Fall Detection Device (F2D) : détection de chute basée sur smartwatch Android et machine learning

MSc HES-SO en Business Administration

Orientation :

Management des Systèmes d'information

Réalisé par Vincent Cuendet

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

AUTHENTIFICATION

Je, soussignée Vincent Cuendet, atteste avoir réalisé seul ce travail et respecté les travaux de tiers en mentionnant expressément, dans le chapitre « Références » ou dans le document, toutes les sources utilisées.

Froideville, 28.08.2015

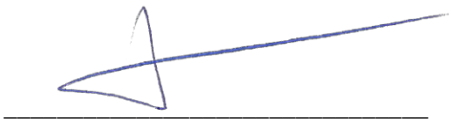


TABLE DES MATIÈRES

« Les détails font la perfection, et la perfection n'est pas un détail. »

Léonard de Vinci (1452-1519)

REMERCIEMENTS

Je tiens particulièrement à remercier Dominique pour son soutien et ses conseils avisés qui m'ont évité bon nombre d'embûches parsemées tout au long de ce travail.

Merci à Marlyse, Daniel et Claude pour leur relecture attentive et leurs commentaires constructifs et enrichissants.

Merci également à Laeticia, Mélina, Lexane et Nathanaël pour leurs sacrifices, leur soutien et leur compréhension dont ils ont fait preuve tout au long de cette formation.

Merci à tous.

INTRODUCTION

Le présent document est la concrétisation du travail réalisé dans le cadre du Master en Business Administration, orientation Management des Systèmes d'Information (MscBA MSI) auprès de la Haute Ecole de Suisse Occidentale (HES-SO).

L'objectif principal du travail est de trouver un modèle, au sens statistique, permettant de détecter les chutes à l'aide de données récoltées via divers capteurs d'une montre intelligente. Il s'agit d'un mandat donné par la FST (Fondation Suisse des Théléthèses). En outre, deux sous-objectifs viennent compléter ce premier élément : il s'agit d'une part de dresser un état de l'art de la connaissance en matière de détection de chutes, et d'autre part de déterminer la meilleure stratégie pour être le plus efficace dans ce dernier domaine.

Au niveau de la structure générale du rapport, il est subdivisé en deux grandes sections avec, tout d'abord les éléments synthétisés ayant mené à la conclusion du travail, puis des compléments détaillés produits durant la réalisation de cette recherche. En outre, il est segmenté de manière à illustrer la démarche entreprise tout au long de cette thèse : dans un premier temps, une analyse plante le décor afin de définir le contexte, arrêter un processus et choisir des outils adaptés. Ceci fait, le sujet principal est abordé, la stratégie de mise en œuvre est décrite en préambule, suivie des étapes nécessaires à la réalisation elle-même pour aboutir à la proposition d'un système de détection de chutes. La partie conclusive qui suit permet de jeter un regard critique sur les actions entreprises et les résultats obtenus tout en présentant succinctement des pistes d'amélioration ouvrant la voie à de futures recherches.

DÉFINITIONS ET CONVENTIONS

Un certain nombre d'acronymes anglais sont utilisés sans traduction dans le texte. Ces dernières se révélant parfois bancales, il a été préféré de garder les références dans leur langue d'origine. De plus, la grande majorité des publications disponibles sont faites dans la langue de Shakespeare.

Les notions de DM (Data Mining) et de KD (Knowledge Discovery) sont utilisées de manière interchangeables, bien que, dans la littérature, la différenciation soit faite, considérant le DM comme une étape du KD (Kurgan & Musilek, 2006). Cette séparation n'est pas remise en cause. Toutefois dans un souci de simplicité, vu que l'objectif de la présente analyse n'est pas la définition des méthodes utilisées dans le domaine de l'extraction de données, les deux termes sont placés sur un pied d'égalité.

La notion de sur-ajustement[1], utilisée dans le document fait référence à la production d'un modèle trop fortement lié aux données utilisées pour sa création. Ainsi ce dernier sera inutilisable dans un cadre différent.

Les notions de « chutes brutales » et de « chutes molles » font référence à la catégorisation définie par la FST, considérant que la première est le résultat d'un événement survenu alors que le sujet se trouvait debout, la deuxième résultant de positions assise ou couchée (Torrent, et al., 2015).

[1] http://www.statsoft.fr/concepts-statistiques/glossaire/s/surajustement.html

RÉSUMÉ

La recherche dans le domaine de la mise en place de systèmes de détection de chutes est un sujet d'actualité et de nombreuses démarches ont été entreprises durant ces 15 dernières années. Selon l'OMS (Organisation Mondiale de la Santé), 28% à 35% des personnes de plus de 65 ans chutent une fois par année et ces accidents représentent 40% des causes de décès parmi la totalité des blessures enregistrées. En outre, la fréquence d'accident augment avec l'âge puisque la proportion des chutes concerne entre 32% et 42% de la population âgée de plus de 70 ans (World Healt Organization, 2007, pp. 1,2). Dès lors, dans une société vieillissante dont le segment des personnes âgées de plus de 80 ans devrait représenter 20% de la population mondiale d'ici 2050 (World Healt Organization, 2007, p. 3), le champ des recherches et des investissements relatifs aux chutes est en plein expansion car il représente un important défi de santé publique.

De nombreuses approches sont proposées, la plupart étant mises en œuvre avec des contraintes particulièrement fortes – par exemple l'utilisation d'un nombre important de capteurs placés sur la poitrine et à l'intérieur de la jambe - qui bloquent totalement un éventuel développement pour le grand public. Toutefois, le niveau de maturité des études les plus récentes est prometteur, permettant d'envisager rapidement une sortie du cénacle académique.

Dans ce contexte, un mandat a été initié par la FST (Fondation Suisse pour les Théléthèses) afin de développer une application exécutée sur un terminal grand public permettant de détecter les chutes. Initialement, la demande portait sur la capacité de détection d'une large gamme d'activités (pour un total de 41) sur une plateforme Android[2]. Toutefois le choix a été fait de ne se concentrer que sur l'analyse de deux classes, à savoir le cas de « chute » et de « non chute » et uniquement pour des cas de « chutes molles », ceci afin de ne pas s'étendre trop largement et de pouvoir aboutir à un résultat exploitable par le mandant. En outre, les cas de « chutes brutales » ont déjà fait l'objet d'une recherche menée par le TaM (équipe « Travelling and Mobility » de l'Université de Genève)[3] ayant débouché sur un modèle utilisant des seuils et obtenant un taux de détection d'une exactitude de 84%.

La partie de mise en œuvre a été soutenue par l'approche préconisée par le processus CRISP-DM et un bon nombre d'outils et de technologies. La plateforme KNIME a notamment été choisie pour la définition du modèle ainsi que sa production au format PMML (Predictive Model Markup Language) permettant une consommation à l'aide d'une librairie Java.

Quatre itérations ont été réalisées permettant de tester un certain nombre d'hypothèses et d'affiner les choix faits tout au long de la démarche. L'algorithme de détection le plus efficace dans ce contexte est un ensemble d'arbres de décision. Un modèle interprétable par la libraire Java « jpmml-evaluator » a été produit. Cette dernière pouvant ensuite aisément être intégrée au sein d'une application Android ouvrant la voie à l'implémentation de la recherche menée dans l'application F2D (Fall Detection Device). Pour terminer, une architecture d'implémentation est proposée, visant la mise en place d'un prototype pouvant être validé sur le terrain.

En conclusion, ce travail permet de faire progresser la détection de chutes molles, premièrement en obtenant un score AUC (Area Under the Curve) de plus de 92% et, deuxièmement, en proposant une implémentation dans l'environnement F2D.

[2] https://www.android.com/
[3] http://tam.unige.ch/projects/f2d.html

ÉTAT DE L'ART

La démarche décrite dans ce chapitre a été menée afin de dresser une image la plus complète possible sur l'état actuel en matière de recherches concernant les systèmes de détection de chutes. Dans une première phase, les opérations de collecte de l'information sont décrites, puis une synthèse est proposée.

MÉTHODOLOGIE

La recherche d'informations s'est faite autour du thème de la détection des chutes à l'aide des mots clefs « fall detection », « fall detection algorithm », « fall detection device », « fall detection with accelerometer », « fall detection with machine learning », « machine learning fall detection », « machine learning algorithms fall detection » sur les plateformes disponibles sur internet que sont Google Scholar[4], Science Direct[5], PubMed[6], et IEEE Xplore[7].

Le choix de la langue anglaise a été retenu car après une rapide évaluation, menée sur Google Scholar, le nombre d'occurrences retournées pour cet idiome surpasse largement celui du français. Ainsi, une recherche du terme « détection de chutes » retourne environ 15 800 résultats alors que l'utilisation de la même combinaison de termes en anglais, soit « fall detection », produit quant à elle plus de 3 340 000 d'entrées, soit un facteur d'environ 200, élargissant d'autant les informations potentiellement intéressantes.

Les plateformes PubMed et IEEE Xplore ont été sélectionnées, car le sujet traite d'une discipline se situant à cheval entre les domaines de l'ingénierie et du médical et ces sources sont spécialisées pour cela. Pour les deux autres, à savoir Google Scholar et Science Direct, il s'agit de portails génériques pouvant mettre en lumière des relations entre des recherches faites en dehors du domaine spécifique de l'ingénierie biomédicale.

Les documents sélectionnés sont ceux ayant les combinaisons de mots recherchés dans le résumé. L'intérêt s'est limité aux articles correspondant à la détection faite à l'aide de senseurs portés, c'est-à-dire que les publications traitant de systèmes basés sur la vidéo, les sons ou tout autre capteur contextuel ont été écartées. Les recherches choisies pour une lecture approfondie sont celles basées sur des mesures d'accélération et optionnellement de position. Pour l'accélération, il s'agit en général de capteurs à trois degrés de liberté – sur trois axes en X, Y et Z - correspondant au système d'enregistrement de la FST.

En outre, pour les recherches faisant état de l'utilisation de techniques de ML (Machine Learning), seules les publications récentes, à savoir postérieures à 2011 ont été retenues. Au total 29 documents ont ainsi été sélectionnés et analysés.

ANALYSE DE L'EXISTANT

Globalement, un système de détection va chercher à identifier les chutes parmi les activités journalières normales, également mentionnées sous l'acronyme ADL (Activities of Daily Living). Dans la littérature, il existe plusieurs moyens de grouper les systèmes de détection de chutes ; certains se basent sur l'identification des phases (Noury, Rumeau, Bourke, OLaighin, & Lundy, 2008), d'autres sur l'usage unique ou non de l'accélération (Mubashr, Ling, & Luke, 2013) ou encore sur le type de capteurs (Perry, et al., 2009).

Dans le présent document, l'approche proposée par Igual, Medrano et Plaza (2013, pp. 3,4) est utilisée comme fil rouge. La manière de classer les systèmes se fait en deux grandes familles que sont, premièrement les systèmes contextuels et deuxièmement les systèmes portables.

[4] https://scholar.google.ch/
[5] http://www.sciencedirect.com/
[6] http://www.ncbi.nlm.nih.gov/pubmed
[7] http://ieeexplore.ieee.org/Xplore/home.jsp

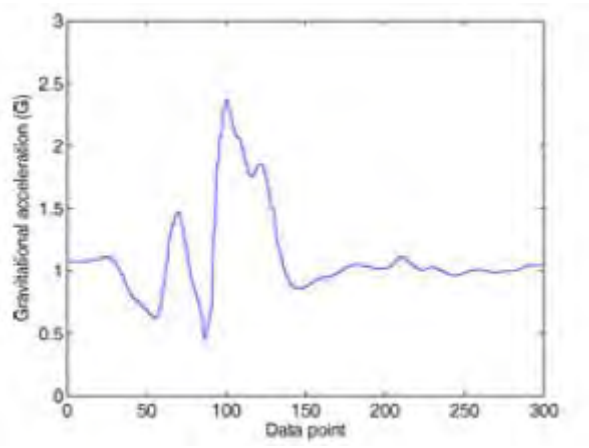
La recherche se focalise uniquement sur la deuxième famille, car la FST a déjà arrêté son choix sur un dispositif qui doit être transportable par le sujet. Cette manière de faire offre une plus grande liberté à ce dernier et ne limite pas le champ de surveillance à un lieu donné.

Les données choisies pour détecter les chutes sont, de manière générale, basées sur celles provenant d'un ou plusieurs accéléromètres (Li, Chen, Shen, Zhu, & Cheng, 2012), (Aguiar, Tiago, Joana, & Inês, 2014), (Bagalà, et al., 2012), (Albert, Kording, Herrman, & Jayaraman, 2012), (Yuanchun Shi & Wang, 2012), (Tan & Tinh, 2014), (Yuwono, Moulton, Su, Celler, & Nguyen, 2012), (Kung, et al., 2010), (Dai, Bai, Yang, Shen, & Xuan, 2010), (Karantonis, Narayanan, Mathie, Lovell, & Celler, 2006), (Abbate, et al., 2012) permettant de capter les mouvements brusques liés à un accident et sont parfois accompagnées de celles d'un gyroscope (He, Hu, & Li, 2013), (Ojetola, Gaura, & Brusey, 2011) ou d'un altimètre (Bianchi, Redmond, Narayanan, Cerutti, & Lovell, 2010) pour définir la position de l'utilisateur. De manière générale, on peut constater que l'ajout de capteurs augmente la capacité de détection. On distingue deux grandes familles pour la détection des événements qui sont celle de l'utilisation de seuils (TBM, Threshold Based Measure), généralement définis empiriquement, et celle basée sur les techniques de ML (MLM, Machine Learning Method) (Igual, Medrano, & Plaza, 2013, p. 10). Il existe également des approches combinant les seuils et la classification à l'aide du ML et, pour terminer, la possibilité d'utiliser le ML plutôt que l'approche empirique pour l'établissement des seuils (Aguiar, Tiago, Joana, & Inês, 2014).

TBM

La définition de seuils se base sur la constatation que les valeurs enregistrées par différents capteurs, principalement l'accéléromètre, vont montrer des amplitudes particulières en cas de chutes. Par exemple, il est possible d'identifier dans un premier temps une décélération, soit une accélération négative tombant sous le seuil de la constante de la gravité terrestre dont la valeur est de 9.81 m/s, suivie d'un enregistrement ayant un valeur importante, puis éventuellement d'une série de moins grande amplitude, pour finir sur un signal presque plat, le sujet étant étendu à terre sans mouvement (Vo, Gueesang, & Deokjai, 2012), (Yuanchun Shi & Wang, 2012), (Ojetola, Gaura, & Brusey, 2011), (Kung, et al., 2010), (Dai, Bai, Yang, Shen, & Xuan, 2010), (Abbate, et al., 2012), (Kau & Chen, 2015). Les seuils cherchent donc à établir des valeurs, deux dans la forme la plus simple, qui seront caractéristiques des événements mentionnés précédemment.

Figure 1 Signal enregistré lors d'une chute. Source: (Kau & Chen, 2015, p. 46).



Historiquement, l'approche TBM est plus ancienne et peut se révéler, de par la nature statique des seuils, peu adaptée à la réalité si un nombre de cas suffisant n'a pas été inclut lors de la phase de définition de ces derniers. Il existe au moins une approche pour déterminer les seuils de manière dynamique. Elle est décrite dans une section suivante -intitulée « Hybrides » - car du ML est utilisé pour déterminer la valeur des seuils.

MLM

Dans le domaine du MLM (Machine Learning Method), plusieurs types d'algorithmes sont utilisés qui vont de SVM (Support Vector Machine) (Albert, Kording, Herrman, & Jayaraman, 2012), aux chaînes de Markov (Mirchevska, Lustrek, & Gams, 2013) en passant par KNN (K-Nearest Neighbors) (Chien-Liang, Chia-Hoang, & Ping-Min, 2010) et MLP (perceptron) (Yuwono, Moulton, Su, Celler, & Nguyen, 2012), (Abbate, et al., 2012) ou encore en comparant différents algorithmes (Luštrek & Kaluža, 2009).

De manière générale, le MLM est plus souvent choisi dans les travaux les plus récents et on peut considérer que cette tendance va se poursuivre (Igual, Medrano, & Plaza, 2013), non seulement car elle permet une détection plus poussée de motifs dans les données, mais également car les notions de DM et de KD sont largement en phase de popularisation.

HYBRIDES

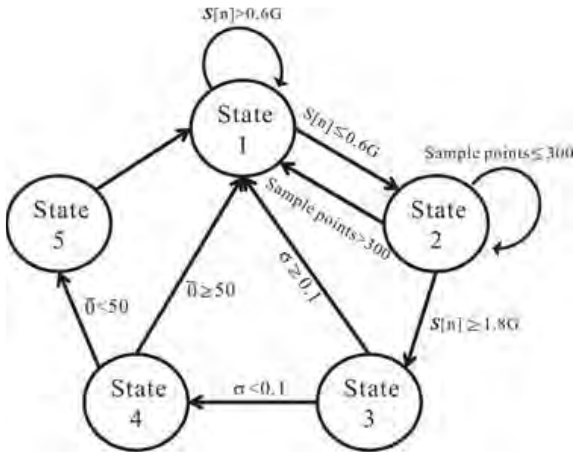
La combinaison des approches TBM et MLM, qualifiable d'hybride, est choisie dans le cadre de systèmes ayant comme objectif une utilisation optimale des ressources, élément particulièrement critique pour les solutions autonomes (Yuanchun Shi & Wang, 2012). En effet, il est souvent avancé que la mise en œuvre d'un modèle produit par ML est plus énergivore que l'utilisation de seuils (Kau & Chen, 2015), (Kostopoulos, Nunes, Slavi, Deriaz, & Torrent, 2014).

Une proposition pour contourner la rigidité de l'approche TBM est d'utiliser des techniques de découverte de connaissances et de ML (CDKML, Combined Domain Knowledge Discovery and Machine Learning) pour adapter dynamiquement les valeurs des seuils (Mirchevska, Lustrek, & Gams, 2013). Dans ce cas, les taux de classification enregistrés se révèlent supérieurs aux méthodes TBM et MLM prises séparément.

Pour Yuanchun Shi *et al.* (2012), une chute peut être découpée en cinq phases, chacune d'elle ayant des caractéristiques particulières. Des seuils sont définis, lesquels atteints, vont produire un score, lui-même déterminant si un cas de chute est suspecté. Dans un cas positif, une transformation des données sous forme de signal est effectuée, ce dernier étant ensuite fourni au classificateur utilisant l'algorithme SVM.

On retrouve chez Kau et Chen (2015) la même logique que décrite précédemment : l'utilisation du classificateur intervient comme étape ultime dans le processus de détection, les étapes précédentes étant basées sur des seuils empiriquement définis. Toutefois ici, il ne s'agit pas d'un score global mais d'une suite de vérifications qui dirige le processus. Ainsi une machine à état sur cinq niveaux est définie pour modéliser le système. Il faut également noter que le signal est traité d'une part avec un filtre haute-fréquence FIR (Finite Imput Response) et de l'autre avec une transformation en ondelettes discrète (DWT, Discrete Wavelet Transform), afin de mener une analyse spectrale plus poussée. La machine à état ainsi construite est présentée ci –dessous ;

Figure 2 Machine à état proposée pour la détection de chutes. Source: (Kau & Chen, 2015).



CONCLUSION

Selon les différentes données recueillies précédemment, ainsi que l'analyse des solutions actuellement disponibles, voici une proposition pour des paramètres à ajuster lors de l'établissement de la stratégie pour la création d'un système de détection;

Etape	Paramètre	Prochaine étape
1	Ressources de la plateforme d'exécution fortement contraintes.	Forte contrainte : Oui ; étape 2a. Non ; étape 2b.
2a	Choix de la méthode TBM ou Hybride.	Méthode choisie : TBM ; étape 4. Hybride ; étape 3.
2b	Choix de la méthode TBM, MLM ou Hybride.	Méthode choisie : TBM ; étape 4. MLM ; étape 3. Hybride ; étape 3.
3	Choix de l'algorithme.	Étape 4.
4	Choix de données.	-

Tableau 1 Etapes pour la définition d'une architecture d'un système de détection de chutes.

