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 :

RIF Hanifa

BENAIED Soumia

THEME :

Etude de la technique « Matrix Profile » pour l'analyse des séries temporelles

Soutenu le : 19 juin 2021

Devant le jury composé de :

HAMAMI Dalila	MCB	Université de Mostaganem	Président
HASSAINE Farida	MAA	Université de Mostaganem	Examinateur
BENAMEUR Abdelkader	MAA	Université de Mostaganem	Encadreur

Année Universitaire 2020-2021

Dédicaces

Nous dédions ce travail qui n'aura jamais pu voir le jour sans les soutiens indéfectibles et sans limite de nos chers parents qui ne cessent de nous donner avec amour le nécessaire pour que nous puisons arriver à ce que nous somme aujourd'hui.

Que dieux vous protège et que la réussite soit toujours à nos portées pour que nous puisons vous combler de bonheur.

Nous dédions aussi ce travail à :

Nos grands-parents.

Nos frères.

Nos oncles, nos tantes et leur famille.

Tous nos cousins et cousines.

Tous nos amis, nos collègues et tous ceux qui nous estiment.

Remerciements

Tout d’abord, nous tenons à remercier le bon Dieu le tout Puissant de nous avoir donné la force et le courage de mener à bien ce modeste travail,

Nous tenons à exprimer toute notre reconnaissance à notre directeur de mémoire, Monsieur BENAMEUR Abdelkader. Nous le remercions de nous avoir encadré, orienté, aidé et conseillé.

Nous adressons nos sincères remerciements à tous les enseignants, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé nos réflexions et ont accepté de nous rencontrer et de répondre à nos questions durant nos recherches.

Nous remercions nos très chers parents, qui ont toujours été là pour nous, et qui nous encouragé et aidé à arriver à ce stade de notre formation.

. Nous remercions nos frères pour leurs encouragements.
Enfin, nous remercions nos amis qui ont toujours été là pour nous. Leur soutien inconditionnel et leurs encouragements ont été d’une grande aide.

À tous ces intervenants, nous présentons nos remerciements, nos respects et nos gratitudes.

Résumé

La technique « *Matrix Profile* » est considérée comme étant l'état de l'art actuel en tout ce qui concerne l'analyse des séries temporelles tels que l'extraction des motifs, la détection d'anomalies, le clustering et autres. Ces tâches trouvent leur utilisation dans divers domaines comme la médecine, le commerce, la robotique ou l'industrie. L'objectif du travail est d'explorer les principes des cette technique et les algorithmes permettant de la mettre en œuvre, pour ensuite l'exploiter afin d'analyser de grandes bases de séries temporelles.

Mots-clés :

Data mining, fouille de données, séries temporelles, Matrix Profile, extraction de motifs.

Abstract

The « *Matrix profile* » technique is actually considered as the state of the art in the field of time series analysis, such in motifs discovery, anomaly detection, clustering and others. This tasks are used in various fields like medicine, commerce, robotic and industry. The aim of this work is to explore the principles of this technique and the algorithms allowing to implement it, and then exploit it to analyse big time series databases.

Keywords :

Data mining, time series mining, Matrix Profile, pattern discovery.

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	Expression Complète	Page
FFT	Fast Fourier Transform	
KDD	Knowledge Discovery in Databases	
IRM	Information Ressource Managment	
ACP	Analyse en Composantes Principales	
MASS	Mueen'sAlgorithm for SimilaritySearch	
STAMP	Scalable Time seriesAnytime Matrix Profile	
STOMP	Scalable Time seriesOrdered Matrix Profile	
MP	Matrix Profile	
TS	Time Series	
DP	Distance Profile	

Table des matières

Introduction générale

Toutes les entreprises de nos jours collectent et stockent des quantités de données très grandes et qui ne cessent d'augmenter jour après jour. Ces « méga» bases de données sont des mines d'informations, elles cachent des connaissances décisives face au marché et à la concurrence, mais elles restent peu exploitées. Pour combler ce besoin, une nouvelle industrie est en train de naître : le data mining, ou extraction de connaissances à partir des données, qui propose d'utiliser un ensemble d'algorithmes issus de différentes disciplines scientifiques tel que les statistiques, l'intelligence artificielle et l'apprentissage automatique afin de construire des modèles à partir des données, autrement dit trouver des schémas intéressants selon des critères fixés au départ et d'extraire un maximum de connaissances utiles à l'entreprise.

L'un des types de données les plus concernés par le data mining est celui des séries temporelles qui sont présentes dans de nombreux domaines d'application. Leur champ d'application est très vaste et s'étend de l'astronomie à l'économie et finance en passant par la biologie, la psychologie, la géophysique ou la théorie du signal, etc. Elles ont donc suscité un très vif intérêt, et ont pour longtemps attirer l'attention des chercheurs, tant pour leur classification, clustering, recherche de motifs, et détection d'anomalies, que pour beaucoup d'autres tâches.

Dans notre travail, nous nous intéressons particulièrement à une technique parue récemment et devenue l'état de l'art actuel dans ce domaine. Cette technique, dite « *Matrix Profile* », se dit efficace pour la plupart des tâches d'analyse des séries temporelles, même avec les grandes bases de données. Le « *Matrix Profile* » est une structure de données et des algorithmes associés qui aident à résoudre différents problèmes comme celui de la détection des anomalies ou de la découverte de motifs. Il est robuste, évolutif et sans paramètre.

L'objectif du travail est d'explorer les fondements de la technique et ses principes, et d'étudier les différents algorithmes permettant de la mettre en œuvre. En fait, depuis l'apparition du « *Matrix Profile* », une succession d'algorithmes pour sa réalisation sont apparus apportant chacun une amélioration par rapport à son prédécesseur, pour arriver jusqu'à franchir la barre d'analyser des millions de séries temporelles.

Nous avons organisé notre rapport de la façon suivante :

Dans le premier chapitre nous allons commencer par introduire le data mining en donnant quelques définitions, un petit historique, ainsi que ses domaines d'application. Ensuite, nous allons expliquer ses tâches, pour conclure par aborder ses différentes méthodes. Nous terminons par les différents types de données de data mining.

Dans le deuxième chapitre nous allons définir les séries temporelles, pour passer ensuite aux détails du « *Matrix Profile* ». Le chapitre se poursuit par la présentation de quelques algorithmes proposés pour mettre en œuvre la technique.

Le troisième chapitre sera consacré à l'introduction de l'approche adoptée, à savoir l'utilisation du « *Matrix Profile* » pour l'extraction des motifs, la détection d'anomalies, et enfin la classification par l'extraction des *shaplets*. L'implémentation de l'approche dans le langage *Python* est détaillée dans une seconde partie du chapitre.

Finalement, nous allons terminer notre rapport par une conclusion générale qui contient les perspectives possibles qui vont guider la suite de notre travail.

Chapitre I :

Le data mining

1.1. Introduction

Ce chapitre est conçu pour la présentation du data mining en premier lieu. Nous allons essayer de le définir et d'effectuer une étude de ses différents domaines d'application. Après, nous allons expliciter ses tâches pour résoudre les difficultés ou besoins de l'entreprise. Enfin, nous allons étudier ses méthodes pour arriver à exploiter les quantités importantes de données.

1.2. Définition et historique

C'est un regroupement de méthodes appliquées aux bases de données volumineuses et complexes. Il s'agit d'accéder à la suppression du caractère aléatoire et de découvrir le modèle caché. Nous utilisons des outils, des méthodologies et des théories d'exploration de données afin de dévoiler des modèles dans les données. Ceci explique clairement que c'est bien la raison pour laquelle ce sujet prend son immense place d'étude [5] [2] [8].

Le *data mining* est définie aussi par :

- L'analyse de grands ensembles de données observationnelles pour découvrir de nouvelles relations entre elles et de les reformuler afin de les rendre plus utilisables de la part de ses propriétaires [2].
- Un processus inductif, itératif et interactif dont l'objectif est la découverte de modèles de données valides, nouveaux, utiles et compréhensibles dans de larges bases de données [4].
- Fouille et découverte de connaissances dans les données [8].
- Modélisation des données [3].
- Repose sur l'apprentissage automatique (*Machine Learning*) pour l'analyse et l'extraction de connaissance à partir de grandes quantités de données.

1.2.1. Historique du data mining

L'expression « *data mining* » est apparue vers le début des années 1960 et avait, à cette époque, un sens péjoratif. En effet, les ordinateurs étaient de plus en plus utilisés pour toutes sortes de calculs qu'il n'était pas envisageable d'effectuer manuellement jusque-là. Le succès de cette démarche empirique ne s'est pas démenti malgré tous les critiques au début [2] [8].

L'analyse des données s'est développée et son intérêt grandissait en même temps que la taille des bases de données. Vers la fin des années 1980, des chercheurs en bases de données, tels que *Rakesh Agrawal*, ont commencé à travailler sur l'exploitation du contenu des bases de données volumineuses, comme celles des tickets de caisses des grandes surfaces, convaincus de pouvoir valoriser ces masses de données dormantes. Ils utilisèrent l'expression « *Database mining* » mais, celle-ci étant déjà déposée par une entreprise (*Database mining workstation*), ce fut « *data mining* » qui s'imposa [2].

En 1989, *Shapiro Piatetski* proposa le terme « *Knowledge discovery* ». Actuellement, les termes *data mining* et *Knowledge Discovery in Databases* (KDD) sont utilisés plus ou moins indifféremment. Nous emploierons par conséquent l'expression « *data mining* », celle-ci étant la plus fréquente dans la littérature [4].

La communauté de data mining a initié sa première conférence en 1995 à la suite de nombreux workshops sur le KDD entre 1989 et 1994. En 1998 s'est créé, sous les auspices de l'*Association for Computing Machinery* (ACM), un chapitre spécial baptisé *ACM-SIGKDD* (*Special Interest Group in KDD*), qui réunit la communauté internationale du *KDD*. La première revue du domaine, *Data Mining and Knowledge Discovery Journal*, est publiée par *Kluwer* depuis 1997 [4].

1.3. Le processus d'ECD

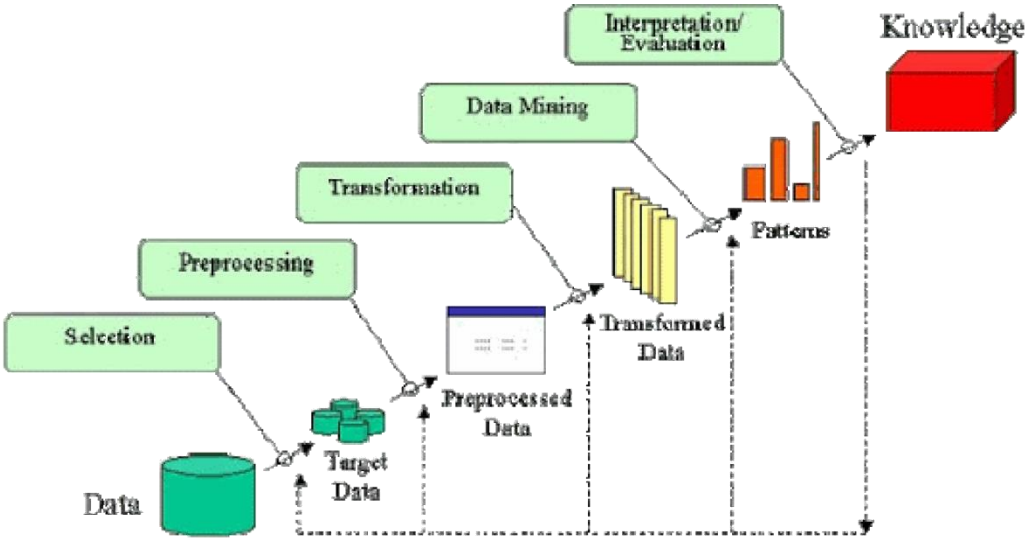

Figure 1 – *Le processus d'Extraction de Connaissances à partir des Données [4]*.

Voici une brève description du processus KDD en neuf étapes, en commençant par une étape de gestion [11] :

1.3.1. Développer une compréhension du domaine d'application

Il s'agit de l'étape préparatoire initiale. Les responsables d'un projet KDD doivent comprendre et définir les objectifs de l'utilisateur final et l'environnement dans lequel le processus de découverte de connaissances se déroulera (y compris les connaissances antérieures pertinentes). Au fur et à mesure que le processus KDD progresse, il peut même y avoir une révision et un ajustement de cette étape. Après avoir compris les objectifs KDD, le prétraitement des données commence, comme décrit dans les trois étapes suivantes (notons que certaines des méthodes présentées ici sont similaires aux algorithmes de Data Mining, mais sont utilisées dans le contexte du prétraitement) :

1.3.2. Sélection et création d'un ensemble de données

Après avoir défini des objectifs, il faut déterminer les données qui seront utilisées pour la découverte de connaissances. Il s'agit de finaliser les données disponibles, d'obtenir les données supplémentaires nécessaires, puis d'intégrer toutes les données pour la découverte de connaissances en un seul ensemble de données, y compris les attributs qui seront pris en compte pour le processus.

