que le capteur de hauteur d'eau, le capteur de précipitations d'eau et le capteur de débit de sortie d'eau en analysant leur comportement et détecter toute panne ou anomalie par des approches basées sur des méthodes statistiques et probabilistes (4).

1.4 Processus de data mining

La fouille de données est en effet un processus complexe et en plusieurs étapes présenté comme suit (figure 4) :

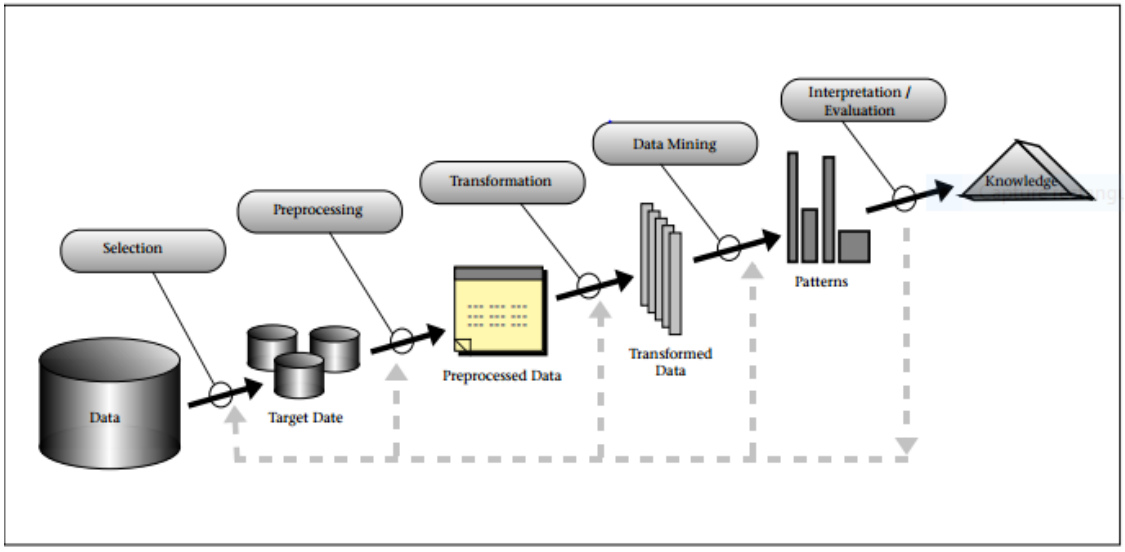


Figure 4 : *Processus de data mining* (5).

1.4.1 Collecte de données

La collecte de données peut nécessiter l'utilisation de matériel spécialisé tel qu'un réseau de capteurs, travail manuel tel que la collecte d'enquêtes auprès des utilisateurs ou outils logiciels tel qu'un moteur d'exploration de documents Web pour collecter des documents. Alors que cette étape est hautement spécifique à l'application et souvent en dehors du domaine de l'analyse de data mining, elle est d'une importance cruciale car de bons choix à ce stade peuvent avoir un impact significatif sur le processus de travail. Après la phase de collecte, les données sont souvent stockées dans une base de données (SQL, etc.), ou plus généralement, un entrepôt de données pour le traitement (2).

1.4.2 Sélection de données

En fonction de la collecte initiale de données, nous pouvons commencer à choisir les données pertinentes pour nos objectifs de data mining. En général, les données peuvent être sélectionnées de deux manières (1) (3) :

- Sélection des enregistrements (lignes) : implique des décisions concernant les objets à inclure.
- Sélection des attributs ou des caractéristiques (colonnes) : implique des décisions concernant l'utilisation de caractéristiques.

1.4.3 Prétraitement

C'est une phase très importante pour mesurer la qualité des données (figure 5) en termes de (1) (2) (3):

- Précision : correcte ou fausse, exacte ou non.
- Intégralité: non enregistrée, indisponible.
- Cohérence : certaines modifiées mais d'autres non, pendantes.
- Ponctualité : mise à jour en temps opportun ?
- Crédibilité : dans quelle mesure les données sont-elles fiables ?
- Interprétabilité : avec quelle facilité les données peuvent-elles être comprises ?

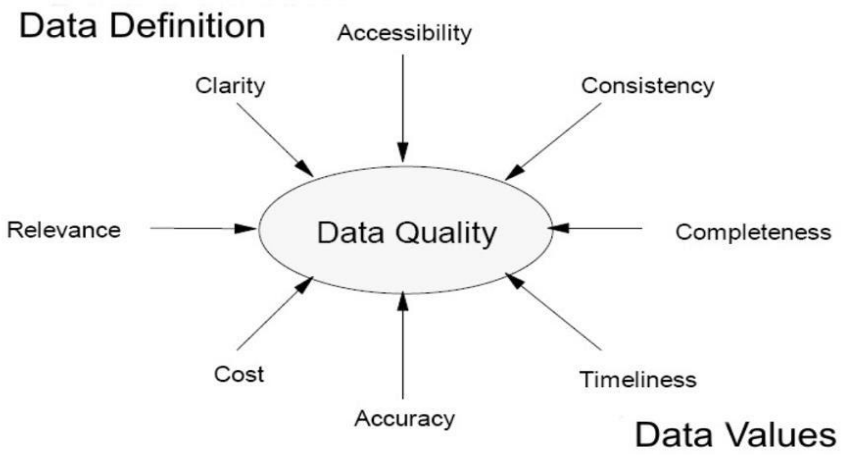


Figure 5 : *Qualité de données* (2).

Lorsque les données sont collectées, elles ne sont souvent pas sous une forme adaptée au traitement. Par exemple, les données peuvent être codées dans des journaux complexes ou des documents de forme libre. Dans de nombreux cas, différents types de données peuvent être arbitrairement mélangés dans un document de forme libre. Pour rendre les données adaptées à traitement, il est essentiel de les transformer dans un format adapté aux algorithmes de fouille de données en appliquant plusieurs tâches comme suit (2) :

1.4.3.1 Data cleaning

Beaucoup de données sont potentiellement (1) (6):

- Incorrectes : à cause d'un instrument défectueux, d'une erreur humaine ou informatique, d'une erreur de transmission, etc.
- Incomplètes : manque de valeurs d'attribut, manque de certains attributs d'intérêt ou ne contenant que des données agrégées (données manquantes).
- Bruyantes : contenant du bruit, des erreurs ou des valeurs aberrantes (une erreur).
- Incohérentes : contenant des divergences dans les codes ou les noms.

Nous gérons les données incomplètes par ignorer le tuple ou le remplir automatiquement avec une constante globale où nous utilisons la moyenne de l'attribut pour tous les échantillons appartenant à la même classe.

1.4.3.2 Réduction des données

Nous utilisons la réduction des données parce que l'analyse de données complexes peut prendre beaucoup de temps pour s'exécuter sur l'ensemble de données complet et nous suivons la stratégie (figure 6 résume les stratégies) (2) :

- Réduction de la dimensionnalité (suppression des attributs sans importance).
- Réduction de la numérotation (réduction des données) par :
  - Modèles de régression et log-linéaires.
  - Histogrammes, clustering, échantillonnage.
  - Agrégation de cubes de données
- Compression de données.

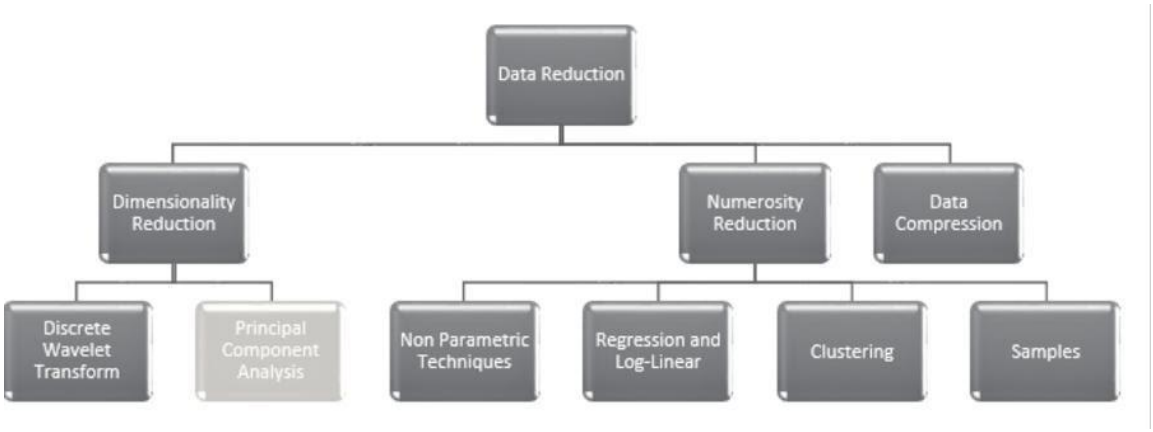


Figure 6 : *Réduction de données* (2).

1.4.4 Transformation et discrétisation des données

La transformation des données est une fonction qui mappe l'ensemble des valeurs d'un attribut donné à un nouvel ensemble de valeurs de remplacement. Chaque ancienne valeur peut être identifiée avec l'une des nouvelles valeurs, parmi les méthodes utilisées :

- Lissage : supprimez le bruit des données.
- Création de caractéristiques : nouveaux attributs construits à partir de ceux donnés.
- Agrégation : résumé, construction de cubes de données.
- Normalisation : mise à l'échelle pour se situer dans une plage spécifiée plus petite.
  - Normalisation min-max : vers [*new_minA, new_maxA*]

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

  - Normalisation du score *z* : (μ : moyenne, σ: écart type):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  - Normalisation par mise à l'échelle décimale

$$v' = \frac{V}{10^J}$$

- Discrétisation : diviser la plage d'un attribut continu en intervalles.

1.4.5 Data Mining

La phase de data mining est une phase cruciale itérative et interactive qui conduit à la découverte de modèles dans un ensemble de données préalablement préparé d'une manière spécifique (1) (3) (2) (6).

1.4.6 Test et validation

Dans une tâche d'apprentissage supervisé ou non supervisé, le but est d'estimer plusieurs modèles afin de prédire au mieux la variable cible pour des données futures. Pour tester le modèle, il faut procéder en distinguant au moins deux ensembles de données :

- Un ensemble de données d'apprentissage.
- Un ensemble de données de validation.

1.4.6.1 K-Validation croisée

La validation croisée (figure 7) consiste à (6) :

- Séparer aléatoirement l'ensemble des données annotées en *k* sous-ensembles.
- Utiliser un sous-ensemble comme ensemble de test *T*.
- Utiliser l'union des *k-1* sous-ensembles restants comme ensemble d'entraînement *E*.
- En changeant chaque fois l'ensemble de validation, on voit qu'une *k* validations croisée permet d'avoir *k* paires d'échantillons (*E, T*) et ainsi que *k* estimations de l'erreur de prédiction.
- On moyenne l'ensemble des *k* mesures d'erreurs afin d'avoir une estimation plus robuste de l'erreur de prédiction.

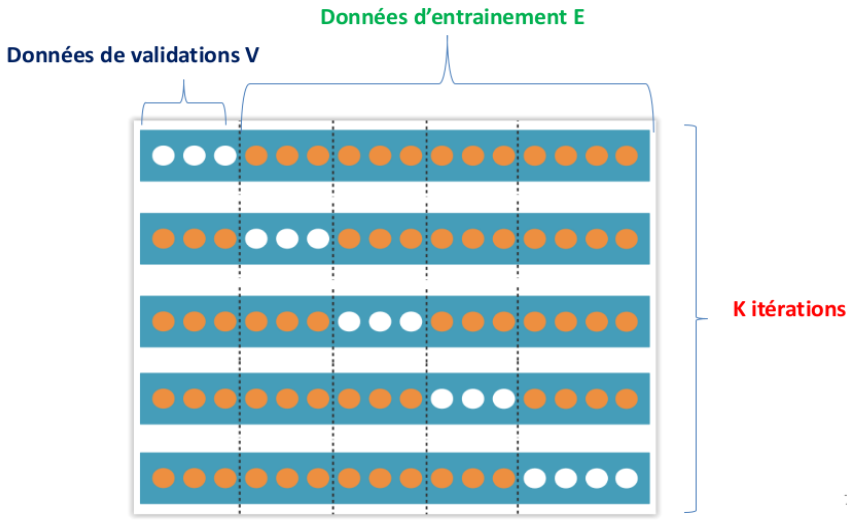


Figure 7 : *K-Validation croisée.*

1.4.6.2 Métriques d'évaluation

On évalue un modèle appris via un algorithme data mining pour (3) (6) :

- Estimer sa performance.
- Déduire la confiance qu'on peut accorder à ce modèle.
- Eviter de tomber sur un sur-apprentissage, sous-apprentissage.

Le processus d'évaluation passe par les 3 étapes:

- Faire les prédictions sur les exemples du corpus de test.
- Comparer les classes prédites avec les vraies classes.
- Calculer des mesures d'évaluation.

Plusieurs métriques d'évaluation pour les différentes tâches :

- Classification: (accuracy, precision, recall, F1-score, ROC, AUC, …)
- Régression (MSE, MAE)
- Statistique (Corrélation)
- Vision par ordinateur (PSNR, SSIM, IoU)
- NLP (Perplexity, BLEU score)
- Deep Learning (Inception score, Frechet Inception distance)

1.4.6.3 Matrice de confusion

Une matrice de confusion est utilisée pour avoir une image complète de la performance d'un modèle. Elle est définie de la manière suivante (figure 8) (6).

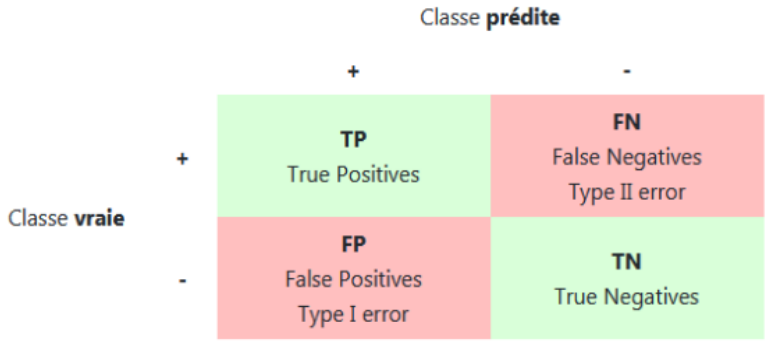


Figure 8: *Matrice de confusion.*

Par rapport à une classe *C*, un échantillon est dit positif s'il appartient à la classe *C*. Sinon il est dit négatif :

- TP (*True positives* / *Vrais positifs*) : les exemples positifs qui sont prédits positifs.
- FP (*False positives / Faux positifs*) : les exemples négatifs qui sont prédits positifs.
- TN (*True negatives / Vrais négatifs*): les exemples négatifs qui sont prédits négatifs.
- FN (*False negatives / Faux négatifs*): les exemples positifs qui sont prédits négatifs.

Il existe plusieurs mesures d'évaluation comme montre la figure ci-dessous :

Indicateur	Formule	Interprétation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Performance globale du modèle
Précision	$\frac{TP}{TP + FP}$	À quel point les prédictions positives sont précises
Rappel Sensibilité	$\frac{TP}{TP + FN}$	Couverture des observations vraiment positives
Spécificité	$\frac{TN}{TN + FP}$	Couverture des observations vraiment négatives
F-mesure	$\frac{2TP}{2TP + FP + FN}$	Indicateur hybride utilisé pour les classes non-balancées

Figure 9 : *Mesures d'évaluation.*

1.5 Techniques de data mining

Les deux objectifs principaux de haut niveau de data mining dans la pratique ont tendance à être la prédiction et la description (6). La prédiction implique l'utilisation de certaines variables ou champs dans la base de données pour prédire des valeurs inconnues au futures, d'autres variables d'intérêt et de description se concentrent sur la recherche de motifs interprétables par l'homme décrivant les données. Bien que les frontières entre prédiction et description ne soient pas nettes (certains des modèles peuvent être descriptifs, dans la mesure où ils sont compréhensibles, et vice versa) la distinction est utile pour comprendre l'objectif global de découverte. L'importance relative de prédiction et de description pour les applications du data mining peuvent varier considérablement. Les objectifs de prédiction et description peuvent être réalisée en utilisant une variété de tâches particulières de data mining (1) (2).

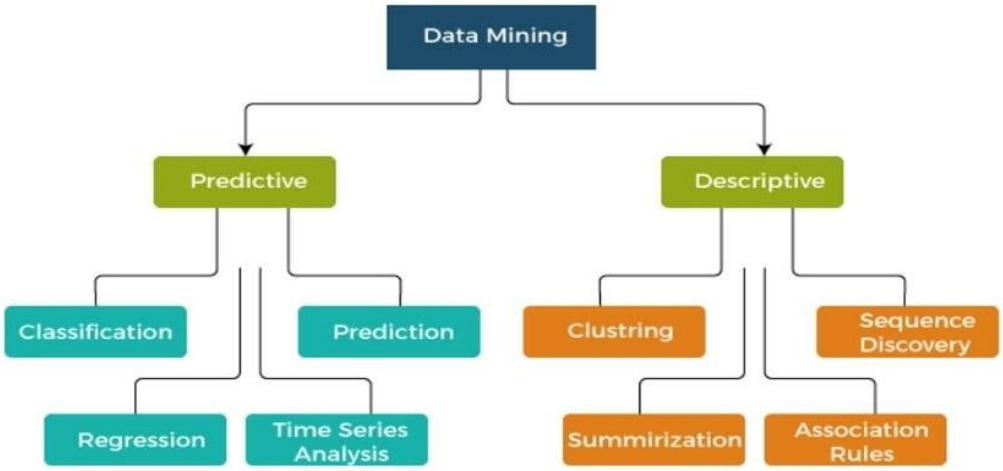


Figure 10 : *Techniques de base de data mining.*

1.5.1 Techniques prédictives

1.5.1.1 Régression

La régression est l'apprentissage d'une fonction qui mappe un élément de données à une variable de prédiction à valeur réelle.