Dans le cadre du présent travail, les ressources à disposition du système sont limitées au niveau de la puissance de calcul ainsi que de celui de l'autonomie, une méthode Hybride est donc privilégiée.

Le choix des données découle des conclusions du chapitre « Compréhension des données » et celui de l'algorithme est le résultat des expériences menées en « Itération 1 ».

MÉTHODES

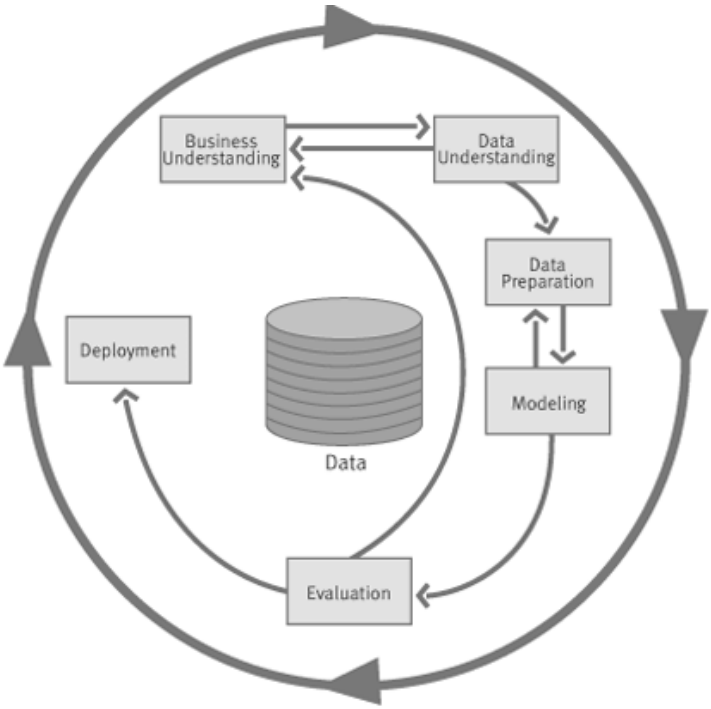
Dans le présent cadre, seuls sont considérés les processus disponibles en dehors des entreprises, bien que de nombreuses manières de faire semblent exister dans les structures utilisant le forage de données. La grande variété dans les approches est due au fait que le travail peut être adapté selon les contextes et les besoins particuliers. D'autant plus que cette forme d'activité est également considérée comme un art requérant de l'analyste flexibilité et créativité.

Les modes opératoires CRISP-DM (CRoss-Industry Standard Process for Data Mining), SEMMA (Sample, Explore, Modify, Model, Assess) et KDD (Knowledge Discovery in Databases) sont ceux le plus souvent référencés dans la littérature (Azevedo & Santos, 2008, p. 182). Il ne s'agit pas à proprement parler de méthodologies, mais sont des guides permettant de poser un fil rouge lors de la mise en œuvre d'une recherche ou d'un projet.

CRISP-DM

Le processus, formalisé par un guide (CRISP-DM 1.0 - Step-by-step data mining guide), se déroule en six étapes et débute par deux phases permettant de comprendre le contexte métier dans lequel s'inscrit la démarche, l'exploration du matériel à disposition en termes de données, ainsi que les objectifs poursuivis. Une fois le cadre établi, les données sont traitées, c'est-à-dire filtrées, formatées ou encore nettoyées, afin de les rendre utilisables en fonction des buts fixés. Les données étant préparées, il est possible de passer à la production de modèles et à leur évaluation. Ce point atteint, il peut être nécessaire de commencer une nouvelle itération ou de passer au déploiement du modèle sélectionné, action généralement réalisée par d'autres personnes que l'analyste (Chapman, et al., 2000).

Figure 3 Diagramme du processus CRISP-DM. Source; http://www.infotechmarketing.net/images/Crisp-dmchartnew.gif, tiré de Chapman, et al. (CRISP-DM 1.0 - Step-by-step data mining guide, 2000, p. 12).



Cette manière de procéder est à la base du découpage du livre « Guide to Intelligent Data Analysis » (Berthold, Borgelt, Höppner, & Klawonn) et également présent au sein du logiciel SPSS Modeler[8].

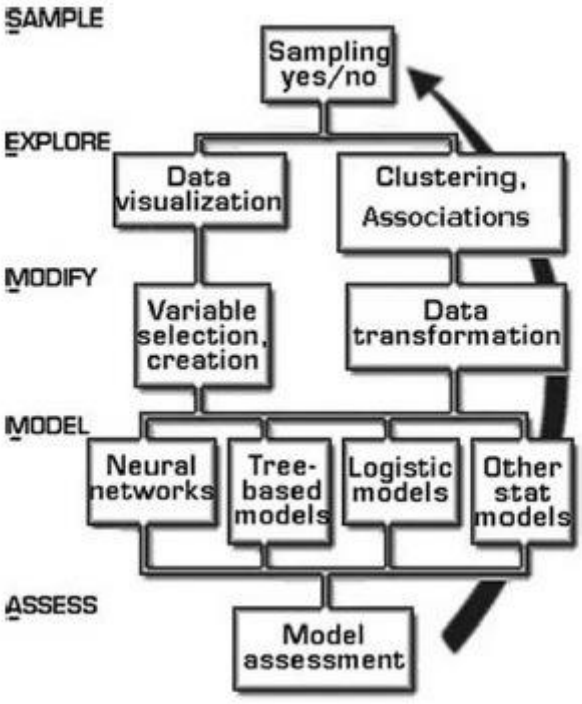
Rohanizadeh et Moghadam (2009) émettent des réserves sur ce processus, car il requiert que toutes les données soient connues au début du projet, que les techniques peuvent être mise en œuvre non pas en raison de leur nécessité, mais parce qu'elles sont disponibles, et que le choix du modèle, bien que présenté comme ayant un impact important et devant être sélectionné rapidement, ne se fait qu'en étape quatre. Toutefois ils concluent que ce processus reste intéressant, car il représente les étapes nécessaires à un travail d'extraction de la connaissance (Rohanizadeh & Moghadam, 2009, pp. 42-43).

SEMMA

Cette approche, qui compte cinq étapes, est liée aux solutions logicielles produites par la société SAS Institute Inc.[9], un leader dans le domaine du logiciel statistique et de l'intelligence métier. Aussi, il n'existe pas de document décrivant ce processus. L'acronyme reprend les différentes actions à entreprendre, à savoir échantillonner (Sample), explorer (Explore), adapter (Modify), modéliser (Model) et évaluer (Access).

Les opérations décrites se focalisent donc sur la partie de forage des données uniquement, laissant de côté les phases préparatoires et finales durant lesquelles, non seulement le contexte est posé, mais également les actions de finalisation sont entreprises. Il peut donc être risqué d'utiliser SEMMA en dehors des solutions logicielles fournies par SAS (Rohanizadeh & Moghadam, 2009, p. 42).

Figure 4 Flux du procesus SEMMA. Source: http://decisionstats.com/2013/04/10/visual-guides-to-crisp-dm-kdd-and-semma/



[8] https://en.wikipedia.org/wiki/SPSS_Modeler
[9] https://en.wikipedia.org/wiki/SAS_Institute_Inc.

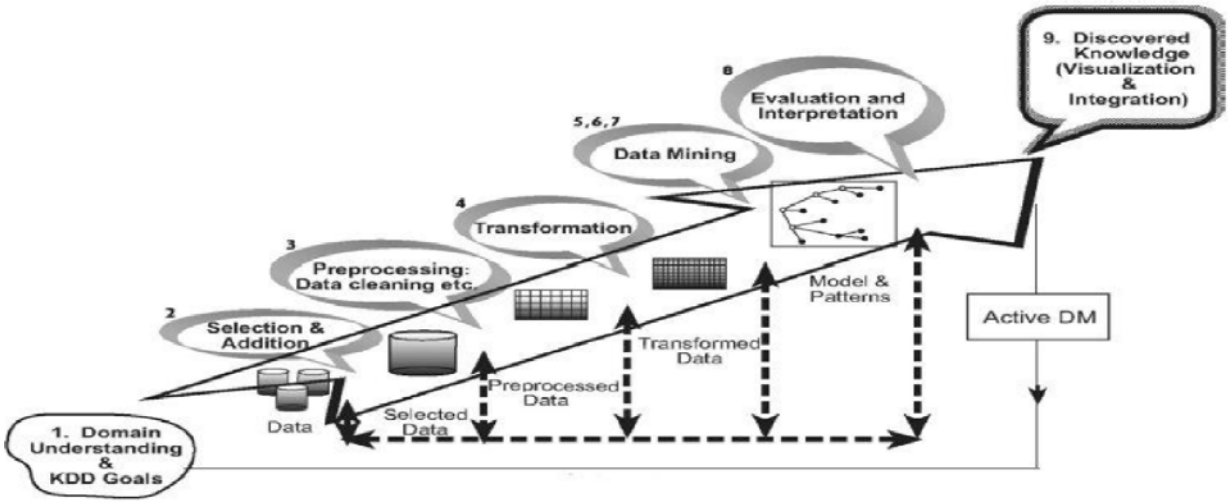
KDD Process

Le processus KDD (Knowledge Discovery in Database), soit littéralement celui de découverte de connaissances dans une base de données, comprend neuf étapes – groupables en sept phases majeures -, faisant de ce dernier le plus complexe parmi les trois approches brièvement discutées ici. Il s'appuie sur une connaissance préalable du domaine et des buts à atteindre. (Fayyad, Piatetsky-Shapiro, & Smyth, 1996).

Dans cette vision, le DM (Data Mining) n'est que le composant technique du processus qui met en œuvre des algorithmes de détection de motifs. Or toutes les étapes placées en amont et en aval sont autant - si ce n'est plus – importantes, afin de pouvoir réellement construire de la connaissance ; tout motif placé hors de son contexte ne véhiculant pas ou peu de sens.

Le processus est une séquence itérative d'étapes, comme pour les deux approches précédemment discutées, mais il permet explicitement de produire des « boucles » plus réduites entre deux étapes si nécessaire (Fayyad, Piatetsky-Shapiro, & Smyth, 1996, p. 42).

Figure 5 Etapes du processus KDD. Source: (Shafique & Qaiser, 2014, p. 218).



CHOIX DE LA MÉTHODOLOGIE

Selon Azevedo et Santos (2008) et Shafique et Qaiser (2014), les différences entre les trois approches brièvement détaillées précédemment ne sont pas grandes, tout au moins dans les concepts. Ainsi mettent-ils en relation les différentes étapes des trois processus et aboutissent à la conclusion que SEMMA et CRISP-DM sont des sous-ensembles, ou implémentations de KDD ; CRISP-DM étant toutefois plus détaillé que SEMMA.

Etape	KDD	SEMMA	CRISP-DM
1	Développement et compréhension du métier.	-	Compréhension du métier.
2	Création d'un jeu de données.	Echantillonner.	Compréhension des données.
3	Nettoyage et prétraitement des données.	Explorer.	
4	Transformation des données.	Modifier.	Préparation des données.
5	Définition du type de forage de données.	Modéliser.	Modélisation.
	Choix de l'algorithme de forage.		
	Mise en œuvre de l'algorithme.		
6	Interprétation des résultats.	Evaluer.	Evaluation.
7	Utilisation de la connaissance acquise	-	Déploiement.

Tableau 2 Mise en relation des différents processus de forage des données. Source: Shafique & Qaiser (2014, p. 221)

Au vu de la remarque relevée plus haut quant à la similitude des processus, le choix s'est porté sur l'approche CRISP-DM, car elle est disponible en dehors du contexte d'un outil particulier, que son utilisation est répandue dans de nombreux projets (43%)[10], qu'elle tient compte du domaine métier et qu'elle sous-tend l'ouvrage « Guide to Intelligent Data Analysis », ce dernier étant focalisé sur une mise en œuvre pratique du DM.

[10] http://www.kdnuggets.com/polls/2014/analytics-data-mining-data-science-methodology.html

OUTILS

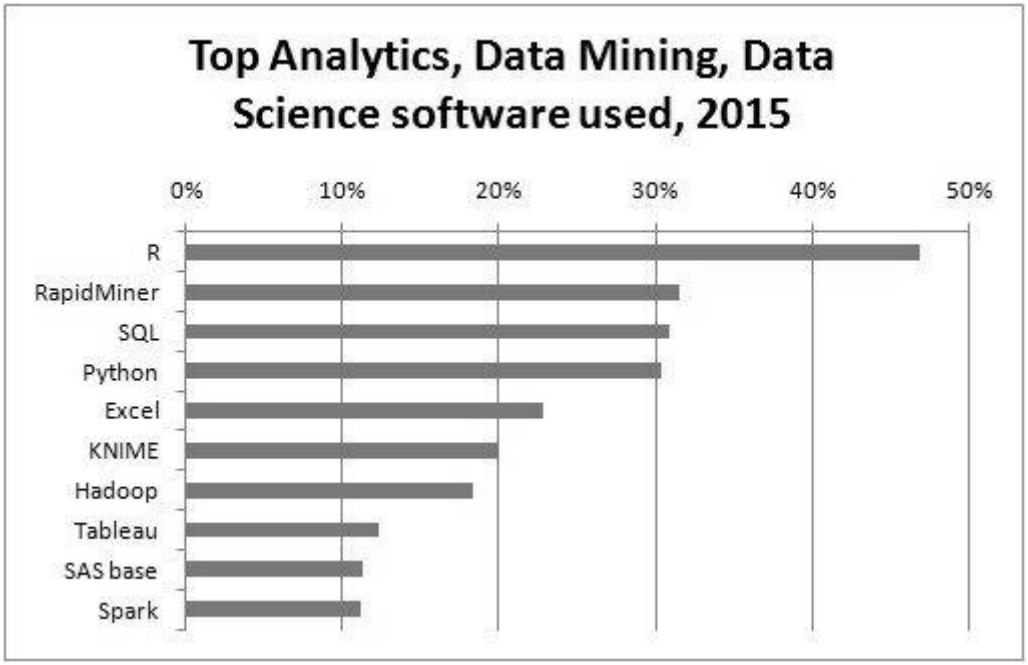
Dans ce chapitre une vue d'ensemble sur les différents outils disponibles est présentée, puis le choix pour le présent travail est défini et argumenté.

OUTILS DISPONIBLES

La boîte à outils de l'analyste des données est bien fournie et peut comprendre de simples « framework » gratuits facilitant la production de scripts comme des suites composées d'une myriade de logiciels et coûtant plusieurs milliers de francs.

En se basant sur le sondage annuel effectué par le site KDnuggets et publié au mois de mai 2015, l'outil le plus utilisé est R[11] avec 46.9% des réponses (KDnuggets, 2015). Il est intéressant de noter que les 10 premières réponses couvrent un spectre d'outils allant d'une solution spécialisée dans le traitement des statistiques, comme R, à une suite logicielle complète dédiée à l'exploration des données telle que RapidMiner[12]. Dans l'intervalle se trouvent, de manière non exhaustive, des langages de programmation - générique (Python) ou spécialisé (SQL) - un système de gestion de base de données (Hadoop[13]), une plateforme open-source (KNIME[14]) ou encore un outil pour la gestion de données tabulaires (Microsoft Excel). Le nombre total d'outils ou de technologies mentionnés dans le précédent sondage est de 81. On peut ainsi constater qu'il est courant d'associer plusieurs briques logicielles selon le type de problème devant être résolu. Par exemple, seuls 3.6% des répondants utilisent R de manière complètement autonome (KDnuggets, 2015).

Figure 6 Classement des 10 premiers outils utilisés dans le domaine du forage de données. Source: (Analytics, Data Mining, Data Science software/tools used in the past 12 months).



[11] https://www.r-project.org/
[12] https://rapidminer.com/
[13] https://hadoop.apache.org/
[14] https://www.knime.org/

CHOIX DE LA PLATEFORME

Dans le cadre du présent travail, le choix de l'outil de forage de données s'est porté sur la plateforme KNIME et s'est fait selon plusieurs critères. Le premier est l'argument du coût. En effet, il n'est pas envisageable d'investir une somme importante pour ce type de mandat, d'autant plus qu'il est initié dans le cadre d'une formation. Le suivant est lié à l'énoncé du thème du travail de Master. Ce dernier mentionne les outils iPython[15] et KNIME comme alternatives crédibles. La troisième raison se situe dans l'effort nécessaire à la maîtrise d'un outil qui n'est pas la finalité même de ce travail. L'initiation déjà reçue sur la plateforme choisie a dès lors permis de limiter le temps nécessaire à sa prise en main, dégageant un maximum de ressources pour l'objectif principal de recherche. Pour finir, le classement de KNIME en sixième position dans le sondage de KDnuggets démontre une adoption importante permettant de disposer de ressources suffisantes en termes de quantité et de qualité au sein d'une communauté de taille suffisante.

[15] http://ipython.org/

EVALUATION DE LA PERFORMANCE D'UN CLASSIFICATEUR

Cette section présente les méthodes disponibles pour pouvoir mesurer l'efficacité d'un classificateur et de pourvoir les comparer entre eux, puis sélectionne la plus adaptée au présent contexte.

MESURES DISPONIBLES

Dans le cadre de l'évaluation de la performance de classificateur, la littérature contient de nombreuses mesures, certaines étant plus adaptées à des domaines spécifiques.

Oufella, Adam, Chatelain, Heute et Lecourtier (ÉVOLUTION DU CONCEPT DE FRONT ROC ET COMBINAISON DE CLASSIFFIEUR, 2008) présentent, dans un cheminement augmentant la pertinence de l'évaluation des performances, tout d'abord les mesures scalaires (le taux d'éléments classés correctement), les méthodes d'évaluation multicritères (la courbe de précision-rappel, la courbe ROC, la F-mesure et l'AUC), puis leur généralisation à des problèmes multi-classes.

Les premières, soit les mesures scalaires, peuvent ne pas se révéler assez pertinentes et doivent être mises en relation avec l'impact (le coût) engendré par une mauvaise classification. (Oufella, Adam, Chatelain, Heute, & Lecourtier, 2008, p. 16). Toutefois cette notion de coût est parfois difficilement calculable, voire souvent même, totalement inconnue, ce qui nécessite l'utilisation de méthodes multicritères.

Dans cette dernière catégorie, on trouve les mesures de précision et de rappel qui sont très utilisées dans le domaine de la recherche documentaire et qui permettent d'évaluer la pertinence d'une requête en terme d'information retournée (Oufella, Adam, Chatelain, Heute, & Lecourtier, 2008, p. 18). Ces mesures s'obtiennent en se basant sur une matrice de confusion, un élément qui regroupe les résultats obtenus suite à une opération de classification dont la forme est la suivante ;

	Décision ; positif	**Décision ; négatif**
Label ; positif	Vrai positif (VP)	Faux négatif (FN)
Label ; négatif	Faux positif (FP)	Vrai négatif (VN)

Tableau 3 Matrice de confusion

Pour permettre de comparer deux classificateurs en se basant sur la précision et le rappel, il est possible d'utiliser le graphique – désigné comme la courbe Précision-Rappel - mettant en relation ces deux mesures ou la F-Mesure, une valeur scalaire obtenue à partir de la précision et du rappel. Il faut toutefois être attentif au fait que les éléments mentionnés ci-dessus sont sensibles au non équilibrage des classes à évaluer (Oufella, Adam, Chatelain, Heute, & Lecourtier, 2008, p. 19), (Fawcett, 2006, p. 865).

$$précision = \frac{VP + VN}{P + N} \qquad rappel = \frac{VP}{P} \qquad F - mesure \frac{2}{1/sensibilité + 1/rappel}$$

Équation 1 Formules permettant de calculer la précision, le rappel et la F-mesure.