Ce processus est très important car le data mining apprend et découvre à partir des données disponibles. C'est la base de preuves pour construire les modèles. Si certains attributs importants manquent, alors l'étude entière peut échouer. Pour le succès du processus, il est bon de considérer autant d'attributs que possible à ce stade. D'autre part, la collecte, l'organisation et l'exploitation d'entrepôts de données complexes sont coûteuses, et il y a un compromis avec la possibilité de mieux comprendre les phénomènes. Ce compromis permet à l'aspect interactif et itératif du KDD de prendre place. On commence avec le meilleur ensemble de données disponible, puis on l'élargit et on observe l'effet en termes de découverte de connaissances et de modélisation.

1.3.3. Prétraitement et nettoyage

Dans cette étape, la fiabilité des données est améliorée. Elle comprend le nettoyage des données, comme le traitement des valeurs manquantes et la suppression du bruit ou des valeurs aberrantes. Plusieurs méthodes sont expliquées dans le manuel, allant de ne rien faire à devenir la partie principale (en termes de temps consommé) d'un processus KDD dans certains projets. Il peut s'agir de méthodes statistiques complexes, ou de l'utilisation d'un algorithme spécifique de data mining dans ce contexte.

Par exemple, si l'on soupçonne qu'un certain attribut n'est pas assez fiable ou a trop de données manquantes, alors cet attribut pourrait devenir l'objectif d'un algorithme de data mining supervisé. Un modèle de prédiction pour cet attribut sera développé, et les données manquantes pourront alors être prédites. L'extension à laquelle on prête attention à ce niveau dépend de nombreux facteurs.

1.3.4. La transformation des données

Dans cette étape, la génération de meilleures données pour le data mining est préparée et développée. Les méthodes utilisées ici comprennent la réduction des dimensions (comme la sélection et l'extraction de caractéristiques et l'échantillonnage d'enregistrements) et la transformation des attributs (comme la discrétisation des attributs numériques et la transformation fonctionnelle). Cette étape est souvent cruciale pour le succès de l'ensemble du projet KDD, mais elle est très spécifique au projet.

Par exemple, dans les examens médicaux, le quotient des attributs peut souvent être le facteur le plus important, et non pas chacun d'entre eux en soi. En marketing, nous pouvons considérer les effets hors de notre contrôle ainsi que des efforts et des questions temporelles (comme l'étude de l'effet de l'accumulation de la publicité). Cependant, même si nous n'utilisons pas la bonne transformation au début, nous pouvons obtenir un effet surprenant qui nous indique la transformation nécessaire (dans l'itération suivante). Ainsi, le processus KDD réagit sur lui-même et conduit à une compréhension de la transformation nécessaire (comme la connaissance concise d'un expert dans un certain document concernant les indicateurs clés avancés).

Après les quatre étapes précédentes, les quatre suivantes sont liées à la partie data mining, où l'accent est mis sur les aspects algorithmiques employés pour chaque projet.

1.3.5. Choisir la tâche de data mining appropriée

Il est temps de décider du type de data mining à utiliser, à savoir, la classification, la régression ou le clustering. Cela dépend surtout des objectifs du KDD, ainsi que des étapes précédentes. Il y a deux objectifs principaux : la prédiction et la description. La prédiction est appelée data mining supervisé, tandis que le data mining descriptif comprend les aspects non supervisés. La plupart des techniques de data mining sont basées sur l'apprentissage inductif, où un modèle est construit explicitement ou implicitement en généralisant à partir d'un nombre suffisent d'exemples d'apprentissage.

L'hypothèse sous-jacente de l'approche inductive est que le modèle formé est applicable aux cas futurs. La stratégie tient également compte du niveau de méta-apprentissage pour l'ensemble particulier de données disponibles.

1.3.6. Choix de l'algorithme de data mining

Ayant la stratégie, on décide maintenant de la tactique. Cette étape comprend la sélection de la méthode septique à utiliser pour la recherche de motifs (y compris les inducteurs multiples). Par exemple, en considérant la précision par rapport à la compréhensibilité, la première est meilleure avec les réseaux neuronaux, tandis que la seconde est meilleure avec les arbres de décision. Pour chaque stratégie de méta-apprentissage, il existe plusieurs possibilités de réalisation. Le méta-apprentissage se concentre sur l'explication des causes du succès ou de l'échec d'un algorithme d'exploration de données dans un problème particulier. Ainsi, cette approche tente de comprendre les conditions dans lesquelles un algorithme de data mining est le plus approprié. Chaque algorithme a des paramètres et des tactiques d'apprentissage (comme la validation croisée à dix reprises ou une autre division pour la formation et le test).

1.3.7. L'utilisation de l'algorithme de data mining

On arrive à la mise en œuvre de l'algorithme de data mining. A cette étape, on aura besoin d'employer l'algorithme plusieurs fois jusqu'à ce qu'un résultat satisfaisant soit obtenu, par exemple en réglant les paramètres de contrôle de l'algorithme, tels que le nombre minimum d'instances dans une seule feuille d'un arbre de décision.

1.3.8. Évaluation

Dans cette étape, on évalue et on interprète les modèles extraits (règles, fiabilité, etc.), par rapport aux objectifs définis dans la première étape. Nous considérons ici les étapes de prétraitement par rapport à leur effet sur les résultats de l'algorithme de data mining (par exemple, l'ajout de caractéristiques à l'étape 4, et la répétition à partir de là).

Cette étape se concentre sur la compréhensibilité et l'utilité du modèle induit. Dans cette étape, les connaissances découvertes sont également documentées pour une utilisation ultérieure. La dernière étape est l'utilisation et le retour d'information global sur les modèles et les résultats de la découverte obtenus par le data mining.

1.3.9. Utilisation des connaissances découvertes

Il est temps maintenant d'incorporer la connaissance dans un autre système pour une action ultérieure. Les connaissances deviennent actives dans le sens où nous pouvons apporter des changements au système et en mesurer les effets. En fait, le succès de cette étape détermine l'efficacité de l'ensemble du processus KDD. Par exemple, les connaissances ont été découvertes à partir d'un certain instantané statique (généralement un échantillon) des données, mais les données deviennent maintenant dynamiques. Les structures de données peuvent changer (certains attributs deviennent indisponibles), et le domaine de données peut être modifié (par exemple, un attribut peut avoir une valeur qui n'était pas supposée auparavant).

1.4. Domaines d'application

Il existe un nombre croissant d'applications réussies dans un large éventail des domaines aussi divers que l'analyse de l'imagerie satellitaire, l'analyse de composés organiques, le résumé automatique, la bio-informatique, les enquêtes criminelles, la gestion de la relation client, la prédiction de charge électrique, les prévisions financières, la détection de fraude, les soins de santé, l'analyse du panier de marché, le diagnostic médical, la conception des produits, l'évaluation immobilière, le marketing ciblé, la synthèse de texte, l'optimisation des centrales thermiques, l'analyse des dangers toxiques, les prévisions métrologiques [5] [2] [8].

1.4.1. Exemples d'applications

- Une chaîne de supermarchés pour optimiser le ciblage de clients de grande valeur.
- Une société de cartes de crédit pour la détection de fraude.
- Prédire la probabilité de défaut des demandes de crédit à la consommation en améliorant la capacité de prédire les créances douteuses.
- Les systèmes d'exploration de données peuvent passer au crible de vastes quantités de données collectées pendant le processus de fabrication des semi-conducteurs pour identifier les conditions qui causent problèmes de rendement.
- Prédire la probabilité qu'un patient cancéreux réponde à la chimiothérapie, réduisant ainsi les coûts des soins de santé sans affecter la qualité des soins.
- Analyse des données de capture de mouvement pour les personnes âgées.

- Extraction de tendances et visualisation dans les réseaux sociaux.
- Analyser les données d'un système de reconnaissance faciale pour localiser un criminel présumé dans une foule.
- Analyser des informations sur une gamme de médicaments et de composés naturels pour identifier des candidats significatifs pour de nouveaux antibiotiques.
- L'analyse des images IRM pour identifier d'éventuelles tumeurs cérébrales.

1.5. Tâches du data mining

Contrairement aux idées reçues, le data mining n'est pas le remède miracle capable de résoudre toutes les difficultés ou besoins de l'entreprise. Cependant, une multitude de problèmes d'ordre intellectuel, économique ou commercial peuvent être regroupés, dans leur formalisation, dans l'une des tâches suivantes [5] [3] :

- Classification
- Estimation
- Prédiction
- Regroupement par similitudes
- Segmentation
- Description
- Optimisation

1.5.1. Classification

La classification se fait couramment depuis déjà bien longtemps pour comprendre et communiquer notre vision du monde (par exemple les espèces animales, minérales ou végétales). « La classification consiste à examiner des caractéristiques d'un élément nouvellement présenté afin de l'affecter à une classe d'un ensemble prédéfini. » [5] [3].

Dans le cadre informatique, les éléments sont représentés par des enregistrements et le résultat de la classification viendra alimenter un champ supplémentaire. La classification permet de créer des classes d'individus (terme à prendre dans son acception statistique). Celles-ci sont discrètes : homme/femme, oui/non, etc.

Les techniques les plus appropriées à la classification sont :

- Les arbres de décision.

- Le raisonnement basé sur la mémoire.
- Éventuellement l'analyse des liens.

1.5.2. Estimation

Contrairement à la classification, le résultat d'une estimation permet d'obtenir une variable continue à travers une ou plusieurs fonctions combinant les données en entrée. Le résultat d'une estimation permet de procéder aux classifications grâce à un barème. Par exemple, on peut estimer le revenu d'un ménage selon divers critères (nombre et type de véhicule, profession ou catégorie socioprofessionnelle, type d'habitation, etc.). Il sera ensuite possible de définir des tranches de revenus pour classifier les individus. La technique la plus appropriée à l'estimation est : les réseaux de neurones [5] [3].

1.5.3. Prédiction

La prédiction ressemble à la classification et à l'estimation mais dans une échelle temporelle différente. Tout comme les tâches précédentes, elle s'appuie sur le passé et le présent mais son résultat se situe dans un futur généralement précisé. La seule méthode pour mesurer la qualité de la prédiction est d'attendre !

Les techniques les plus appropriées à la prédiction sont :

- L'analyse du panier de la ménagère
- Le raisonnement basé sur la mémoire.
- Les arbres de décision.
- Les réseaux de neurones.

1.5.4. Regroupement par similitudes

Le regroupement par similitudes consiste à grouper les éléments qui vont naturellement ensemble. La technique la plus appropriée au regroupement par similitudes est : l'analyse du panier de la ménagère

1.5.5. Segmentation

L'analyse des clusters consiste à segmenter une population hétérogène en sous-populations homogènes. Contrairement à la classification, les sous-populations ne sont pas préétablies. La technique la plus appropriée au clustering est : l'analyse des clusters.

1.5.6. Description

C'est souvent l'une des premières tâches demandées à un outil de data mining. On lui demande de décrire les données d'une base complexe. Cela engendre souvent une exploitation supplémentaire en vue de fournir des explications. La technique la plus appropriée à la description est : l'analyse du panier de la ménagère.

1.5.7. Optimisation

Pour résoudre de nombreux problèmes, il est courant pour chaque solution potentielle d'y associer une fonction d'évaluation. Le but de l'optimisation est de maximiser ou minimiser cette fonction. Quelques spécialistes considèrent que ce type de problème ne relève pas du data mining. La technique la plus appropriée à l'optimisation est : les réseaux de neurones.

1.6. Méthodes du data mining

Pour arriver à exploiter ces quantités importantes de données, le data mining utilise des méthodes d'apprentissage automatique. Il existe deux types de méthodes [5] :

1.6.1. Méthodes descriptives

Le principe de ces méthodes est de pouvoir mettre en évidence les informations présentes dans l'entrepôt de données (Data Warehouse) mais qui sont masquées par la masse de donnée. Parmi les techniques et algorithmes utilisés, on cite :

- Analyse factorielle (ACP et ACM).
- Méthode des centres mobiles.
- Classification hiérarchique.
- Classification neuronale (réseau de Kohonen).
- Recherche d'association.

1.6.2. Méthodes prédictives

Contrairement à l'analyse descriptive, cette technique fait appel à l'IA. L'analyse prédictive, est comme son nom l'indique une technique qui va essayer de prévoir une évolution des événements en se basant sur l'exploitation de données stockés dans le data warehouse. Parmi les techniques utilisées dans l'analyse prédictive, on cite :

- Arbre de décision ;
- Réseaux de neurones ;
- Régression linéaire ;
- Analyse discriminante de Fisher ;
- Analyse probabiliste.

1.6.3. 1er exemple : les arbres de décision

Les arbres de décision sont un type spécifique de modèle prédictif qui permet aux entreprises d'exploiter efficacement leurs données. Techniquement, un arbre de décision fait partie du *machine learning*, mais il est plus connu sous le nom de *test en boîte blanche* en raison de sa nature extrêmement simple.

Un arbre de décision permet aux utilisateurs de comprendre clairement comment les entrées de données affectent les sorties. Lorsque différents modèles d'arbres de décision sont combinés, ils créent des modèles prédictifs connus sous le nom de *random forest*. Les modèles de *random forest* complexes sont considérés comme des techniques de *machine learning* en boîte noire, car il n'est pas toujours facile de comprendre les sorties en fonction des entrées. Dans la plupart des cas, cependant, cette forme de base de modélisation d'ensemble est plus précise que l'utilisation exclusive d'arbres de décision [13].