Les applications de régression sont nombreuses, par exemple, estimant la probabilité qu'un patient survivra dépend des résultats d'un ensemble de tests de diagnostic, prédisant la demande des consommateurs pour un nouveau produit en tant que fonction des dépenses publicitaires, et prédiquer des séries temporelles où les variables d'entrée peuvent être des versions décalées dans le temps de la prédiction variable. La figure 11 montre le résultat de simple régression linéaire où la dette totale est ajustée comme une fonction linéaire du revenu : car seule une faible corrélation existe entre les deux variables (5).

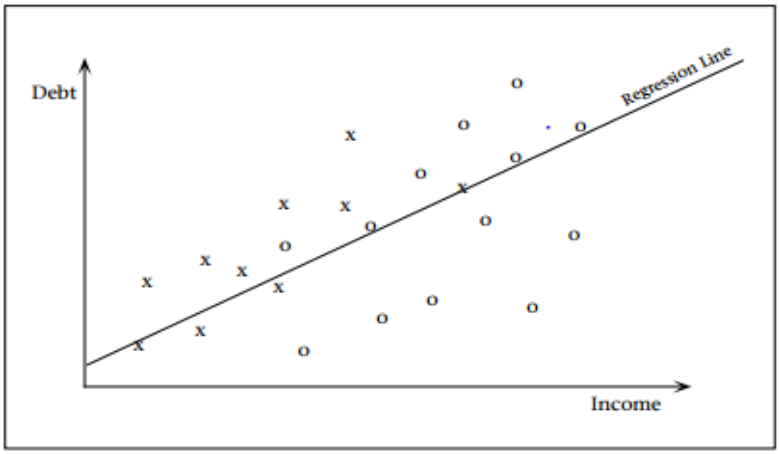


Figure 11 : *Régression (5).*

1.5.1.2 Classification

La classification apprend une fonction qui mappe (classe) un élément de données dans l'une des classes prédéfinies (1) (2) (3).

La figure 12 montre un partitionnement simple des données de prêt en deux régions de classe ; notons qu'il n'est pas possible de séparer parfaitement les classes à l'aide d'un linéaire limite de décision. La banque pourrait vouloir utiliser les régions de classification pour automatiquement décider si les futurs demandeurs de prêt seront prêtés ou non (5).

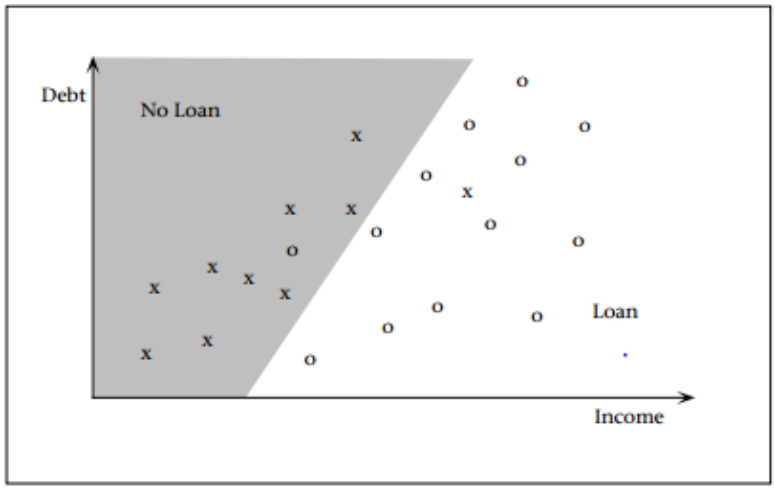


Figure 12 : *Classification (5).*

1.5.2 Techniques descriptives

1.5.2.1 Clustering

Le clustering est une tâche descriptive courante où l'on cherche à identifier un ensemble fini de catégories ou clusters pour décrire les données. Les catégories peuvent être mutuellement exclusives et exhaustives ou consister en une représentation, telle que hiérarchique ou sur catégories de rodage. Exemples d'applications du clustering dans un contexte de découverte de connaissances incluant la découverte de sous-populations homogènes pour les consommateurs dans les bases de données marketing et l'identification de sous catégories de spectres à partir de mesures infrarouges du ciel (1) (3).

La figure 13 montre un cluster possible, la répartition de l'ensemble de données sur les prêts en trois groupes ; notons que les clusters se chevauchent, permettant aux données (points) d'appartenir à plus d'un cluster(5).

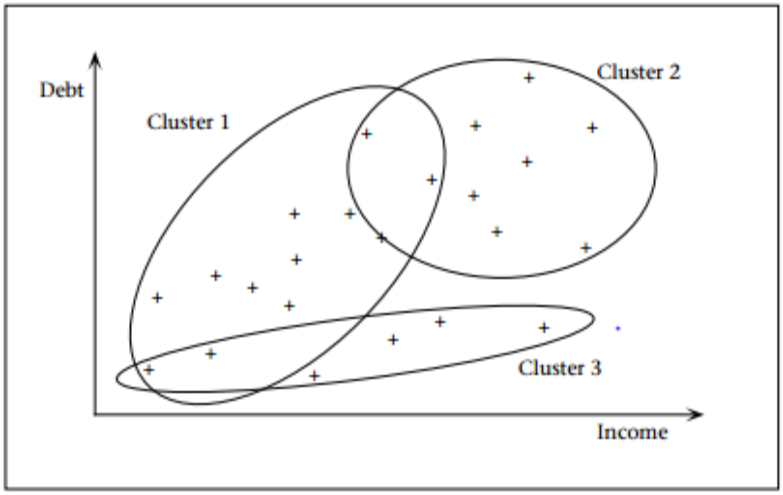


Figure 13 : *Clustering (5).*

1.5.2.2 Règles d'association

Une règle d'association est un pattern qui prédit qu'un événement va se réaliser avec une certaine probabilité (6). Les règles d'association sont souvent des conditions logiques (if / then) qui permettent d'une part, de trouver des relations entre les différentes composantes des données éparpillées dans une base de données et d'autre part, de trouver les relations entre les objets fréquemment utilisés ensemble (1) (2).

Pour calculer les règles de décision on utilisera 2 critères : le support et la confiance.

Le support : il représente la fiabilité. Ce critère permet de fixer un seuil en dessous duquel les règles ne sont pas considérées comme fiables. Le support d'une règle $F_1 \rightarrow F_2$ correspond à la probabilité $P(F_1 \cap F_2)$.

- *Support(X) = Nombre de transactions où X apparaît / Nombre total de transactions*

La confiance : elle représente la précision de la règle et peut être vue comme la probabilité conditionnelle $Confiance\ (F_1 \rightarrow F_2) = P(F_2 | F_1)$.

- *Confiance (XY) =Support (XY) / Support(X)*

Plus la confiance est élevée, meilleure est la règle d'association (6).

Un exemple classique des règles d'association: analyse du panier de marché (prédire les constituants du panier d'un client) comme montre la figure 14 (2) (6).

Figure 14 : *Analyse du panier de marché.*

1.6 Méthodes de data mining

Il existe une grande variété de méthodes de data mining, mais ici, nous nous concentrons uniquement sur un sous-ensemble des plus populaires. Chaque méthode est discutée dans le contexte de la représentation du modèle, modèle d'évaluation et recherche.

1.6.1 Arbres de décision

Un arbre de décision est une représentation hiérarchique de la structure des données sous forme des séquences de décisions (tests) en vue de la prédiction d'un résultat ou d'une classe. Chaque individu (ou observation), qui doit être attribué(e) à une classe, est décrit(e) par un ensemble de variables qui sont testées dans les nœuds de l'arbre. Les tests s'effectuent dans les nœuds internes et les décisions sont prises dans les nœuds feuille (6).

Il existe deux principaux types d'arbres de décision : les arbres de classification (*Classification Tree*) permettant de prédire à quelle classe la variable cible appartient, dans ce cas la prédiction est une étiquette de classe, et arbres de régression (*Regression Tree*) qui permettent de prédire une quantité réelle (par exemple, le prix d'une maison ou la durée de séjour d'un patient dans un hôpital), dans ce cas la prédiction est une valeur numérique (2).

1.6.1.1 Algorithme ID3

L'algorithme commence par le placement de tous les exemples d'apprentissage dans le nœud racine (6). Ensuite, chaque nœud est coupé sur un des attributs restants (qui n'a pas encore été testé). Le choix de cet attribut se fait à travers l'entropie et le gain d'information.

- Entropie du jeu de données *S* (l'incertitude) (*p* : positif, *n* : négatif) :
  - $Entropie(S) = (-p)/(p+n)\ log_2(p/(p+n))\ -n/(p+n)log_2(n/(p+n))$
- Entropie d'une partition (ou attribut) :
  - $I(Attribut) = \sum(p_i\text{-}n_i)/(p+n)\ .\ Entropie\ (A)$
- Gain d'information (différence d'entropie avant et après avoir divisé l'ensemble) :
  - $Gain = Entropie(S) - I(Attribut)$
- Quand $p = n$, $Entropie(S) = 1$ et lorsque $p = 0$ ou $n = 0$, $Entropie(S) = 0$

1.6.1.2 Fonctionnement simplifié d'ID 3

1. Calculer l'entropie de l'ensemble de données *Entropie(S).*
2. Pour chaque attribut :
   2.1. Calculer l'entropie pour toutes les valeurs *Entropie(A).*
   2.2. Calculer la moyenne d'information pour cet attribut.
   2.3. Calculer le gain.
3. Choisir l'attribut avec le plus grand gain.
4. Répéter les étapes jusqu'à obtenir l'arbre de décision désiré.

1.6.2 Les Réseau de Neurones

C'est une transposition simplifiée des neurones du cerveau humain. Dans leur variante la plus courante, les réseaux de neurones apprennent sur une population d'origine puis sont capables d'exprimer des résultats sur des données inconnues. Ils sont utilisés dans la prédiction et la classification dans le cadre de découverte de connaissances dirigées. Certaines variantes permettent l'exploration des séries temporelles et des analyses non dirigées (réseaux de Kohonen) (7).

1.6.2.1 Fonctionnement

- La construction de la structure du réseau (généralement empirique).
- La constitution d'une base de données de vecteurs représentant au mieux le domaine à modéliser. Celle-ci est scindée en deux parties: une partie servant à l'apprentissage du réseau (on parle de base d'apprentissage) et une autre partie aux tests de cet apprentissage (on parle de base de test).
- Le paramétrage du réseau par apprentissage. Au cours de 1'apprentissage, les vecteurs de données de la base d'apprentissage sont présentés séquentiellement et plusieurs fois au réseau. Un algorithme d'apprentissage ajuste le poids du réseau afin que les vecteurs soient correctement appris. L'apprentissage se termine lorsque l'algorithme atteint un état stable.
- La phase de reconnaissance consiste à présenter au réseau chacun des vecteurs de la base de test. La sortie correspondante est calculée en propageant les vecteurs à travers le réseau. La réponse est lue directement sur les unités de sortie et comparée à la réponse attendue. Une fois que le réseau présente des performances acceptables, il peut être utilisé pour répondre au besoin qui a été à l'origine de sa construction.

1.6.2.2 Architecture de base des réseaux de neurones

Il y a deux types des réseaux de neurones monocouches et multicouches. Dans le seul réseau en couches, un ensemble d'entrées est directement mappé sur une sortie en utilisant une variation généralisée d'une fonction linéaire. Dans le 2ème type, les neurones sont disposés en couches, dans lesquelles les couches d'entrée et de sortie sont séparées par un groupe de couches cachées appelée feed-forward (7).

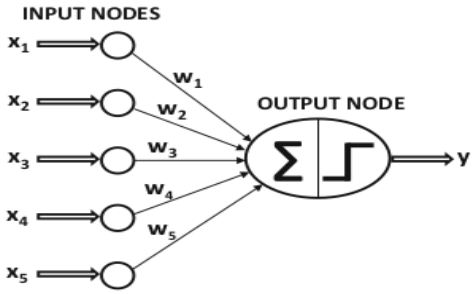


Figure 15 : *Architecture de base des réseaux de neurones (5).*

1.6.2.3 Réseaux de neurones multicouches (MLP)

Les réseaux de neurones multicouches (figure 16) contiennent plusieurs couches de calcul ; les couches intermédiaires supplémentaires (entre l'entrée et la sortie) sont appelées couches cachées car les calculs effectués ne sont pas visibles par l'utilisateur. L'architecture spécifique des réseaux de neurones multicouches est appelée réseaux d'alimentation directe, car les couches successives s'alimentent les unes les autres dans le sens direct de l'entrée à la sortie. L'architecture par défaut des réseaux *feed-forward* suppose que tous les nœuds d'une couche sont connectés à ceux de la couche suivante. Par conséquent, l'architecture du réseau de neurones est presque entièrement définie, une fois que le nombre de couches et le nombre/type de nœuds dans chaque couche ont été définis. Le seul détail restant est la fonction de perte qui est optimisée dans la couche de sortie (6).

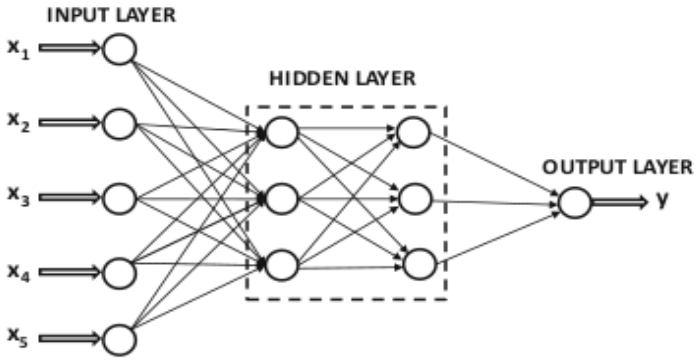


Figure 16 : *Réseaux de neurones multicouches (6).*

1.6.3 Naïves bayes

Naïve Bayes est une approche très populaire connue sous le nom de théorie des probabilités pour trouver le plus probable des classifications possibles, c'est à dire elle donne un moyen de combiner la probabilité *a priori* et les probabilités conditionnelles dans une seule formule, que nous pouvons utiliser pour calculer tour à tour la probabilité de chacune des classifications possibles. Ayant fait cela, nous choisissons la classification avec la plus grande valeur. Le premier mot dans le nom à consonance plutôt péjoratif *Naïve Bayes* se réfère à l'hypothèse que fait la méthode, que l'effet de la valeur d'un attribut sur la probabilité d'une classification donnée est indépendant des valeurs des autres attributs (6).

- **Algorithme de Naïve Bayes :**

Étant donné un ensemble de *k* classifications mutuellement exclusives et exhaustives c_1, c_2, ...,c_k, qui ont des probabilités a priori $P(c_1)$, $P(c_2)$, ..., $P(c_k)$, respectivement, et *n* attributs a_1, a_2, ..., a_n qui pour une instance donnée ont les valeurs v_1, v_2, ..., v_n respectivement, la probabilité *a posteriori* que la classe c_i se produise pour l'instance spécifiée peut être montrée comme étant proportionnelle à :

$$P(c_i) \times P(a_1 = v_1 \text{ et } a_2 = v_2 \ldots \text{ et } a_n = v_n \mid c_i)$$

En supposant que les attributs sont indépendants, la valeur de cette expression peut être calculée en utilisant le produit :

$$P(c_i) \times P(a_1 = v_1 \mid c_i) \times P(a_2 = v_2 \mid c_i) \times \ldots \times P(a_n = v_n \mid c_i)$$

Nous calculons ce produit pour chaque valeur de *i* de 1 à *k* et choisissons la classification qui a la plus grande valeur (6).

1.6.4 Nearest Neighbor Classification (KNN)

La classification du plus proche voisin est principalement utilisée lorsque toutes les valeurs d'attribut sont continues, bien qu'elle puisse être modifiée pour traiter des attributs catégoriques. L'idée est d'estimer la classification d'une instance invisible en utilisant la classification de l'instance ou des instances qui en sont les plus proches (6) (2).

L'algorithme basée sur :

- Trouvez les *k* instances d'entraînement les plus proches de l'instance invisible.
- Prenez la classification la plus courante pour ces *k* instances.

Nous utilisons des mesures pour calculer la distance entre deux instances :

- La distance euclidienne : distance qui calcule la racine carrée de la somme des différences carrées entre les coordonnées de deux points.
- Distance Manhattan: calcule la somme des valeurs absolues des différences entre les coordonnées de deux points.

- Distance Hamming: la distance entre deux points donnés est la différence maximale entre leurs coordonnées sur une dimension.

Le choix de la valeur à utiliser pour effectuer une prédiction avec *K-NN* varie en fonction du jeu de données.

1.7 Conclusion

Dans ce chapitre nous avons présenté un aperçu sur les techniques de fouille de données, le processus d'extraction de connaissances, les tâches et les différents algorithmes existants. Cette première étape sert comme une base permettant de nous positionner dans un domaine qui englobe notre problématique et d'entamer la prochaine phase de l'étude.

Chapitre 2

Les séries temporelles : un aperçu

2.1 Introduction

Dans ce chapitre nous allons se projeter sur un type de données particulier de data mining, il s'agit des séries temporelles (chronologiques) où nous découvrons surtout de nouveaux aspects théoriques proposé par des chercheurs comme solution pour extraire et analyser les motifs fréquents sur la base d'observations passées.