Dans la catégorie des méthodes multicritères, on trouve également l'analyse ROC (Receiver Operating Characteristics) qui est efficace dans l'évaluation d'un modèle pour lequel l'évaluation des coûts n'est pas connue (Oufella, Adam, Chatelain, Heute, & Lecourtier, 2008, p. 20). Elle met en lumière la capacité d'un classificateur en fonction des taux « vrai positif » et « faux négatif ». Son grand avantage par rapport à la courbe Précision-Rappel vient du fait que les valeurs utilisées pour établir le graphique sont insensibles à la répartition des classes. En effet, elles se basent sur une seule colonne de la matrice de confusion et ne sont donc pas liées à la distribution des classes (Fawcett, 2006, p. 864), (Oufella, Adam, Chatelain, Heute, & Lecourtier, 2008, p. 21). Les valeurs utilisées vont aussi avoir une influence sur le score obtenu. Ainsi une courbe ROC avec 100% de réussite de détection ne donne pas forcément une précision de 100%.

$$tvp = \frac{VP}{P} \qquad tfp = \frac{FP}{N}$$

Équation 2 Formules pour le calcul des taux « vrai positif » et « faux négatif ».

La comparaison visuelle de deux courbes ROC ne permet pas de clairement les départager ; aussi il existe la mesure AUC (Area Under the Curve). Cette valeur étant discrète, elle peut dès lors être utilisée comme point de comparaison (Oufella, Adam, Chatelain, Heute, & Lecourtier, 2008, p. 22). Elle s'obtient par le calcul de l'aire couverte par la courbe ROC et on peut la décrire comme une « mesure générale de la capacité de prédiction » (Fawcett, 2006, p. 869).

Au niveau de KNIME, les mesures suivantes sont disponibles ;

Nœud KNIME	Mesures disponibles
Scorer	Rappel
	Précision
	Sensitivité
	Spécificité
	F-mesure
	Exactitude
	Kappa de Cohen
ROC Curve	AUC

Tableau 4 Mesures disponibles avec les nœuds KNIME.

CHOIX DE LA MÉTHODE D'ÉVALUATION

En regard des différents éléments amenés précédemment, tant sur le plan de l'évaluation de la performance que des mesures à disposition, il a été choisi de comparer les modèles à l'aide du score AUC. Il s'agit en effet de la mesure la plus efficace pour le problème traité, le coût de mauvaise classification n'étant pas connu, qu'il s'agit d'un problème impliquant deux classes uniquement et surtout que les données présentent un fort déséquilibre des classes.

MODÉLISATION

Dans ce chapitre, une revue des méthodes de modélisation est effectuée afin de déterminer leurs forces et faiblesses et ainsi pouvoir effectuer les meilleurs choix dans la phase de mise en œuvre.

Le nombre d'algorithmes à disposition est important et ils sont subdivisés en deux grandes familles ; la première regroupe ceux permettant une interprétation des données sous-jacentes, alors que la deuxième, qui peut parfois donner de meilleurs résultats, ne livre que ces derniers. Ces deux divisions sont respectivement appelées « transparente » et « opaque » ci –après, car les algorithmes faisant partie de la famille « opaque » sont parfois désignées sous le terme « black box models » (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 10).

Dans les méthodes de modélisation « transparentes », on trouve l'arbre de décision, la classification Bayésienne, la régression et l'approche basée sur les règles. A l'opposé, celles qui sont considérées comme « opaques » sont basées sur les plus proches voisins, les réseaux de neurones, les machines à support de vecteur (SVM, Support Vector Machines), et les méthodes d'ensemble.

ARBRE DE DÉCISION

L'arbre de décision va chercher à classer les données selon une structure hiérarchique. Il s'agit d'une méthode peu sensible au bruit et qui ne requiert donc pas nécessairement une normalisation. Son objectif est la visualisation des dépendances entre les données. Toutefois il est sensible aux changements et présente donc une grande fragilité si les données varient, par exemple entre les valeurs collectées pour l'apprentissage et le système de production (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 245). Afin de réduire la sensibilité des arbres de décision vis-à-vis des données, il est possible de les utiliser sous forme d'une forêt (ou ensemble) qui entraînera plusieurs modèles sur des sous-ensembles de données, puis regroupera les résultats.

L'implémentation de l'algorithme dans KNIME est celle popularisée par Ross Quinlan et connue sous l'appellation « Quinlan c4.5 »[16]. Cette implémentation va chercher à déterminer l'impureté d'une branche par rapport à une classe ; la capacité de séparation se base sur le calcul de « l'entropie relative », séparant en deux groupes chaque attribut (Berthold, Borgelt, Höppner, & Klawonn, 2010, pp. 211-215). Dans cette implémentation, l'impact des valeurs manquantes n'est pas un problème et est peu coûteux (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 217).

CLASSIFICATION BAYÉSIENNE

Le classificateur de Bayes est un modèle de base, pas toujours performant, mais qui peut se révéler utile pour se faire une idée sur le potentiel de détection et les relations entre les données, du fait de l'utilisation de simples probabilités. Globalement, tout modèle plus complexe doit être capable d'obtenir des résultats meilleurs. En effet, le classificateur bayésien se trouve limité dans sa capacité à mettre en lumière des relations complexes, car il se base sur l'hypothèse que les attributs sont indépendants (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 208). Toutefois, la classification Bayésienne fonctionne de la même manière indépendamment de la quantité de données d'entraînement, ce qui la rend opérable dans une situation où peu de données sont disponibles.

RÉGRESSION

Cette approche cherche à minimiser les erreurs entre les valeurs prédites et réelles, par exemple en cherchant la plus faible somme des carrés de la distance entre les valeurs prédites et celles observées (technique des « moindres carrés ordinaires » pour une régression linéaire). Elle permet de mettre en lumière les interactions entre les attributs et la valeur trouvée, car l'accès à chaque coefficient attribué à ces derniers est possible (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 208). Une limitation importante de ces algorithmes est leur besoin de valeurs numériques.

[16] https://en.wikipedia.org/wiki/C4.5_algorithm

MODÈLE DE RÈGLES

Cette manière de procéder va chercher à trouver les règles générales pouvant expliquer les données, mais il n'existe à ce jour pas d'algorithme pouvant s'appliquer de manière générale. Une approche consiste, par exemple, à reprendre les valeurs produites par un arbre de décision, de les formuler sous forme de règles, puis de réduire ces dernières à celles couvrant un maximum de données (Berthold, Borgelt, Höppner, & Klawonn, 2010, pp. 245-246).

Bien que l'établissement de règles représente la manière la plus naturelle pour l'être humain de grouper des éléments, il est vite très complexe de pourvoir déterminer quels sont les éléments ayant un réel poids, particulièrement dans un contexte dans lequel les données contiennent du bruit et des attributs hautement corrélés.

VOISINS PLUS PROCHES

Cette méthode se base sur la mesure de la distance entre deux éléments et est donc facilement compréhensible. L'appartenance d'un nouveau cas à une classe établie va être définie selon son positionnement vis-à-vis de cette dernière. Il existe plusieurs manières de calculer la distance, mais également pour établir le point servant de mesure, par exemple s'il faut considérer le centre ou la limite périphérique d'un élément. Ainsi cette approche ne cherche pas à modéliser les relations entre les attributs et ne donne donc pas d'indication sur les éléments qui vont influencer la construction des classes (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 259).

La première difficulté liée à cette approche est que la comparaison à un seul et unique voisin ne fait pas de sens ; aussi le choix du nombre de voisins à considérer est extrêmement important : une quantité trop faible rend l'algorithme peu robuste, tandis que dans le cas opposé, les limites des classes deviennent floues (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 262). La deuxième réside dans l'incapacité à établir des classes représentatives dans un contexte ne comptant pas beaucoup de données.

Enfin, cette méthode est résistante aux valeurs extrêmes et ce, d'autant plus si les distances sont pondérées.

RÉSEAUX DE NEURONES

Cette approche s'appuie initialement sur une tentative de définir des modèles flexibles et apprenants en s'inspirant des structures biologiques présentes dans la nature (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 269).

Le type le plus populaire est le MLP (Multi Layer Perceptron) qui agit comme une évaluation hiérarchique, chaque couche agissant comme filtre, réduisant ou amplifiant le « signal ». Le perceptron désigne ici le type de réseau de neurones le plus simple. Le MLP se base ainsi sur le même mécanisme que les neurones ; ces derniers forment un réseau transmettant un signal électrique et agissant comme inhibiteur ou excitateur sur celui-ci. Par rapport à un seuil donné ainsi que les conditions chimiques du point de connexion un signal, de type excitateur ou inhibiteur, sera transmis aux éléments connectés. L'implémentation du MLP reprend ce principe en définissant une couche d'entrée, un certain nombre de couches cachées (qui ne sont pas influencées par l'environnement) et une couche de sortie.

Le fonctionnement même - en multiples couches - de ce genre d'approche, limite fortement la capacité de déterminer quel impact a un attribut soumis en entrée sur le résultat produit en sortie. Ceci peut augmenter le risque de produire un modèle sur-ajusté aux données de test (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 260).

Machines à support de vecteur

Les méthodes de régression, telles que succinctement décrites ci-dessus, sont limitées dans leur capacité à séparer des classes présentant des frontières complexes. Les machines à support de vecteur permettent donc de dépasser ces limitations en projetant dans un espace de plus grande dimension les données, afin de pouvoir utiliser des classificateurs linéaires (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 278). Les machines à support de vecteur sont donc globalement efficaces, mais de par leur projection dans des dimensions d'ordre plus grand ne permettent pas de comprendre les mécanismes définissants la création des classes. En outre, le choix de la fonction noyau permettant d'effectuer la séparation des classes – aussi désignée comme fonction « kernel » -, est crucial et peut être lié au domaine de recherche.

Méthodes d'ensemble

Cette dernière approche se base sur la combinaison de différentes méthodes décrites dans les pages précédentes - s'appuyant, si possible, sur des propriétés distinctes - et sur une manière de combiner les prédictions obtenues. La construction de tels ensembles vise une amélioration des performances et va logiquement faire diminuer la capacité de compréhension des attributs ayant une influence sur ce dernier.

Choix de la méthode de modélisation

Après avoir passé en revue les algorithmes présents dans les deux familles, il apparaît que ceux appartenant à la première, catégorisée comme « transparente », vont être utiles dans la première phase de découverte des données. La deuxième pourra fournir des méthodes éventuellement plus efficaces pour la suite du processus. A cette étape de la démarche il n'est pas possible de fixer un choix sur l'algorithme, celui-ci sera fait en « Itération 1 »

MÉTHODOLOGIE DE MISE EN ŒUVRE

Selon le panorama dressé au préalable, tant sur le plan des techniques mises en œuvre pour résoudre les problèmes de détection de chutes, que sur celui des processus et des outils à disposition, il a été décidé de mettre en œuvre une analyse exploratoire des données en segmentant la démarche en itération. Chacune d'entre elles définissant au moins un objectif principal et éventuellement des sous-objectifs, mais considérant, de manière générale, que tout objectif découle du résultat de l'action précédente.

Pour la première itération, les valeurs en entrée sont le produit de la phase initiale de compréhension du contexte, menée tant sur le plan du métier que sur celui des données. Au niveau de chaque itération, une première section présente les éléments s'appliquant de manière générale aux différentes catégories étudiées et détaille les éventuelles hypothèses émises. Pour celles-ci, un résumé des démarches entreprises est présenté, comprenant notamment des informations sur l'implémentation. Pour terminer, une section conclut l'itération synthétisant les résultats obtenus et présentant les pistes à poursuivre dans la prochaine. Le matériel détaillé contenant les données et informations enregistrées et produites est disponible séparément dans les annexes afin de ne pas alourdir inutilement le présent rapport.

Le dernier sous-chapitre de la section « Mise en œuvre », intitulé « Déploiement » présente le système final proposé, tenant compte du résultat de toutes les itérations, ainsi que des contraintes ou souhaits mentionnés en amont.

De manière générale, les chapitres suivants reflètent, assez fidèlement, les différentes étapes traversées au cours de l'analyse, pour déboucher sur la proposition du système qui représente l'aboutissement et la synthèse des étapes précédentes.

MISE EN ŒUVRE

Dans la méthodologie CRISP-DM, il est recommandé de définir le contexte d'application selon les axes suivants (Chapman, et al., 2000, p. 7) :

- Domaine d'application,
- type de problème de forage,
- aspect technique,
- outils et technique.

Pour le projet courant, les éléments suivants ont été identifiés. Ils ne s'appliquent que pour le présent travail et n'ont pas vocation à produire un processus générique pour le domaine :

Domaine d'application	Type de problème	Aspects techniques	Outils et technique
Détection des chutes.	Classification, deux classes.	Capteurs inactifs.	KNIME
		Fréquence d'enregistrement non stable.	SQL (MySQL)
		Classes non- équilibrées.	Groovy
		Données à « pivoter » pour un traitement en séries temporelles.	Excel + Excel Power Query

Tableau 5 Outils et techniques utilisées selon les domaines d'applications.

Pour rappel, voici les six phases de la méthodologie CRISP-DM (Chapman, et al., 2000, pp. 10-11) détaillées :

Phase	Nom	Objectifs
1	Compréhension du métier	Définition des objectifs et des besoins d'un point de vue métier. Conversion de ces derniers en un problème de forage et établissement d'un plan initial.
2	Compréhension des données	Récolte et exploration des données pour se familiariser avec ces dernières. Formulation et validation d'éventuelles hypothèses.
3	Préparation des données	Sélection, nettoyage et construction du jeu de données à utiliser pour l'élaboration du modèle.
4	Modélisation	Application, y compris affinement des paramètres, de différentes techniques de modélisation.
5	Évaluation	Passage en revue des étapes nécessaires à la production du ou des modèles et de l'adéquation vis-à-vis des besoins métiers primairement identifiés. Cette phase débouche sur la décision de terminer le processus de recherche de modèle ou de poursuivre.
6	Déploiement	Mise en œuvre du modèle choisi.

Tableau 6 Les six phases détaillées du processus CRISP-DM. Source : (Chapman, et al., 2000)

COMPRÉHENSION DU MÉTIER

D'un point de vue métier, l'objectif est de mettre en place un système de détection des chutes utilisant un dispositif d'enregistrement des mouvements placé au poignet, éventuellement couplé avec un appareil effectuant le traitement des données. La finalité du projet est d'obtenir une capacité de détection maximale, tout en minimisant les fausses alertes, ceci à l'aide de matériel disponible sur le marché, abordable et non stigmatisant.

Au niveau des ressources, du côté du mandant, l'équipe composée de Julien Torrent, Mohamed Trikit, Célin Chassot et de Romain Edelman s'est montrée disponible pour fournir toutes les explications nécessaires et même une version revue et adaptée des données.

Le matériel mis à disposition comprend l'analyse du contexte effectué par la FST, la présentation du projet, la définition des types de chutes (nomenclature), ainsi que les données comprenant des activités normales et des chutes simulées. En outre, les éléments de recherche ainsi que les résultats de la détection par seuils obtenus par le TaM (Université de Genève) ont été fournis.

L'aspect des coûts est couvert par le budget du projet, pour un montant de 542 840 CHF. Quant à la partie bénéfice, elle n'est pas quantifiable aussi précisément. Toutefois il faut noter que les différents éléments suivants sont des bénéfices qui seront produits par le système F2D : tout d'abord la capacité d'intervention rapide en cas de chute permettra de réduire les séquelles, voire les cas de décès, donc par-là, les frais liés à l'hospitalisation ou aux moyens auxiliaires supplémentaires ; ensuite, une confiance accrue de la part des personnes âgées et de leurs proches rendra envisageable un maintien prolongé à domicile réduisant les coûts inhérents au placement en établissement médico-social (Torrent, et al., 2015).

En termes de calendrier, le travail se déroule entre le 01.04.2015 et le 31.08.2015 et les livrables produits seront, d'une part le rapport de travail, et d'autre part le modèle de détection le plus abouti sous un format exploitable dans l'application F2D.

D'un point de vue du forage des données, le plan est de déterminer, empiriquement, comment maximiser la détection de chutes, en limitant les contraintes dans un premier temps. Il s'agit donc de la capacité à classifier un événement, composé d'une série d'enregistrements provenant des capteurs du terminal comme étant un cas de chute, respectivement de non chute. Le choix de se focaliser sur les cas de chutes molles a été motivé et justifié plus haut.

COMPRÉHENSION DES DONNÉES

Les données qui ont été enregistrées par la FST et fournies brutes présentent des disparités ; les fréquences d'enregistrement ne sont pas constantes, les valeurs des trois capteurs – l'accéléromètre, l'accélération linéaire et le gyroscope - pas toujours présentes, et les plages enregistrées sont de longueurs variables. En outre, il faut noter que les cas de chutes ainsi qu'une partie des activités normales sont des données de laboratoire ne couvrant bien évidemment pas la totalité des événements possibles.

Afin de pouvoir mieux appréhender le matériel transmis, une base de données MySQL a été créée. Les avantages de placer les informations dans une structure SQL sont multiples ; tout d'abord le langage SQL fournit des méthodes d'agrégation permettant de produire des statistiques sur les données dans un environnement existant. Il est composé de plusieurs outils et d'un langage qui permettent d'interroger la base sans besoin de créer des programmes ad-hoc. La totalité des informations se trouve ainsi réunie en un et seul même lieu physique. En outre, le schéma de la base renforce la cohérence des types de données et offre un cadre de normalisation. De plus, il est possible de produire facilement des fichiers agrégés au format CSV (Comma-Separated Values)[17], par exemple, et de les exploiter par la suite dans Excel ou dans KNIME. Pour terminer, il est possible de connecter la source de données à KNIME directement. Plus de détails concernant la modélisation de la base de données, ainsi

[17] https://en.wikipedia.org/wiki/Comma-separated_values

que le chargement de contenu dans de cette dernière, sont disponibles dans le chapitre « Préparation des données ».

La compréhension des données s'est également faite au moyen du tableur Microsoft Excel avec les extensions « Power Query »[18] et « MySQL »[19] permettant d'extraire les informations de différentes sources, de les manipuler et d'en produire des représentations graphiques.

La dimension temporelle (timestamp) des données enregistrées par les senseurs provient de la méthode System.currentTimeMillis[20]. Cette méthode retourne le temps courant par rapport à la valeur 0 (1er janvier 1970). Ce choix a été motivé par le besoin de pouvoir connaître la date d'enregistrement. Toutefois au vu de l'utilisation de cette donnée dans le cadre de la détection, la méthode System.nanoTime[21] - qui fournit une estampille plus précise - aurait pu être utilisée, car il ne s'agit pas de mesurer un temps absolu, mais écoulé[22].

Voici un tableau présentant les différentes données à disposition et leurs caractéristiques :

Désignation	Format	Valeurs possibles	Type
Type de chute	Alphanumérique, 2 symboles.	B1, B2, B3, M1, M2, M3, NO, FR	Catégorie
Sujet	Alphabétique, 2 caractères	AA, BA, BO, FA, GU, KU, LA ; NI, PI, TO, TR, UN, VA	Catégorie
Age	Numérique, 2 digits	[22, 93]	Continu
Sexe	Alphabétique, 1 caractère	F, M	Catégorie
Moyen auxiliaire	Alphanumérique, 2 symboles	00, CA, DR, DS	Catégorie
Accélération linéaire, axe des X.	Numérique	[-34.86532974243164, 33.24461364746094]	Continu
Accélération linéaire, axe des Y.	Numérique	[-44.05729675292969, 42.206565856933594]	Continu
Accélération linéaire, axe des Z.	Numérique	[-29.68331527709961, 35.54317855834961]	Continu
Accélération, axe des X	Numérique	[0,0]	Continu
Accélération, axe des Y	Numérique	[-39.08319854736328, 39.369998931884766]	Continu
Accélération, axe des Z	Numérique	[-38.91899871826172, 39.82899856567383]	Continu
Gyroscope, axe des X	Numérique	[-38.91109848022461, 39.54209899902344]	Continu
Gyroscope, axe des Y	Numérique	[-39.08319854736328, 39.369998931884766]	Continu
Gyroscope, axe des Z	Numérique	[-13.002599716186523, 14.506699562072754]	Continu

Tableau 7 Nature des données fournies par la FST

Les données collectées sont de trois types différents, soit l'accélération linéaire, l'accélération et le gyroscope. Ces deux dernières proviennent de capteurs physiques, alors que la première, l'accélération linéaire, est une valeur calculée, obtenue en soustrayant la gravité terrestre (9.81 m/s^2) à l'accélération enregistrée par l'accéléromètre[23].