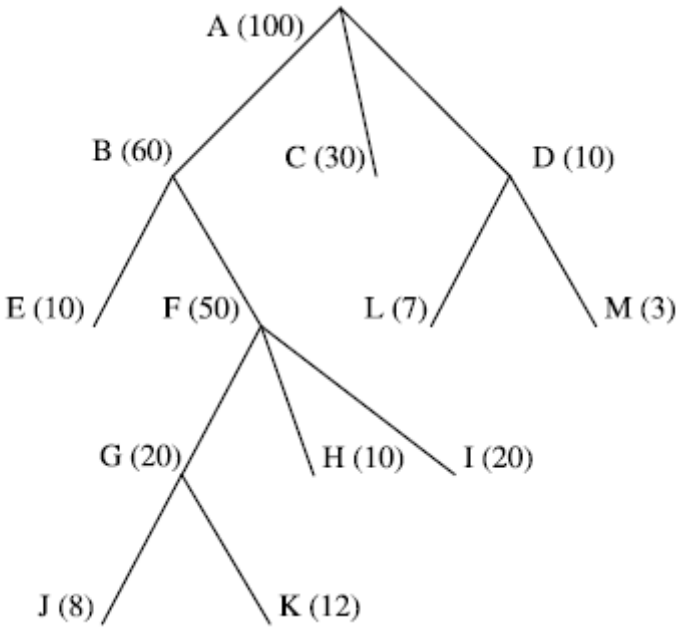

Figure 2 – *Exemple d'un Arbre de décision [12].*

1.6.4. 2ème exemple : les réseaux de neurones

Un réseau de neurones est un type spécifique de modèle de machine learning, souvent utilisé avec l'intelligence artificielle et le deep learning. Nommés ainsi car ils présentent différentes couches qui ressemblent à la façon dont les neurones fonctionnent dans le cerveau humain, les réseaux de neurones sont l'un des modèles de machine learning les plus précis utilisés aujourd'hui.

S'ils sont un outil puissant pour l'exploration de données, les entreprises doivent faire preuve de prudence lorsqu'elles les utilisent : certains de ces modèles de réseaux de neurones sont incroyablement complexes. Il est souvent difficile de comprendre comment un réseau neuronal a déterminé un résultat donné [14].

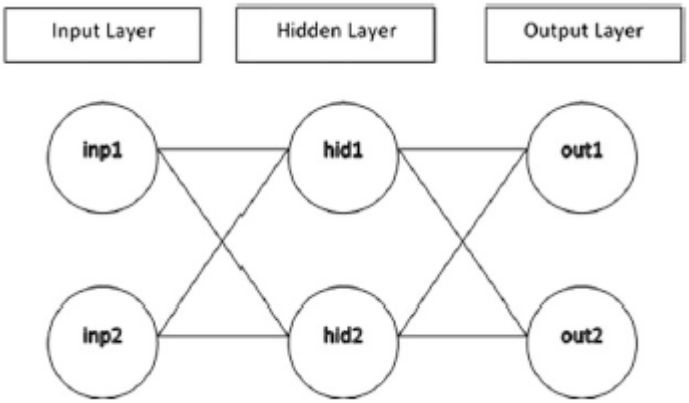

Figure 3 - *Exemple de réseaux de neurones [15]*

Le choix de l'architecture de base du réseau est une question d'expérience plutôt qu'une science exacte. Après l'avoir fait et avoir généré les valeurs initiales des poids par un processus aléatoire, les instances d'un ensemble d'apprentissage sont maintenant présentées au réseau une par une. Chaque instance comprend une valeur à donner à chaque nœud d'entrée et la valeur cible à obtenir pour chaque nœud de sortie pour ces entrées. Au fur et à mesure du traitement de chaque instance, les valeurs des nœuds dans une couche et les poids des liens vers la couche suivante sont utilisées pour calculer les valeurs à attribuer aux nœuds de la couche suivante.

Ce processus se poursuit couche par couche jusqu'à ce que des valeurs aient été attribuées aux nœuds de la couche de sortie. Ce processus s'appelle la propagation vers l'avant et ce type de réseau est appelé réseau neuronal à action directe. Il est presque certain que les valeurs de sortie calculées ne correspondent pas aux valeurs cibles.

Auquel cas les erreurs dans les nœuds de sortie (les écarts entre les valeurs cibles et les valeurs calculées) sont utilisées pour remonter le réseau couche par couche et calculer les ajustements à apporter à chacun des poids. Ce processus s'appelle la rétro-propagation. La rétro-propagation est une méthode d'apprentissage supervisé largement utilisée avec les réseaux neuronaux à action directe pour former les poids. Elle peut être utilisée pour la classification ou la prédiction numérique.

La rétro-propagation appliquée aux grands réseaux neuronaux multicouches à action directe est souvent décrite comme un apprentissage profond. Le traitement d'une instance à travers le réseau (propagation vers l'avant suivie d'une rétro-propagation) est appelé une passe. Lorsqu'un passage est terminé, l'instance suivante est présentée au réseau et la passe suivante commence. Le traitement de toutes les instances d'un ensemble d'apprentissage une par une est appelé une époque.

Le processus se poursuit, les poids changeant lentement jusqu'à ce qu'un niveau de précision acceptable soit atteint. Cela peut prendre plusieurs époques pour y parvenir, souvent des dizaines ou des centaines de milliers, voire plus. Dans sa forme finale, le réseau est considéré comme entraîné, c'est-à-dire que la propagation des valeurs d'entrée dans le réseau donnera une approximation proche de la réalité.

1.7. Types de données du data mining

1.7.1. Données stockées dans une base de données

Une base de données ou plus précisément un système de gestion de base de données (SGBD) stocke des données qui sont liées les unes aux autres d'une manière ou d'une autre. Il dispose également d'un ensemble de logiciels qui permettent de gérer les données et d'y accéder facilement. Ces logiciels servent à de nombreuses fins, notamment la définition de la structure de la base de données, la garantie que les informations stockées restent sécurisées et cohérentes et la gestion de différents types d'accès aux données, tels que partagés, distribués et simultanés.

Une base de données relationnelle a des tables qui ont des noms et des attributs différents et peuvent stocker des lignes ou des enregistrements d'ensembles de données volumineux. Chaque enregistrement stocké dans une table a une clé unique. Le modèle entité-relation est créé pour fournir une représentation d'une base de données relationnelle qui présente des entités et les relations qui existent entre elles [16].

1.7.2. Données stockées dans un entrepôt de données

Un entrepôt de données est un emplacement de stockage de données unique qui collecte des données provenant de plusieurs sources, puis les stocke sous la forme d'un plan unifié. Lorsque les données sont stockées dans un entrepôt de données, elles sont nettoyées, intégrées, chargées et actualisées. Les données stockées dans un entrepôt de données sont organisées en plusieurs parties [16].

1.7.3. Données transactionnelles

La base de données transactionnelle stocke les enregistrements capturés en tant que transactions. Ces transactions incluent la réservation de vol, l'achat du client, le clic sur un site Web, etc. Chaque enregistrement de transaction a un identifiant unique. Il répertorie également tous les éléments qui en ont fait une transaction [16].

1.7.4. Autres types de données

Il existe également beaucoup d'autres types de données qui sont connus pour leur structure, leurs significations sémantiques et leur polyvalence. Ils sont utilisés dans de nombreuses applications. Parmi ces types de données : les flux de données, les données de conception technique, les données de séquence, les données de graphique, les données spatiales, les données multimédias, etc. [16].

1.8. Conclusion

Dans ce chapitre, nous avons introduit le domaine du data mining, en donnant quelques-unes de ses définitions, un petit tour historique, et ses principaux domaines d'application. Ensuite, nous avons consacré une section pour définir ses tâches, ainsi qu'une autre pour survoler ses principales méthodes. Dans le chapitre suivant, nous allons se focaliser sur l'analyse d'un type particulier des données du data mining : les séries temporelles.

Chapitre II : Matrix Profile

2.1. Introduction

L'objectif de ce chapitre consiste à présenter les séries temporelles, et d'introduire l'une des techniques les plus récentes pour les l'analyser : le Matrix Profile, ainsi que les différents algorithmes permettant de la mettre en œuvre.

2.2. Les séries temporelles

Une série temporelle *T* est une suite (X_1, ..., Xn) de données indexées par le temps. L'indice temps peut être selon les cas la minute, l'heure, le jour, l'année etc. Le nombre « *n* » est appelé la longueur de la série. Il est la plupart du temps bien utile de représenter la série sur un graphe construit de la manière suivante : en abscisse le temps, en ordonnée la valeur de l'observation à chaque instant. Pour des questions de lisibilité, les points ainsi obtenus sont reliés par des segments de droite. Le graphe apparaît donc comme une ligne brisée [5][10].

Une série temporelle est définie aussi comme une collection d'observations faites séquentiellement dans le temps. Plus que la plupart des types de données, les séries temporelles se prêtent à l'inspection visuelle et aux intuitions.

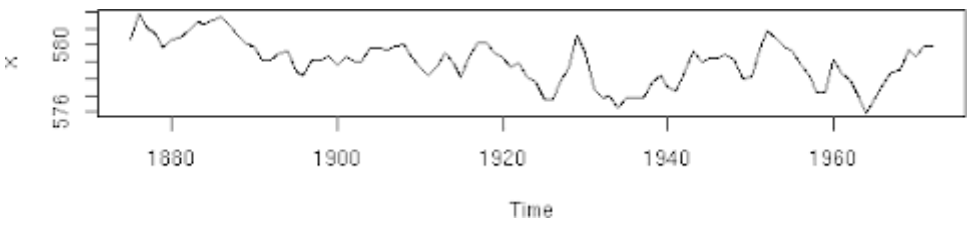

Figure 4 – *Un exemple de série temporelle.*

2.2.1. Exemple de séries temporelles

L'observation des chiffres du vecteur bleu ci-dessous ne dit rien. Mais après avoir tracé ses données, on peut reconnaître un battement de cœur, et peut-être même diagnostiquer la maladie de cette personne. Quand les observations sont uniformément échantillonnées, l'indice d'observation peut remplacer le temps d'observation. Les observations peuvent avoir une unité [17].

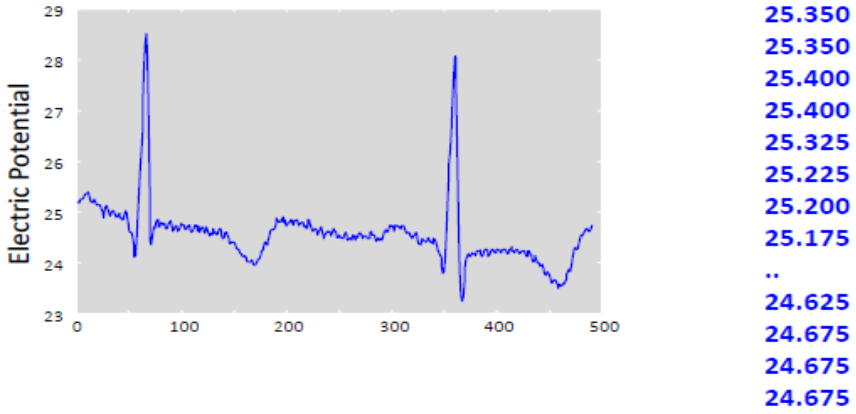

Figure 5 - *Une série temporelle issue d'un Electrocardiogramme*

2.3. Mesures de similarité pour les séries temporelles

Une mesure de similarité compare deux séries et produit un nombre représentant leur similitude. Une mesure de distance est l'opposé d'une mesure de similarité [10].

2.3.1. Mesures de verrouillage

- Distance euclidienne.
- Coefficient de corrélation.
- Similitude cosinus.

2.3.2. Mesures élastiques

- Déformation temporelle dynamique.
- Modifier la distance.
- Sous-séquence commune la plus longue.

2.4. Le Matrix Profile

Le « *Matrix Profile* » est une technique parue récemment et devenue l'état de l'art actuel dans le domaine d'analyse des séries temporelles. Cette technique se dit efficace pour la plupart des tâches d'analyse des séries temporelles, même avec les grandes bases de données. Le « *Matrix Profile* » est une structure de données et des algorithmes associés qui aident à résoudre différents problèmes comme celui de la détection des anomalies ou de la découverte de motifs. Il est robuste, évolutif et sans paramètre.

Commençons par donner quelques concepts et définitions de base.

2.4.1. Sous-séquence d'une série temporelle

Une sous-séquence T_{i+n} est un sous-ensemble continu des valeurs de T de longueur m à partir de la position i. Formellement, $T_{i+n} = t_i, t_{i+1}, \ldots, t_{i+n-1}$, ou $1 \leq i \leq n+1$ Nous pouvons prendre n'importe quelle sous-séquence d'une série temporelle et calculer sa distance par rapport à toutes les autres sous-séquences. Nous appelons un vecteur ordonné de telles distances un *profile de distance* (*distance profile*) [9].