2.2 Séries temporelles

2.2.1 Définition

Une série temporelle est issue d'un processus d'analyse d'une observation de points de données collectés sur une période de temps. Dans l'analyse des séries temporelles, les analystes de données enregistrent les observations de données à intervalles constants pour un ensemble de périodes de temps au lieu d'enregistrer les observations de données de manière aléatoire. Le taux d'observation (intervalle de temps) peut aller de quelques millisecondes à plusieurs années afin de prédire les résultats futurs sur la base de données antérieures tout en assurant la cohérence et la fiabilité (1) (2).

2.2.2 Types d'attributs de séries temporelles

2.2.2.1 Attributs contextuels

Ce sont les attributs qui définissent le contexte sur la base dont les dépendances implicites se produisent dans les données. Par exemple, dans le cas de données du capteur, l'horodatage auquel la lecture est mesurée peut être considéré comme l'attribut contextuel.

Parfois, l'horodatage n'est pas explicitement utilisé, mais une position d'indice est utilisée. Alors que le type de données de série temporelle ne contient qu'un seul attribut contextuel, d'autres types de données peuvent avoir plus d'un attribut contextuel (2).

2.2.2.2 Attributs comportementaux

Ils représentent les valeurs mesurées dans un contexte. Dans l'exemple du capteur, la température est la valeur de l'attribut comportemental. Il est possible d'avoir plus d'un attribut comportemental. Par exemple, si plusieurs capteurs enregistrent les lectures à des horodatages synchronisés, il en résulte un résultat multidimensionnel de données de séries temporelles. Les attributs contextuels ont généralement un fort impact sur les dépendances entre les valeurs des attributs comportementaux dans les données (1) (2).

2.3 Domaines d'application

2.3.1 Prévisions météorologiques

Ce concept a commencé avec le philosophe grec Aristote qui faisait des recherches météorologiques afin d'identifier les causes et les effets des changements climatiques. Plus tard, les scientifiques ont commencé à accumuler des données météorologiques à l'aide de l'instrument « *baromètre* » pour calculer l'état des conditions atmosphériques sur des intervalles d'une heure ou d'une journée et les ont conservées à différents endroits.

Avec le temps, les prévisions météorologiques personnalisées ont commencé à être imprimées dans les journaux et, plus tard, avec les progrès de la technologie, les prévisions actuellement dépassent les conditions météorologiques générales. Afin d'effectuer des mesures atmosphériques avec des méthodes de calcul pour des compilations rapides, de nombreux gouvernements ont établi des milliers de stations de prévision météorologique à travers le monde. Ces stations sont équipées d'appareils hautement fonctionnels et sont interconnectées les unes aux autres pour accumuler des données météorologiques à différents emplacements géographiques et prévoir les conditions météorologiques à chaque instant selon les besoins.

2.3.2 Astronomie

L'astronomie est basée sur le traçage d'objets, de trajectoires et de mesures précises, et de ce fait, les experts en astronomie maîtrisent les séries temporelles pour évaluer les instruments et étudier les objets de leur intérêt. Au cours des siècles passés, l'analyse des séries temporelles a été utilisée pour découvrir des étoiles variables qui sont utilisées pour estimer les distances stellaires, et observer des événements transitoires tels que les *Supernova* pour comprendre le mécanisme de l'évolution de l'univers avec le temps (2).

2.3.3 Finance

L'analyse de séries temporelles est un processus de prévision essentiel pour expliquer le comportement dynamique et influe des marchés financiers. En examinant les données financières, un expert peut prédire les taux d'intérêt, le risque de change, la volatilité des marchés boursiers et bien d'autres et de ce fait. Les décideurs politiques peuvent utiliser les prévisions financières pour prendre des décisions concernant la production, les achats, la durabilité du marché, l'allocation des ressources, … etc (1).

2.3.4 Développement commercial

La prévision des séries temporelles aide les entreprises à prendre des décisions commerciales éclairées, car le processus analyse les modèles de données passés, il peut être utile pour prévoir les possibilités et les événements futurs des manières suivantes (2) (3):

- Fiabilité : lorsque les données intègrent un large éventail d'intervalles de temps sous la forme d'observations massives sur une période plus longue, les prévisions de séries temporelles sont très fiables.
- Croissance : afin d'évaluer la performance et la croissance financières globales ainsi que endogènes, les séries temporelles sont l'actif le plus approprié.
- Estimation des tendances : des méthodes de séries temporelles peuvent être utilisées pour découvrir des tendances, par exemple, ces méthodes inspectent les observations de données pour identifier quand les mesures reflètent une diminution ou une augmentation des ventes d'un produit particulier.

2.4 Un état de l'art

Dans cette section, nous présentons quelques travaux de recherches qui se sont intéressés aux séries temporelles en adoptant chacun, une approche différente.

Dans (4), les chercheurs proposent une approche basée sur des méthodes statistiques et probabilistes pour contrôler les lectures de capteurs en analysant leur comportement et détecter toute panne ou anomalie. Cette approche est appliquée sur l'étude de cas industrielle du barrage de *Cecebre* en *Espagne* qui est équipé de capteurs sans fil qui mesurent les apports d'eau au barrage tel que le capteur de hauteur d'eau, le capteur de précipitations d'eau et le capteur de débit de sortie d'eau. Le travail est organisé en plusieurs sections comme suit :

La première section consiste à collecter des traces des capteurs qui sont sous forme de données de séries temporelles pour faire ensuite un prétraitement des données qui englobe une étape cruciale dans l'étude qui est la discrétisation. L'idée est de mapper des valeurs continues à des intervalles prédéfinis et fixe afin de les rendre discret en utilisant la méthode *EWD* (*Equal Width Discretisiation*) (8) en raison de sa simplicité. En se focalisant sur le capteur de hauteur d'eau, les chercheurs ont distingué cinq niveaux pour la discrétisation des données. En dernier ils ont généré un fichier de distribution des capteurs en comptant l'occurrence de chaque niveau de hauteur d'eau chaque jour.

La deuxième section consiste à spécifier un modèle approximatif du comportement du capteur à partir de ces observations passées. Ce dernier est basé sur le langage *BIP* pour la modélisation de distribution des jours en fonction de distribution d'eau.

La troisième section consiste à vérifier par *SMC* exprimé en *SBIP* si le modèle satisfait une propriété *LTL* donnée avec une certaine probabilité.

La quatrième section représente la conclusion générale où les chercheurs font un récapitulatif sur les modèles et les méthodes utilisé pour l'étude du comportement des capteurs et ils prévoient à l'avenir d'améliorer l'approche proposée en analysant la cohérence entre les comportements d'un ensemble de capteurs et en exprimant les propriétés inter-capteurs.

Dans (9), les chercheurs ont adopté pratiquement la même approche que l'article (4) avec un exemple différent. Il s'agit de la surveillance des épidémies pour faire des prédictions sur les infections virales telles que le choléra en République démocratique du Congo. Le travail est organisé en plusieurs sections comme suit :

La deuxième section définit l'étude de cas au niveau du système de santé de la *RDC* qui est sous forme hiérarchique, organisé en trois niveaux. Le niveau de la mise en œuvre *DS* où une équipe de district gère un réseau de centres de santé. Le niveau intermédiaire, chargé de l'appui technique et logistique, est géré par *DSP*. Le niveau central, appelé aussi niveau national, joue le rôle des décideurs. Les données de santé sont essentiellement produites au niveau de la mise en œuvre où chaque district sanitaire collecte, agrège et transmet les données *PHD* de ses centres de santé. Pour tester leur approche de prédiction, les chercheurs ont extrait les données de cinq zones *Kabalo, Kalemie, Manono, Nyemba et Nyunzu* à partir d'une base de données nationale sur le *choléra* sur dix ans enregistrés dans un fichier Excel. Parmi les raisons de propagation, les zones sont traversées par les fleuves *Lukunga, Luvua* et *Congo* où les gens se lavent même boivent l'eau des rivières, ce qui facilite la propagation de l'épidémie de *choléra*.

La troisième section consiste à utiliser vérificateur de modèle statistique *BIP* pour spécifier le comportement des infections virales en agrégeant d'abord les données des zones en semaines pour chaque année, puis les discrétiser en utilisant la méthode *EWD* en sept niveaux incrémentiels selon le nombre d'infections représentant où L_1 est le niveau d'infections le plus bas et L_7 est le niveau le plus élevé. Ensuite, utiliser *PMF* pour générer des distributions hebdomadaires de probabilité d'infections *WDZ* en comptant l'occurrence de chaque niveau.

La quatrième section consiste à utiliser le l'outil *BIP* permettant de simuler le modèle et d'exprimer certaines propriétés pour analyser les infections virales dans plusieurs zones de la *RDC* dans le temps à partir de traces d'exécution. Une variante *PBLTL* est utilisée comme formalisme pour décrire les propriétés temporelles linéaires probabilistes sur le modèle.

La sixième section représente la conclusion générale où les chercheurs parlent de l'utilité des prévisions fournies par leur approche pour les professionnels de la santé car elles indiquent la période approximative au cours de laquelle des cas suspects de maladies infectieuses peuvent survenir et aider à anticiper la préparation pour répondre efficacement au bon moment. Ils prévoient aussi au futur d'étendre leur approche pour prendre en compte d'autres facteurs importants influençant la dynamique des épidémies de nature à la fois biomédicale et socio-économique et également d'appliquer leur approche à d'autres études de cas de maladies infectieuses telles que COVID-19.

Dans (10), les chercheurs abordent un problème difficile, il s'agit de la découverte de motifs séquentiels et proposent plusieurs solutions inspirées de l'algorithme *Apriori*.

L'algorithme *Apriori* (11) est un algorithme de data mining qui sert à reconnaître des propriétés qui reviennent fréquemment ensemble dans une base de données de transactions client et d'en déduire une catégorisation. Pour résoudre ce problème, la tâche d'exploration de motifs séquentiels a été proposée. Elle consiste à découvrir des sous-séquences intéressantes dans un ensemble de séquences et mesurer différents critères en termes de sa fréquence d'occurrence, sa longueur et son profit.

L'exploration de modèles séquentiels peut être appliquée à deux types de données : une séquence qui est une liste ordonnée de valeurs nominales (symboles) et une série temporelle qui est une liste ordonnée de nombres. Cette dernière est convertie en séquences à l'aide de techniques de discrétisation comme algorithmes *SAX* et *iSAX* (12). Le but est de découvrir des relations séquentielles entre des éléments intéressants pour l'utilisateur, il y a toujours une seule réponse correcte à un problème d'extraction de motifs séquentiels ou de nombreux algorithmes ont été conçus pour le résoudre. Certains des plus populaires sont *GSP, Spade, PrefixSpan, Spam, CM-Spam et CM-Spade*.

L'algorithme *GSP* utilise un paradigme par niveau, il s'agit de découvrir d'abord tous les éléments fréquents d'une manière par niveau. Cela signifie simplement compter les occurrences de tous les éléments singleton dans la base de données. Ensuite, les transactions sont filtrées en supprimant les éléments non fréquents. A l'issue de cette étape, chaque transaction n'est constituée que des éléments fréquents qu'elle contenait à l'origine.

La base de données modifiée devient une entrée pour l'algorithme *GSP*. Ce processus nécessite un passage sur toute la base de données. Néanmoins, *GSP* a plusieurs limitations importantes comme l'analyse multiple de la base de données pour calculer le support des modèles candidats. Cela peut être très coûteux pour une grande base de données, perte de temps en générant des candidats inexistants et consommer une énorme quantité de mémoire en gardant toutes les séquences fréquentes de longueur *k* en mémoire pour pouvoir générer des motifs de longueur *k + 1*.

Spade inspiré de l'algorithme d'*Eclat* est un algorithme alternatif qui utilise une recherche en profondeur avec une présentation verticale de la base de données qui a deux propriétés intéressantes pour l'extraction de motifs séquentiels : *IDList* de n'importe quel pattern permet de calculer directement son support qui est simplement le nombre d'identificateurs de séquence distincts dans sa liste *IDL* et que l'*IDList* de tout modèle peut être créé sans scanner la base de données d'origine en effectuant la jointure d'*IDLists* sans conserver un grand nombre de motifs en mémoire contrairement aux algorithmes de recherche en largeur. Néanmoins, les *IDLists* peuvent être très volumineux lorsque des motifs apparaissent dans de nombreuses séquences en particulier dans les bases de données denses ou contenant de longues séquences, et que l'application de l'opération de jointure des *IDLists* est coûteuse car elle nécessite de comparer les éléments de deux listes *IDL*.

Spam et *BitSpadeest* sont des versions de l'algorithme *Spade* utilisant des vecteurs de bits. De plus, un algorithme inspiré de *Spam* appelé *Fast* a introduit le concept d'*IDListes* creuses indexées, pour calculer plus rapidement le support des candidats et réduire l'utilisation de la mémoire. Une version de *Spade* utilisant des vecteurs de bits appelée *Prism* a également introduit le concept de codage *Prime-block*. Il a été démontré que *Spam*, *BitSpade* et *Prism*, qui utilisent la représentation vectorielle de bits, sont plus d'un ordre de grandeur plus rapides que l'algorithme *Spade* original, tandis que l'algorithme *Fast* s'est avéré plus rapide que *Spam* mais n'a pas été comparé à *Spade* ou à *BitSpade* amélioré.

CM-Spam et *CM-Spade* sont des versions améliorées de *Spam* et *BitSpade* ou ils cherchent à réduire le nombre d'opérations de jointure en introduisant le concept de taille de cooccurrence. Il consiste à parcourir initialement la base de données pour créer une structure appelée *CMAP* qui stocke toutes les 2-séquences fréquentes. Ensuite, pour chaque

motif qui est pris en compte par la procédure de recherche si les deux derniers éléments ne sont pas des 2-séquences fréquentes, le motif peut être directement éliminé sans construire son *IDList* (donc sans effectuer l'opération de jointure). Il a été démontré que *CM-Spam* et *CM-Spade* surpassaient *GSP*, *Spam*, *BitSpade* et *PrefixSpan* de plus d'un ordre de grandeur. *CM-Spade* est considéré comme l'algorithme le plus rapide actuellement.

PrefixSpan inspiré de l'algorithme *FPGrowth* pour l'extraction d'éléments procède comme suit : il explore l'espace de recherche des motifs séquentiels en utilisant une recherche en profondeur d'abord pour calculer le support des éléments uniques et identifier les éléments fréquents. Il commence à partir de modèles séquentiels contenant un seul élément et explore des modèles plus larges en ajoutant de manière récursive des éléments aux modèles pour créer des modèles plus grands. Pour s'assurer qu'aucun modèle n'est généré deux fois, les éléments sont ajoutés aux modèles en fonction d'un ordre total, qui peut être l'ordre lexicographique ou tout autre ordre total.

Il a l'avantage de n'exporter que les modèles apparaissant dans la base de données. Cependant, un inconvénient de *PrefixSpan* est qu'il peut être coûteux d'analyser à plusieurs reprises la base de données et de créer des projections de base de données, en termes d'exécution. De plus, en termes de mémoire, la création de projections de base de données peut consommer une énorme quantité de mémoire si elle est implémentée naïvement, car dans le pire des cas, elle nécessite de copier la quasi-totalité de la base de données pour chaque projection de base de données. En pratique, il a été démontré que *CMSpade* surpassé *PrefixSpan* de plus d'un ordre de grandeur. La raison en est que le coût de la numérisation de la base de données et de la réalisation de projections peut être assez élevé.

En conclusion, tous les algorithmes d'exploration de motifs séquentiels explorent l'espace de recherche des motifs séquentiels en effectuant des opérations pour générer un ($k+1$)-séquence (une séquence contenant $k + 1$ articles) d'un k-séquence et diffèrent selon s'ils utilisent une recherche en profondeur d'abord ou en largeur d'abord, le type de représentation de base de données qu'ils utilisent en interne ou en externe, la façon dont ils génèrent ou déterminent les prochains motifs à être explorées dans l'espace de recherche, et en dernier comment elles comptent le support des modèles pour déterminer s'ils satisfont la contrainte de support minimum.

Dans (13)**,** les chercheurs tentent de la prédiction basée sur des règles dans les séries à valeurs réelles en utilisant (*MDL*) qui évalue les règles candidates en fonction de leur capacité à compresser les données. Les chercheurs définissent le cadre de règles. Tout d'abord, ils mesurent la distance entre deux sous-séquences et sortent avec 5 définitions :

1. un antécédent candidat doit se produire au moins deux fois pour être considéré comme précurseur de règle.
2. une sous-séquence soit un résultat significatif qui doit se produire dans un délai acceptable après que l'antécédent de la règle a été détecté.
3. le *maxlag* est le nombre maximum de pas de temps autorisé entre une détection antécédent et son conséquent.
4. une règle des séries temporelles, *R*, est un *4-uplet* de {*Ra, Rc, maxlag, t*}.
5. le point final de l'antécédent est le début du conséquent.

En raison de l'intention d'utiliser *MDL* pour mesurer les mérites relatifs des règles candidates, il faut d'abord transformer la série temporelle à valeurs réelles en un espace discrétisé en la quantifiant dans des bacs de tailles uniforme et pour une longueur de sous-séquence ρ, normalisée en *z* toutes les sous-séquences possibles de cette longueur et en enregistrant les valeurs minimales et maximales sur les sous-séquences normalisées. Après avoir atteint la valeur minimale globale et la valeur maximale globale dans toutes les sous-séquences, définir des limites de bacs de taille uniforme entre *minimum* et *maximum*. La largeur de bac résultante est alors : *(maximum minimum) / cardinalité*.