[18] https://support.office.com/en-us/article/Introduction-to-Microsoft-Power-Query-for-Excel-6e92e2f4-2079-4e1f-bad5-89f6269cd605
[19] https://www.mysql.com/why-mysql/windows/excel/
[20] http://developer.android.com/reference/java/lang/System.html#currentTimeMillis%28%29
[21] http://developer.android.com/reference/java/lang/System.html#nanoTime%28%29
[22] https://blogs.oracle.com/dholmes/entry/inside_the_hotspot_vm_clocks
[23] http://developer.android.com/guide/topics/sensors/sensors_motion.html

Les enregistrements ont été réalisés au moyen de deux dispositifs différents ; à savoir les smart Watches LG G[24] et Moto 360[25].

Selon la documentation officielle d'Android[26], le système de coordonnées des axes est définit relativement à l'écran du téléphone, l'axe des abscisses (X) pointant vers la droite, celui des ordonnées (Y) vers le haut et celui des Z traversant le dispositif.

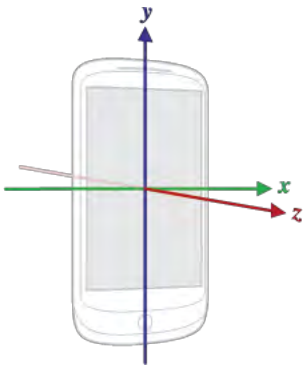


Figure 7 Axes d'enregistrement des données. Source : http://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-coords

QUALITÉ DES DONNÉES

Pour commencer, les 1075 enregistrements concernant les activités quotidiennes collectés à l'aide du dispositif placé à la hanche sont écartés, car le présent travail se concentre sur la détection à l'aide d'une smart Watch. En outre, il n'existe pas de chutes - brutales ou molles - enregistrées avec ce moyen de capture.

Pour les chutes molles, il existe 475 jeux de données dont le début et la fin sont identifiés – labellisés - sur un total de 3126 cas soit 15.19% des jeux. Les chutes comptent pour 145 865 enregistrements sur 12 990 094, ce qui représente un taux de 1.12%. On est donc dans une situation de déséquilibre important entre les classes. De manière globale, les jeux de données fournies comptent un nombre moyen d'enregistrements de 4188.76 et ont une durée moyenne de 24.53 secondes.

Cent vingt-trois jeux ayant une durée de moins d'une seconde et concernant des activités de type « Freestyle » ou « Normale » ont été supprimés car inexploitables. Ces cas sont connus par la FST qui a constaté des problèmes lors de la capture des données avec, parfois, un système se trouvant dans un état de blocage partiel, les valeurs retournées par les capteurs n'étant pas mises à jour.

[24] https://support.google.com/androidwear/answer/6056447?hl=en
[25] https://support.google.com/androidwear/answer/6088867?hl=en
[26] http://developer.android.com/reference/android/hardware/SensorEvent.html

Au niveau des fréquences d'enregistrement, ces dernières ne sont pas régulières et de grandes variations sont présentes, comme il est possible de le voir dans le graphique ci-dessous, et ce, malgré la configuration de l'application Android collectant les données à une valeur de 50 Hz (Torrent, et al., 2015, p. 18). Outre le graphique, les valeurs scalaires suivantes ont été calculées : la fréquence moyenne est de 989.32 Hz et la médiane 199.5975799560547 Hz. La borne supérieure des fréquences est de 35 714.28515625 Hz, et la borne inférieure de 4.152244567871094 Hz.

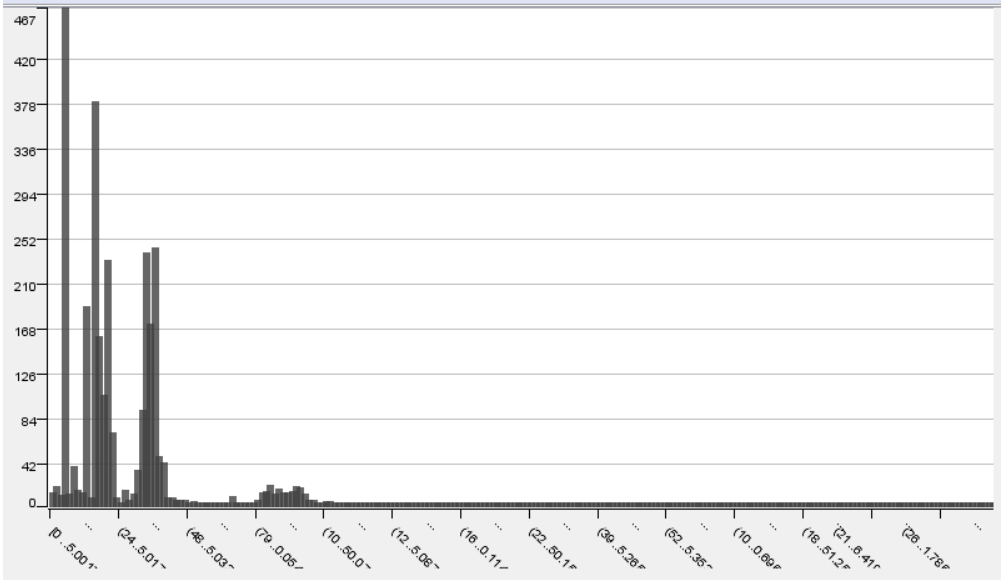


Figure 8 Répartition des fréquences d'enregistrement

L'absence d'une dimension temporelle connue ne permet pas le traitement des suites de données et doit être impérativement corrigée. Cette opération d'alignement des séquences a été réalisée par Mohamed Trikit du FSTLab à l'aide d'un programme en Python. Suite à cette correction, il a été possible de poursuivre avec des données de bonne qualité.

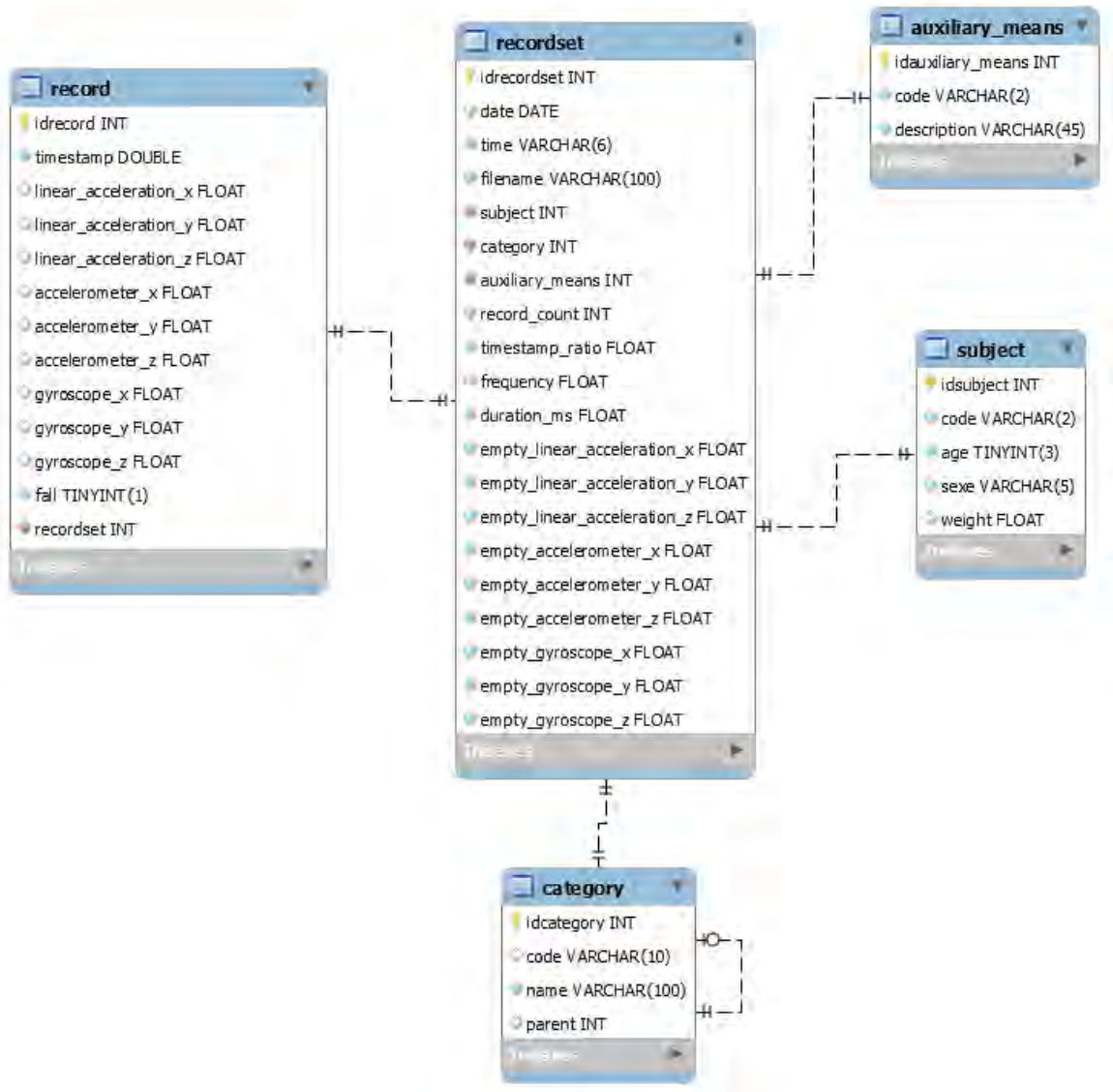
PRÉPARATION DES DONNÉES

Comme expliqué plus haut, le contenu des fichiers CSV fournis par la FST a été importé dans une base MySQL pour faciliter manipulation et compréhension. Ce chargement a été effectué à l'aide de quatre scripts Groovy[27] créés pour l'occasion. Ce langage a été choisi, car il permet une manipulation aisée – à l'aide de méthodes sur des collections d'instances d'objets - des éléments extraits depuis un fichier CSV pour les insérer dans la base de données. Outre l'import, un script a également été produit pour marquer – labelliser - les enregistrements ayant été identifiés comme appartenant à une séquence de chute. Au final, la base de données contient plus d'1 Go de données.

[27] http://www.groovy-lang.org/

La modélisation du schéma de base de données s'est faite en suivant les bonnes pratiques en vigueur, en identifiant les différentes entités présentes et en les liant à l'aide d'identifiants techniques quand nécessaire. Ainsi, la notion centrale est le « recordset », ou jeu d'enregistrement, toutes les autres valeurs gravitant autour de cette dimension. Les catégories d'activités sont détaillées et leur relation hiérarchique est caractérisée à l'aide d'une clef étrangère sur la même table. Outre les données fournies, un certain nombre de champs – record_count, timestamp_ratio, frequency, duration_ms, etc.-, calculés après l'insertion initiale, ont été ajoutés pour obtenir des informations supplémentaires.

Figure 9 Schéma initial de base de données.



Les jeux de données utilisés dans les différentes itérations sont toujours des exports de la base de données sous format CSV. De manière générale, seuls les enregistrements des activités normales - dont le code commence par N - et ceux des chutes molles - dont le code commence par M - identifiés comme chute - dont la valeur de la colonne « fall » est égale à un - sont inclus.

Comme expliqué précédemment, le choix de ne conserver que les chutes molles vient tout d'abord du besoin de focaliser la recherche sur un aspect pour lequel l'algorithme mis en place par le TaM donne les résultats les moins bons et deuxièmement, afin de pouvoir dans le temps imparti mettre en œuvre les six phases du processus CRISP-DM. La capacité de détection de l'approche TBM du TaM est de 84% pour les chutes brutales est nulle pour les cas de chutes molles, comme le montre le graphique ci-dessous.

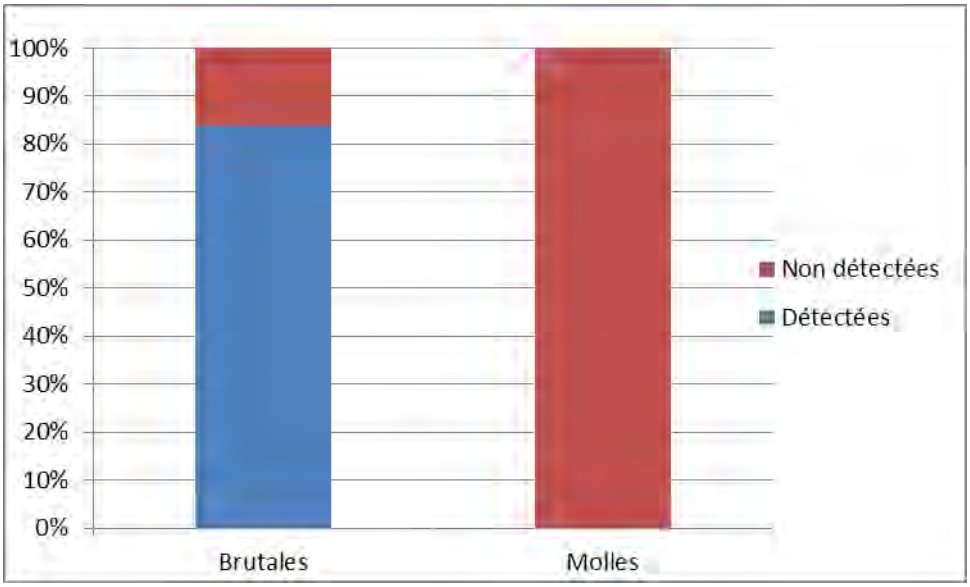


Figure 10 Résultats de détection obtenus à l'aide de la méthode des seuils. Source : FST

Outre la volonté de ne traiter que les chutes molles, il a été choisi de ne pas faire de distinction entre les différentes typologies de chutes, mais de se limiter à l'aspect binaire, à savoir un état « chute » ou « non chute ». Ceci rend le modèle moins complexe tout en restant aligné avec l'objectif final fixé par la FST, à savoir d''être capable de détecter la survenance d'un accident.

Les valeurs des senseurs utilisées pour la détection sont celles provenant de l'accéléromètre linéaire pour plusieurs raisons. La première est liée à la disponibilité des données. En effet, une part importante - 2030 sur 3126, soit 64.9% - des enregistrements ne contient pas la totalité des valeurs des trois senseurs, même en fixant un taux maximal de 20% d'enregistrements non conformes. La deuxième s'appuie sur les constatations faites par la FST qui identifie ces données comme étant les plus prometteuses (Torrent, et al., 2015, p. 18), mais également sur le fait que cette approche a été choisie, entre autres, par Wang et Bai (2013, p. 618). Enfin, seul trois senseurs ont été retenus, afin de limiter la quantité de données à traiter, car cela représente une part importante dans le calcul du coût du modèle et son déploiement dans une plateforme mobile.

ITÉRATION 1

CONTEXTE

Dans cette première itération, divers objectifs ont été définis ; premièrement, le but a été d'évaluer les potentielles informations présentes au sein de données, puis deuxièmement, de choisir l'algorithme le plus adapté. Pour rappel, le score AUC est utilisé pour l'évaluation des modèles, les autres mesures, sensibles à un non équilibrage des classes, ne peuvent s'appliquer dans le cas présent en raison de la surreprésentation des activités normales par rapport aux chutes molles.

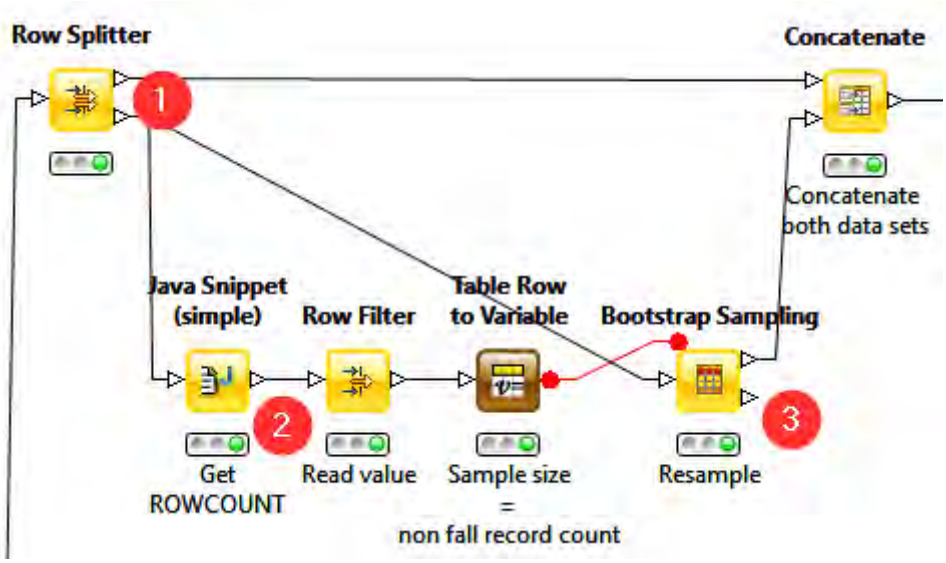
Dans la volonté de trouver des motifs sans dépenser trop de temps dans cette phase, ceci afin de mettre l'accent sur l'efficacité, les données ont été considérées « brutes », c'est-à-dire sans dimension temporelle.

De manière générale, les données ont été extraites de la base MySQL à l'aide d'instructions SQL et stockées sous format CSV ; tous les enregistrements pour les activités de type « normal » et « chute molle » sont considérés, ce qui représente 1 090 946 lignes. Cette manière de faire, à savoir l'export, a été choisie afin de limiter le nombre de nœuds nécessaires pour la préparation du jeu de données nécessaires dans KNIME. La colonne stockant le label de la chute ou de non chute est convertie en chaîne de caractères pour permettre son exploitation par un algorithme d'arbre décisionnel. La séparation des données entre le jeu d'entraînement et de test est faite de manière simple en utilisant des valeurs relatives. Les classes présentes dans les données d'entraînement sont équilibrées afin de s'affranchir des faiblesses de certains algorithmes, tel que l'arbre de décision, lors de leur exploitation dans des situations de déséquilibre. Ce dernier est présent au niveau des cas de chutes molles. En conséquence, des données additionnelles représentant ces activités sont générées à l'aide de la méthode de « boostrapping »[28]. Au niveau de KNIME, ce concept est disponible avec le nœud « Bootstrap Sampling », basé sur la méthode « bagging » (Boostrap aggregating)[29] développée par Breiman (Bagging Predictors, 1994).

[28] http://en.wikipedia.org/wiki/Bootstrapping_%28statistics%29
[29] http://en.wikipedia.org/wiki/Bootstrap_aggregating

L'implémentation se fait tout d'abord en séparant les données entre les deux classes (1), puis la population la plus petite – déterminée au préalable – se voit augmenter pour atteindre la même taille (2) que la classe contenant le plus d'enregistrements ; le choix des valeur se faisant grâce au nœud « Bootstrap Sampling »(3). Les données sont ensuite agrégées pour produire le matériel utilisé pour l'établissement du modèle. Il faut noter que cette approche va produire une population totalement équilibrée.

Figure 11 Détail de l'implémentation du « bootstrapping » dans KNIME.



Les algorithmes suivants ont été testés :

- Arbre de décision,
- classificateur naïf de Bayes,
- K plus proches voisins,
- réseau de neurones,
- et SVM.

Les deux premiers algorithmes produisent des résultats interprétables et dont la logique de production est identifiable, par exemple en visualisant l'arbre généré. Ils sont donc tout désignés pour définir le point de départ et remplir l'objectif d'identification de motifs. Les trois autres approches sont de type « boîte noire », limitant l'interprétation du modèle mais augmentant généralement sa capacité (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 84).

Les attributs utilisés sont les valeurs de l'accéléromètre sur les trois axes, c'est-à-dire :

- linear_acceleration_x,
- linear_acceleration_y,
- et linear_acceleration_z.

Arbre de décision

Pour cet algorithme, plusieurs paramètres ont été considérés dans le but d'augmenter sa capacité de classification :

- La mesure de qualité,
- le nombre minimal d'enregistrements par nœud,
- la méthode d'élagage.