2.4.2. Le profil de distance ou Distance Profile

Nous commençons par définir ce qui est un *profil de distance* (*Distance Profile*). Un profil de distance D est un vecteur des distances euclidiennes entre une requête donnée (une séquence d'une série temporelle) et chaque sous-séquence de l'ensemble de toutes les sous- séquences de la même ou d'une autre série temporelle.

Notons que nous supposons que la distance est mesurée à l'aide de la distance euclidienne entre les sous-séquences *z-normalisées* (normalisées en utilisant la *transformé en Z*, c-à-dire avec une moyenne de zéro, et un écart-type de 1). Le profil de distance peut être considéré comme une méta-série temporelle qui annote la série temporelle T qui a été utilisée pour le générer [9].

La figure suivante illustre les définitions précédentes :

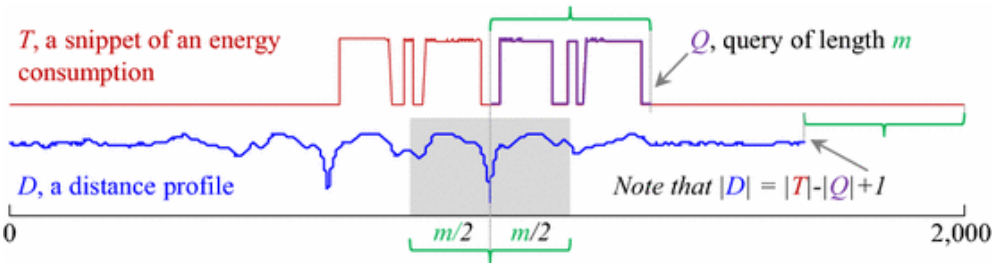

Figure 6 - *Une sous-séquence Q extraite de T est utilisée comme requête à toutes les sous- séquences. Le vecteur de toutes les distances est le profile de distance.*

2.4.3. Ensemble de toutes les sous-séquences d'une série temporelle

L'ensemble A de toutes les sous-séquences d'une série temporelle T est un ensemble ordonné de toutes les sous-séquences possibles de T obtenues en faisant glisser une fenêtre de longueur m sur T :

$A = \{T_{1,m}, T_{2,m}, \ldots, T_{n-m+1,m}\}$, où m est la longueur de sous-séquence définie par l'utilisateur. Nous utilisons $A[i]$ pour désigner $T_{i,m}$.

Nous nous intéressons à la relation du plus proche voisinage (c'est-à-dire *1NN*) entre les sous-séquences ; cependant nous définissons une fonction de jonction *1NN* qui indique la relation du plus proche voisinage entre les deux sous-séquences d'entrée [10].

2.4.4. Fonction de jointure 1NN

Etant donné deux ensembles de toutes les sous-séquences *A* et *B* et deux sous-séquences *A[i]* et *B[j]*, une fonction de jointure *1NN :* θ_{1nn} *(A[i], B[j])* est une fonction booléenne qui ne renvoie « vrai » que si *B[j]* est le plus proche voisin de *A[i]* dans l'ensemble *B*. Avec la fonction de jointure ainsi définie, un ensemble de jointure de similarité peut être généré en appliquant l'opérateur de jointure de similarité sur deux ensembles de toutes les sous- séquences d'entrée [9].

2.4.5. Ensemble de jointure de similarité

Etant donné les ensembles de toutes les sous-séquences *A* et *B*, un ensemble de jointure de similarité J_{AB} de *A* et *B* est un ensemble contenant des paires de chaque sous-séquence de *A* avec sa plus proche voisine en *B* : $J_{AB} = \{(A[i], B[j]) \mid \theta_{1nn}(A[i], B[j])\}$. Nous le désignons formellement comme $J_{AB} = A \bowtie_{\theta\ 1nn} B$

Nous mesurons la distance euclidienne entre chaque paire dans un ensemble de jointures de similarité et nous stockons les résultats dans un vecteur ordonné. Nous appelons profil de matrice (*Matrix profile*) le vecteur résultant [9].

2.4.6. Le Matrix Profile

Le *Matrix profile* ou *profil matriciel* (ou simplement un profil) P_{AB} est un vecteur des distances euclidiennes entre chaque paire dans J_{AB} où $P_{AB}[i]$ contient la distance entre *A[i]* et son plus proche voisin dans *B*. Nous appelons ce vecteur le profil matriciel car une manière (inefficace) de le calculer serait de calculer la matrice de distance complète de toutes les sous-séquences d'une série temporelle avec toutes les sous-séquences d'une autre série temporelle et d'extraire la plus petite valeur de chaque ligne (la plus petite valeur non diagonale pour le cas d'auto-jointure) [9][10].

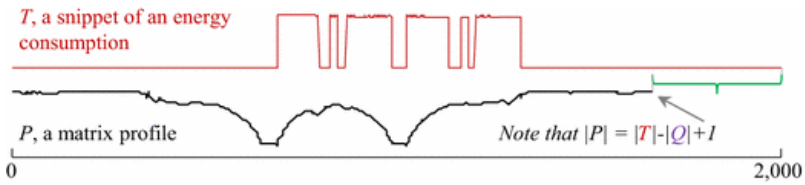

Figure 7 – *Exemple de « matrix profile ».*

Comme le profil de distance, le matrix profile peut-être considéré comme une méta-série annotant la série temporelle *T* s'il est généré en joignant *T* avec elle-même. Le profile a un bon nombre de propriétés intéressantes et exploitables. Par exemple, le plus haut point dans le profile correspond au discord de la série, et le point le plus bas correspond au meilleur motif, et sa variance correspond à la complexité de *T*.

2.4.7. Ensemble d'autosimilarité

Un ensemble d'auto-similarité J_{AA} est le résultat de jointure de similarité de l'ensemble *A* avec lui-même. Il est indiqué en tant que $J_{AA} = \boldsymbol{A} \bowtie_{\theta\, 1nn} \boldsymbol{A}$. Nous désignons le profil matriciel correspondant ou le profil de jointure d'auto-similarité comme P_{AA}.

Notons que les correspondances triviales sont exclues lorsque la jointure d'autosimilarité est effectuée, c'est-à-dire que si *A[i]* et *A[j]* sont des sous-séquences du même ensemble de sous séquences *A*, $\theta_{1nn}(A[i], B[j])$ est « faux » lorsque *A[i]* et *A[j]* sont une paire triviale.

Le $i^{ème}$ élément du profil matriciel indique la distance euclidienne entre la sous-séquence de *T* à partir de *i* et son plus proche voisin, mais, il ne dit pas où se trouve ce voisin. Ces informations sont enregistrées dans l'index des profils matriciels [9].

2.4.8. L'index du matrix profile

Un index de profil matriciel I_{AB} d'un ensemble de jointure de similarité J_{AB} est un vecteur d'entiers où $I_{AB}\ [i] = j$ si $\{A[i], B[j]\} \in J_{AB}$. En stockant les informations des voisines de cette manière, nous pouvons retrouver efficacement le plus proche voisin de *A[i]* en accédant au $i^{ème}$ élément de l'index du matrix profile. Notons que la fonction qui calcule l'ensemble des jointures de similarité de deux séries temporelles d'entrée n'est pas symétrique ; par conséquent, $J_{AB} \neq J_{BA}$, $P_{AB} \neq P_{BA}$, et $I_{AB} \neq I_{BA}$.

Notons enfin que nous avons limité notre travail au cas unidimensionnel ; cependant, rien n'exclut les généralisations aux données multidimensionnelles.

2.5. Algorithmes pour calculer le matrix profile

La méthode la plus triviale pour calculer le matrix profile est de procéder au calcul des distances une à une, cependant cette façon simpliste s'avère très couteuse en termes de ressources. Il existe alors plusieurs algorithmes performants pour procéder.

2.5.1. L'algorithme MASS

MASS (*Mueen's Algorithm for Similarity Search*) ne permet pas seulement de trouver le plus proche voisin d'une requête et renvoie sa distance, mais il retourne la distance à chaque sous-séquence. En particulier, il calcul le profile de distance. Il nécessite un temps de l'ordre de *O(n logn)* en exploitant la *transformée de fourrier* (*FFT*) pour calculer le produit de convolution entre la requête et toutes les sous-séquences d'une série temporelle [9][10].

Algorithme : Calcul du produit scalaire glissant
Entrée : une requête *Q* et une série temporelle *T* fournie par l'utilisateur **Sortie :** le produit scalaire entre *Q* et toutes les sous-séquences de *T*
1. $n \leftarrow Longueur\ (T),\ m \leftarrow Longueur(Q)$
2. $T_a \leftarrow$ *Concaténer T avec nzéros*
3. $Q_r \leftarrow Inverser(Q)$
4. $Q_{ra} \leftarrow$ *Concaténer Qr avec 2n – m zéros*
5. $Q_{raf} \leftarrow FFT(Q_{ra}),\ T_{af} \leftarrow FFT(T_a)$
6. $QT \leftarrow$ *Inverse FFT (Multiplication-élémentaire* (Q_{raf}, T_{af})*)*
7. **Retourner** *QT*

Tableau 1 – *Le calcul du produit scalaire glissant.*

La première ligne détermine la longueur de la série *T* et de la requête *Q*. En utilisant ses deux informations, nous concaténons *T* avec le même nombre de zéros dans la ligne 2. Dans la ligne 3, on obtient l'image miroir de la requête *Q*, puis, dans la ligne 4, nous la concaténons avec des zéros pour obtenir autant d'éléments que *T* (c'est-à-dire *2n* éléments). Dans la ligne 5, l'algorithme calcule la *transformée de fourrier* des deux séries résultantes. Enfin, un produit de convolution classique est effectué sur les deux vecteurs qui résultent.

Exemple :

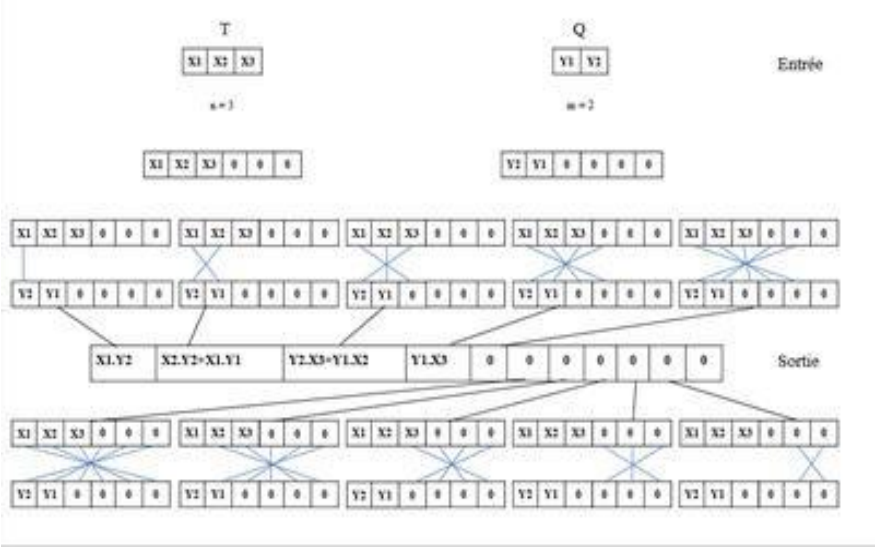

Figure 8 - *Exemple pour calculer le produit de convolution.*

Algorithme : MASS
Entrée : une requête Q et une série temporelle T fournie par un utilisateur **Sortie :** un profil de distance D de la requête Q ***1.*** $QT \leftarrow$ *Produit-Scalaire-Glissant (Q, T)* ***2.*** $\mu_Q, \sigma_Q, M_T, \Sigma_Q \leftarrow$ *CalculerMoyenneEcart-type (Q,T)* ***3.*** $D \leftarrow$ *CalculerDistanceProfile (QT,* $\mu_Q, \sigma_Q, M_T, \Sigma_Q$*)* **Retourner** D

Tableau 2 – *L'algorithme MASS.*

Dans l'algorithme principal, nous invoquons le code précédent, Nous calculons la distance euclidienne entre Q et chaque $T[i]$ avec la formule suivante:

$$D[i] = \sqrt{2m\left(1 - \frac{QT[i] - m\mu_Q M_T[i]}{m\sigma_Q \Sigma_T[i]}\right)}$$

Où m est la longueur de la sous-séquence, μ_Q est la moyenne de Q, $M_T[i]$ est la moyenne de $(T[i], m)$, σ_Q est l'écart type de Q, et $\Sigma_T[i]$ est l'écart type de $(T[i],m)$.

2.5.2. L'algorithme STAMP

Nous passons à l'algorithme de jointure *STAMP*, *Scalable Time seriesAnytime Matrix Profile*. Il utilise *MASS* comme algorithme de base pour calculer la similitude entre la requête et la série de référence (le profil de distance *DP*). Le *MP* (matrix profile) ultime vient de la fusion du minimum élément par élément de tous les *DP* possibles. Cet algorithme a la complexité temporelle de $O(n^2 logn)$ et la complexité spatiale de $O(n)$ [9] [10].