Essentiellement, l'algorithme de recherche de règles comporte deux parties : une fonction de notation et un algorithme de recherche qui invoque à plusieurs reprises cette fonction de notation lors de la recherche de règles de haute qualité. Considérant le cas où *maxlag* est contraint à *zéro* : la fonction de notation *MDL* reçoit deux entrées : une série temporelle candidate et une valeur *maxlag*. La fonction renvoie alors trois choses : un antécédent, un conséquent et le score de qualité de la règle résultante. Notons que l'antécédent concaténé avec le conséquent est simplement la série temporelle d'entrée, *R*, mais le point de partage n'est pas connu à l'avance.

Pour l'évaluation expérimentale, les chercheurs fournissent deux sources d'évaluation pour la qualité. Ils montrent que les règles sont significatives en considérant l'annotation disponible par des étiquettes externes d'un certain type. Dans le cas plus général, ils utilisent la distance euclidienne entre le conséquent prédit et le *F* correspondant aux emplacements où la règle s'est déclenchée.

Les chercheurs cherchent à trouver des règles dans des exemples différents comme la vocalisation des oiseaux, la désagrégation énergétique, et dans un ensemble de données d'activité. La découverte de motifs de séries temporelles et l'extraction des règles par application de *MDL* permettent de classer et de comparer de manière significative des règles de longueur variée et des règles avec différents niveaux.

2.5 Conclusion

Dans ce chapitre nous avons fait une projection plus spécifique d'un type de donnée de data mining en relation avec notre problématique, il s'agit de séries temporelles où nous avons présenté un aperçu sur leur aspect algorithmique et méthodique.

Chapitre 3

Extraction et analyse des motifs

3.1 Introduction

Dans ce chapitre, nous exposons notre approche pour l'analyse des séries temporelles issues, le cas échéant, de capteurs qui surveillent un équipement ou un procédé particulier. En fait, l'idée de l'approche est de transformer les séries initiales à valeurs réelles, en des séries à valeurs discrètes, puis de rechercher à extraire des séquences qui se répètent dans plusieurs séries. Ces séquences, peuvent être utilisées ensuite pour identifier un comportement normal ou anormal du procédé ou de l'équipement, ou bien pour détecter carrément un dysfonctionnement du capteur.

3.2 Description de l'approche

Comme le montre la figure ci-dessous, notre approche se compose de cinq étapes se succédant dans un processus linéaire itératif et interactif. Il s'agit en effet d'un processus d'extraction de connaissance adapté à l'extraction et l'analyse des motifs :

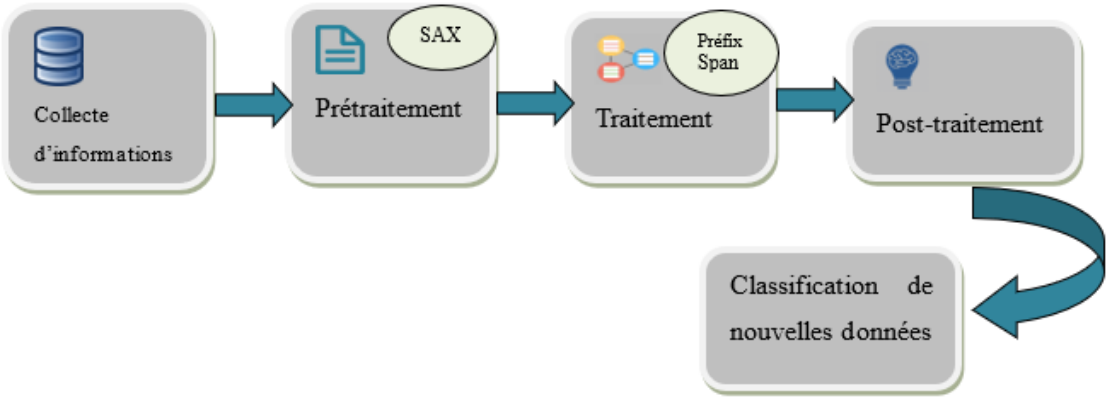


Figure 17 : *Approche générique pour l'extraction et l'analyse des motifs.*

Dans une première étape, nous commençons par collecter les traces des capteurs, qui sont enregistrées de manière continue dans un système automatique, souvent un ordinateur. Des difficultés liées à l'accès aux données peuvent être rencontrées à cause de la fermeture du système de surveillance, et sa soumission à des droits de propriété.

Les données collectées se retrouvent souvent dans des fichiers aux formats variés se qui induit une série d'opérations de prétraitement permettant de transformer les données brutes en des données exploitables. Ces opérations sont les mêmes rencontrées dans un processus d'extraction de connaissances, comme le nettoyage (traitement des données manquantes et des données aberrantes), le filtrage, ou l'agrégation. A la fin de cette étape, nous prévoyons faire une transformation en utilisant une méthode de réduction de dimension, de symbolisation, ou les deux. Parmi les méthodes qui peuvent être utilisées, nous pouvons citer : *PAA (Piecewise Aggregate Approximation)* (11), *SAX (Symbolic Aggregation approXimation)* (12), ou *EWD (Equal Width Discretization)* (13).

A l'issue de l'étape de prétraitement, nous obtenons un ensemble de séquences et sur la base de ces séquences, nous spécifions une méthode d'extraction et d'analyse du comportement des capteurs. Il s'agit des méthodes connues pour l'extraction des motifs, comme celles inspirées d'*Apriori* (14), ou celles inspirées de *FP-Growth* (15), ou d'autres retrouvées dans la littérature de spécialité. Un exemple des ces méthodes est *PrefixSpan*.

Le nombre de motifs extraits peut être facilement de l'ordre des centaines, voir des milliers, nous proposons une étape de post-traitement pour optimiser les motifs ainsi obtenus, et pour enfin pouvoir les analyser et classer les nouvelles séquences.

3.3 Cas d'application

Pour illustrer notre approche, nous présentons d'abord notre cas d'application. Nous considérons les données issues de trois capteurs déployés dans le barrage de *Cecebre* dans la ville de *la Corogne* en *Espagne*. Ces capteurs sont utilisés pour mesurer la hauteur d'eau (*WH*), la précipitation de pluie (*RP*) et le débit de sortie d'eau (*WF*). Les données recueillies par les capteurs sont utilisées pour contrôler l'ouverture du clapet de décharge afin de s'assurer que l'eau n'atteint pas un niveau maximum dans le barrage (figure 18).

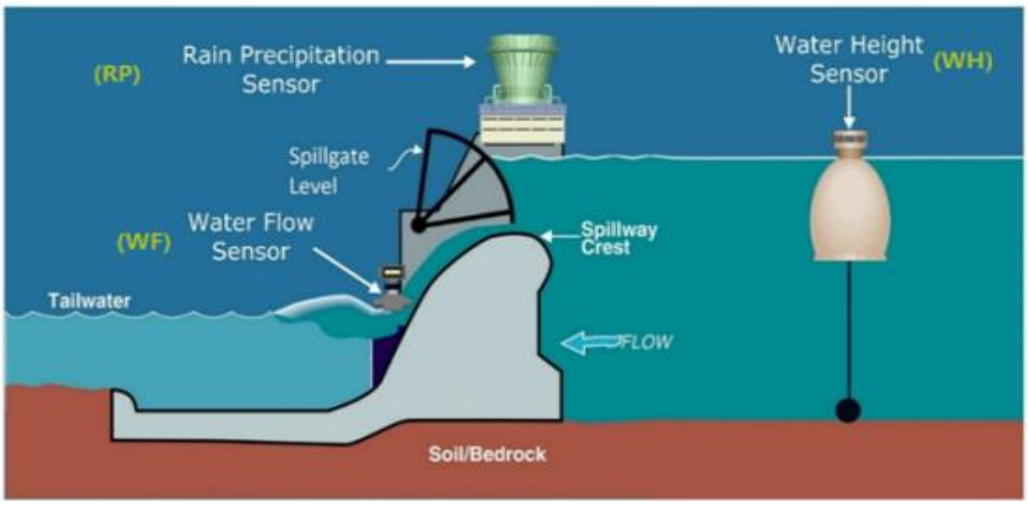


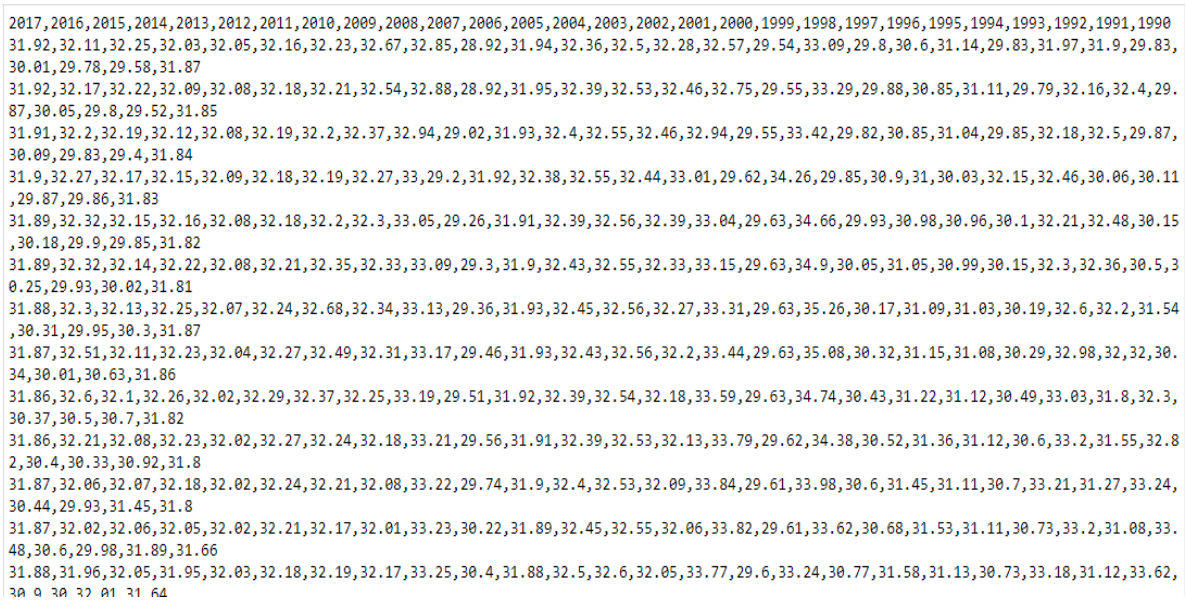
Figure 18 : *Infrastructure de barrage* (4).

Le comportement anormal de ces capteurs peut influencer le bon fonctionnement du système de barrage. Notre objectif est d'appliquer notre approche pour spécifier le comportement normal des capteurs et détecter toute panne ou anomalie en analysant un ensemble de motifs fréquents extraits sur la base d'observations passées.

3.3.1 Collecte de données

Une trace des données enregistrées par chaque capteur chaque jour de 1990 à 2017 a été collectée. Nous avons réorganisé la trace d'origine en créant un fichier *CSV*. Le nouveau fichier contient les lectures des capteurs par jour pendant 28 ans (figure19).

Un fichier CSV est un simple fichier texte dans lequel les valeurs sont séparées par des virgules, 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. À la place des virgules, les données peuvent également être séparées par des points-virgules, des tabulations ou par le symbole « | » et, là encore, elles peuvent être entre guillemets. Il n'existe pas de structure standard pour les fichiers *CSV*. En revanche, la majorité des logiciels et systèmes l'utilisent et le prennent en charge.



```
2017,2016,2015,2014,2013,2012,2011,2010,2009,2008,2007,2006,2005,2004,2003,2002,2001,2000,1999,1998,1997,1996,1995,1994,1993,1992,1991,1990
31.92,32.11,32.25,32.03,32.05,32.16,32.23,32.67,32.85,28.92,31.94,32.36,32.5,32.28,32.57,29.54,33.09,29.8,30.6,31.14,29.83,31.97,31.9,29.83,
30.01,29.78,29.58,31.87
31.92,32.17,32.22,32.09,32.08,32.18,32.21,32.54,32.88,28.92,31.95,32.39,32.53,32.46,32.75,29.55,33.29,29.88,30.85,31.11,29.79,32.16,32.4,29.
87,30.05,29.8,29.52,31.85
31.91,32.2,32.19,32.12,32.08,32.19,32.2,32.37,32.94,29.02,31.93,32.4,32.55,32.46,32.94,29.55,33.42,29.82,30.85,31.04,29.85,32.18,32.5,29.87,
30.09,29.83,29.4,31.84
31.9,32.27,32.17,32.15,32.09,32.18,32.19,32.27,33,29.2,31.92,32.38,32.55,32.44,33.01,29.62,34.26,29.85,30.9,31,30.03,32.15,32.46,30.06,30.11
,29.87,29.86,31.83
31.89,32.32,32.15,32.16,32.08,32.18,32.2,32.3,33.05,29.26,31.91,32.39,32.56,32.39,33.04,29.63,34.66,29.93,30.98,30.96,30.1,32.21,32.48,30.15
,30.18,29.9,29.85,31.82
31.89,32.32,32.14,32.22,32.08,32.21,32.35,32.33,33.09,29.3,31.9,32.43,32.55,32.33,33.15,29.63,34.9,30.05,31.05,30.99,30.15,32.3,32.36,30.5,3
0.25,29.93,30.02,31.81
31.88,32.3,32.13,32.25,32.07,32.24,32.68,32.34,33.13,29.36,31.93,32.45,32.56,32.27,33.31,29.63,35.26,30.17,31.09,31.03,30.19,32.6,32.2,31.54
,30.31,29.95,30.3,31.87
31.87,32.51,32.11,32.23,32.04,32.27,32.49,32.31,33.17,29.46,31.93,32.43,32.56,32.2,33.44,29.63,35.08,30.32,31.15,31.08,30.29,32.98,32,32,30.
34,30.01,30.63,31.86
31.86,32.6,32.1,32.26,32.02,32.29,32.37,32.25,33.19,29.51,31.92,32.39,32.54,32.18,33.59,29.63,34.74,30.43,31.22,31.12,30.49,33.03,31.8,32.3,
30.37,30.5,30.7,31.82
31.86,32.21,32.08,32.23,32.02,32.27,32.24,32.18,33.21,29.56,31.91,32.39,32.53,32.13,33.79,29.62,34.38,30.52,31.36,31.12,30.6,33.2,31.55,32.8
2,30.4,30.33,30.92,31.8
31.87,32.06,32.07,32.18,32.02,32.24,32.21,32.08,33.22,29.74,31.9,32.4,32.53,32.09,33.84,29.61,33.98,30.6,31.45,31.11,30.7,33.21,31.27,33.24,
30.44,29.93,31.45,31.8
31.87,32.02,32.06,32.05,32.02,32.21,32.17,32.01,33.23,30.22,31.89,32.45,32.55,32.06,33.82,29.61,33.62,30.68,31.53,31.11,30.73,33.2,31.08,33.
48,30.6,29.98,31.89,31.66
31.88,31.96,32.05,31.95,32.03,32.18,32.19,32.17,33.25,30.4,31.88,32.5,32.6,32.05,33.77,29.6,33.24,30.77,31.58,31.13,30.73,33.18,31.12,33.62,
30.9,30.32,31.31.64
```

Figure 19 : *Fichier CSV contenant une trace des données enregistrées par chaque capteur chaque jour de 1990 à 2017.*

3.3.2 Prétraitement de données

Le prétraitement se fait en deux étapes :

- Nettoyage des données : il consiste à supprimer les données manquantes comme la ligne du 29 du mois de février afin d'harmoniser la base de données.
- Discrétisation des données : dans notre étude, nous utilisons la méthode *SAX (Symbolic Aggregation approXimation)* pour discrétiser les données de séries temporelles en une chaîne symbolique qui réduit la dimensionnalité tout en étant indexée par une mesure de distance à limite inférieure.

SAX permet à une série temporelle de longueur arbitraire n d'être réduite à une chaîne de longueur arbitraire w, ($w < n$). La taille de l'alphabet est également un entier arbitraire a, où $a > 2$ (12).

Les principales notations utilisées par la suite sont :

- C une série temporelle $C = c_1, \ldots, c_n$
- $\bar{C}$ une approximation agrégée par morceaux d'une série temporelle $\bar{C} = \bar{c}_1, \ldots, \bar{c}_w$

- $\hat{C}$ une représentation symbolique d'une série temporelle $\hat{C} = \hat{c}_1, \ldots, \hat{c}_w$
- w le nombre de segments *PAA* représentant la série C
- a Taille de l'alphabet (par exemple, pour l'alphabet = $\{a, b, c\}$, $a = 3$)

La procédure de discrétisation est unique en ce qu'elle utilise une représentation intermédiaire entre la série temporelle brute et les chaînes symboliques. Elle transforme d'abord les données en représentation par *PAA* (11), puis symbolise la représentation *PAA* en une chaîne discrète. Il y a deux avantages importants à faire cela :

- Réduction de la dimensionnalité : utiliser le pouvoir de réduction de dimensionnalité bien défini et bien documenté de *PAA*, et la réduction est automatiquement reportée sur la représentation symbolique (12).
- Borne inférieure : prouver qu'une mesure de distance entre deux chaînes symboliques limite inférieurement la vraie distance entre les séries temporelles d'origine n'est pas triviale. L'observation qui a permis de prouver les bornes inférieures est de se concentrer sur la preuve que la distance symbolique mesure les bornes inférieures de la mesure de distance *PAA* (12).