La mesure de qualité va définir à l'aide de quel algorithme, ici « gain ratio » ou « Gini index », va être déterminée l'appartenance d'une valeur à un ensemble. Ceci va influer sur la création des branches. Dans le cadre actuel, la détermination de la qualité à l'aide du « gain ratio » est plus efficace.

Modifier le nombre minimal d'enregistrements par nœud va forcer l'arbre à posséder des branches plus grosses en groupant les valeurs proches. Dans le cas présent, vu la grande diversité des données, ce paramètre a fortement joué sur la performance.

L'élagage MDL (Minimum Description Length), qui permet non seulement de réduire la taille de l'arbre, mais également d'éviter les problèmes de sur-ajustement en supprimant les branches n'apportant pas de grande valeur ajoutée (Mehta, Rissanen, & Agrawal, 1995), a été utilisé et a permis d'augmenter la capacité du modèle. Tous les scores présentés ci-dessous utilisent l'élagage à l'aide de MDL.

Mesure de la qualité	Nombre minimal d'enregistrements par nœud	Score AUC
Gini Index	2	0.5928
Gini Index	100	0.7477
Gain ratio	100	0.7493

Tableau 8 Scores arbre de décision en fonction de la mesure de qualité et du nombre minimal d'enregistrements par nœud.

Le meilleur score AUC obtenu avec un arbre de décision seul est de 0.7493 en utilisant la méthode « gain ratio », une taille minimale de nœuds de 100 enregistrements et l'élagage MDL.

Ensemble d'arbres de décision

Les paramètres à disposition au niveau des arbres de décision uniques ayant été modifiés et le résultat obtenu ne pouvant être amélioré, la suite logique a été d'utiliser une combinaison d'arbres – ou ensemble d'arbres -, permettant ainsi d'offrir une plus grande stabilité au modèle (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 217). Un ensemble d'arbre va produire plusieurs modèles basés, chacun utilisant un sous-ensemble des données. Les différents modèles sont ensuite compilés en ne retenant que les éléments ayant obtenu le plus de voix.

Le seul paramètre avec lequel l'investigation a été menée est celui du nombre de modèles. En effet, celui du critère de séparation a déjà été retenu suite aux expérimentations faites avec un arbre de décision seul ; à savoir l'utilisation de l'option « information gain ratio ».

Le nombre de modèles a été varié entre 10 et 300 et, sans grande surprise, le score de détection augmente de concert avec ce paramètre. Le tableau ci-dessous offre une vue compacte des scores par rapport à la variation du nombre de modèles inclus.

Nombre de modèles	Score AUC
10	0.6759
100	0.7889
300	0.8118

Tableau 9 Scores obtenus en faisant varier le nombre de modèles pour une forêt d'arbres de décision.

Le meilleur score obtenu est 0.8118, ceci avec 300 modèles et un critère de séparation basé sur la mesure de gain d'information (« information gain ratio »).

Classificateur naïf de Bayes

Au vu de sa capacité limitée à trouver des relations complexes, l'algorithme de classification bayésienne n'a été utilisé qu'une seule fois, dans le but d'évaluer grossièrement les potentiels de détection dans les données. Vu la grande disparité des enregistrements, il n'a pas pu se montrer particulièrement efficace.

Avec un paramètre de probabilité de 0.1, le score obtenu est de 0.5427.

K PLUS PROCHES VOISINS

L'utilisation de l'algorithme KNN (K-Nearest Neighbors) montre les mêmes faiblesses que dans le cadre de l'arbre de décision seul, à savoir qu'en raison de la grande dispersion des données cela implique de considérer un grand nombre de voisins pour obtenir des scores suffisamment bons.

Pour les voisins plus proches, voici les paramètres étant intervenus lors de l'évaluation des différentes possibilités :

- le nombre de voisins considérés,
- le poids des voisins en fonction de leur distance.

Selon les différents scénarios, on constate globalement que plus le nombre de voisins est élevé, plus le taux de détection est élevé, et que la prise en compte de la distance des voisins influe également de manière importante.

Nombre de voisins	Poids selon la distance	Score AUC
3	Non	0.5986
7	Non	0.6413
7	Oui	0.6452
15	Non	0.6899
50	Non	0.7665
50	Oui	0.8215

Tableau 10 Scores KNN en fonction du nombre de voisins et de leur distance.

Le meilleur score – 0.8215 - a été obtenu avec 50 voisins en tenant compte de leur distance.

RÉSEAUX DE NEURONES

Le réseau de neurones le plus répandu, à savoir MLP a été utilisé, et les éléments suivants ont été pris en compte lors des différents essais :

- Le nombre de neurones,
- la normalisation des données.

Selon (Berthold, Borgelt, Höppner, & Klawonn, 2010, p. 273), une règle de base pour déterminer le nombre de neurones dans la couche cachée, en général unique, s'obtient de la manière suivante :

$$\frac{1}{2}(i+o)$$

Équation 3 Formule pour déterminer le nombre de neurones avec MLP.

Avec *i* représentant le nombre d'attributs en entrée et *o* ceux en sortie. Appliquer cette formule dans le cas présent donne un résultat de trois, or les essais menés concluent qu'un nombre plus grand de neurones donne de meilleurs résultats, ceci étant sans doute lié au sur-ajustement.

En outre, l'utilisation des données normalisées, bien que recommandée avec cet algorithme, péjore le score au lieu de l'améliorer. Pour finir, il a été constaté que l'augmentation du nombre de neurones augmente la capacité détective du modèle, comme montré dans le tableau suivant :

Nombre de neurones	Normalisation	Score AUC
3	Non	0.5964
3	Oui	0.5781
10	Non	0.6562
10	Oui	0.6489
15	Non	0.6585
15	Oui	0.6572
30	Non	0.6732
30	Oui	0.6554

Tableau 11 Score MLP en fonction du nombre de neurones et de la normalisation des données.

Le meilleur score obtenu avec le MLP est de 0.6732, ceci avec 30 neurones et sans appliquer de normalisation.

SVM

Le temps de calcul nécessaire à la mise en œuvre de cet algorithme s'est révélé trop long, aucun résultat n'ayant pu être fourni en 24 heures avec la configuration actuelle, ceci en raison du nombre important de données. Il a donc été écarté dans cette phase initiale.

Il faut toutefois noter qu'au vu des résultats relativement bons obtenus avec les autres méthodes, par exemple l'ensemble d'arbres de décision et KNN, la séparation des classes est possible sans avoir besoin de SVM.

Conclusion

Un certain nombre d'algorithmes ont été évalués afin de déterminer, tout d'abord, si les données fournies présentaient des motifs détectables, et deuxièmement quelle méthode est potentiellement la plus intéressante. Sur le premier point, les essais démontrent parfaitement que les enregistrements à disposition sont utilisables pour poursuivre dans la détection de chutes molles. Au niveau du deuxième objectif, deux algorithmes sortent du lot, à savoir « l'ensemble d'arbres de décision » et « K plus proches voisins ».

Algorithme	Meilleur score AUC
Arbre de décision	0.7493
Forêt d'arbres de décision	0.8118
Classificateur de Bayes	0.5427
K plus proches voisins	0.8215
Réseau de neurones	0.6732
SVM	-

Tableau 12 Liste des meilleurs scores obtenus par algorithme.

En ne tenant compte que des scores, le choix de l'algorithme va logiquement à « K plus proches voisins », toutefois il a été écarté de l'itération suivante. En effet, cet algorithme est efficace pour autant que la masse de données à disposition soit suffisamment importante, ce qui est le cas dans cette première itération, puisque le nombre d'enregistrements est supérieur au million. Or lors de la deuxième itération, ce nombre va tomber en dessous de 1000, l'algorithme perdant dès lors de sa capacité (Berthold, Borgelt, Höppner, & Klawonn, 2010). Cette assertion a été vérifiée en fin d'itération 3, l'algorithme KNN obtenant un score AUC inférieur de 12.91% à un ensemble d'arbres de décision, ce avec les mêmes valeurs pour les paramètres communs.

ITÉRATION 2

CONTEXTE

Le but de cette deuxième itération est d'améliorer la quantité d'informations trouvées en traitant les données sous format de séries temporelles. En effet, dans la première phase, la recherche du modèle de classification avait été faite en ne prenant en compte les valeurs de l'accélération linéaire que pour un point donné dans le temps. L'objectif est donc de déterminer quel est le nombre d'enregistrements optimal pour détecter une chute molle.

Comme pour l'itération précédente, les données ont été extraites, sous format CSV, de la base MySQL à l'aide d'instructions SQL. Une condition supplémentaire a été ajoutée afin de ne garder que les enregistrements labellisés comme chute pour les événements de type « chute molle ».

Pour pouvoir travailler sur une série temporelle, les données collectées sur les trois axes de l'accéléromètre doivent être transformées en un signal d'une seule dimension. Le SVM (Signal Vector Magnitude) permet de mesurer l'intensité des mouvements en dérivant des valeurs des trois axes collectées (Karantonis, Narayanan, Mathie, Lovell, & Celler, 2006) et a déjà été largement utilisé dans différentes études de chutes (Bianchi, Redmond, Narayanan, Cerutti, & Lovell, 2010), (Karantonis, Narayanan, Mathie, Lovell, & Celler, 2006), (Wang & Bai, 2013), (Bersch, Azzi, Khusainov, Achumba, & Ries, 2014), (Kau & Chen, 2015). Le vecteur contenant les données des senseurs est calculé à l'aide de la formule suivante :

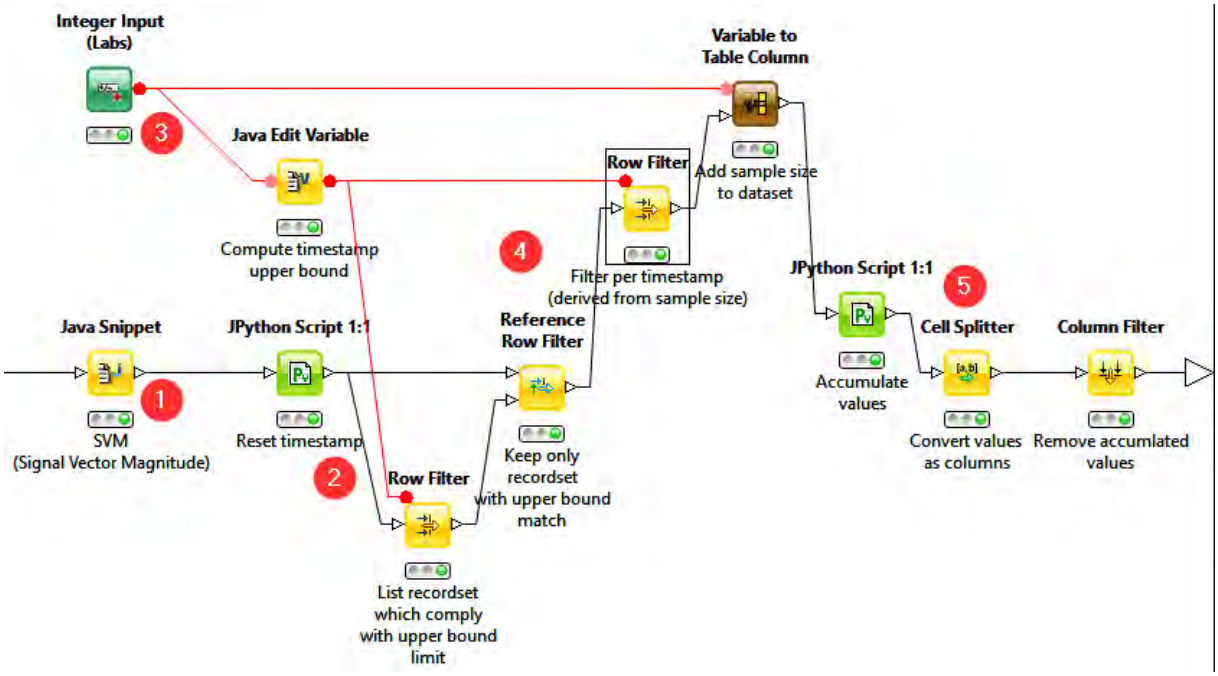
$$SVM = \sqrt[2]{linear\_acc\_x^2 + linear\_acc\_y^2 + linear\_acc\_z^2}$$

Équation 4 Formule pour calculer le SVM.

L'implémentation dans KNIME s'est faite à l'aide d'un nœud « Java Snippet », ce dernier offrant toute liberté pour programmer un traitement appliqué à chaque ligne d'une table.

Le format des données, tel que fourni, ne permet pas un traitement en série. En effet, chaque point de temps représente une ligne dans le fichier d'entrée. Or, pour pouvoir utiliser ces valeurs comme paramètres du modèle, il est impératif de pouvoir en disposer sous forme de colonnes, groupées par enregistrement. Cette phase de transformation a été réalisée avec un nœud « JPython Script », afin de pouvoir grouper les lignes par jeu d'enregistrements (5). Au préalable, les valeurs de l'estampille temporelle ont dû être réalignées (2) en raison du filtre mis en place lors de l'export des données depuis la base sur les chutes molles qui ne prend que les enregistrements labellisés. La capacité de paramétrage de la longueur de la série se fait en étape (3), puis les jeux d'enregistrements possédant une estampille temporelle équivalente à la borne supérieur définie sont filtrés (4). Bien entendu la tâche de transformation en signal SVM (1) est effectuée en amont.

Figure 12 Détail des modifications apportées sur les données dans KNIME pour pouvoir traiter les séries temporelles.



La troisième contrainte liée à l'exploitation de données temporelles a été d'inclure la capacité de définir le nombre de valeurs qui doivent être considérées afin de contrôler les attributs du modèle. La durée, donc le nombre des enregistrements utilisés pour la détection, est déterminée de manière empirique. Des tests ont étés effectués avec des valeurs allant de 10 à 400 enregistrements, ce qui correspond, pour des données collectées, à une fréquence de 50 Hz, à une durée s'étalant de 200 ms à 8 s. Le tableau ci-dessous présente les performances obtenues en utilisant un ensemble d'arbres de décision avec 1000 modèles :

Durée	Nombre d'enregistrements	% de chutes molles	AUC
200ms	10	46.64%	0.6855
2000ms	100	43.30%	0.8510
4000ms	200	23.70%	0.8993
8000ms	400	7.78%	0.8732

Tableau 13 Scores de classification en fonction des durées choisies.

On peut constater que les meilleurs résultats s'obtiennent pour les durées de 4 secondes, ce qui représente 200 enregistrements. Cette valeur a donc été retenue pour la suite des investigations.

Une analyse détaillée a été menée sur les 134 cas mal classés pour la durée de 4 secondes afin de déterminer la nature des erreurs. Elle a été réalisée à l'aide d'un fichier Excel selon plusieurs axes ; le premier est le calcul de la proximité du score obtenu par rapport au seuil de décision, fixé à 0.5. L'écart de +/-5% a été choisit comme limite,

ce qui fait que les scores compris ou égaux à l'intervalle entre 0.45 et 0.55 tombent dans cette catégorie. Le deuxième est la comparaison du signal, sous forme de graphique, d'un enregistrement représentant le même type d'activité mais ayant été classé correctement. Le troisième est la production d'un histogramme permettant une analyse du spectre.

Les trois approches ont permis de relever les différences notables produites ou représentées par les jeux incorrectement classifiés, mais également les éléments présentant des divergences importantes par rapport à des activités « normales ». Ainsi, neuf jeux ont été écartés, suite à leur revue par la FST, leurs valeurs montrant des incohérences par rapport à l'activité attendue.

Selon les résultats obtenus, les activités normales (dont le code est « NOM ») représentent la majorité des cas mal classés avec une part de 25.19%, comme présenté dans le tableau ci-dessous.

Activité	% de cas mal classés
NOC	6,67%
NOED	9,63%
NOEM	5,19%
NOM	25,19%
M1AR	5,19%
M1AV	11,85%
M2AR	11,11%
M2AV	11,11%
M3AR	6,67%
M3AV	6,67%
	99,26%

Tableau 14 Répartition des cas mal classés en itération 2 selon le type d'activité.

Conclusion

Le changement des paramètres du modèle, en passant des trois valeurs collectées par l'accéléromètre aux 200 points de la série temporelle, ont permis d'augmenter significativement la capacité détective du modèle, en passant de 0.8118 à 0.8993, soit une augmentation de 8.75%. En outre, l'analyse poussée des jeux de données a permis de déterminer le point d'amélioration pour l'itération suivante.

ITÉRATION 3

CONTEXTE

Suite à l'analyse des erreurs de classification rencontrées dans l'itération précédente, il apparaît que la majorité de ces dernières ne peut pas être attribuée à des problèmes de mesure. Au contraire, cette revue a permis de déterminer un choix d'optimisation se portant sur les activités générant des signaux répétitifs, telles que la course ou la marche.

Pour traiter ces cas, la méthode consiste à mener une analyse spectrale de la série temporelle à l'aide d'une technique parmi les suivantes : la transformée de Fourrier discrète (DFT, Discrete Fourier Transform), la transformée en ondelettes discrète (DWT, Discrete Wavelet Transform), la décomposition en valeurs singulières (Zhu, 2004) (Korn, Jagadish, & Faloustos, 1997), l'approximation par segments constants de longueur adaptés (APCA) (Keogh, Kaushik, Pazzani, & Mehrotra, 2001), ou encore l'approximation par segments linéaires (PLA) , (Morinaka, Yoshikawa, Amagasa, & Uemura, 2001). Parmi les différentes techniques citées, les plus abordables sont les transformées de Fourrier et en ondelettes, raison pour laquelle ces dernières sont retenues. Le choix de comparer deux approches vient du fait que selon Chan & Fu (Efficient TIme Series Matching by Wavelets, 1999), la DWT est plus efficace que la DFT, une assertion qu'il a été décidé de vérifier dans le cas présent.

La méthode la plus ancienne est celle de la transformée de Fourrier et son évolution est incarnée par la transformée en ondelettes. Il s'agit de techniques permettant de conserver l'information véhiculée dans les données quasiment sans perte (Zhu, 2004). Ces deux approches peuvent être mises en œuvre dans KNIME à l'aide de la bibliothèque Java « jWave Toolbox »[30] de manière assez triviale à l'aide d'un nœud « Java Snippet ». Les deux algorithmes utilisés pour produire les transformées sont la FFT (Fast Fourier Transform) et la FWT (Fast Wavelet Transform) respectivement qui sont des méthodes efficaces de transformation permettant de réduire la complexité du calcul.

Pour les deux cas, la résolution du signal est augmentée en doublant sa longueur, les intervalles nouvellement créés étant remplis avec des zéros (Zero Padding Applications, 2015).

Dans le cadre d'une transformée – qu'elle soit de Fourrier ou en ondelettes -, il va falloir déterminer le nombre de coefficients à conserver pour analyse. Dans le cas présent, le choix de la meilleure valeur va se faire de manière empirique. Selon (Agrawal, Faloustos, & Swami, 1993), un nombre de coefficients inférieurs à cinq est suffisant pour obtenir des résultats de bonne qualité. Toutefois, suite aux expérimentations faites, il apparaît que ce nombre doit être plus élevé, comme le montre le tableau ci-dessous ;

	Score (précision)	
Coefficients	FFT	FWT
4	31.429	33.257
5	34.857	36.477
6	37.943	37.371
7	38.857	36
8	39.086	37.029
9	40.343	37.371
10	41.6	37.143
25	37.6	39.314
50	42.743	42.4
75	41.257	42.4
125	41.143	42.057

Tableau 15 Résultats de détection obtenus avec un nombre donné de coefficients.

[30] https://code.google.com/p/scalalab/downloads/detail?name=jWave.jar&can=2&q=

La mesure de comparaison est ici la précision, car le score AUC n'est pas disponible pour une évaluation multi classe, la classification étant fait selon les activités. Pour ce faire, il faudrait utiliser la notion de VUS (Volume Under the ROC Hyper Surface) (Landgrebe & Duin, 2006) (Oufella, Adam, Chatelain, Heute, & Lecourtier, 2008, pp. 28-33), valeur non fournie dans KNIME directement et donc pas aisément exploitable dans le présent cas.