Algorithme : STAMP

Entrée : une série temporelle *T* fournie par l'utilisateur et la longueur désirée *m*
Sortie : un profil matriciel *P* et un indice de profil matriciel *I*

1. *n ← Longueur(T)*
2. *P ← infs, I ← zéros, idxes ← 0 : n –m*
3. ***Pour chaque*** *idx* ***dans*** *idxes* ***faire***
    4. *D ← MASS (T[idx], T)*
    5. *P, I ← ElementWiseMin (P, I, D, idx)*
6. ***FinPour***
7. ***Retourner*** *P,I*

Tableau 3 – *L'algorithme STAMP.*

L'algorithme est décrit comme suit : à la ligne 1, nous extrayons la longueur de *T*. À la ligne 2, nous allouons la mémoire au profil matriciel initial *P* et l'indice de profil matriciel *I*. De la ligne 3 à la ligne 6, nous calculons les profils de distance *D* en utilisant chaque sous-séquence $T[idx]$ dans la série temporelle *T*.

Ensuite, nous effectuons un minimum par paire pour chaque élément de *D* avec l'élément apparié dans *P* (c'est-à-dire min ($D[i]$; $P[i]$) pour $i = 0$ à length$(D) - 1$). Nous mettons également à jour $I[i]$ avec *idx* lorsque $D[i] \leq P[i]$ lorsque nous effectuons l'opération de minimum par paire. Les correspondances triviales sont ignorées dans *D* lors de l'exécution de *ElementWiseMin* à la ligne 5.

Enfin, le résultat est retourné à la ligne 7.

2.5.2.1. La propriété *Anytime* de STAMP

Bien que l'algorithme exact présenté dans la section précédente soit extrêmement évolutif, il y aura toujours des ensembles de données pour lesquels le temps nécessaire à une solution exacte est intenable. Nous pouvons atténuer ce problème en calculant les résultats « *à tout moment* », ce qui permet des solutions approximatives rapides. Pour ajouter la nature « *anytime* » à l'algorithme *STAMP*, il suffit d'assurer un ordre aléatoire lorsque nous sélectionnons les sous-séquences de *T* à la ligne 2 de l'algorithme *STAMP*.

Nous pouvons calculer une mesure (*post-hoc*) de la qualité d'une solution *anytime* en mesurant l'erreur moyenne quadratique (*RMSE*) entre le vrai profil de matrice et le meilleur profil de matrice actuel. Comme le suggère la figure ci-dessous, avec une expérience sur des données de marche aléatoire, l'algorithme converge très rapidement.

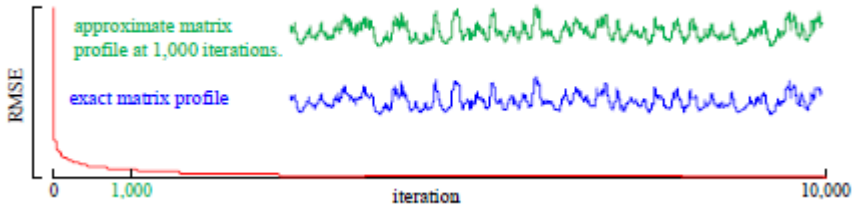

Figure 9 - *La diminution de RMSE lorsque l'algorithme STAMP met à jour le profil matriciel avec le profil de distance calculé à chaque itération.*

Puisque le profil de distance de chaque sous-séquence est limité en dessous par le profil de matrice exact, la mise à jour d'un profil de matrice approximatif avec un profil de distance avec l'opération *pairwise minimum* permet soit de rapprocher la solution approximative de la solution exacte, soit de conserver la solution approximative actuelle. Ainsi, nous avons garanti la monotonicité.

2.5.2.2. Peut-on améliorer STAMP ?

STAMP calcule des profils de distance pour des lignes dans un ordre aléatoire. Chaque profil de distance est calculé indépendamment. Cependant, les lignes successives sont des profils de deux requêtes qui se chevauchent dans m-1 observations. Nous pouvons exploiter le chevauchement entre requêtes successives lors du calcul de leurs profils de distance pour construire un algorithme ($n2$) temps, $O(n)$espace [20].

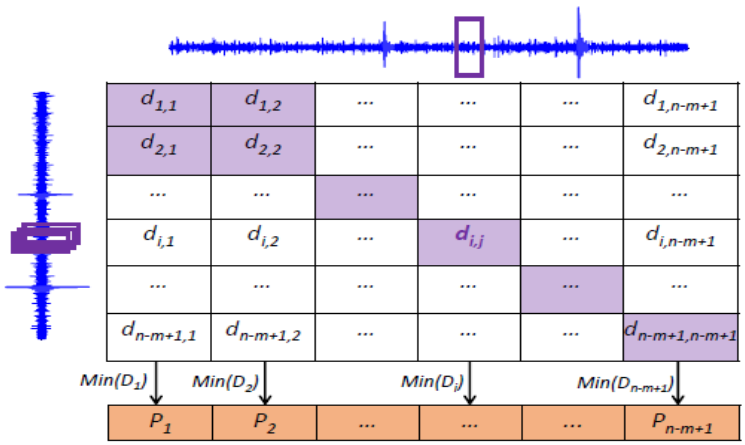

Figure 10 – *Amélioration de STAMP*

2.5.3. L'algorithme STOMP

STOMP (*Scalable Time series Ordered Matrix Profile*) était le deuxième algorithme utilisé pour calculer le *MP*. Il utilise également *MASS* pour calculer le *DP*, mais uniquement pour la première itération de chaque lot. Les chercheurs ont remarqué qu'ils pouvaient réutiliser le calcul des valeurs du premier *DP* pour effectuer un calcul plus rapide dans les itérations suivantes. Il en résulte une complexité temporelle de $O(n^2)$, en gardant la même complexité spatiale de $O(n)$ [9][10].

Le principal inconvénient de *STOMP* par rapport à *STAMP* est l'absence de la propriété « *anytime* ». Dans les scénarios où une convergence rapide est nécessaire, il peut ne suffire que de 5% du calcul *MP* pour fournir une approximation très précise du résultat final. Aussi impressionnante que soit l'efficacité temporelle de *STAMP*, nous pouvons créer un algorithme encore plus rapide si nous sommes prêts à sacrifier l'une des caractéristiques de *STAMP* : son caractère instantané.

C'est un compromis que de nombreux utilisateurs peuvent être prêts à faire. Comme cette variante de *STAMP* effectue une recherche ordonnée (et non aléatoire), nous l'appelons *STOMP*, *Scalable Time series Ordered Matrix Profile*. L'algorithme *STOMP* est très similaire à *STAMP* et, au moins en principe, il s'agit toujours d'un algorithme *anytime*. Cependant, comme *STOMP* doit calculer les courbes de distance dans un ordre de gauche à droite, il est vulnérable à un ensemble de séries temporelles qui ont des motifs uniquement du côté droit, et des données aléatoires du côté gauche.

Algorithme : STOMP

Entrée : une série temporelle *T* fournie par l'utilisateur et la longueur désirée *m*

Sortie : un profil matriciel *P* et un indice de profil matriciel *I*

1. $n \leftarrow Longueur\ (T)$
2. $M_T, \Sigma_T \leftarrow ComputeMeanStd(T)$
3. $D, QT \leftarrow MASS\ (T_{1,m}, T)$
4. $QT_{première} \leftarrow QT$
5. P ← *D, I* ← *uns*
6. ***Pour*** *i* ← *1* ***jusqu'à*** *n – m* ***faire***

   7. ***Pour*** *j* ← *n – m* ***jusqu'à*** *j* ← *1* ***faire***

      8. $QT\ [j] \leftarrow QT\ [j - 1] - T\ [i - 1] \times T\ [j - 1] + T\ [i + m - 1] \times T\ [j + m - 1]$

   9. ***FinPour***
   10. $QT\ [0] \leftarrow QT_{première}[i]$
   11. $D \leftarrow CalculateDistanceProfile\ (QT, i, M_T, \quad T)$
   12. *P, I* ← *ElementWiseMin (P, I, D,i)*

13. **Finpour**
14. Retourner *P,I*

Tableau 4 – *L'algorithme STOMP.*

2.6. Conclusion

Dans ce chapitre nous nous sommes focalisés sur la technique dite Matrix Profile pour l'analyse des séries temporelles. Nous avons explicité ses bases théoriques, et nous avons introduit quelques algorithmes permettant de la mettre en œuvre. Cette étude va nous permettre de bien aborder la suite du travail pour approfondir notre étude sur la technique.

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

3.1. Introduction

Dans ce chapitre, nous allons essayer de mettre en œuvre la *technique matrix* profile pour analyser des séries temporelles. Nous allons étudier le cas de la classification à travers la découverte des « *shapletes* ». Au final, l'implémentation d'une petite application qui permet d'analyser les séries temporelles et les classer.

3.2. Application à la tâche de classification

3.2.1. Classification des séries temporelles

Il existe plusieurs algorithmes dédiés à la classification des séries temporelles ce qui explique qu'il n'est pas toujours indispensable de résoudre chaque tâche de classification de séries temporelles à travers l'apprentissage en profondeur. Le contenu de notre projet s'intéresse beaucoup plus à la classification basée sur les *shapelets* [21].

3.2.2. C'est quoi une *Shapelet* ?

Dans la classification des séries temporelles, les *shapelets* sont des sous-séquences avec un pouvoir discriminant élevé. La recherche combinatoire pour la découverte de *shapelets* est effectuée par les méthodes existantes. Même avec des heuristiques d'accélération telles que l'élagage, le clustering et la réduction de dimensionnalité, la recherche reste coûteuse en calcul [22].

3.2.3. Classification basée sur les *Shapelets*

Le Matrix Profile sert à identifier rapidement les bons *shapletes*. Le mentionnement de cette idée fut au passage dans [19] et comme l'espace manquait, le développement ainsi que l'évaluation n'ont pas été entièrement réalisés [23].

Example :

Nous introduisons notre approche avec le jeu de données *GunPoint*. Cet ensemble de données à deux classes, *Gun* et *NoGun* (*NoGun* est également connu sous le nom de *Point*, d'où le nom *GunPoint*). Comme le montre la figure, nous construisons des séries temporelles T_A en concaténant toutes les instances de la classe *Gun*, et nous construisons des séries temporelles T_B en concaténant toutes les instances de la classe *NoGun*.

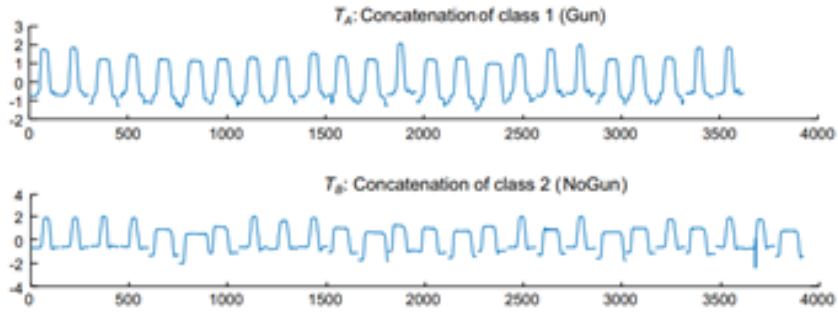

Figure 11 – *Les deux séries temporelles A et B formées par la concaténation des instances de chaque classe de l'ensemble de données GunPoint.*

Nous insérons une valeur *NaN* entre deux instances concaténées pour éviter d'introduire des modèles artificiels qui ne sont pas présents dans la série temporelle d'origine. Nous calculons ensuite deux profils matriciels, P_{BB} et P_{BA}. Pour simplifier, nous utilisons une longueur de sous-séquence de *m*, qui est la longueur de la meilleure *shapletes* rapportée pour cet ensemble de données.

Intuitivement, P_{BB} sera plus petit que P_{BA} car nous nous attendrions à ce que les sous-séquences d'une même classe soient plus similaires que celles des différentes classes. La différence entre P_{BA} et P_{BB} (nous la désignons par $P = P_{BA} - P_{BB}$), comme le montre la figure ci-dessous, est généralement en accord avec cette intuition :

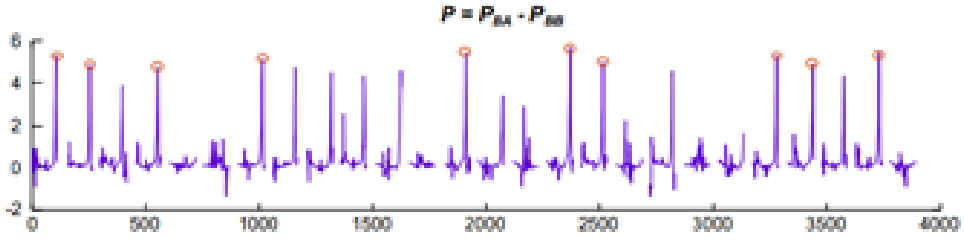

Figure 12 - *La différence entre le P_{BA} et P_{BB}. Les 10 valeurs de pics les plus élevées (surlignées de cercles rouges) suggèrent de bonnes shapelets*

Nous proposons que les valeurs de pics de *P* soient des indicateurs de bons candidats de *shapelets*, car elles suggèrent des modèles qui sont bien conservés dans leur propre classe mais qui sont très différents de leur correspondance la plus proche dans l'autre classe. Nous choisissons les 10 meilleurs candidats de T_B (de manière analogue, nous pouvons trouver les 10 meilleurs candidats de T_A si nous considérons la différence entre P_{AB} et P_{AA}).