Commençons par la réduction de la dimensionnalité via *PAA* :

Une série temporelle C de longueur n peut être représentée dans a $w-$espace dimensionnel par un vecteur $\bar{C} = \bar{c}_1, \ldots, \bar{c}_w$. En termes simples, pour réduire la série temporelle de n dimensions à w dimensions, les données sont divisées en w « *trames* » de taille égale. La valeur moyenne des données se trouvant dans une *trame* est calculée et un vecteur de ces moyennes devient la représentation réduite des données. La représentation peut être visualisée comme une tentative d'approximation de la série temporelle d'origine avec une combinaison linéaire de fonctions de base, comme le montre la figure 20. Pour plus de simplicité et de clarté, on suppose que n est divisible par w (12).

La réduction de dimensionnalité *PAA* est intuitive et simple, mais il a été démontré qu'elle rivalise avec des techniques de réduction de dimensionnalité plus sophistiquées telles que les transformées de Fourier et les ondelettes (12).

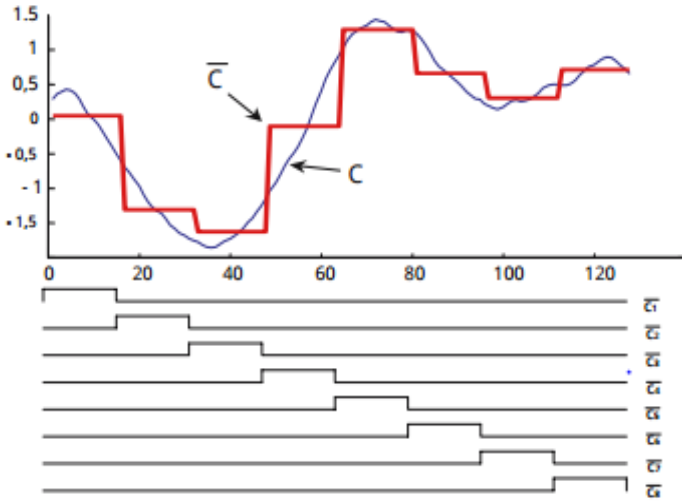


Figure 20 : *Représentation PAA* (11).

Rappelons que nous normalisons chaque série pour avoir une moyenne de zéro et un écart type d'un avant de la convertir en représentation *PAA*, car il est bien entendu qu'il est inutile de comparer des séries avec différentes échelles et amplitudes (11).

Après avoir transformé une base de données de séries temporelles en *PAA*, nous pouvons appliquer une transformation supplémentaire pour obtenir une représentation discrète. Il est souhaitable de disposer d'une technique de discrétisation qui produise des symboles avec équiprobabilité. Ceci est facilement réalisé puisque les séries temporelles normalisées ont une distribution gaussienne (figure 21).

Un diagramme de probabilité normale est une technique graphique qui montre si les données sont à peu près normalement distribuées : une ligne approximativement droite indique que les données sont approximativement distribuées normalement. Comme le montre la figure 21, la nature hautement linéaire des tracés suggère que les données sont approximativement normales. Pour une grande famille de données de séries temporelles dont nous disposons, nous remarquons que l'hypothèse gaussienne est en effet vraie. Pour le petit sous-ensemble de données où l'hypothèse n'est pas respectée, l'efficacité est légèrement détériorée ; cependant, l'exactitude de l'algorithme n'est pas affectée (12).

L'exactitude de l'algorithme est garantie par la propriété de borne inférieure de la mesure de distance dans l'espace symbolique.

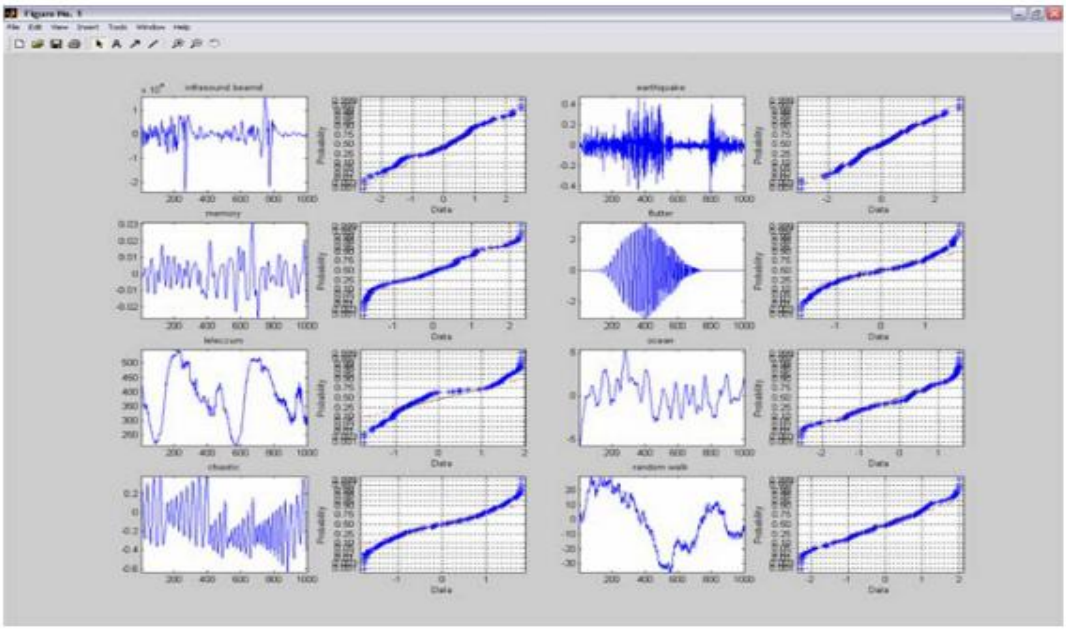


Figure 21 : *Un tracé de probabilité normale de la distribution des valeurs des sous-séquences de longueur 128 à partir de huit ensembles de données différents* (12).

Étant donné que les séries temporelles normalisées ont une distribution hautement gaussienne, nous pouvons simplement déterminer les « *points de rupture* » qui produiront *a* aires de taille égale sous la courbe gaussienne (12).

SAX repose sur les définitions suivantes :

- Définition 1. Les *points de rupture* sont une liste triée de nombres $B = \beta_1, \ldots, \beta_{a-1}$ telle que la zone sous une courbe de Gauss de β_i à $\beta_{i+1} = 1 / a$ (β_0 et β_a sont définis comme $-\infty$ et $+\infty$, respectivement), a étant la taille de l'alphabet.

Ces *points de rupture* peuvent être déterminés en les recherchant dans un tableau statistique. Par exemple, la figure 22 donne les points de rupture pour les valeurs de 3 à 10.

Une fois les points de rupture obtenus, nous pouvons discrétiser une série temporelle de la manière suivante. Nous obtenons d'abord une représentation *PAA* de la série. Tous les coefficients *PAA* inférieurs au plus petit point de rupture sont mappés sur le symbole *a*, tous les coefficients supérieurs ou égaux au plus petit point d'arrêt et inférieurs au deuxième plus petit point d'arrêt sont mappés sur le symbole *b*, etc.

βi \ a	3	4	5	6	7	8	9	10
β_1	−0.43	−0.67	−0.84	−0.97	−1.07	−1.15	−1.22	−1.28
β_2	0.43	0	−0.25	−0.43	−0.57	−0.67	−0.76	−0.84
β_3		0.67	0.25	0	−0.18	−0.32	−0.43	−0.52
β_4			0.84	0.43	0.18	0	−0.14	−0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Figure 22 : *Points de rupture qui divisent une distribution gaussienne en un nombre arbitraire (de 3 à 10) de régions équiprobables* (12).

Notez que dans cet exemple, les trois symboles *a*, *b* et *c* sont à peu près équiprobables comme nous le souhaitions. On appelle la concaténation de symboles qui représentent une sous-séquence un *mot*.

- Définition 2. Une sous-séquence C de longueur n peut être représentée comme un *mot* $\hat{C} = \hat{c}_1, \ldots, \hat{c}_w$ comme suit. Soit $alpha_i$ dénoter le $i^{ème}$ élément de l'alphabet, c'est-à-dire $alpha_1 = a$ et $alpha_2 = b$. Ensuite, la correspondance à partir d'une approximation *PAA* $\bar{C}$ à un mot $\hat{C}$ est obtenue comme suit :

$$\hat{C}_i = alpha_j, \beta_{j-1} \leq \bar{C}_i < \beta_j$$

Nous avons maintenant défini notre représentation symbolique (la représentation *PAA* n'est qu'une étape intermédiaire nécessaire pour obtenir la représentation symbolique).

En résumé, une série d'entrées est d'abord normalisée et divisée en un nombre de segments horizontaux fourni par l'utilisateur. Une moyenne des valeurs dans chaque segment est prise, et un symbole est ensuite attribué, en fonction de la plage de valeurs qui contient la moyenne. Les intervalles de valeur de symbole (c'est-à-dire la taille de segment vertical ou points de rupture) sont calculés en fonction de l'affectation égale de l'aire sous une courbe gaussienne, qui, à son tour, dépend de la taille de l'alphabet fournie (12).

Remarque, les symboles peuvent être alphanumériques, produisant des chaînes *SAX*, *baabccbc* comme le montre la figure 23, ou *21323*, comme indiqué sur la figure 24.

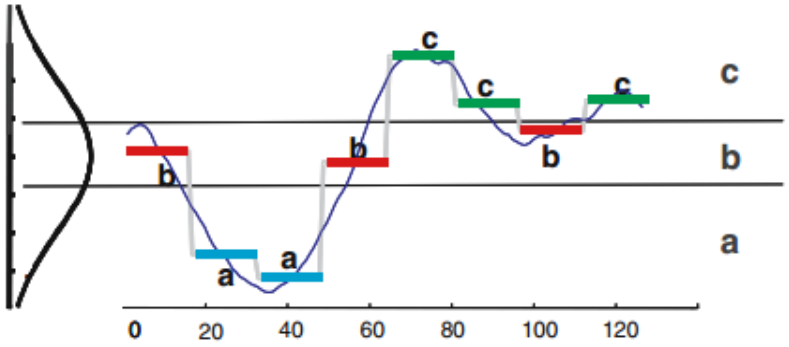


Figure 23 : *Traduction de série brute en une chaîne symbolique (alphanumérique) à l'aide de SAX* (12).

Les points de rupture sont déterminés pour faire correspondre les coefficients *PAA* en symboles *SAX*. Dans l'exemple ci-dessus (figure 23), avec $n = 128$, $w = 8$ et $a = 3$, la série temporelle est mappée sur le mot *baabccbc*.

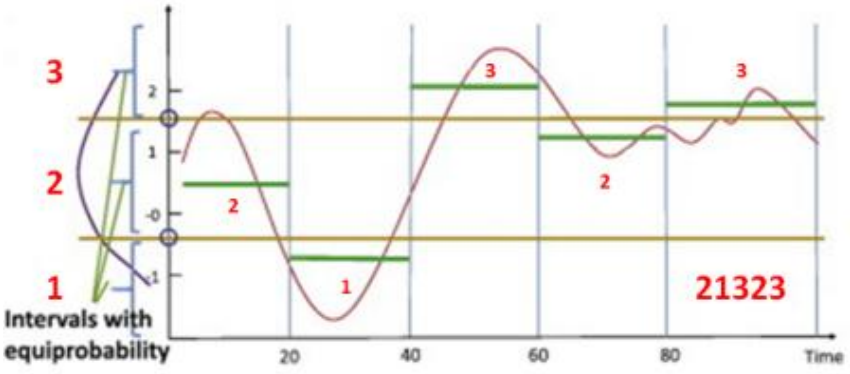


Figure 24 : *Traduction de série brute en une chaîne symbolique (numérique) à l'aide de SAX* (12).

Les points de rupture pour la désignation des symboles sont représentés par des lignes horizontales brunes et les lignes vertes indiquent les valeurs de série moyennes dans un segment de série (séparées par des lignes verticales noires figure 24).

3.3.3 Traitement des données

Dans notre étude de cas, nous nous intéressons aux séquences, car c'est le type de données utilisé dans la fouille de motifs séquentiels.

Le but est de découvrir des sous-séquences intéressantes dans un ensemble de séquences, où l'intérêt d'une sous-séquence peut être mesuré en termes de différents critères tels que son support, sa longueur et son profit.

Le support d'une séquence S_a dans une base de données de séquences est défini comme le nombre de séquences qui contiennent S_a, et est noté par $Support(S_a)$. Une séquence est dite être une séquence fréquente ou un motif séquentiel si et seulement si $Support(S_a) \geq minSup$, pour un seuil *minSup* défini par l'utilisateur (14).

Le problème d'extraction de motifs est irréaliste pour la plupart des bases de données de séquences réelles. Il est donc nécessaire de concevoir des algorithmes efficaces pour éviter d'explorer l'espace de recherche de toutes les sous-séquences possibles.

De nombreux algorithmes ont été conçus pour découvrir des modèles séquentiels dans des bases de séquences, parmi les plus connus *PrefixSpan* dont nous nous appuyons dans notre étude de cas. *PrefixSpan* consiste à explorer l'espace de recherche des motifs en utilisant une recherche en profondeur d'abord : il commence à partir de modèles contenant un seul élément et explore des modèles plus larges en ajoutant de manière récursive des éléments aux modèles pour créer des modèles plus grands. Pour s'assurer qu'aucun modèle n'est généré deux fois, les articles sont ajoutés aux modèles en fonction d'un ordre total sur les articles, qui peut être l'ordre lexicographique ou tout autre ordre total.

La première opération effectuée par *PrefixSpan* consiste à scanner la base de données de séquences d'origine pour calculer le support des éléments uniques et identifier les éléments fréquents (ceux dont le support n'est pas inférieur au seuil *minSup*). Ensuite, *PrefixSpan* génère chacun de ces éléments sous forme de modèles séquentiels fréquents et considère ces modèles comme des graines pour poursuivre la recherche en profondeur. Lors de la recherche en profondeur d'abord, pour un motif séquentiel donné S_a de longueur k, *PrefixSpan* crée d'abord la base de données projetée du motif S_a. Ensuite, *PrefixSpan* analyse la base de données projetée de S_a pour compter le support des éléments afin de trouver les éléments qui peut être ajouté à sa *i-extension* ou *s-extension* pour former des motifs séquentiels $(k + 1)$. Ce processus est puis répété de manière récursive comme une recherche en profondeur d'abord pour trouver tous les modèles séquentiels fréquents.

Tableau 1: *Base de données de séquences.*

SID	Séquence
1	⟨{a, b}, {c}, {f, g}, {g}, {e}⟩
2	⟨{a, d}, {c}, {b}, {a, b, e, f}⟩
3	⟨{a}, {b}, {f}, {e}⟩
4	⟨{b}, {f, g}⟩

Tableau 2 : *La base de données de séquences projetées du motif ⟨{a}⟩.*

SID	Séquence
1	⟨{-, b}, {c}, {f, g}, {g}, {e}⟩
2	⟨{-, d}, {c}, {b}, {a, b, e, f}⟩
3	⟨{b}, {f}, {e}⟩
4	⟨{b}, {f, g}⟩

Tableau 3 : *La base de données de séquences projetées du motif ⟨{a, b}⟩.*

SID	Séquence
1	⟨{c}, {f, g}, {g}, {e}⟩
2	⟨{-, e, f}⟩

Nous illustrons ces étapes par un court exemple :

Considérons *minSup = 2* et en scannant la base de données d'origine (tableau 1) *PrefixSpan* trouvera par exemple que les *1-séquences fréquentes* sont *⟨{a}⟩, ⟨{b}⟩, ⟨{c}⟩, ⟨{e}⟩, ⟨{f}⟩,* et *⟨{g}⟩.* Ces séquences seront ainsi sorties. Ensuite, en supposant l'ordre lexicographique des éléments, *PrefixSpan* considérera d'abord l'élément *a* pour essayer de trouver des séquences fréquentes plus grandes commençant par le préfixe *⟨{a}⟩. PrefixSpan* va donc scanner la base de données d'origine pour construire la base de données projetée de *⟨{a}⟩,* illustrée dans le tableau 2. La base de données projetée du modèle *⟨{a}⟩* est l'ensemble des séquences où le motif *⟨{a}⟩* apparaît, mais où tous les éléments et ensembles d'éléments apparaissant avant la première occurrence de *⟨{a}⟩* ont été supprimées.

Ensuite, pour trouver des modèles séquentiels fréquents commençant par le préfixe *⟨{a}⟩* contenant un élément de plus, l'algorithme *Prefixspan* lira la base de données projetée de *⟨{a}⟩* et comptera le support de tous les éléments apparaissant dans cette base qui pourraient être ajoutés soit par *i-extension* ou *s-extension* à *⟨{a}⟩*. Par exemple, les *2-séquences* qui sont des *i-extensions* ou des *s-extensions* de *⟨{a}⟩* sont : *⟨{a,b}⟩ : 2, ⟨{a,d}⟩ : 1, ⟨{a},{a}⟩ : 1, ⟨{a,b}⟩ : 3, ⟨{a},{c}⟩ : 2, ⟨{a},{e}⟩ : 3, ⟨{a, f}⟩ : 4*, et *⟨{a},{g}⟩ : 2*, où pour chaque motif, le nombre après les deux points (:) indique son support. Ensuite, *PrefixSpan* sortira les motifs séquentiels (ceux ayant un support supérieur ou égal à 2), c'est-à-dire : *⟨{a,b}⟩, ⟨{a},{b}⟩, ⟨{a},{c}⟩, ⟨{a},{e}⟩, ⟨{a},{f}⟩, et ⟨{a},{g}⟩.*

PrefixSpan poursuivra son exploration approfondie de l'espace de recherche en tentant de trouver des modèles séquentiels commençant par le préfixe *⟨{a,b}⟩. PrefixSpan* analysera la base de données du modèle *⟨{a}⟩* pour créer la base de données projetée de *⟨{a,b}⟩.* Cette base de données est décrite dans le tableau 3. Lors de la création de la base de données projetée de *⟨{a,b}⟩, PrefixSpan* comptera le support des éléments pouvant étendre *⟨{a,b}⟩* par *i-extension* de *s-extension*. Ce processus se poursuivra ensuite de la même manière (en poursuivant la recherche en profondeur d'abord) jusqu'à ce que tous les modèles séquentiels aient été trouvés.