On constate tout d'abord que de manière générale, la transformée de Fourrier produit dans presque tous les cas de meilleurs résultats que la transformée en ondelettes. Ensuite, la précision de classification augmente jusqu'au nombre de 50 coefficients puis diminue lentement au-delà.

Au vu des résultats obtenus, deux approches ont été ensuite détaillées ; la première détermine quelles sont les activités qui peuvent être écartées dans une première phase à l'aide d'un modèle spécifique, puis les cas résultants sont soumis à un deuxième modèle – il s'agit ici de classification en cascade - . La deuxième consiste à augmenter les paramètres du modèle avec les coefficients obtenus après transformation.

Approche « classification en cascade »

Pour la première approche, l'identification des activités pouvant être détectées à l'aide d'un modèle spécifique s'est faite de manière empirique, tout d'abord en analysant la matrice de confusion du meilleur classificateur, soit celle de la FFT pour 50 coefficients, en sélectionnant les meilleurs candidats, puis en groupant les activités et en produisant des modèles dont la performance est évaluée.

	NOAL	NOC	NOED	NOEM	NOM	M1AR	M1AV	M2AR	M2AV	M3AR	M3AV
NOAL	47	0	0	3	8	0	3	1	0	0	0
NOC	0	48	0	0	0	0	0	0	0	0	0
NOED	2	0	27	3	27	0	3	2	1	0	0
NOEM	3	0	5	21	41	0	0	1	0	0	0
NOM	30	2	15	21	159	1	5	0	0	0	0
M1AR	6	0	0	1	5	0	1	0	0	0	0
M1AV	0	0	6	4	8	0	14	12	3	0	0
M2AR	2	1	4	1	5	1	14	12	1	0	0
M2AV	0	1	2	2	7	0	11	2	2	2	0
M3AR	0	0	2	0	3	0	5	4	1	0	0
M3AV	0	0	0	0	3	1	1	0	0	1	0
Justes / faux	0,478	0,077	0,557	0,625	0,403	1	0,754	0,647	0,75	1	0

Tableau 16 Matrice de confusion en utilisant une FFT dont 50 coefficients sont conservés pour la classification.

La matrice présentée ci-dessus permet d'identifier les activités dont le potentiel d'amélioration est les plus grand, soit celles pour lesquelles il y a un nombre très important de cas mal classifiés. Il s'agit ici des activités « NOC » - activité normale de course –et « NOM » - activité normale de marche – avec des scores respectifs de 0.077 et de 0.403. Ces deux catégories ont donc été extraites et évaluées, tout d'abord en les combinant puis en les considérant individuellement.

Les résultats sont les suivants avec un ensemble d'arbres de décision comprenant 1000 modèles :

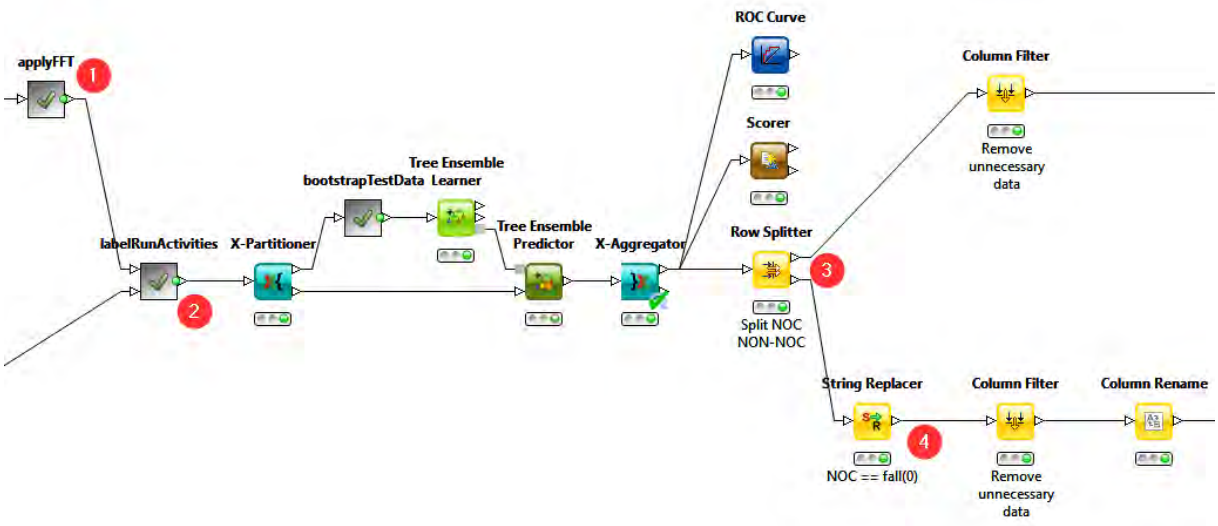
Classes	Score AUC
NOC-NOM, OTHER	0.7971
NOC, NON-NOC	1.000
NOM, NON-NOM	0.8111

Tableau 17 Score dans l'identification d'activités particulières.

L'activité catégorisée « NOC », à savoir de type course, est clairement identifiable à l'aide des 50 coefficients obtenus après la transformation de Fourrier. Il s'agit donc de la meilleure candidate pour être filtrée en amont. L'implémentation dans KNIME s'est faite à l'aide d'une collection de nœuds, encapsulés dans un container (méta nœud).

Dans ce dernier, outre la transformation de Fourrier (1), les activités sont labélisées entre activités de course et de non course (2), puis soumises à un modèle de classification. Une fois la séparation faite, les jeux ayant été identifiés comme des activités de course sont retirées des données (3) soumises au deuxième classificateur. L'évaluation de la performance de cette classification en cascade se fait tout à la fin en groupant et standardisant (4) les résultats classés afin de pouvoir les évaluer de manière consolidée.

Figure 13 Détail de l'implémentation de la détection des activités de type "course".



Les résultats suivants ont été obtenus :

Coefficients	Score AUC modèle 1 (détection NOC)	Score AUC
10	0.9985	0.9102
25	0.9996	0.9161
50	1	0.9146
75	1	0.9104

Tableau 18 Scores obtenus avec des modèles en cascade en fonction des coefficients.

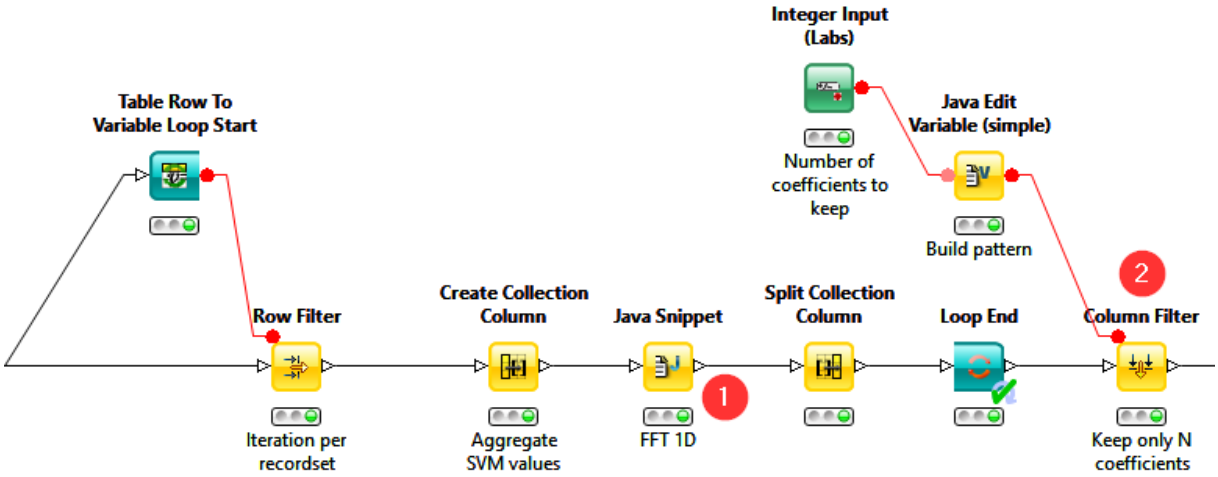
Tous les scores globaux, ainsi que ceux du premier modèle, se trouvent sur un intervalle peu étendu ; le nombre de coefficients ne joue donc pas un rôle crucial dans ce modèle. Toutefois, par rapport au résultat de l'itération précédente, on constate un léger gain de 1.68%.

APPROCHE « EXTENSION DES PARAMÈTRES »

La réduction des données, telle que résultant des transformées DFT ou DWT, permet d'ajouter des paramètres au modèle de classification sans le surcharger, pour autant que leur nombre ne soit pas trop conséquent. Idéalement, les valeurs ajoutées véhiculent de l'information supplémentaire permettant de mieux départager les classes.

L'implémentation dans KNIME s'est faite par l'ajout de la transformée de Fourrier (1), un paramètre permettant de contrôler le nombre de coefficients devant être conservés (2). Ces valeurs sont ensuite jointes comme paramètres au nœud produisant le modèle.

Figure 14 Détail de l'extraction des coefficients à ajouter comme paramètres au modèle.



Les tests suivants, mettant l'accent sur une petite valeur pour le nombre de coefficients, ont été réalisés ;

Nombre de coefficients	Score AUC
4	0.9166
5	0.9247
6	0.9239
7	0.9218
8	0.9215
9	0.9234
10	0.9231
75	0.9226

Tableau 19 Scores obtenus par l'ajout des coefficients après FFT à la série temporelle.

On constate que l'augmentation du nombre de coefficients ne modifie pas fondamentalement les capacités de classification, les scores des différents essais ayant un écart maximum de 0.0081%. Toutefois, contrairement à l'analyse précédente, ici l'intérêt est de minimiser le nombre de paramètres nécessaires pour augmenter le taux de classification. Le meilleur résultat obtenu permet tout de même d'obtenir un score AUC de 92.47%, tout en n'ajoutant que cinq paramètres au modèle, ce qui reste relativement modeste. En comparaison des résultats obtenus à l'itération précédente, la progression est de 2.54%.

CONCLUSION

L'introduction d'une analyse spectrale permet de faire progresser légèrement la capacité détective du modèle. L'approche plaçant deux modèles en cascade se montre moins performante que l'ajout de paramètres, cette dernière étant donc la méthode retenue.

ITÉRATION 4

CONTEXTE

Le filtrage des données, dans sa première version, présentait un bug. La conséquence de ce problème est une mauvaise attribution des valeurs du SVM aux jeux d'enregistrements. Cette erreur ayant été détectée durant l'itération 3, les résultats de cette dernière ont été corrigés directement ; ils sont donc cohérents par rapport aux choix établis. Toutefois, ce n'est pas le cas des conclusions tirées de l'itération 2, raison pour laquelle une itération corrective a vu le jour. L'objectif de cette phase est d'effectuer la même analyse qu'en itération 2, de comparer les résultats obtenus et d'identifier de potentiels problèmes.

En premier lieu, l'analyse de la durée optimale a été rejouée, et les scores suivants ont été obtenus :

Durée	Nombre de jeu d'enregistrements	% de chutes molles	Score AUC	Score AUC itération deux
200ms	1012	46.64%	0.7928	0.6855
2000ms	926	43.30%	0.9118	0.8510
4000ms	637	23.70%	0.9108	0.8993
8000ms	527	7.78%	0.8544	0.8732

Tableau 20 Scores obtenus pour déterminer la meilleur durée comparés aux éléments obtenus en itération 2.

Selon le tableau précédent, le meilleur résultat est obtenu avec une durée de 2 secondes, soit 100 enregistrements. Comparé aux précédentes valeurs, on constate, non seulement une meilleure détection, mais également le besoin de moins d'enregistrements pour analyser l'activité.

Une partie de l'analyse détaillée – portant sur les erreurs rencontrées - menée en itération 2, a été reconduite, et la constatation suivante a été faite : les activités les plus difficilement classifiables sont celles de type « chute molle avec faible activité résiduelle » (M2AR et M2AV). Ces dernières représentant entre les deux 28.74% des erreurs de classification.

Le choix d'optimisation fait en itération 3 portait sur l'amélioration de la détection des activités de course, car, selon les conclusions tirées précédemment, la marge de progression pour ces erreurs était conséquente. Or, suite à la découverte du bug, les données correctes ont été utilisées pour le calcul des scores en itération 3, ce qui explique la faible progression de la capacité additionnelle de détection. L'utilisation de 100 coefficients avec le modèle le plus efficace de la phase précédente montre que le score obtenu, 0.9104, n'est pas meilleur, démontrant par là l'erreur de ciblage.

CONCLUSION

L'erreur commise en itération 2 a eu comme conséquence de choisir comme cible d'optimisation un type d'activité ne présentant pas le meilleur potentiel d'amélioration. La suite de l'analyse doit se concentrer sur les cas de chutes molles avec faible activité résiduelle, ces derniers étant mal identifiés comme tels par le modèle actuel.

DÉPLOIEMENT

Le format PMML[31] (Predictive Model Markup Language) est un standard d'échange des modèles, et KNIME possède un certain nombre de nœuds permettant, non seulement d'exporter les modèles, mais également les traitements effectués dans le flux dans ce format XML (Morent, Stathatos, Lin, & Berthold, 2011).

Il existe de nombreuses solutions pour l'évaluation des modèles au format PMML ; une liste[32] en recensant plus de 20 est disponible sur le site du DMG (Data Mining Group). Le DMG est l'organisation, regroupant plusieurs vendeurs de solutions de DM dont KNIME, qui formalise la norme PMML. Le prototype réalisé dans le présent travail pour valider la consommation du modèle utilise une librairie Java appelée « jpmml-evaluator »[33] - présente dans la liste précédemment citée - qui fournit un module d'exemple facile à utiliser. Cette dernière a l'avantage d'être disponible librement puisque soumise à la licence « GNU Affero General Public License »[34] qui autorise toute modification et redistribution. Elle peut donc être facilement incluse dans le système F2 et son support des arbres de décision, mais également de la majorité des algorithmes la rend intéressante car élargissant la gamme des options disponibles. En outre, l'application F2D étant produite pour Android, l'intégration d'une librairie Java est aisée.

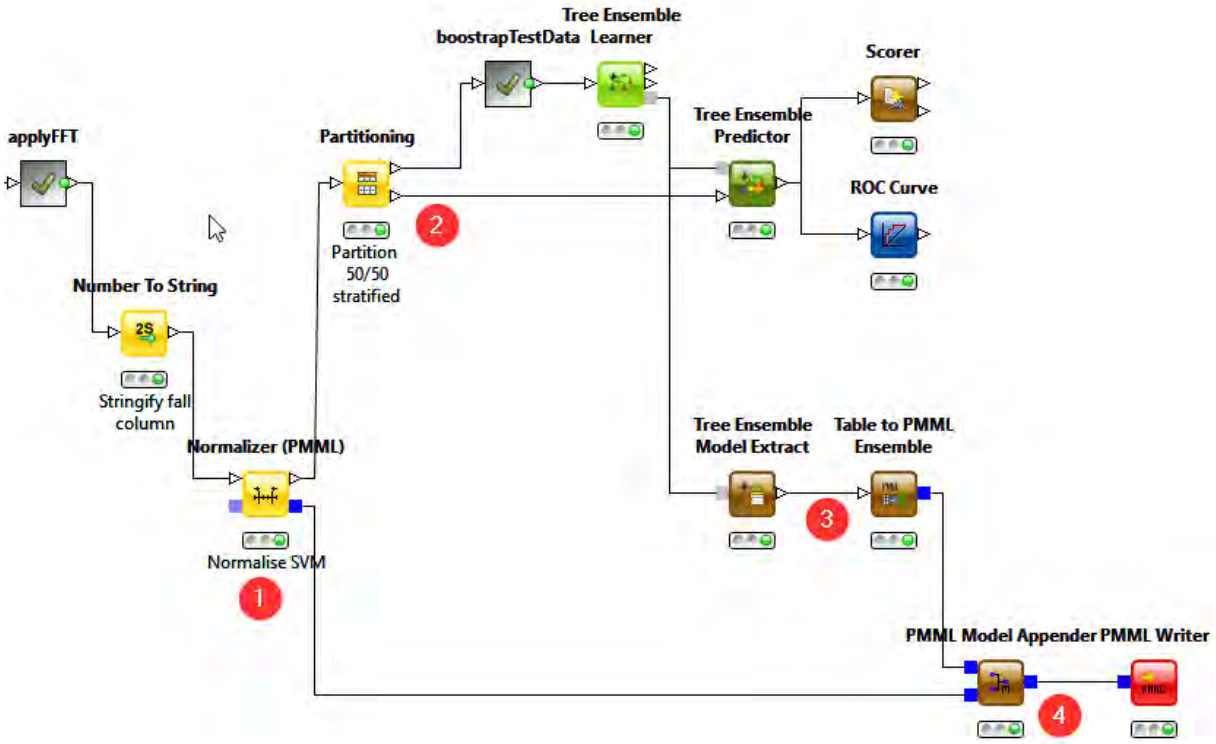
En vue d'exporter le flux KNIME sous un format PMML, celui produit en itération 3 a été copié et adapté. En effet, du fait de l'impossibilité d'inclure sous format PMML directement depuis KNIME, le traitement effectué au niveau des nœuds « Java Snippet », l'ordre des opérations de préparation a été modifié afin de pouvoir grouper les traitements exportables en PMML et ainsi les inclure dans le fichier XML. Ceci a pour but de simplifier au maximum le code devant être appliqué en amont de l'évaluation.

Outre l'ordre des éléments, les nœuds permettant d'effectuer l'évaluation des modèles successifs en partitionnant les données d'entraînement et de test dans une approche de validation croisée, ne font plus de sens dans la présente phase. Les nœuds « X-Partitioner » et « X-Aggregator » ont donc été supprimés et la division des données remplacée par une subdivision en deux jeux de taille égale et une méthode de répartition par strates sur les activités permettant une meilleure répartition des différents cas.

[31] https://en.wikipedia.org/wiki/Predictive_Model_Markup_Language
[32] http://www.dmg.org/products.html
[33] https://github.com/jpmml/jpmml-evaluator
[34] http://www.gnu.org/licenses/agpl-3.0.en.html

Il en résulte donc le flux suivant : la normalisation du SVM (1) est effectuée après la production de la transformée de Fourrier. Les données sont séparées en deux populations de taille égale (2), le jeu d'entraînement est soumis à un modèle d'ensemble d'arbres comptant 1000 modèles. Le modèle généré est d'une part évalué et d'autre part préparé pour l'extraction en PMML (3). Ce dernier est ensuite couplé avec l'opération de transformation réalisée en étape 1 (4)[35] puis exporté dans un fichier.

Figure 15 Flux produit pour l'export du modèle sous format PMML.



Le fichier PMML ainsi généré a une taille d'un peu plus de 40 Mo, ce qui est assez important. Toutefois, son format texte permet d'obtenir des taux de compressions extrêmement bons - de l'ordre de 6% -. Cette opération menée produit une archive de 2 Mo environ. Bien entendu le modèle devra être décompressé et chargé en mémoire, ce qui aura un impact sur les ressources consommées par l'application. C'est dans ce sens que la notion de partitionnement a été conservée. En effet, un modèle généré sur la totalité des jeux d'enregistrements délivre un modèle sous format PMML plus volumineux, de l'ordre de 70 Mo.

Une fois le modèle disponible sous forme de fichier, l'application fournie en exemple dans la librairie « jpmml-evaluator » a été compilée en une archive Java auto exécutable comprenant toutes les librairies dépendantes. Le JAR ainsi obtenu a une taille de 7.45 Mo. Cette dernière peut être réduite en supprimant les librairies tierces utilisées pour paramétrer l'application ou encore pour produire des mesures de performance. Toutefois, dans le cas présent, le but est d'utiliser le modèle en dehors de KNIME afin de valider son interprétation par une potentielle forme d'implémentation dans le système F2D.