Sur la figure à gauche ci-dessous, nous pouvons voir que toutes les 10 premières *shapelets* donnent une précision de classification très élevée à la fois sur les données d'apprentissage que sur les données de test. Parmi elles, nous choisissons celle qui rend la précision de classification la plus élevée sur l'ensemble d'apprentissage (la 6ème *shapelet*) et nous le montrons sur la figure à droite. Cette *shapelet* donne une précision de 93,33% sur les données de test, ce qui est supérieur à la précision de 91,33% de la mesure de distance 1-Nearest-Neighbor avec DTW, avec un avantage supplémentaire, celui de temps de classification nettement inférieur.

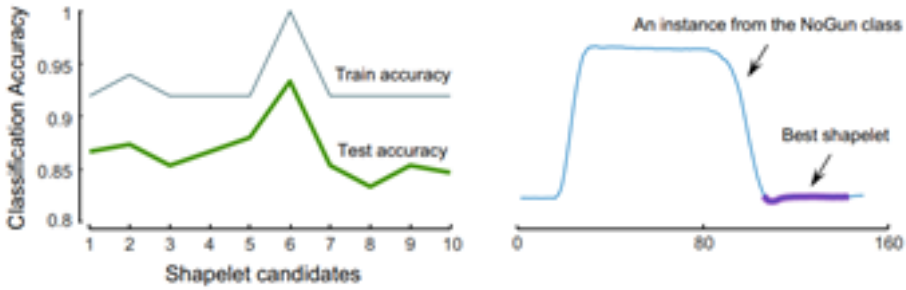

Figure 13 - *Précision de classification des dix meilleurs candidats de forme.*

Rétrospectivement, cette *shapelet* atteint également la même précision de classification sur les données de test que l'algorithme de forme originale [19]. Cependant, contrairement à l'algorithme de forme classique, qui évalue de manière exhaustive la puissance de classification de chaque shaplete possible dans l'ensemble de données, le MP fournit facilement les meilleurs *shapletes* gratuitement.

3.2.4. Découverte de motifs basée sur le profil

Depuis leur introduction en 2003, les motifs de séries temporelles sont devenus l'une des primitives les plus fréquemment utilisées dans l'exploration de données de séries temporelles, avec des applications dans des dizaines de domaines [].

Plusieurs définitions ont été proposées pour les motifs de séries temporelles, mais [9] soutiennent que si l'on peut résoudre la variante la plus élémentaire, la paire de sous-séquences la plus proche (non triviale), toutes les autres variantes ne nécessitent que quelques calculs supplémentaires mineurs. Il est à noter que les emplacements des deux valeurs minimales (liées) du profil de la matrice sont exactement les emplacements de la paire de motifs la plus proche (non triviale).

L'algorithme exact le plus rapide connu pour le calcul des motifs de séries temporelles est l'algorithme MK [9]. Notons, cependant, que la performance temporelle de cet algorithme dépend de la série temporelle elle-même. En revanche, l'algorithme *PBMD* (*Profile-Based Motif Discovery*) est indépendant de la série temporelle. *PBMD* prend du temps indépendamment des données. On a comparé les deux approches sur un électrocardiogramme de longueur 65 536. Dans la Figure ci-dessous en haut, on se demande ce qui se passe lorsqu'on recherche des motifs de plus en plus longs. Dans la figure en bas, on se demande ce qui se passe si la longueur du motif est fixée à *m = 512*, mais que les données deviennent de plus en plus bruyantes [24].

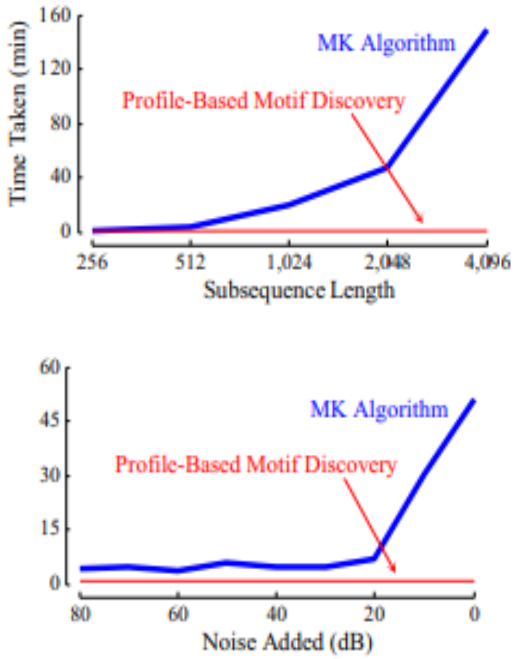

Figure 14 - *Le temps nécessaire pour trouver les paires top-motif dans une série temporelle de longueur 65 536 pour de plus longueurs de motifs (à gauche), et pour une longueur fixée à 512, mais face à des niveaux de bruit croissants (à droite).*

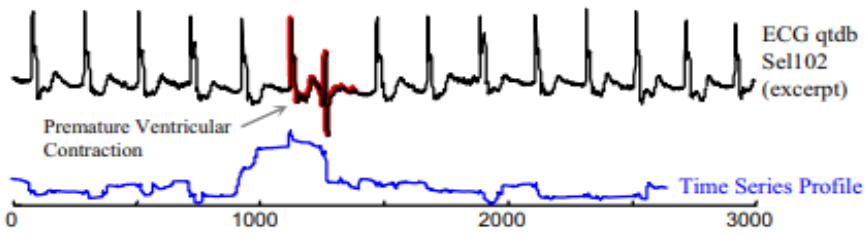

Figure 15 - *(Haut) Un extrait d'un ECG incorporant une contraction ventriculaire prématurée (rouge/gras). (En bas) Le profil de la série temporelle culmine exactement au début du PVC*

Ces résultats montrent que, même dans le meilleur des cas pour *MK*, *PBMD* est compétitif. Mais lorsqu'on a des requêtes plus longues et/ou des données plus bruyantes, son avantage devient irréprochable. De plus, *PBMD* hérite de la parallélisabilité et de la calculabilité incrémentale de STOMP.

3.2.5. Les anomalies ou discordes

Une anomalie ou discorde de série temporelle est la sous-séquence qui présente la distance maximale par rapport à son voisin le plus proche. Bien qu'il s'agit d'une définition simple, les discordes de séries temporelles sont connues pour être très compétitives en tant que détecteurs de nouveauté/anomalie. Très compétitifs en tant que détecteurs de nouveauté/anomalie, par exemple, on a effectué une évaluation empirique approfondie et on a constaté que « sur 19 ensembles de données différents accessibles au public, la comparaison de 9 techniques différentes (les discordes de séries temporelles) est la meilleure technique globale » [19].

Notons que, comme le montre la figure ci-dessous, le discorde des séries temporelles est codé comme la valeur maximale dans un profil matriciel. De plus, les chercheurs ont récemment noté l'utilité des algorithmes anytime pour trouver des valeurs aberrantes dans des archives de données vraiment massives.

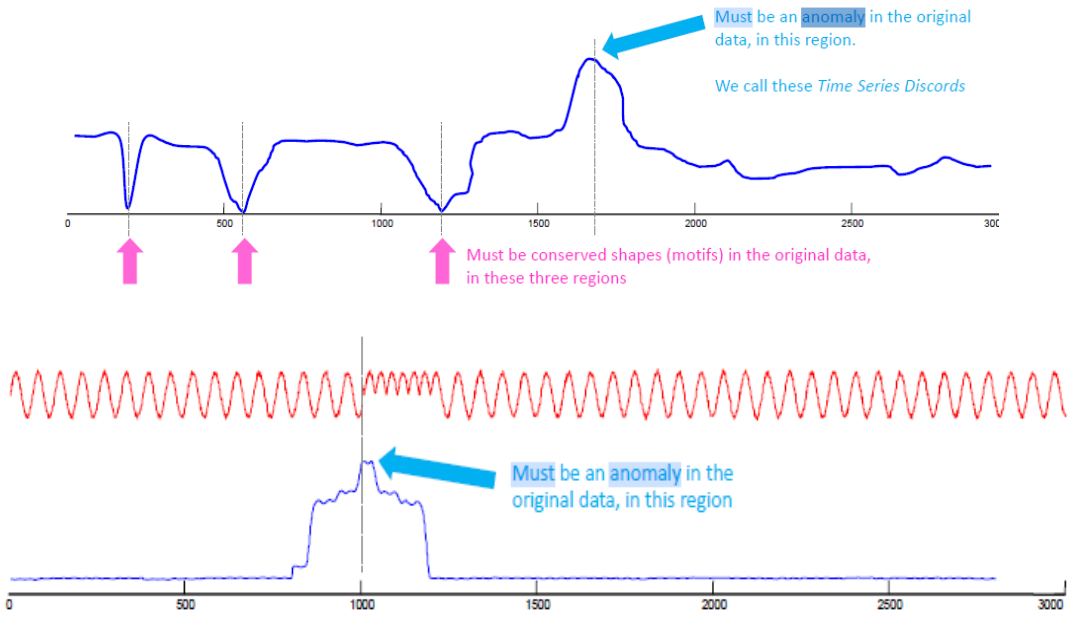

Figure 16 - *Anomalie ou discorde d'une série temporelle [25]*

3.2.6. Maintien incrémentiel des motifs et des discordances

Nous avons montré la capacité de détecter les motifs et les discordes des séries temporelles en utilisant le profil matriciel dans les deux sections précédentes. Cependant, nous avons supposé que l'ensemble de la série temporelle fût disponible au préalable. Ici, nous supprimons cette hypothèse et montrons comment STOMPI nous permet de maintenir de façon incrémentale les motifs et les discordes en ligne. Il existe de nombreuses tentatives pour l'une ou les deux [9] de ces tâches dans la littérature, mais elles sont toutes approximatives et permettent de faux négatifs.

La capacité à maintenir de façon incrémentale implique la capacité de maintenir exactement le motif de la série temporelle et/ou le discorde des séries temporelles dans les données en continu. Il nous suffit de garder la trace des valeurs extrêmes du profil matriciel à croissance incrémentale, de signaler une nouvelle paire de valeurs extrêmes et d'en faire la synthèse. Incrémentalement, de signaler une nouvelle paire de motifs lorsqu'une nouvelle valeur minimale est détectée, et de signaler un nouveau discorde lorsque nous voyons une nouvelle valeur minimale.

Sur le jeu de données suivant, et bien qu'il s'agisse d'un jeu de données réel, il manque d'annotation de vérité de terrain. Nous l'avons donc légèrement modifié afin de pouvoir vérifier la plausibilité des résultats.

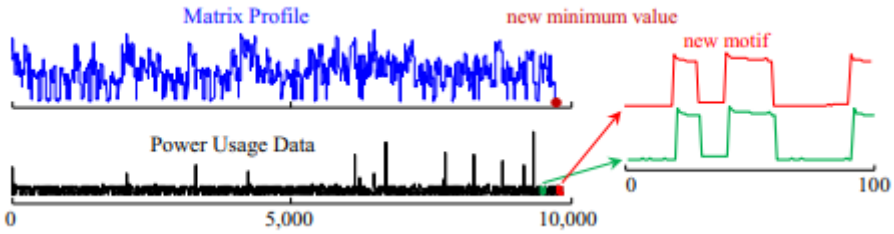

Figure *17* - *Le profil matriciel des 9864 premières minutes de données.*

En haut, le profil matriciel des 9864 premières minutes de données. En bas, la valeur minimale du profil matriciel correspond à une paire de motifs de séries temporelles dans les données de consommation d'énergie. A droite, le motif de la série temporelle détecté.

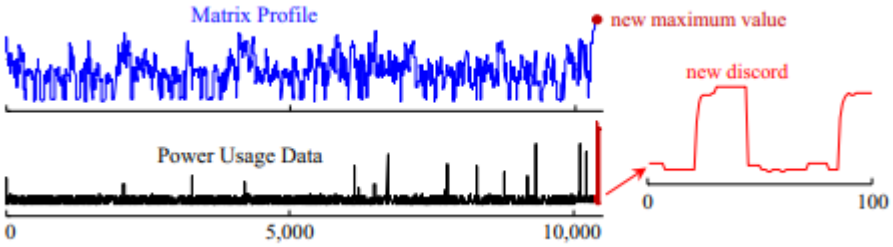

Figure 18 - *Le profil de la matrice pour les 10 473 premières minutes*

En haut, le profil matriciel pour les 10 473 premières minutes. En bas, la valeur maximale du profil matriciel correspond à un discorde de série temporelle. A droite, le discorde de la série temporelle détecté est la première occurrence du motif de la pompe à chaleur dans l'ensemble de données.

Pour gérer les données nouvellement arrivées, chaque fois qu'on détecte une nouvelle valeur extrême, on signale un événement. Le premier événement se produit à la 9864e minute (6 jours 20 h 24 min). Comme le montre la figure 18, une nouvelle valeur minimale est détectée, ce qui indique un nouveau motif de série temporelle. Le modèle de 100 minutes qui vient d'arriver ressemble beaucoup à un autre modèle qui s'est produit cinq heures plus tôt. Bien qu'il y ait beaucoup de régularité dans les données des réfrigérateurs en général, la similitude exceptionnelle observée ici suggère un mécanisme physique sous-jacent qui a causé un modèle si parfaitement conservé, peut-être un « sous-programme » de fabrication de glace mécanique.