L'approche de croissance de modèles de *PrefixSpan* a l'avantage de n'explorer que les modèles apparaissant dans la base de données (contrairement à de nombreux autres algorithmes d'extraction de modèles séquentiels).

3.3.4 Post-Traitement des motifs

Plusieurs motifs obtenus à partir de la méthode *PréfixSpan* sont inclus l'un dans l'autre. Par exemple, avec *minSup = 28, minLeng = 5* et *maxLeng = 10,* les séquences :

- [4,4,4,4,3] ⊂ [4,4,4,4,3,0] ⊂ [4,4,4,4,3,0,0]
- [4,4,4,4,0] ⊂ [4,4,4,4,0,0]
- [4,4,4,3,0] ⊂ [4,4,4,3,0,0]

Le but est de faire une optimisation de tel sorte que nous gardons seulement les motifs différents. En utilisant l'algorithme de *PostTrain*, le résultat s'affiche comme suit :

- [4,4,4,4,3,0,0]
- [4,4,4,4,0,0]
- [4,4,4,3,0,0]
- [4,4,4,0,0]
- [4,4,3,0,0]

Nous remarquons dans cet exemple que les motifs sont passés de 9 à 5 (réduction de 4 motifs).

```
ENTRE : a[]// Tableau of all extracted patterns
Variables : f , p , i
SORTIE : post[]// empty list to add different patterns
DEBUT
        while (i < Len(a) ) :
                f = a[i]
                p = a[i+1]
                if ( Len(f)  >= Len(p) || NOT Contains(f, p) ):
                     post.append(f)
                else :
                      // Ignore Pattern
               i++
         END While
FIN
```

Example :

a[0] = [1,1,1]
a[1] = [1,1,1,2]
a[2] = [1,1,1,2,2]
a[3] = [1,1,3]

PostTrain(a) :

post[0] = [1,1,1,2,2]
post[1] = [1,1,3]

Le principe de fonctionnement de l'algorithme Post-Train se fait comme suit :

- l'algorithme prend en entrée un tableau de tous les motifs obtenus dans la phase de traitement et en sortie une liste vide qui va etre remplis par les motifs différents.
- Soit f une variable qui resoit un tableau *a[]* à la position (i) et p une variable qui resoit un tableau *a[]* à la position (i+1)
- Si la langueur de f est superieur ou egale a la langueur de p ou bien f n'est pas inclut dans p donc le motif f va etre ajouter a la liste post [] sinon le motif va etre ignoré
- Incrémenté l'indice i et répéter les étapes jusqu'à i atteint la langueur du tableau a

3.3.5 Classification de nouvelles données

Dans cette étape, nous cherchons à classer de nouvelles données à partir de la base de données d'origine traitée de tel sorte que nous prouvons qu'il existe réellement une similarité de motifs entre eux.

```
ENTRE : l1,l2// l1 for newData , l2 for patterns
Variables : i,j // i,j to browse l1 and l2
SORTIE : pattern[] // list to add patterns if exist in newData
DEBUT
        For each pattern do
        while (i < len(l1) and j < len(pattern)  ) :
                x = l2[j]
                if ( l1[i] == l2[j] ):
                        j++
                        pattern.append(x)
                else :
                        j = 0
                        pattenr.clear()
                i++
         END While
FIN
```

```
Example :
l1 = [1,1,1,2,2,4,4,1,1]
l2[0] = [2,2,4,4]
Classification(l1, l2[0]) :
l2[0] exist in l1

Classification(l1, l2[1]) :

l2[1] = [2,1,1,4]
l2[1] not exist in l1
```

L'algorithme prend en entrée deux listes, l1 pour la nouvelle donnée que nous voulons classer et l2 pour les motifs de notre base de données et en sortie une liste vide qui va contenir les motifs existants dans la nouvelle donnée ll.

Pour chaque motif tant que l'indice i est inferieur a la langueur de la nouvelle donnée l1 et l'indice j est inferieur a la langueur de motif de l2 répéter :

Si le motif l2 à la position (j) existe dans la nouvelle donnée l1 à la position (i), ajouter l2[j] a la liste pattern [] sinon remettre l'indice j de motif l2 a 0 et incrémenter l'indice i.

3.4 Conclusion

Dans ce chapitre, nous avons illustré notre approche sur un cas d'application en mettant en valeur le processus de travail qui est devisé en cinq phases importantes pour l'analyse et l'extraction de motifs à partir des données de capteur. Il s'agit de la collecte de données, prétraitement, traitement, post-traitement et classification où nous avons justifié le choix de méthodes utilisées et leur principe de fonctionnement.

Chapitre 4

Solution logicielle

4.1 Introduction

Ce chapitre porte sur la construction d'une application qui permet d'analyser et d'extraire les motifs fréquents dans les données de capteurs et de les classer en utilisant des approches de représentations de séries temporelles à l'aide des outils qui nous ont permis d'accomplir le travail. Les détails de ces choix feront l'objet de ce chapitre.

4.2 Environnement de travail

4.2.1 Python

Python est le langage de programmation choisit pour notre travail en raison de son interprétabilité de haut niveau et sa simplicité. C'est un langage de programmation open source multi-plateformes et orienté objet. Principalement utilisé pour le machine learning et la data science grâce à ses nombreuses bibliothèques telles *Pandas*, *Numpy*, *Scipy*, *Scrapy*, *Matpotlib*, *Scikit-Learn* ou encore *TensorFlow*, *Python* offre une grande flexibilité dans les tâches à effectuer et une grande compatibilité quelle que soit la plateforme utilisée.

4.2.2 Visual studio

Visual Studio est un éditeur de code open-source, gratuit et multi-plateforme (*Windows*, *Mac* et *Linux*), développé par *Microsoft* qui offre un moyen pratique et facile de créer des applications informatiques pour les systèmes d'exploitation *Microsoft Windows*. Il utilise plusieurs langages de programmations tels que *C++*, *C#*, *Python*, *VB.net*, etc.

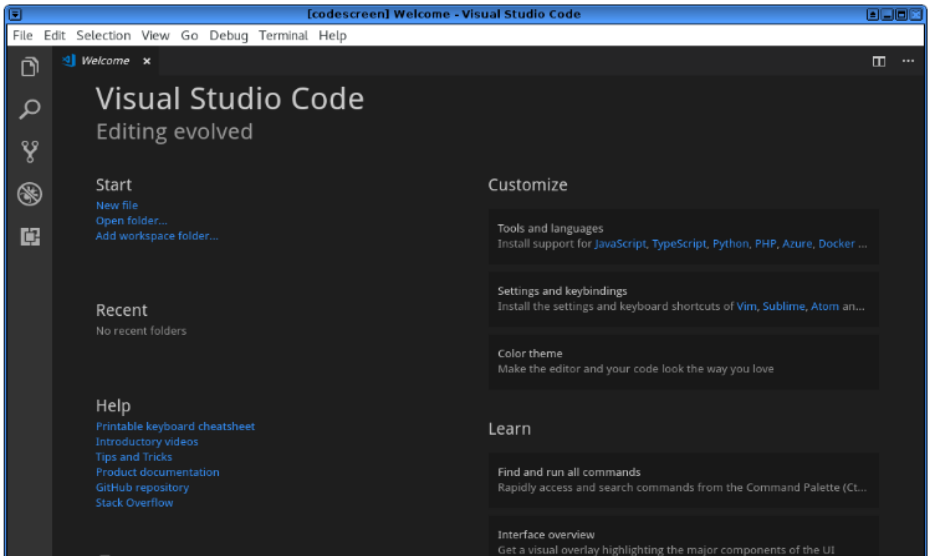


Figure 25 : *Visual Studio Code.*

Figure 26 : *Extensions de Python sur Visual Studio Code.*

4.2.3 Jupyter Notebook

Jupyter Notebook offre un moyen pratique de combiner un code *Python* (ou d'autres langages) avec un texte *Markdown* dans un seul canevas appelé *notebook*. L'avantage de *Jupyter Notebook* est qu'il permet d'exécuter et de modifier facilement des parties du code de manière sélective, sans avoir à exécuter le programme dans son intégralité. De plus, on peut intégrer du texte formaté (et des chiffres) dans son fichier, ce qui permet aux autres de lire et de modifier directement le même code.

Figure 27 : *Extension de Jupyter sur VS Code.*

4.2.4 Qt Designer

Qt Designer est un outil qui fournit une interface utilisateur pour créer des interfaces graphiques pour les applications *PyQt* de manière productive et efficace. Avec cet outil, on peut créer des interfaces graphiques en faisant glisser et en déposant des objets *QWidget* sur un formulaire vide. Après cela, on peut les organiser dans une interface graphique cohérente à l'aide de différents gestionnaires de mise en page. *Qt Designer* est indépendant de la plate-forme et du langage de programmation. Il ne produit pas de code dans un langage de programmation particulier, mais il crée des fichiers *.ui*. Ces fichiers sont des fichiers *XML* avec des descriptions détaillées sur la façon de générer les interfaces.

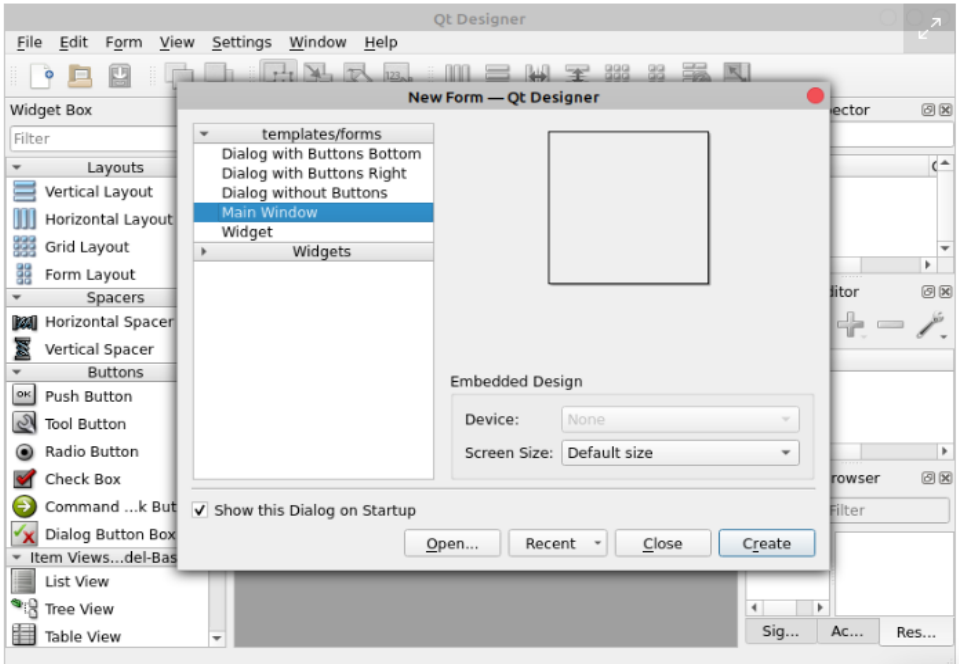


Figure 28 : *Qt Designer.*

4.3 Description de la solution logicielle

4.3.1 Interface d'accueil

L'interface principale s'affiche lors du double-clique sur l'exécutable du projet. Dans cette fenêtre, l'utilisateur doit choisir la source de données.

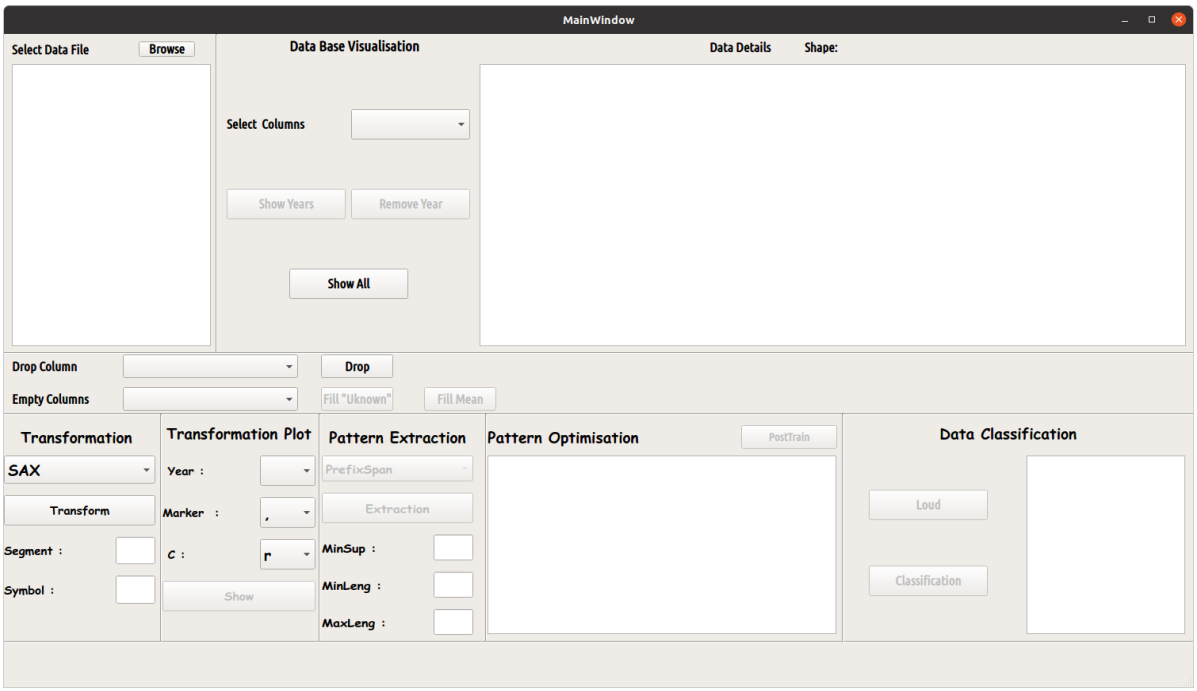


Figure 29: *Interface principale.*

4.3.2 Interface de chargement de données

En faisant une clique sur le bouton *Browse*, l'utilisateur peut sélectionner un fichier *CSV* pour le visualiser. Ce dernier s'affiche sous forme de table qui se trouve dans le côté droit accompagner des détails sur sa forme (lignes/colonnes) affichés au-dessus d'elle, ainsi que le type de données sur la table qui se trouve dans le côté gauche de l'interface.

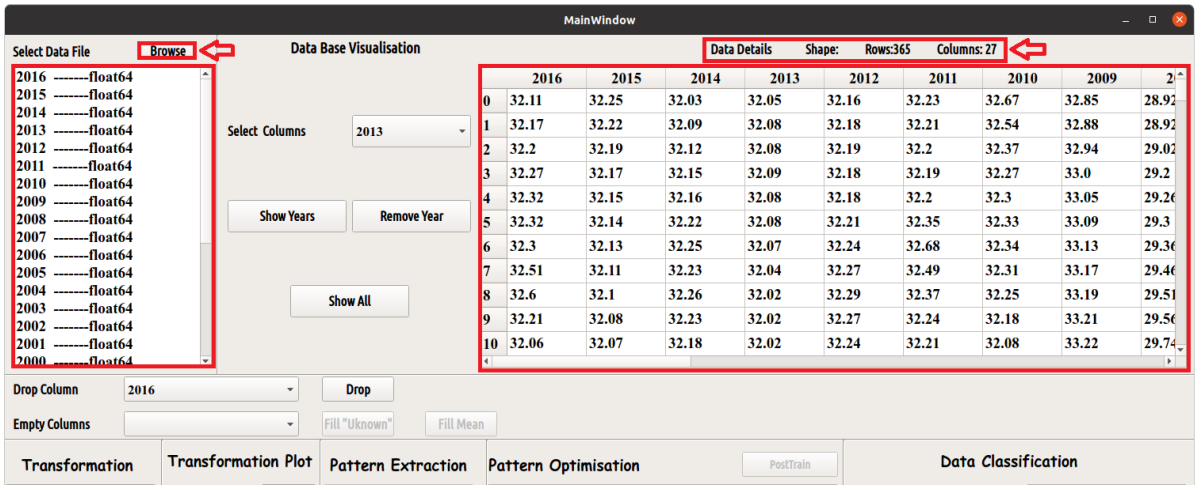


Figure 30 : *Chargement de données.*

4.3.3 Interface de visualisation des données

L'utilisateur peut aussi visualiser les données sous forme de courbe (figure 32) en cliquant sur le bouton *ShowAll* du panneau *Data Base Visualisation* (figure 31).