[35] https://tech.knime.org/forum/knime-labs-general/how-to-save-tree-ensemble-model-to-pmml

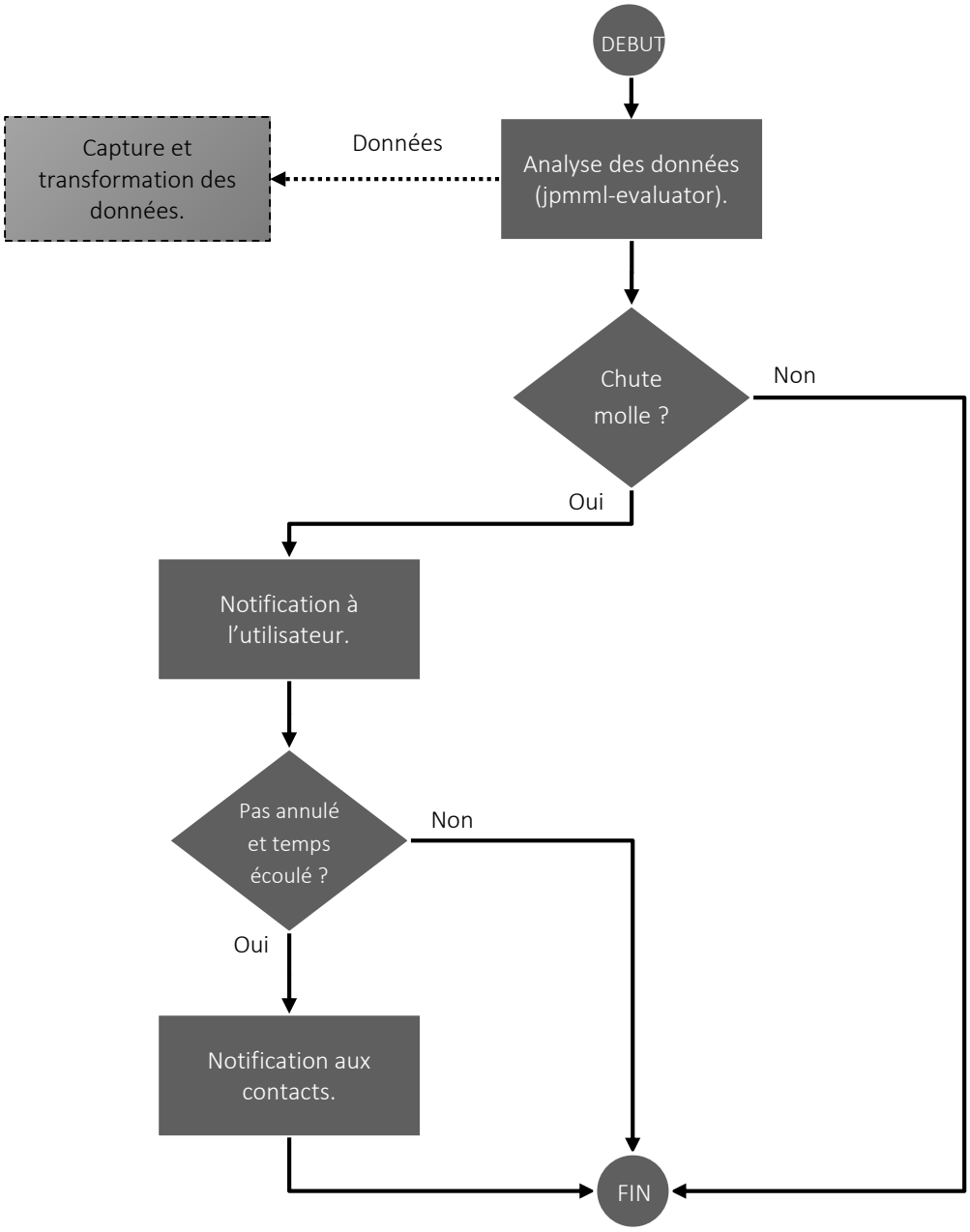
Les données préparées dans KNIME qui sont disponibles avant la normalisation ont été premièrement filtrées au niveau des colonnes pour ne conserver que celles nécessaires à l'évaluation du modèle, puis exportées dans un fichier CSV. L'application « jpmml-evaluator-example », avec le modèle généré et les données ainsi soumises, s'est montrée fonctionnelle, toutefois la normalisation n'a pas été appliquée sur les données. Ce disfonctionnement a obligé la mise à disposition de l'application Java des données déjà normalisées dans KNIME. Sur ce point, les investigations n'ont pas été conduites plus avant, mais selon la documentation de la librairie, le support des transformations, tout du moins dans la version précédente[36], est fonctionnel.

L'export du modèle tel que réalisé avec la librairie Java précitée permet d'établir de manière plus précise l'architecture du système tel que proposé. Il faut préciser que le système décrit ci-dessous ne contient que la partie de détection des chutes molles, l'application développé par le TaM pouvant sans problème être exécutée en parallèle, une extension du système aux chutes brutales pouvant être envisagée dans une phase ultérieure.

[36] https://github.com/jpmml/jpmml

Le système fonctionne comme suit : un processus s'exécutant en tâche de fond enregistre les données d'accélération linéaire provenant du dispositif. Il s'assure que la fréquence d'enregistrement est bien de 50 Hz, calcule le SVM, le normalise et applique la transformée de Fourrier. La durée des données ainsi conservées est de 4 secondes. Un second processus s'exécutant de manière programmée toutes les 2 secondes évalue les données présentes dans le tampon en considérant toujours la totalité de son contenu. Si le résultat de l'analyse est positif, une notification est émise à l'adresse de l'utilisateur. Ce dernier peut ainsi stopper la logique d'alerte en cas de fausse détection ou s'il n'a subi aucun traumatisme. Les notifications sont émises à l'adresse des contacts préalablement choisis si le sujet ne les a pas annulées et que le temps d'attente, par exemple 15 secondes, est dépassé. Durant ces deux dernières étapes, le processus évaluant les données du tampon est stoppé.

Figure 16 Schéma du système proposé.



CONCLUSION FINALE

Après quatre itérations, le modèle produit permet une détection à plus de 92% des cas de chutes molles ce qui représente une augmentation importante des capacités de l'environnement F2D. Combiné au système proposé et développé par le TaM, l'application sera capable de produire des résultats parfaitement exploitables dans la réalité et répondant aux besoins formulés par la FST.

Bien entendu, de nombreuses voies d'améliorations existent. Parmi elles, l'extension de la technique de MLM à toutes les chutes, car les résultats obtenus sur les chutes molles, plus difficilement détectables, sont de très bonne qualité. Le fait d'intégrer la détection de tous les cas permettrait non seulement de diminuer les ressources nécessaires à l'application – en ne conservant qu'un processus exécuté en tâche de fond par exemple - mais également de repenser globalement les déclencheurs d'analyse.

La conclusion de l'itération 4 met en lumière une moins bonne performance de détection des cas de chutes molles avec faible reprise de l'activité. Ces types d'événements, qui peuvent représenter un part importante des cas réels, doivent être analysés plus finement afin de déterminer le meilleur moyen de les détecter. Ici un modèle spécifique utilisant un plus grand nombre d'enregistrements comme paramètres peut être une option intéressante.

Pour l'affinage des résultats de détection, l'ajout de capteurs est un choix intéressant. Comme démontré par He, Hu & Li (An Autonomous Fall Detection and Alerting System based on Mobile and Ubiquitous Computing, 2013), l'ajout d'un gyroscope peut amener une finesse supérieur de détection en se basant sur la position finale de l'utilisateur à l'aide de ces derniers senseurs.

L'implémentation, telle que proposée, souffre de plusieurs faiblesses : tout d'abord au niveau des ressources consommées ; le fait d'avoir un processus qui soit programmé pour être exécuté régulièrement n'est pas la solution la moins gourmande et d'autres stratégies pour son déclanchement doivent être trouvées. De plus, la transformation des données, ici effectuée en continu, devrait se produire le plus tard possible, afin d'économiser le traitement nécessaire à ces opérations. Ici encore, le traitement ne devrait être déclenché que si un cas de chute est suspecté.

Pour finir, ce travail ne peut prétendre livrer toutes les clefs pour la mise en production d'un système de détection de chutes, mais doit servir au développement d'un prototype, celui-ci permettant la mise en place d'un système efficace, efficient et robuste. Ceci, notamment grâce aux retours en provenance du terrain, permettant d'amener des améliorations au système, tant sur le plan de l'ergonomie que des capacités détectives.

RÉFÉRENCES

Abbate, S., Avvenuti, M., Bonatesta, F., Cola, G., Corsini, P., & Vecchio, A. (2012). A smartphone-based fall detection system. *Pervasive and Mobile Computing*, 883-899.

Agrawal, R., Faloustos, C., & Swami, A. (1993). *Efficient Similarity Search In Sequence Databases.* San Jose, CA : IBM Almaden Reasearch Center.

Aguiar, B., Tiago, R., Joana, S., & Inês, S. (2014). Accelerometer-based fall detection for smartphones. *IEEE International Symposium on Medical Measurements and Applications (MeMeA)* (pp. 1-6). Lisboa: IEEE.

Albert, M. V., Kording, K., Herrman, M., & Jayaraman, A. (2012, May 7). Fall Classification by Machine Learning Using Mobile Phones. (U. H. Christian Lovis, Ed.) *PLos ONE*, pp. 1-6.

Azevedo, A., & Santos, M. F. (2008). KDD, SEMMA AND CRISP-DM: A PARALLEL OVERVIEW . *IADIS European Conference on Data Mining* (pp. 182-185). IEEE.

Bagalà, F., Becker, C., Capello, A., Chiari, L., Aminian, K., Hausdorf, J. M., . . . Klenk, J. (2012, May). Evaluation of Accelerometer-Based Fall Detection Algorithms on Real-World Falls. *PLoS ONE, 7*(5), pp. 1-9.

Bersch, S. D., Azzi, D., Khusainov, R., Achumba, I. E., & Ries, J. (2014). Sensor Data Acquisition and Processing Parameters for Human Activity Classification. *Sensors, 14*, pp. 4239-4270. doi:10.3390/s140304239

Berthold, M. R., Borgelt, C., Höppner, F., & Klawonn, F. (2010). *Guide to Intelligent Data Analysis.* Springer.

Bianchi, F., Redmond, S. J., Narayanan, M. R., Cerutti, S., & Lovell, N. H. (2010, December). Barometric Pressure and Triaxial Accelerometry-Based Falls Event Detection. *IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING,, 18*, pp. 619-627.

Breiman, L. (1994). *Bagging Predictors.* Berkley, California 94720: University of California, Department of Statistics.

Chan, K.-p., & Fu, A. W.-c. (1999). Efficient TIme Series Matching by Wavelets. *15th IEEE International Conference on Data Engineering* (pp. 126-233). Melbourne, Australia: IEEE.

Chapman, P., Clinton, J., Kerber, R., Khbaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0 - Step-by-step data mining guide.* SPSS.

Chien-Liang, L., Chia-Hoang, L., & Ping-Min, L. (2010). A fall detection system using k-nearest neighbor classifier. *Expert Systems with Applications*, 7174–7181.

Dai, J., Bai, X., Yang, Z., Shen, Z., & Xuan, D. (2010). PerFallD: A Pervasive Fall Detection System Using Mobile Phones. (IEEE, Ed.) Colombus, Ohio, USA.

Efron, B. (1979, January). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics, Vol 7, No. 1*, pp. 1-26.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, pp. 861-874.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 37-53.

He, J., Hu, C., & Li, Y. (2013). An Autonomous Fall Detection and Alerting System based on Mobile and Ubiquitous Computing. *International Conference on Ubiquitous Intelligence & Computing and 2013 IEEE 10th International Conference on Autonomic & Trusted Computing* (p. 539). IEEE.

Igual, R., Medrano, C., & Plaza, I. (2013). Challenges, issues and trends in fall detection systems. *BioMedical OnLine*, 12:66.

Karantonis, D. M., Narayanan, M. R., Mathie, M., Lovell, N. H., & Celler, B. G. (2006, January). Implementation of a Real-Time Human Movement Classifier Using a Triaxial Accelerometer for Ambulatory Monitoring. *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, 10*, pp. 156-167.

Kau, L.-J., & Chen, C.-S. (2015, Januay). A Smart Phone-Based Pocket Fall Accident Detection, Positionning, and Rescue System. *IEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL 19, NO 1*, 44-56.

KDnuggets. (2015, May). *Analytics, Data Mining, Data Science software/tools used in the past 12 months*. Retrieved August 7, 2015, from KDnuggets: http://www.kdnuggets.com/polls/2015/analytics-data-mining-data-science-software-used.html

Keogh, E., Kaushik, C., Pazzani, M., & Mehrotra, S. (2001). Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record*, (pp. 151-162).

Korn, F., Jagadish, H. V., & Faloustos, C. (1997). Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequence. *SIGMOD*, (pp. 298-300). Tucson, Arizona.

Kostopoulos, P., Nunes, T., Slavi, K., Deriaz, M., & Torrent, J. (2014). *Increased Fall Detection Accuracy in an Accelerometer-Based Algorithm Considering Residual Movement.* fstlab.ch.

Kung, H.-Y., Ou, C.-Y., Li, S.-D., Lin, C.-H., Chen, H.-J., Hsu, Y.-L., . . . Wu, C.-I. (2010). Efficient Movement Detection for Human Actions Using Triaxial Accelerometer. Pingtung, Taiwan.

Kurgan, L. A., & Musilek, P. (2006). A survey of Knowledge Discovery and Data Mining process models. (C. U. Press, Ed.) *The Knowledge Engineering Review, 21:1*, 1-24.

Landgrebe, T., & Duin, R. P. (2006). A simplifed extension of the Area under the ROC to the multiclass domain. *17th annual Symposium of the Pattern Recognition Association of South Africa*, (pp. 241-245). Citeseer.

Li, Y., Chen, G., Shen, Y., Zhu, Y., & Cheng, Z. (2012). ACCELEROMETER-BASED FALL DETECTION SENSOR SYSTEM FOR THE ELDERLY. *International Conference on Cloud Computing and Intelligence Systems* (pp. 1216-1220). IEEE.

Luštrek, M., & Kaluža, B. (2009). Fall Detection and Activity Recognition with Machine Learning. *Informatica*(33), pp. 205-212.

Mehta, M., Rissanen, J., & Agrawal, R. (1995). *MDL-based Decision Tree Pruning.* AAAI Press.

Mirchevska, V., Lustrek, M., & Gams, M. (2013, May). Combining domain knowledge and machine learning for robust fall detection. *Expert Systems*, pp. 163-175.

Morent, D., Stathatos, K., Lin, W.-C., & Berthold, M. (2011). Comprehensive PMML Preprocessing in KNIME. *Proceedings of the 2011 workshop on Predictive markup language modeling* (pp. 28-31). ACM.

Morinaka, Y., Yoshikawa, M., Amagasa, T., & Uemura, S. (2001). The L-index: An indexing structure for efficient subsequence matching in time sequence databases. *Proceedings of 5th PacificAisa Confereve on Knowledge Discovery and Data Mining*, (pp. 51-60).

Mubashr, M., Ling, S., & Luke, S. (2013, January 16). A survey on fall detection: Principles and approaches. *Neurocomputing*, pp. 144-152.

Noury, N., Rumeau, P., Bourke, A. K., OLaighin, G., & Lundy, J. E. (2008, December 6). A proposal for the classification and evaluation of fall detectors. *IRBM*, pp. 340-349.

Ojetola, O., Gaura, E. I., & Brusey, J. (2011). Fall Detection with Wearable Sensors—SAFE (SmArt Fall dEtection). *2011 Seventh International Conference on Intelligent Environments* (pp. 318-321). IEEE.

Oufella, Y., Adam, S., Chatelain, C., Heute, L., & Lecourtier, Y. (2008, Septembre 2). ÉVOLUTION DU CONCEPT DE FRONT ROC ET COMBINAISON DE CLASSIFFIEUR. Rouen, France.

Perry, J. T., Kellog, S., Vaidya, S. M., Jong-Hoon, Y., Ali, H., & Sharif, H. (2009). Survey and evaluation of real-time fall detection approaches. *Poceedings of the 6th International Symposium High-Capacity Optical Networks and Enabling Technologies* (pp. 158-164). Alexandria: Institute of Electrical and Electronics Engineers.

Rohanizadeh, S. S., & Moghadam, M. B. (2009). A Proposed Data Mining Methodology and its Application to Industrial Procedures. *Journal of Industrial Engineering*(4), 37-50.

Shafique, U., & Qaiser, H. (2014). A Comparative Study of Data Mining Process Models (KDD, CRISP-DM and SEMMA). *International Journal of Innovation and Scientific Research*, 217-222.

Tan, T. D., & Tinh, N. V. (2014). Reliable Fall Detection System Using an 3-DOF accelerometer and Cascade Posture Recognitions. *APSIPA 2014.* Angkor Wat: APSIPA .

Torrent, J., Triki, M., Kühner, D., Nicolet, M., Sieber, G., Vallat, M., . . . Nedjmeddin, B. (2015). *Développement d'un dispositif innovant de détection de chute à l'aide d'une smartwatch Android.* Neuchâtel: FST - Fondation Suisse pour les Téléthèses.

Vo, V. Q., Gueesang, L., & Deokjai, C. (2012). Fall Detection based on Movement and Smart Phone Technology. (IEEE, Ed.) Gwangju, South Korea.

Wang, Y., & Bai, X.-y. (2013). Research of Fall Detection and Alarm Applications for the Elderly. *2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)* (pp. 615 - 619). Shenyang, China: IEEE.

World Healt Organization. (2007). *WHO Global Report on Falls Prevention in Older Age.* World Health Organization.

Yuanchun Shi, Y., & Wang, X. (2012). Fall Detection on Mobile Phones Using Features from A Five-phase Model. *2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing* (pp. 951-956). IEEE.

Yuwono, M., Moulton, B. D., Su, S. W., Celler, B. G., & Nguyen, H. T. (2012, 02 06). Unsupervised machine-learning method for improving the performance of ambulatory fall-detection systems. *BioMedical Engineering OnLine*, pp. 2-11.

Zero Padding Applications. (2015, Juillet 25). *Zero Padding Applications*. Retrieved from DSP Related: http://www.dsprelated.com/freebooks/mdft/Zero_Padding_Applications.html

Zhu, Y. (2004). *High Performance Data Mining in Time Series: Techniques and Case Studies.* New York.

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ÉQUATIONS

ANNEXES

MÉDIA CONTENANT LES SOURCES

Le DVD-ROM accompagnant ce travail contient tous les fichiers qui ont été utilisés et produits, excepté la présentation utilisée lors de la défense. Au premier niveau se trouve le découpage par chapitre, proche de celui présent dans ce document. On trouve ainsi les sections suivantes :

Dossier	Contenu
00_knowledge_base	Subdivisé selon les domaines de connaissance, ce dossier contient tous les documents, sous format électronique, consultés et analysés lors de ce travail. On trouve notamment les publications ayant servi de références dans le chapitre « État de l'art ».
01_database	Ce second dossier regroupe les éléments produits en relation avec la base de données. On y trouve ainsi le fichier source du schéma de base de données, les scripts Groovy pour l'import des données, les données brutes fournies par la FST ainsi que le script de création de la base.
02_data_comprehension	Les artéfacts produits lors de la phase de découverte des données sont placés dans ce dossier, notamment le flux KNIME.
03_iteration_1	Ce dossier contient les mesures obtenues lors des différents essais menés pour trouver le meilleur algorithme ainsi que les flux KNIME ayant servis lors de cette première itération.
04_iteration_2	Les éléments produits lors de l'itération 2 sont, non seulement des mesures de score et des flux KNIME, mais également un fichier Excel spécifiquement créé pour l'analyse détaillée et contenant des macros VBA permettant la génération automatique des différents axes d'inspection.
05_iteration_3	Dans cette troisième itération, outre les éléments généralement rencontrés lors des phases précédentes, il existe le détail des scores obtenus après la transformation du signal pour l'analyse spectrale (FFT et FWT).
06_iteration_4	Les artéfacts générés durant cette courte itération sont les courbes ROC liés aux essais menés pour déterminer la durée optimale nécessaire à la classification, ainsi que le fichier d'analyse détaillée.
07_deployment	L'exemple fourni par la librairie PMML est exploitable à l'aide du fichier « evaluate.cmd ». Ce dernier utilise le modèle exporté depuis KNIME (« model.pmml ») et effectue la classification des données présentes dans le fichier « data.csv ».