Le deuxième événement se produit à la 10 473e minute (7 jours 6 h 33 min). Comme le montre la figure 19, une nouvelle valeur maximale est détectée, ce qui indique un nouveau discorde de série temporelle. La discorde de série temporelle correspond à la première occurrence d'un modèle de pompe à chaleur dans les données de consommation électrique.

Le temps maximum nécessaire pour traiter un seul point de données avec *STOMPI* dans cet ensemble de données est de 0,0003 s, ce qui est inférieur à 0,004 % du taux d'échantillonnage des données. Ainsi, sur ce jeu de données on a pu continuer la surveillance avec l'algorithme *STOMPI* pendant plusieurs décennies avant de manquer de temps ou de mémoire.

3.3. Implémentation

3.3.1. Outils de développement

Le développement du l'application a nécessité l'utilisation de quelques outils. Dans ce qui suit, nous citons les principaux outils qui ont été utilisés.

3.3.1.1. Langage de programmation Python

C'est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structuré, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à *Perl*, *Ruby*, *Scheme*, *Smalltalk* et Tcl [27].

3.3.1.2. PyCharm

C'est un environnement de développement intégré (IDE) utilisé dans la programmation informatique, spécifiquement pour le langage *Python*. Il est développé par la société tchèque *JetBrains* (anciennement *IntelliJ*) [5]. Il fournit une analyse de code, un débogueur graphique, un testeur d'unité intégré, une intégration avec des systèmes de contrôle de version (*VCS*) et prend en charge le développement Web avec *Django* ainsi que la science des données avec *Anaconda*.

PyCharm est multiplateforme, avec les versions *Windows*, *MacOS* et *Linux*. L'édition communautaire est publiée sous la licence *Apache*, [7] et il existe également une édition professionnelle avec des fonctionnalités supplémentaires - publiée sous une licence propriétaire [26].

3.3.2. Présentation de l'application

3.3.2.1. L'interface principale

L'entrée à l'application est réalisée à travers l'interface principale. L'utilisateur doit charger les fichiers au format *csv* qui contiennet les séries temporelles, pour les analyser, à travers les deux boutons *Charger S1* et *Charger S2*.

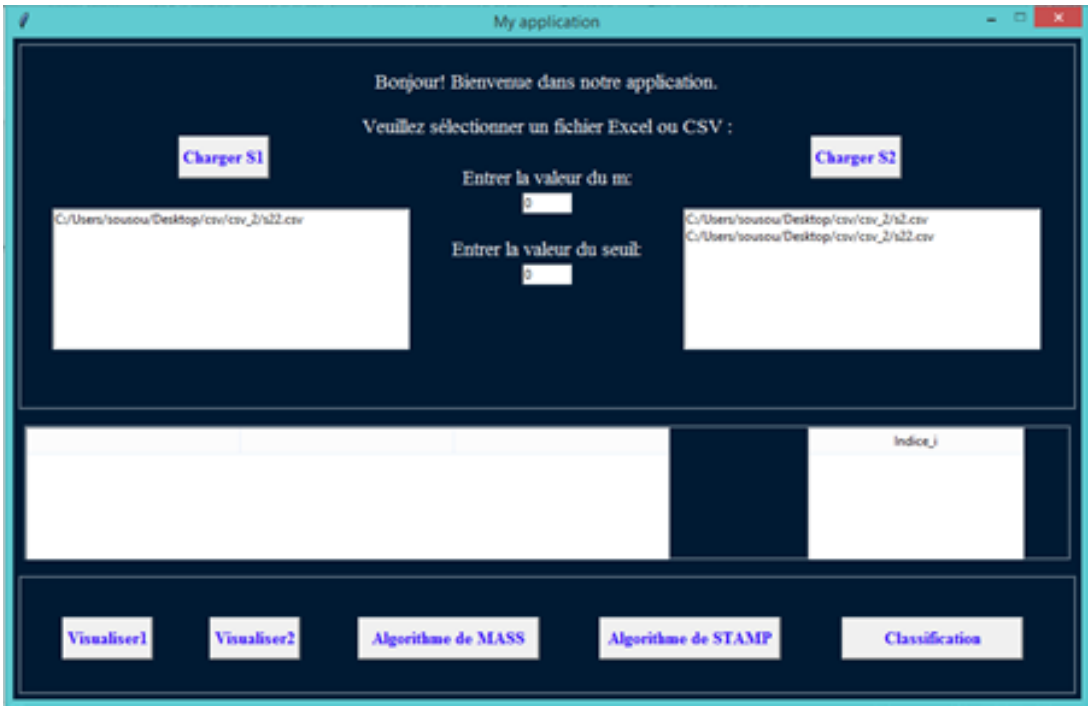

Figure 19 - *Interface principale de l'application*

3.3.2.2. Le chargement des données

Le tableau permet de visualiser les données des fichiers csv qu'on a chargé en faisant un double-clique sur le fichier qu'on veut afficher :

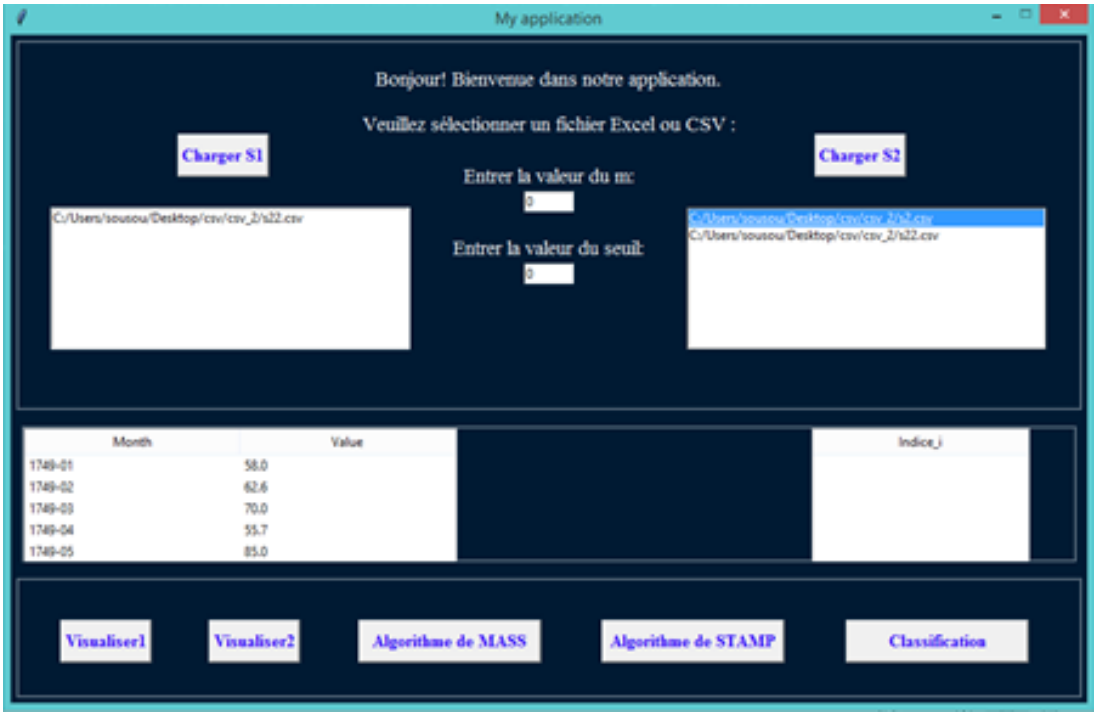

Figure 20 – *La visualisation des données*

3.3.2.3. La première visualisation des données

Le bouton *Visualiser 1* permet de visualiser un seul graphe en cliquant une seule fois sur le fichier trouvé sur *Charger S1,* comme on peut également changer de fichier en cliquant une fois toujours sur le fichier trouvé sur *Charger S2* et le résultat apparait en cliquant sur *Visualiser 1*. Ce qui caractérise ce bouton c'est la lecture d'un seul fichier, il n'est pas possible que ce bouton visualise plusieurs fichiers en même temps :

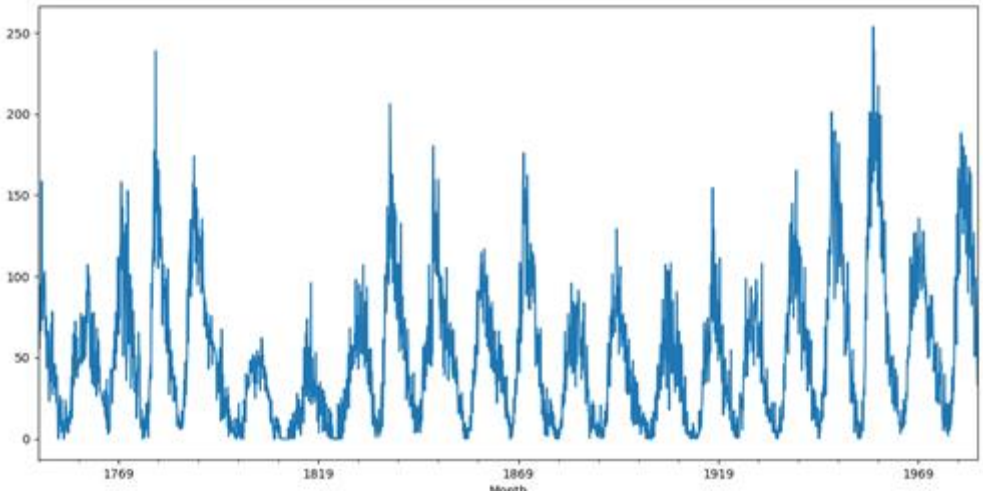

Figure 21 – *Première visualisation du graphe*

3.3.2.4. La deuxième visualisation des données

Le bouton *Visualiser 2* permet de visualiser deux fichiers, un de *Charger S1* et un de *Charger S2* au même temps et cela contrairement au bouton *Visualiser 1* dont le résultat est par sélection du fichier qu'on a voulu étudié.

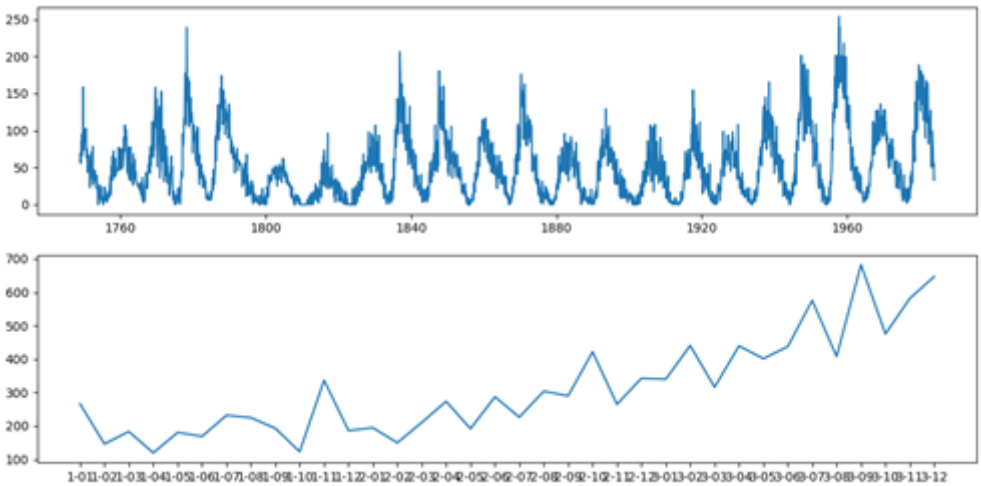

Figure 22 - *Deuxième visualisation du graphe*

3.3.2.5. Le calcule du distance profile

Cette figure représente le distance profile calculé à l'aide d'algorithme *MASS* avec *m* « la longueur de la sous séquence », à travers la concaténation de tous les fichiers *csv* dans chaque liste.

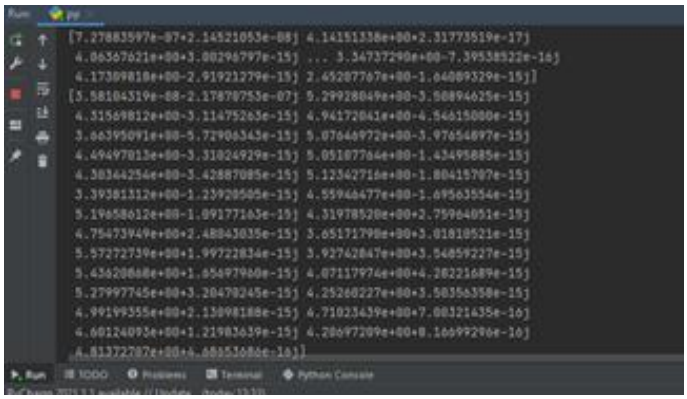

Figure 23 - *Le distance profile*

3.3.2.6. Le calcule du matrix profile

Même chose pour le bouton de l'algorithme de STAMP mais au lieu de calculer la distance, nous calculons le matrix profile, en entrant la valeur du m.