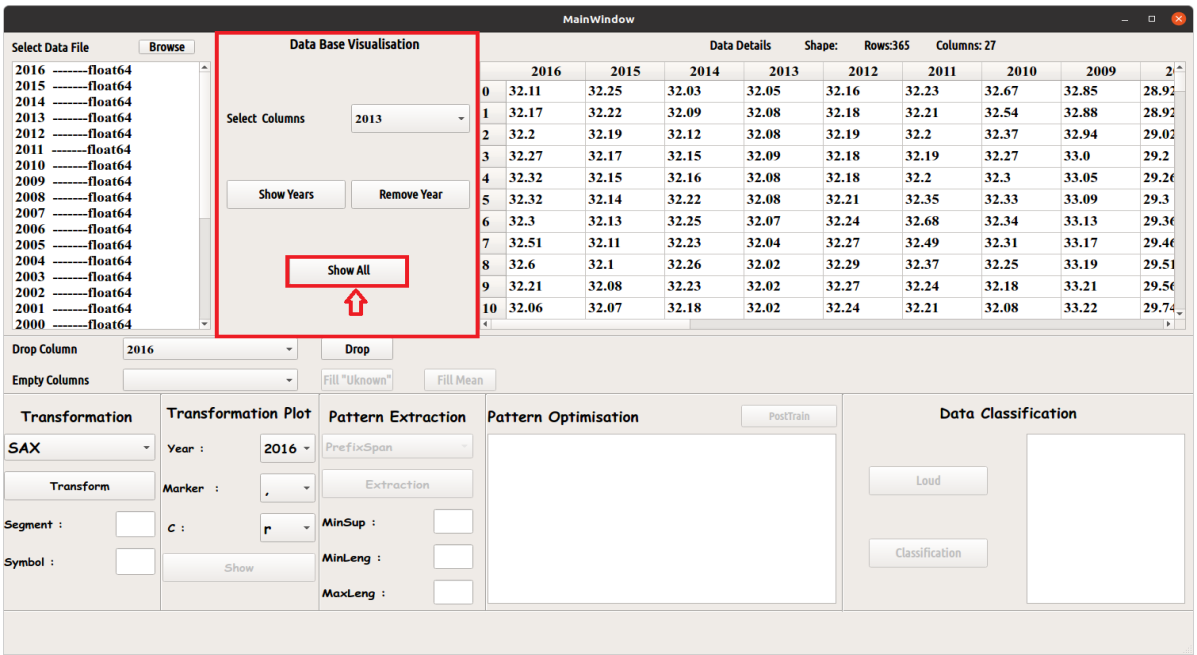


Figure 31 : *L'emplacement de bouton Database Plot.*

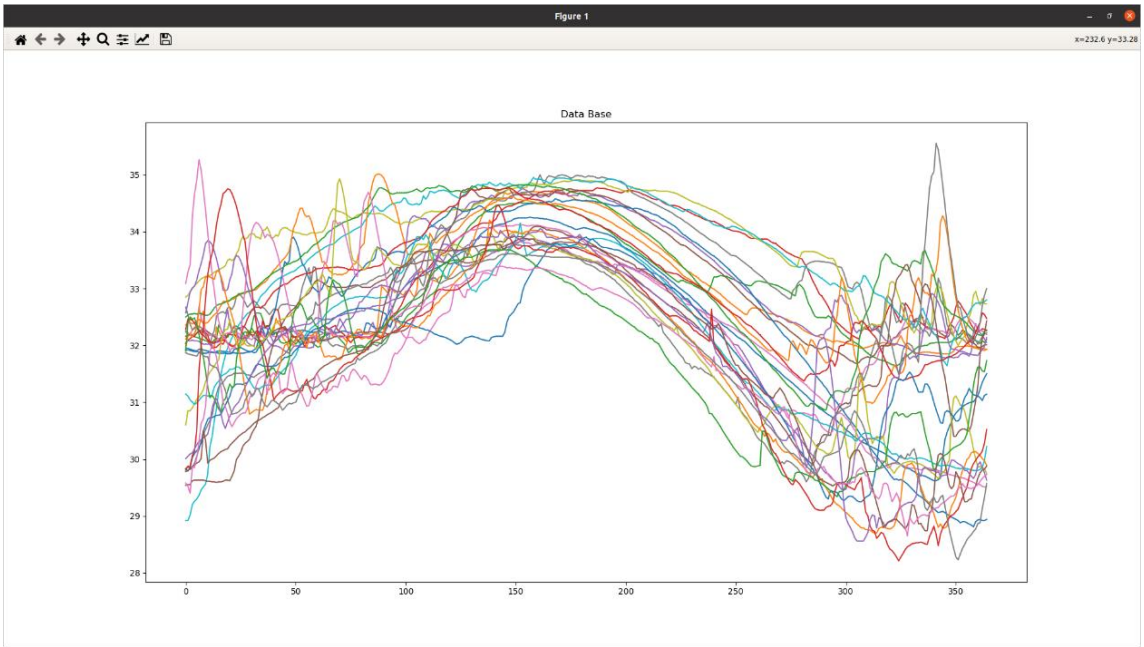


Figure 32 : *Database Plot.*

L'utilisateur peut sélectionner une ou plusieurs années à partir de la table gauche pour visualiser les données en faisant un double clic sur chaque série souhaitée qui va être chargée par la suite dans la partie *Select Columns* (figure 33). Si l'utilisateur se trompe ou veut annuler une série de la liste, il n'a qu'à cliquer sur le bouton *Remove year* avant de cliquer sur *Show years* pour visualiser les données sélectionnées (figures 34 et 35). Cette étape débloque la fonctionnalité de la transformation qui se trouve juste après.

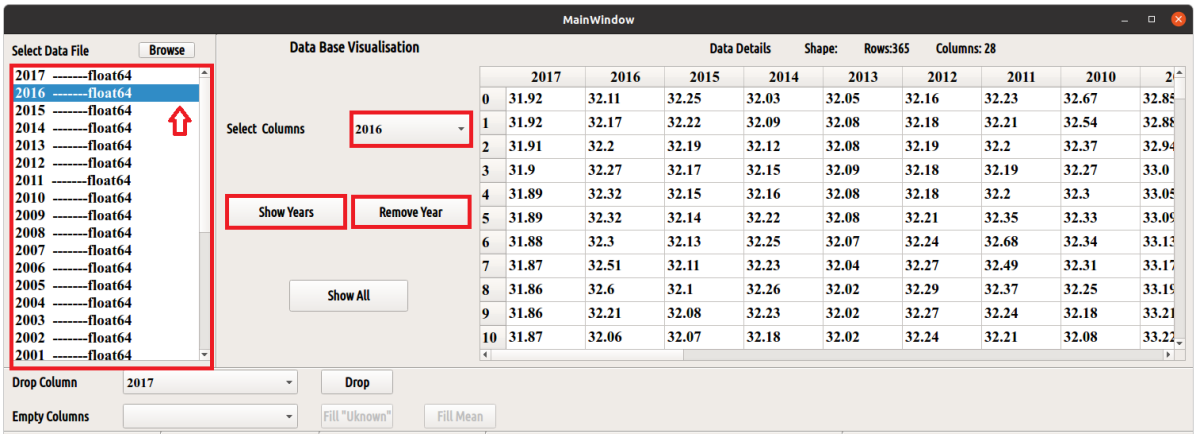


Figure 33 : *Démonstration pour visualisation les données sélectionnées.*

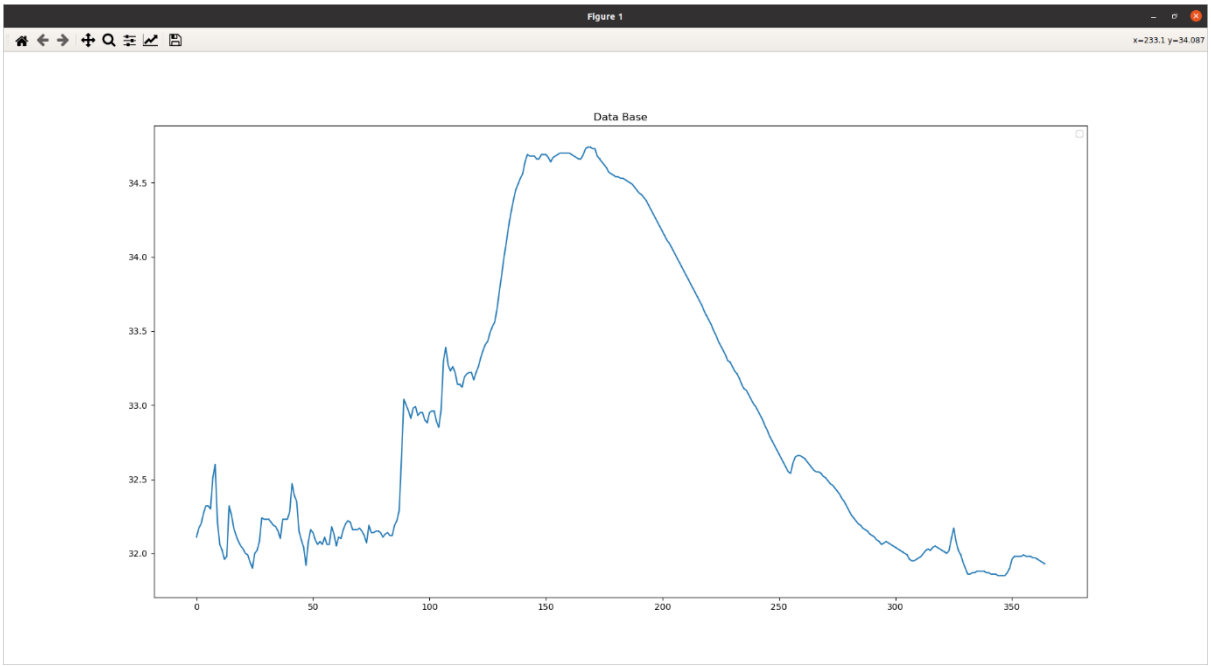


Figure 34 : *Visualisation d'une série particulière.*

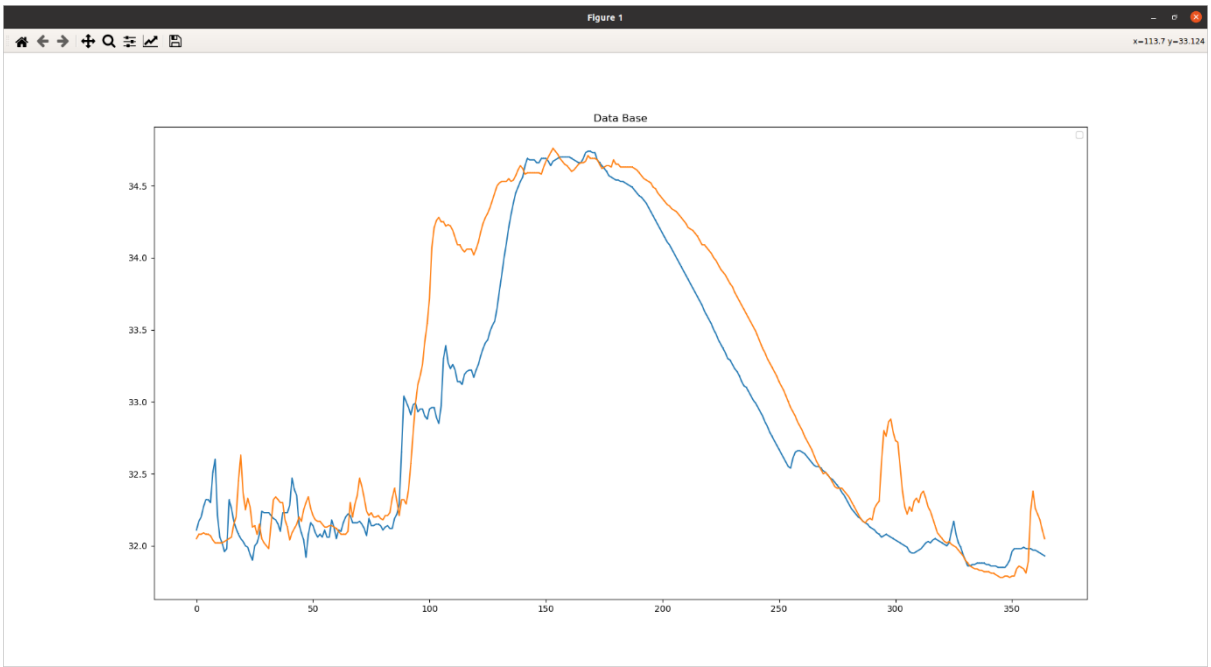


Figure 35 : *Visualisation de deux séries selon la sélection de l'utilisateur.*

4.3.4 Prétraitement des données

Apres l'étape de visualisation, l'utilisateur peut faire une transformation de données en les discrétisant en une chaîne symbolique afin de réduire la dimensionnalité tout en étant indexée par une mesure de distance à limite inférieure. Cette étape est particulièrement efficace pour la découverte de motifs. Tout d'abord, l'utilisateur sélectionne la méthode *SAX*, puis, il clique sur le bouton *Transform* qui permet d'afficher une boite de dialogue à remplir avec le nombre de segments et de symboles. Ensuite il clique sur le bouton *Ok* afin de renvoyer ces paramètres aux cases vides de *Segment* et *Symbole* dans la partie transformation qui se trouvent dans l'interface principale. Enfin, en cliquant sur le bouton *Show*, les symboles s'affichent dans la table qui se trouve à droite (figure 37) et débloque les boutons de l'étape suivante (*Pattern Extraction*).

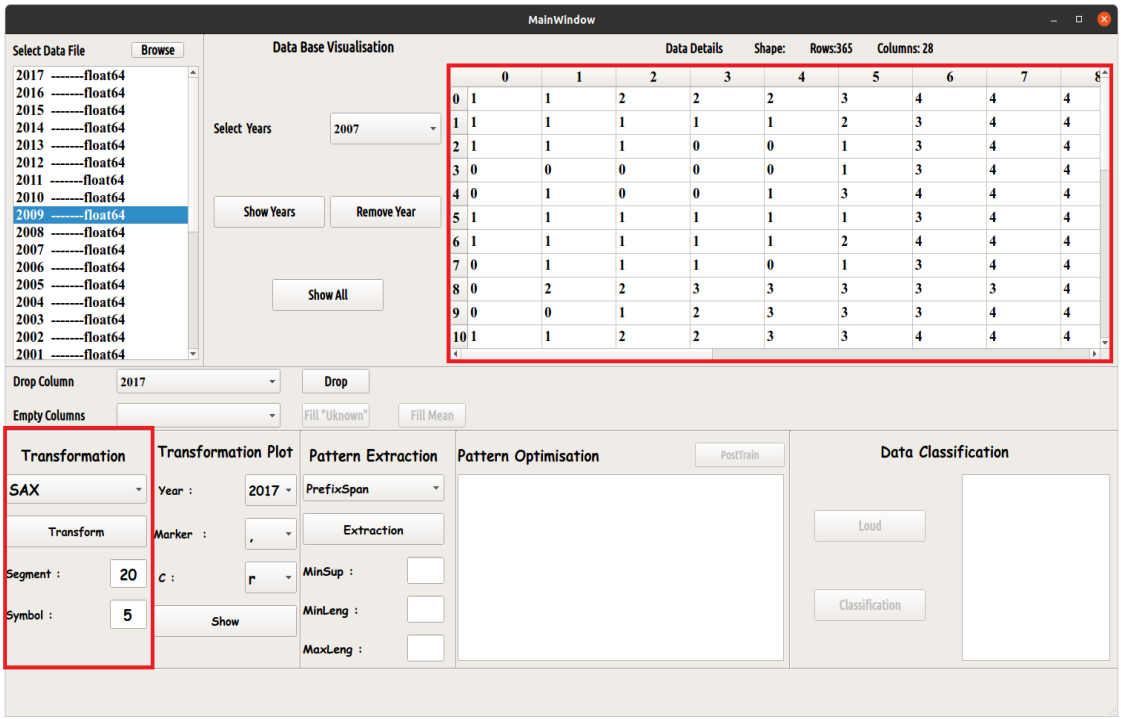


Figure 36 : *Visualisation sous forme de table de la transformation avec SAX.*

L'utilisateur peut aussi visualiser la transformation des données sous forme graphique en sélectionnant la série désirée, en cliquant sur le bouton *Show* (figure 38).

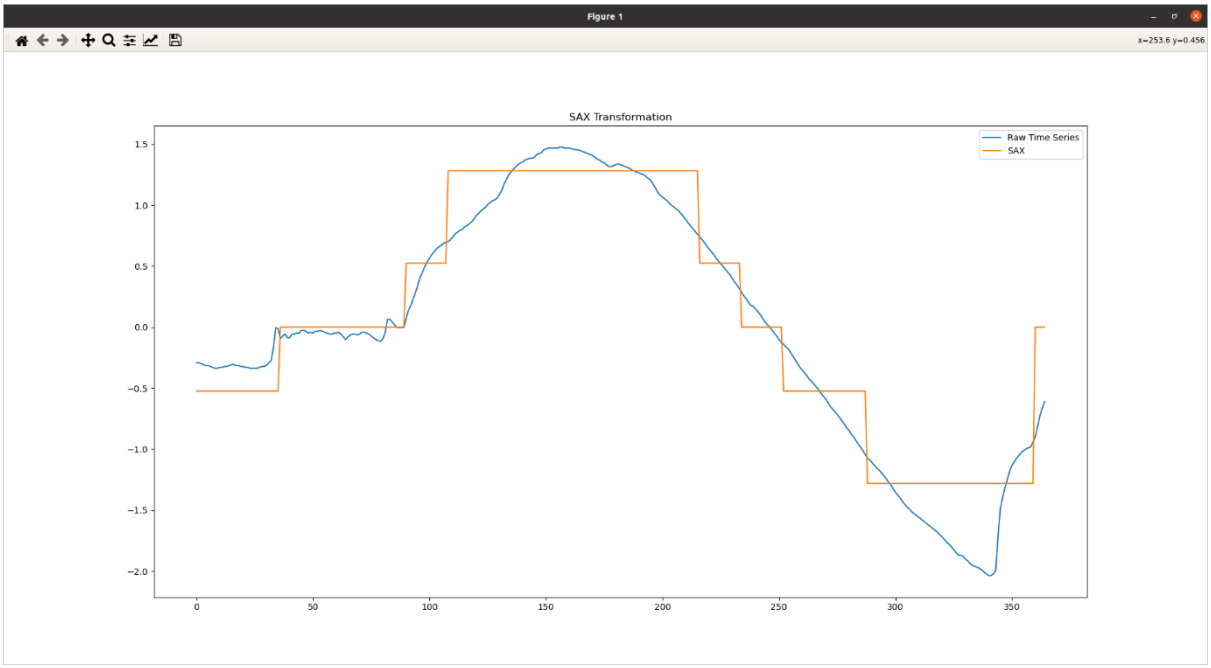


Figure 37 : *Visualisation du résultat de la méthode SAX.*

4.3.5 Traitement des données

Après la transformation, l'utilisateur peut faire un traitement afin d'extraire les motifs fréquents cachés dans la base de données en sélectionnant la méthode *PréfixSpan* et en cliquant sur le bouton *Extraction* qui permet d'afficher une boite de dialogue (figure 39) à remplir avec le minimum support de la séquence, c'est-à-dire sa fréquence d'apparition minimale, ainsi qu'ajouter sa langueur minimale et maximale dans les cases *MinLeng* et *MaxLeng* respectivement en terminant par un simple clic sur le bouton *Ok* afin de les renvoyer aux cases vides correspondantes qui se trouvent dans l'interface principale. Les résultats sont affichés sur la table qui se trouve juste à côté.

Exemple démonstratif :

Soit *MinSup = 28*, *MinLeng = 5* et *MaxLeng = 10* saisis par l'utilisateur. Les résultats sont affichés comme suit (figure 39)

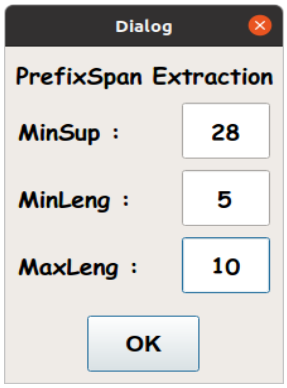


Figure 38 : *Paramètres de PrefixSpan.*

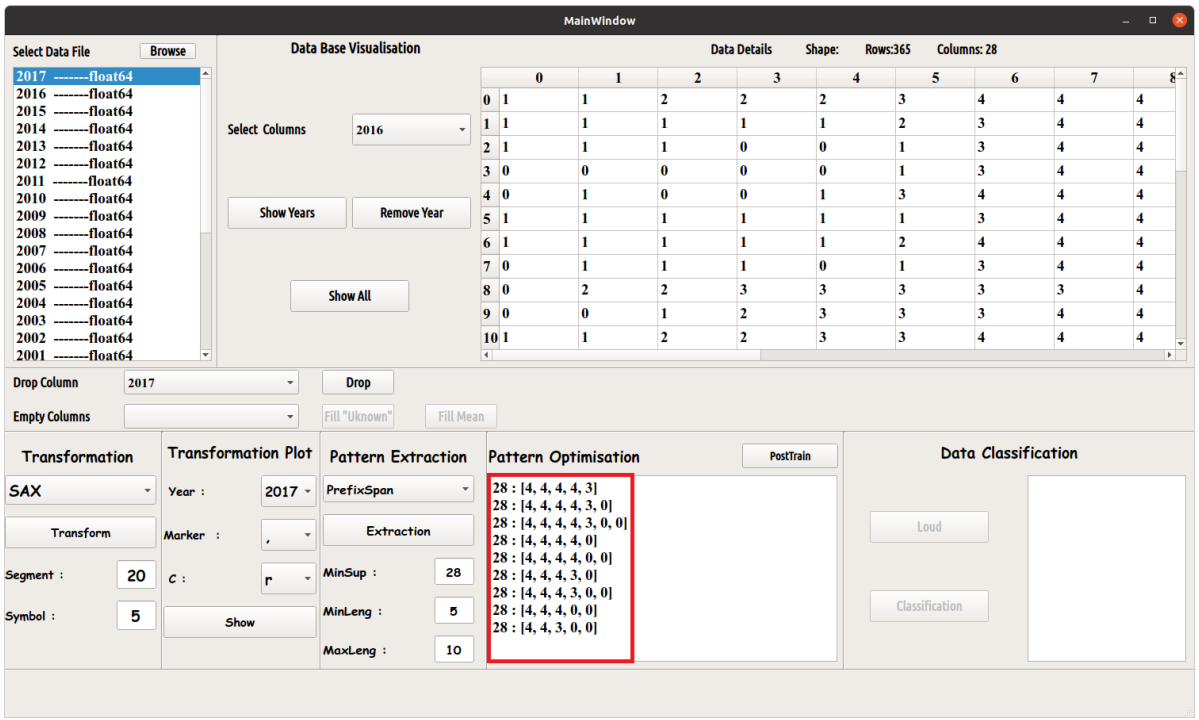


Figure 39 : *Extraction de motifs avec PrefixSpan.*

4.3.6 Post-Traitement de données

En cliquant sur le bouton *PostTrain*, le résultat s'affiche. Nous remarquons dans cet exemple que les motifs sont passés de 9 à 5 (réduction de 4 motifs).

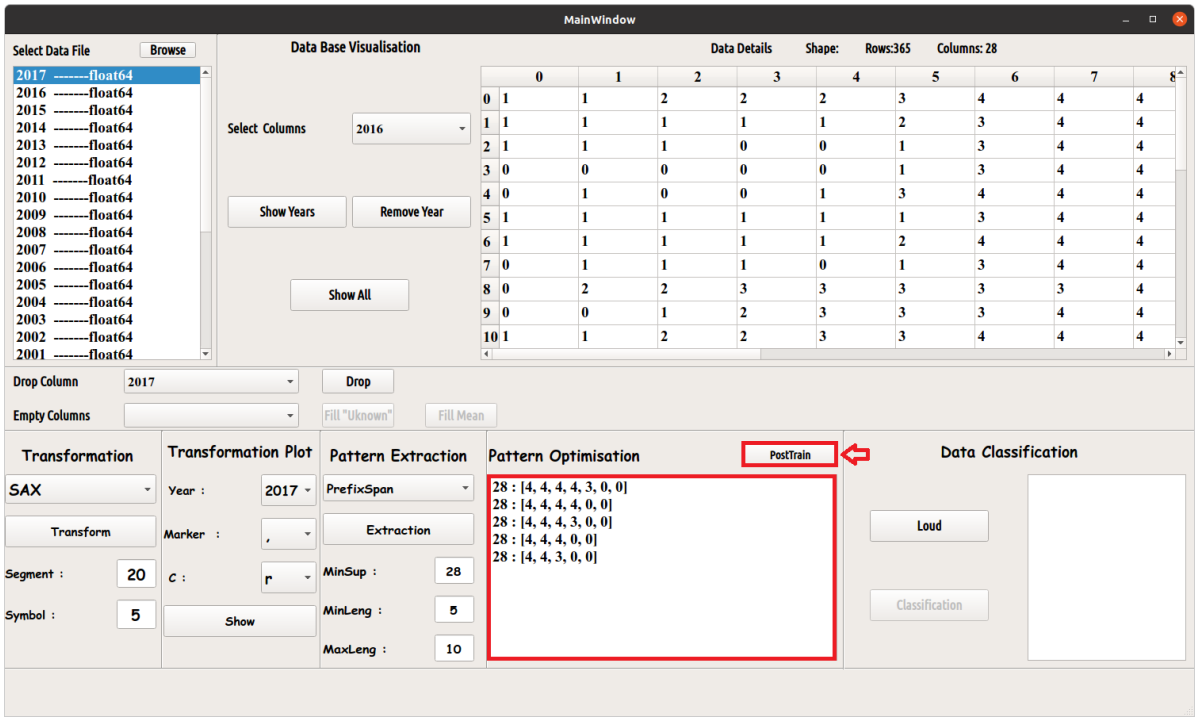


Figure 40 : *Optimisation de motifs avec PostTrain.*

4.3.7 Classification

En cliquant sur le bouton *Load*, l'utilisateur peut sélectionner une nouvelle donnée de séries temporelles qui s'affiche dans une table (figure 41) qui peut être classée par la suite en cliquant sur le bouton *Classification* pour vérifier qu'il existe un motif dans cette dernière similaire à l'un des motifs de la base de données d'origine (figure 42).

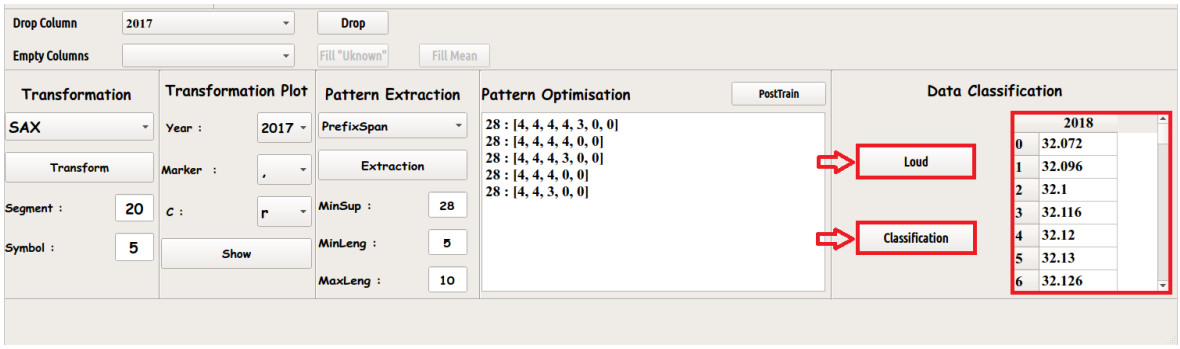


Figure 41 : *Classification de nouvelles données.*

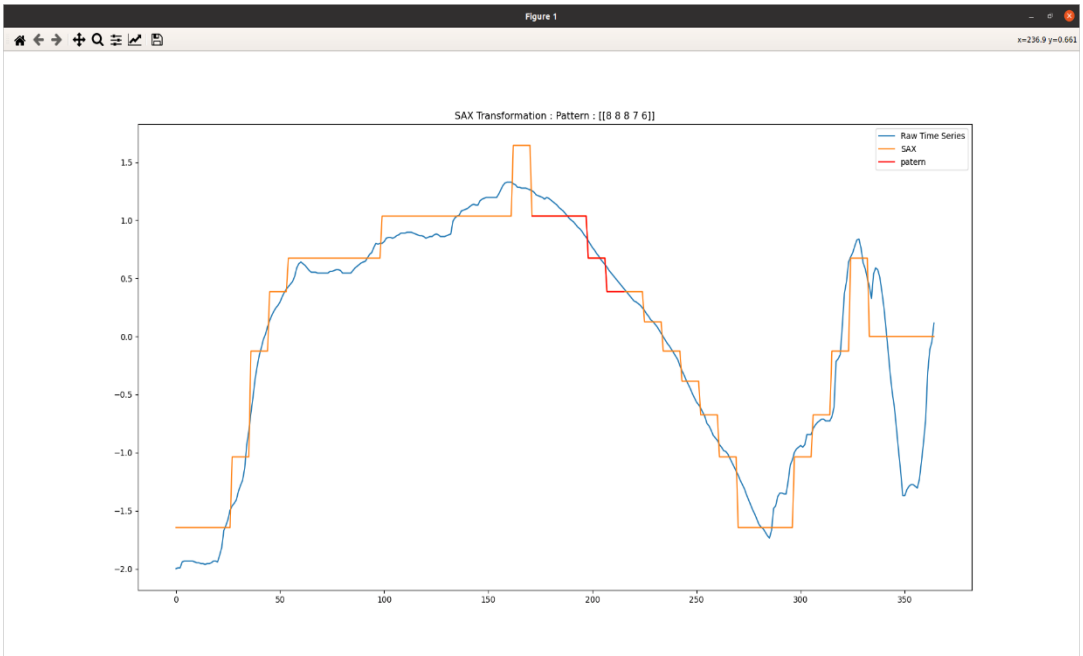


Figure 42 : *Visualisation de la nouvelle donnée en faisant comparaison avec un motif de base de données d'origine.*

4.4 Conclusion

Dans ce chapitre, nous avons décrit les étapes de la réalisation de notre projet, l'environnement matériel et logiciel, ainsi les détails des interfaces développées. En vue de ce qui a été fait dans l'application, on peut dire que notre application répond aux certaines caractéristiques suivantes :

- Une interface intuitive et simple à utiliser.
- Rapidité d'exécution.
- Une bonne performance des résultats.
- Un minimum d'erreurs.

Conclusion Générale

Les systèmes de prévision jouent un rôle important dans divers domaines et sont au cœur d'une grande partie de la science au cours de la dernière décennie en raison de leur capacité à découvrir des motifs qui peuvent être cachés dans de grandes bases de données et qui sont interprétables par l'homme, et donc utiles pour comprendre les données et pour la prise de décision. Ceci peut également avoir une grande utilité et une valeur réelle dans le domaine des séries temporelles issues des capteurs. Ce type de données est le plus difficile en raison des complexités créées par les relations préexistantes entre les valeurs adjacentes enregistrées par un capteur et qui varient progressivement dans le temps.

Le défi que nous avons rencontré réside dans la manière d'utiliser ce facteur explicitement traité dans le data mining en ciblant l'étape de prétraitement des données, ainsi que celle de traitement central. Cette dernière permet l'extraction de motifs proprement dite pour obtenir des résultats contextuels significatifs, comprendre le comportement de l'équipement surveillé par des capteurs durant son fonctionnement, prédire sa défaillance à travers les prochaines mesures qui aide à la prise de décision pour l'amélioration de fonctionnement de l'ensemble du système.

Au cours de ce projet nous avons fourni à la fois un aperçu sur le domaine incluant notre problématique, il s'agit du data mining et un aperçu d'un type particulier de ce dernier, les séries temporelles en faisant le point sur plusieurs techniques et approches qui ont essayés de résoudre ce problème, sur lesquels nous nous sommes basés afin de proposer une solution qui nous garantit des meilleures performances.

La suite de ce travail peut concerner la modélisation du comportement du capteur par des motifs, en détectant d'abord les motifs aberrants, puis de construire un ensemble de motifs exclusifs et exhaustifs décrivant le comportement du système.

Bibliographie

[1]. **H.Jiawei, K.Micheline et P.Jian.** *"Data Mining : Concepts and Techniques (Third edition)"*. s.l. : Morgan Kaufmann, 2012.

[2]. **C.Aggarwal.** *"Data Mining : The Textbook"*. Cham, Switzerland : Springer, 2015. 978-3-319-14141-1.

[3]. **H.Witten, F.Eibe et H.Mark.** *"Data Mining: Practical Machine Learning Tools and Techniques (Fourth edition)"*. s.l. : Morgan Kaufmann, 2017.

[4]. **S. Chehida, A. Baouya, S. Bensalem, and M. Bozga.** *"Learning and analysis of sensors behavior in IoT systems using statistical model checking"***.** Software Quality Journal : Elsevier, 2021, Vol. 01. 11219-021-09559.

[5]. **U.Fayyad, P.Shapiro, Gregory et S.Padhraic."***From Data Mining to Knowledge Discovery in Databases"*. Providence, Rhode Island : s.n., July 27–31, 1997, AAAI.

[6]. **M.Bramer.** *"Principles of Data Mining. London"*.United Kingdom : Springer, 2020. 978-1-4471-7492-9.

[7]. **C.Aggarwal.** *"Neural Networks and Deep Learning : A Textbook"*. Cham, Switzerland : Springer, 2018. 978-3-319-14141-1.

[8]. **D.James, K.Ron et S.Mehran."***Supervised and unsupervised discretization of continuous features"*. San Francisco, CA : Morgan Kaufmann, 1995. Proceedings of the Twelfth International Conference on Machine Learning. pp. 194-202. 978-1-55860-377-6.50032-3.

[9]. **S.Chehida, J.Claude T.Mfumu."***Analysis and Prediction of Viral Infections using Statistical"*. Grenoble, France : s.n.

[10]. **R.Thomas, Y.S.Koh,P.F.Viger,J.Chun-Wei Lin,R.U.Kiran.** "*A Survey of Sequential Pattern Mining*". February 2017.

[11]. **Agrawal, Rakesh et Srikant, Ramakrishnan."***Fast algorithms for mining association rules in large databases"*. San Mateo : Morgan Kaufmann, 1994. Proceedings

of the 20th international conference on very large databases (VLDB94). pp. 487–499. 1558601538.

[12]. **J.Lin, E.Keogh, L.Wei et S.Lonardi.** *"Experiencing SAX : a novel symbolic representation of time series"*. 2007, Data Mining and Knowledge Discovery, pp. 15:107–144.

[13].**M.Shokoohi-Yekta, Y.Chen, B.Campana, Z.Bing-Hu-Jesin et E. Keogh.***"Discovery of Meaningful Rules in Time Series"*. University of California, Riverside : s.n.

[14]. **E.Keogh.** *"Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases"***.** 2001, Knowledge Information Systems, pp. 3:263–286.

[15].**H.Jiawei, P.Jian et Y.Yiwen.***"Mining frequent patterns without candidate generation"*. s.l. : ACM Press, 2000.

[16]. **H.Trevor, R.Tibshirani et J.Friedman.** *"The Elements of Statistical Learning"*. s.l. : Springer, 2009.