Ce bouton permet aussi d'afficher les valeurs du matrix profile en graphe :

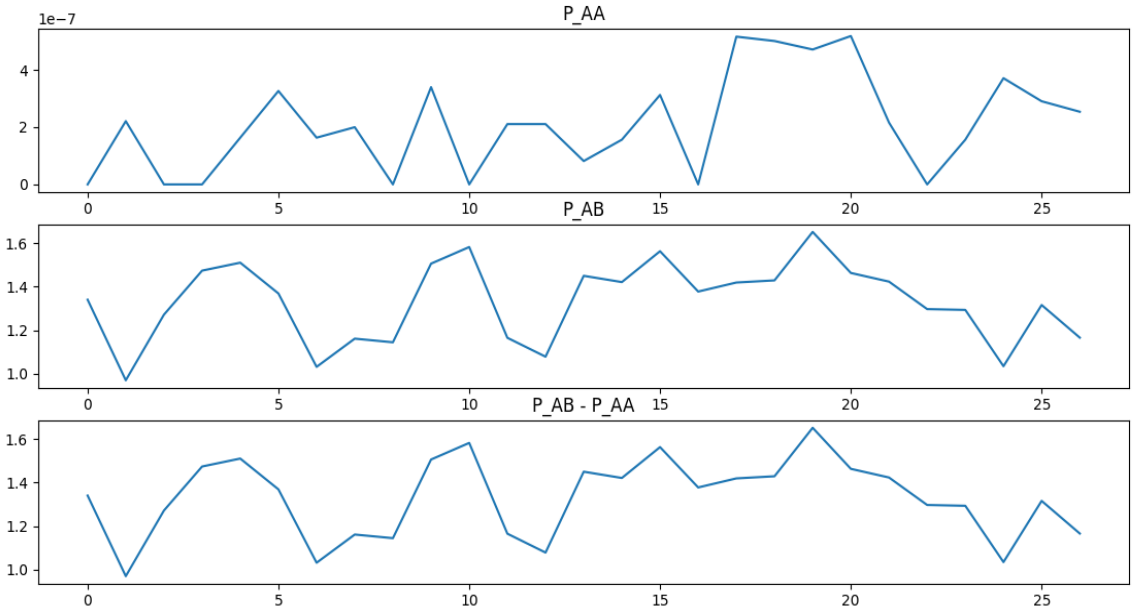

Figure 24 – *Les graphes du matrix profile*

Cela étant, nous comparons la différence des deux matrices qui est représenté dans le 3ème graphe de la figure ci-dessus avec la valeur du seuil qu'on a saisie. Si elle est supérieure au seuil nous la mettons dans le tableau « *Classe_A* » :

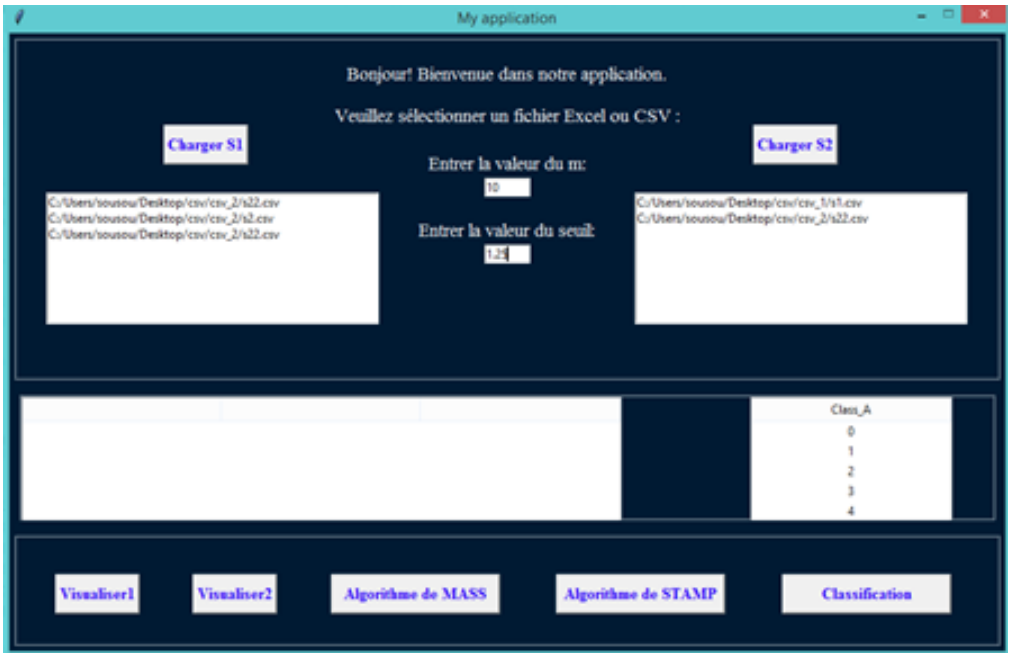

Figure 25 - *L'extraction des shaplets « class_A »*

On peut visualiser les *shapelets* si on fait un double clic sur un élément de la « *Classe_A* », par exemple :

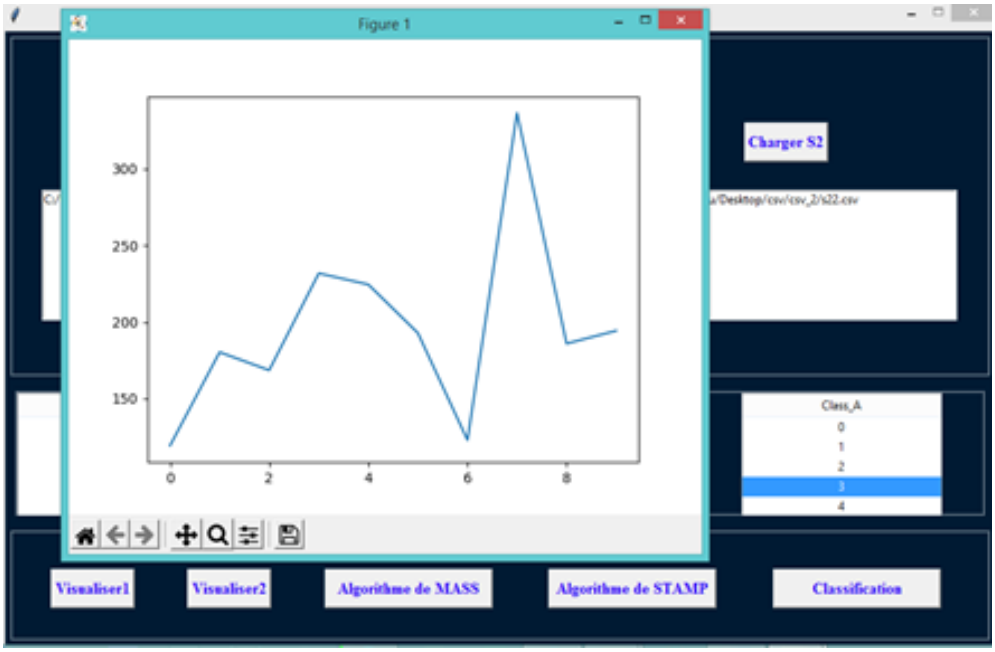

Figure 26 - *Visualisation d'une shapelet*

3.3.2.7. La classification

Dans notre application, nous pouvons charger un autre fichier et le comparer avec la *shapelet* sélectionnée, pour voir s'ils sont dans la même classe ou non.

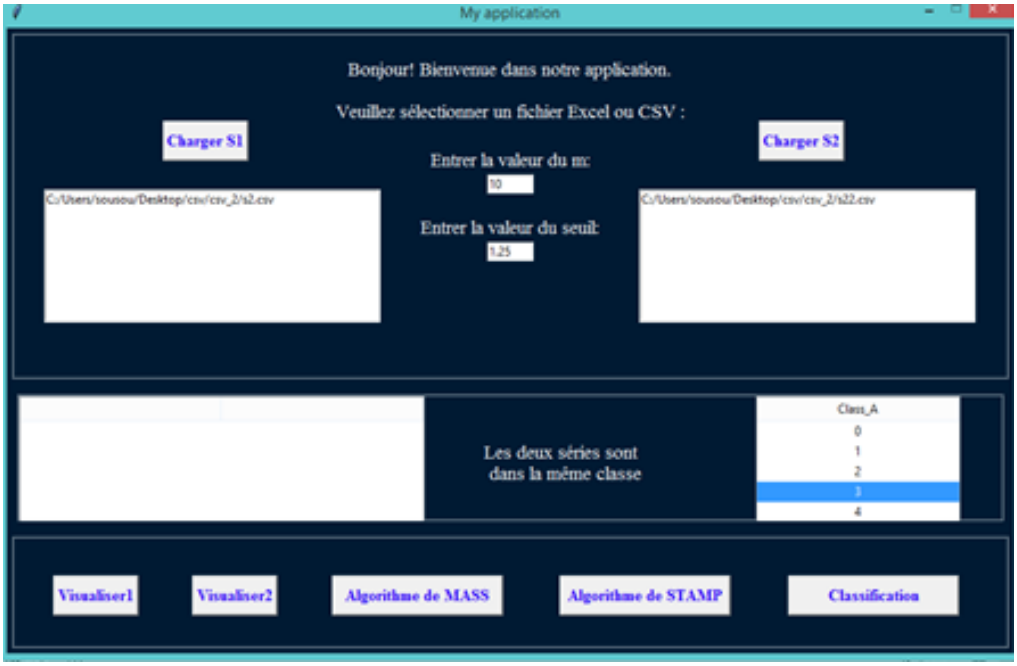

Figure 29 – *Le résultat de la classification*

3.4. Conclusion

Le but de notre travail et d'explorer les principes de la technique « Matrix Profile » et les algorithmes permettant de la mettre en œuvre, pour ensuite l'exploiter afin d'analyser de grandes bases de séries temporelles. Dans ce chapitre, nous avons fait une démonstration de ces pouvoirs pour la classification, l'extraction de motifs, et la détection d'anomalies ou de discordes.

Conclusion générale

Dans le cadre de ce projet, nous avons essayé d'aborder le problème d'analyse des séries temporelles, un problème traité dans le cadre du data mining. Pour cela nous avons présenté le data mining qui représente un ensemble de techniques d'exploration de données permettant d'extraire d'une base de données des connaissances sous la forme de modèles de description par exemple, ainsi que les différentes tâches que peut effectuer un système de data mining et les différentes méthodes qui peuvent être appliqué pour accomplir ces tâches.

Nous avons vu aussi la notion de série temporelle qui est une suite de valeurs numériques représentant l'évolution d'une quantité spécifique au cours du temps. De telles suites de variables aléatoires peuvent être exprimées mathématiquement afin d'en analyser le comportement, généralement pour comprendre son évolution passée et pour en prévoir le comportement futur. Une telle transposition mathématique utilise le plus souvent des concepts de probabilités et de statistiques. Cependant, nous nous sommes intéressés à une technique récemment proposée dite technique « Matrix profile ».

Notre travail consistait à bien étudier cette technique, comprendre ses fondements et ses principes, ainsi que les algorithmes permettant de la mettre en œuvre. Une telle étude approfondie nous a permet d'explorer les capacités de la technique pour les tâches connues du data mining, notamment la classification. Dans ce sens, nous avons essayé d'utiliser la technique pour la classification, supervisée ou non, des bases de séries temporelles, et d'apporter les modifications nécessaires pour arriver à cette fin.

Le constat que nous pouvons faire est que cette technique permet un cadre unique pour les différentes tâches d'analyse des séries temporelles, que ce soit l'extraction des motifs, la détection d'anomalies, l'extraction des shaplets, pour ensuite les mettre en œuvre dans les différentes tâches de data mining en relation avec les séries temporelles.

Ce travail a ouvert la voie vers plusieurs perspectives, à savoir la généralisation de l'étude sur le reste des tâches de data mining, comme le clustering (classification non supervisée) ou l'extraction des motifs séquentiels fréquents dans le domaine d'extraction des règles d'association.

Bibliographie

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

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

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

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

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

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

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

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

[9] Yeh, C.-C.-M., Zhu, Y., Ulanova1, L., Begum, N., Ding, Y., Anh Dau, H., Zimmerman1, Z., Silva, D.-F., Mueen, A., Keogh, E. : Time series joins, motifs, discords and shapelets : a unifying view that exploits the matrix profile, *Data Mining and Knowledge Discovery*, 32, 2018, pp83-123.

[10] Dieter D.-P., Sander V.-H., Bram S., Filip D.-T., Femke O., Olivier J., Sofie V.-H., A generalized matrix profile framework with support for contextual series analysis, *Engineering Applications of Artificial Intelligence* 90 (2020) 103487.

[11] Mahtab M., Yousef K., Lars K., James B., An Automated Matrix Profile for Mining Consecutive Repeats in Time Series, *X. Geng and B.-H. Kang (Eds.) : PRICAI* 2018, LNAI 11013, pp. 192–200, 2018.

[12] Chin-Chia M.-Y., Towards a Near Universal Time Series Data Mining Tool : Introducing the Matrix Profile. *A Dissertation of the requirements for the degree of Doctor of Philosophy in Computer Science* by September 2018.

Webographie

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

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

[15] https://www.cs.ucr.edu/~eamonn/Matrix_Profile

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

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

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

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