Au niveau supérieur, outre les dossiers, se trouvent les documents rédigés pour la présentation du travail, dont le présent rapport.

Dans chaque archive de flux KNIME se trouve un fichier au format SVG permettant d'obtenir une visualisation de ce dernier sans effectuer l'import à l'aide de la plateforme.

SCRIPT D'EXPORT DES DONNÉES DE LA BASE

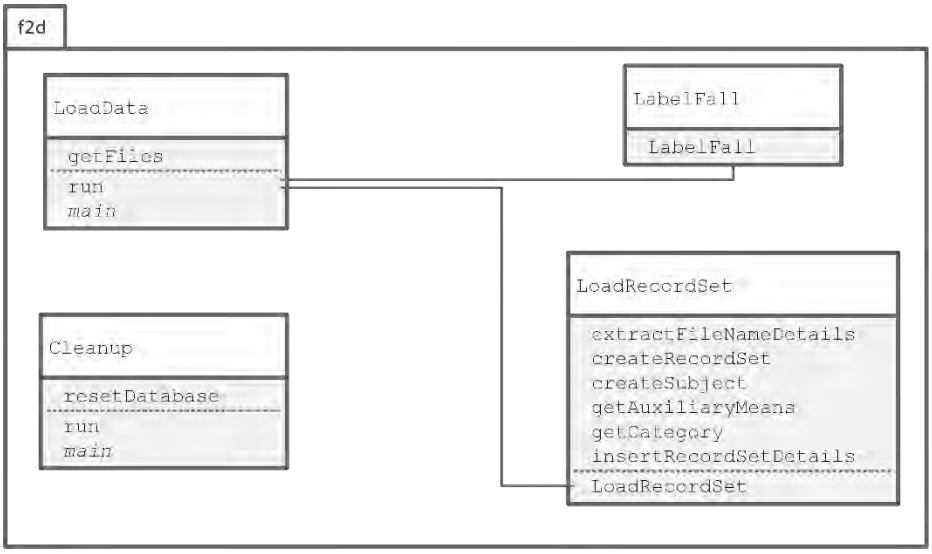
```
-- Administrative privilege to write on file system
-- USE mysql;
-- UPDATE user SET File_priv = 'Y' WHERE User = 'f2d';
-- FLUSH PRIVILEGES;

SELECT
    'recordset',
    'timestamp',
    'linear_acceleration_x',
    'linear_acceleration_y',
    'linear_acceleration_z',
    'fall',
    'activity'

UNION ALL SELECT
    r0.recordset,
    r0.timestamp,
    r0.linear_acceleration_x,
    r0.linear_acceleration_y,
    r0.linear_acceleration_z,
    r0.fall,
    c.code
FROM
    record r0
        INNER JOIN
    recordset rs ON r0.recordset = rs.idrecordset
        INNER JOIN
    category c ON rs.category = c.idcategory
WHERE
    recordset IN (SELECT
            rs.idrecordset
        FROM
            recordset rs
                INNER JOIN
            category c ON rs.category = c.idcategory
        WHERE
            c.code LIKE 'N%'
                OR (c.code LIKE 'M%' AND r0.fall = 1))
ORDER BY recordset, convert(timestamp,SIGNED) INTO OUTFILE '@path/data.csv' FIELDS
ENCLOSED BY '"' TERMINATED BY ';' ESCAPED BY '"' LINES TERMINATED BY '';
```

CHARGEMENT DES DONNÉES

Le chargement des données dans la base s'est fait au moyen de classes Groovy dont le schéma UML est présent ci-dessous.

Les deux classes pouvant être exécutées sont « LoadData » et « Cleanup ». La première va itérer sur la liste des fichiers CSV présents dans le dossier paramétré. Une fois chargées, les données de chaque « recordset » sont passées à l'objet « LoadRecordSet » qui s'occupera d'insérer les informations dans les tables respectives de la base de données. Les méthodes privées – « extractFileNameDetails », « createRecordset », « createSubject », « getAuxiliaryMeans », « getCategory » et « insertRecordDetails » - correspondent à des traitements spécifiques liés à des entités de la base. Une fois tous les fichiers insérés dans la base de données, le contenu du fichier comprenant les valeurs de début et de fin pour chaque chute est lu. Les enregistrements compris entre la borne supérieure et la borne inférieure – comprises - sont mis à jour et un rapport est affiché permettant de vérifier le bon déroulement du traitement.

La seconde classe (« Cleanup ») effectue un nettoyage complet de la base de données en supprimant la totalité des informations contenues dans les tables « record », « recordset » et « subject ». Les autres tables contenant des éléments « constants ».



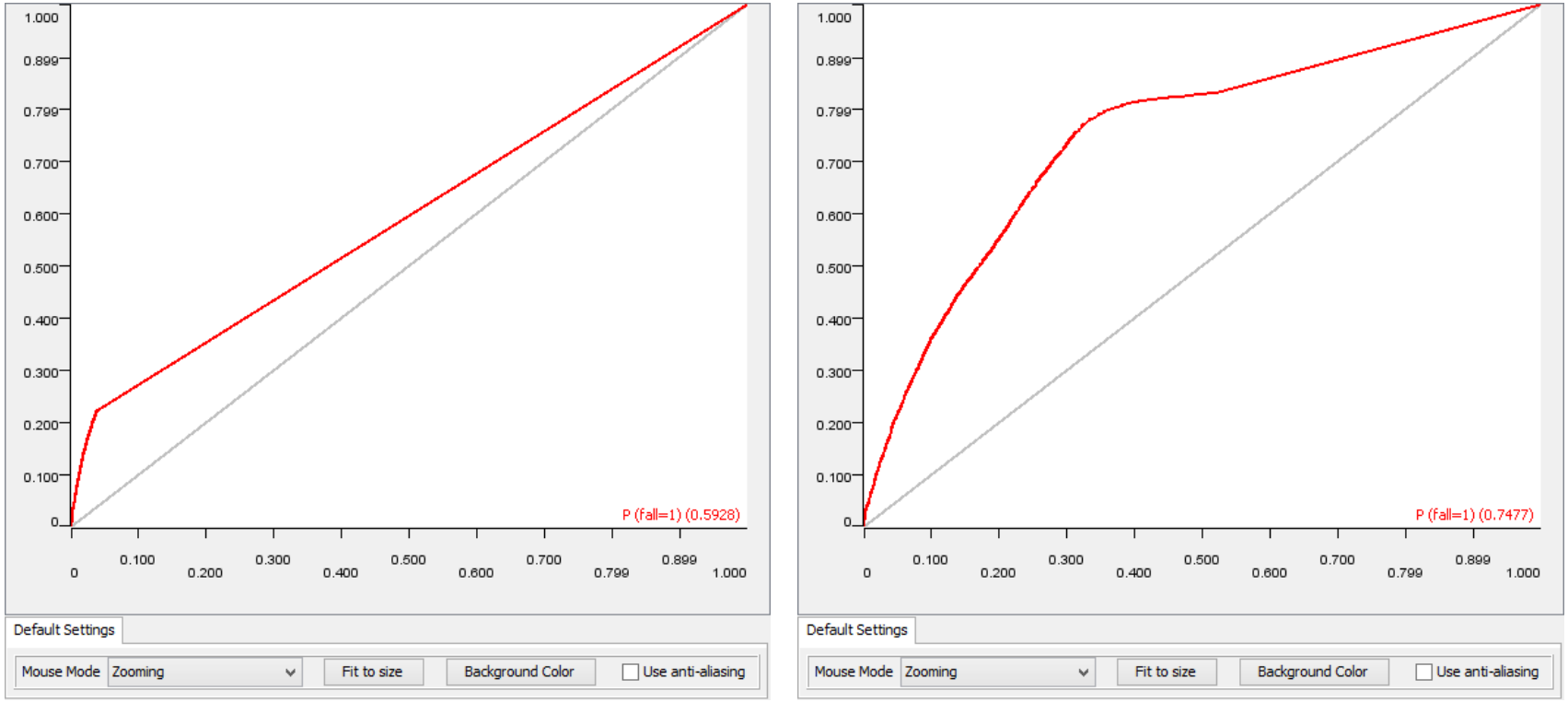
ITÉRATION 1

RÉSULTATS DES ESSAIS

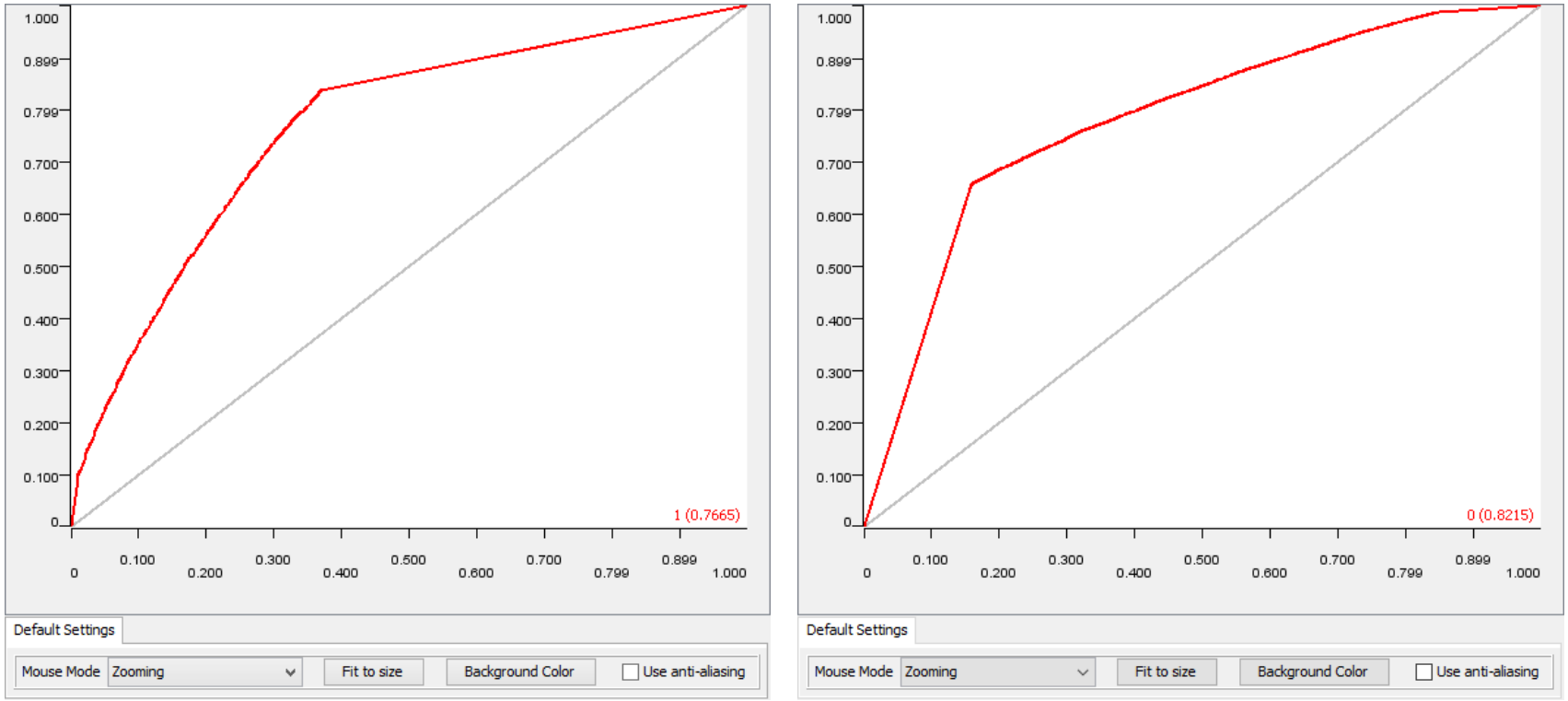
No	Nom	Classés juste	Classés faux	Précision	Erreur	VP	FN	VN	FP	K de Cohen	Sensibilité	Spécificité	AUC
1	Arbre de décision. Gain ratio. Taille minimale d'un nœud; 100	538264	116304	82,232	17,768	531910	6354	6536	109768	505,576	0,987861364	0,054718313	0,7493
2	Ensemble d'arbres de décision. 10 modèles	640777	13791	97,893	2,107	639110	1667	11223	2568	6,058	0,982742687	0,393624557	0,6759
3	Ensemble d'arbres de décision. 100 modèles	642499	12069	98,156	1,844	640982	1517	11373	696	-5,376	0,982566241	0,685494803	0,7889
4	Ensemble d'arbres de décision. 300 modèles	642540	12028	98,162	1,838	641028	1512	11378	650	-5,524	0,98255994	0,699352451	0,8118
5	k-NN défaut	613036	41532	93,655	6,345	609875	3161	9856	31676	-76,236	0,984096326	0,09073686	0,5986
6	k-NN avec k = 50	430585	223983	65,782	34,218	420174	10411	2606	221377	357,138	0,993836038	0,044916044	0,7665
7	k-NN avec k = 15	525389	129170	80,26	19,734	518293	7105	5912	123258	-319,539	0,988721969	0,054501661	0,6899
8	k-NN avec k = 7	578475	76093	88,375	11,625	573560	4915	8102	67991	34,425	0,986070948	0,067415576	0,6413
9	k-NN avec k = 7 et poids des voisins	583100	71468	89,082	10,918	578185	4915	8102	63366	458,889	0,98618083	0,071981957	0,6452
10	k-NN avec k = 5 et poids des voisins	598182	56386	91,386	8,614	594038	4144	8873	47513	19,821	0,985283068	0,080221461	
11	MLP défaut 10 neurones	438241	216327	66,951	33,049	431195	7046	5971	210365	-40,269	0,986341573	0,032408664	0,6562
12	MLP 3 neurones	426792	227776	65,202	34,798	420508	6284	6733	221043	-137,127	0,984240745	0,027642999	0,5964
13	MLP 15 neurones	402845	251723	61,544	38,456	394946	7899	5118	246605	-166,926	0,987207047	0,03103684	0,6585
14	MLP 10 neurones, normalisation	434468	220100	66,375	33,625	427551	6917	6103	213997	-106,178	0,985926568	0,031310827	0,6489
15	SVM default										#DIV/0!	#DIV/0!	

No	Nom	Classés juste	Classés faux	Précision	Erreur	VP	FN	VN	FP	K de Cohen	Sensibilité	Spécificité	AUC
16	k-NN avec k = 50 et poids des voisins	473973	180595	72,41	27,59	464398	9575	3445	177150	-87,86	0,992636419	0,051278618	0,8215
17	MLP 3 neurones, normalisation	429566	211982	66,55	33,45	429566	6052	6968	211982	308,589	0,984037899	0,027757139	0,5781
18	MLP 15 neurones, normalisation	408705	245863	62,439	37,561	401169	7536	5484	240379	-140,762	0,986514301	0,030397515	0,6572
19	MLP 30 neurones	406842	247726	62,154	37,846	398786	8056	4961	242765	-53,758	0,987712602	0,032118523	0,6732
20	MLP 30 neurones, normalisation	395947	258621	60,49	39,51	388049	7898	5122	253499	-165,109	0,98697259	0,030214578	0,6554
21	Bayes default	382264	272304	58,399	41,601	375759	6505	6385	265919	273,414	0,983291639	0,023878219	0,5427
22	Arbre de décision. Gini Index												0,5928
23	Arbre de décision. Gini Index. Taille minimale d'un nœud; 100												0,7477

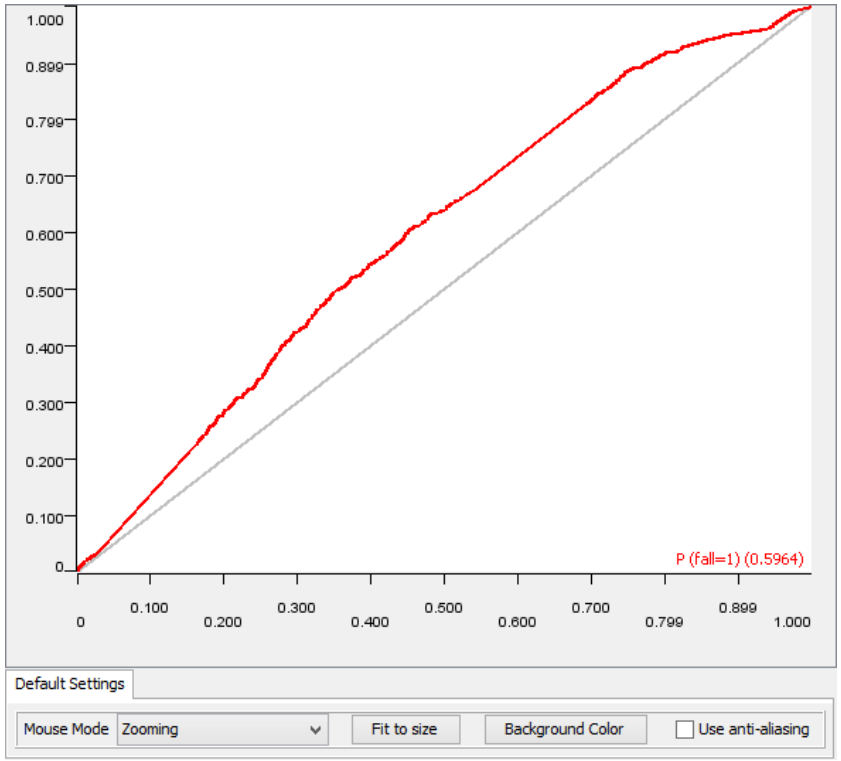
COURBES ROC (PARTIEL)

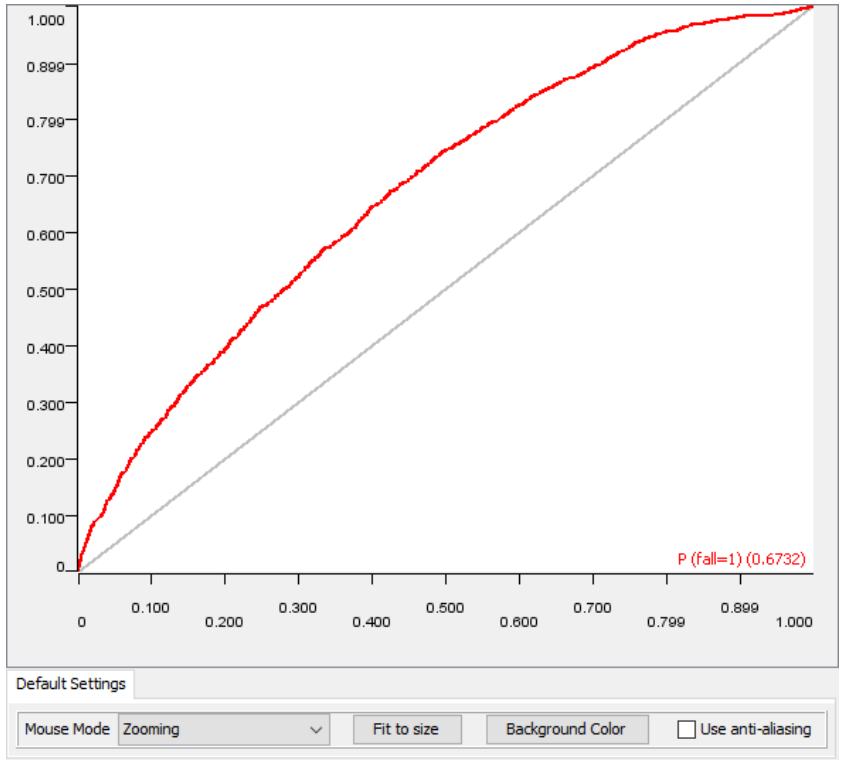


Impact du paramètre de taille minimale d'un nœud sur les courbes ROC obtenues avec un arbre de décision ; à droite sans contrainte, à gauche avec une contrainte de 100 éléments



Impact de la prise en compte de la distance avec l'algorithme KNN sur les courbes ROC ; à gauche sans prise en compte, à droite avec.





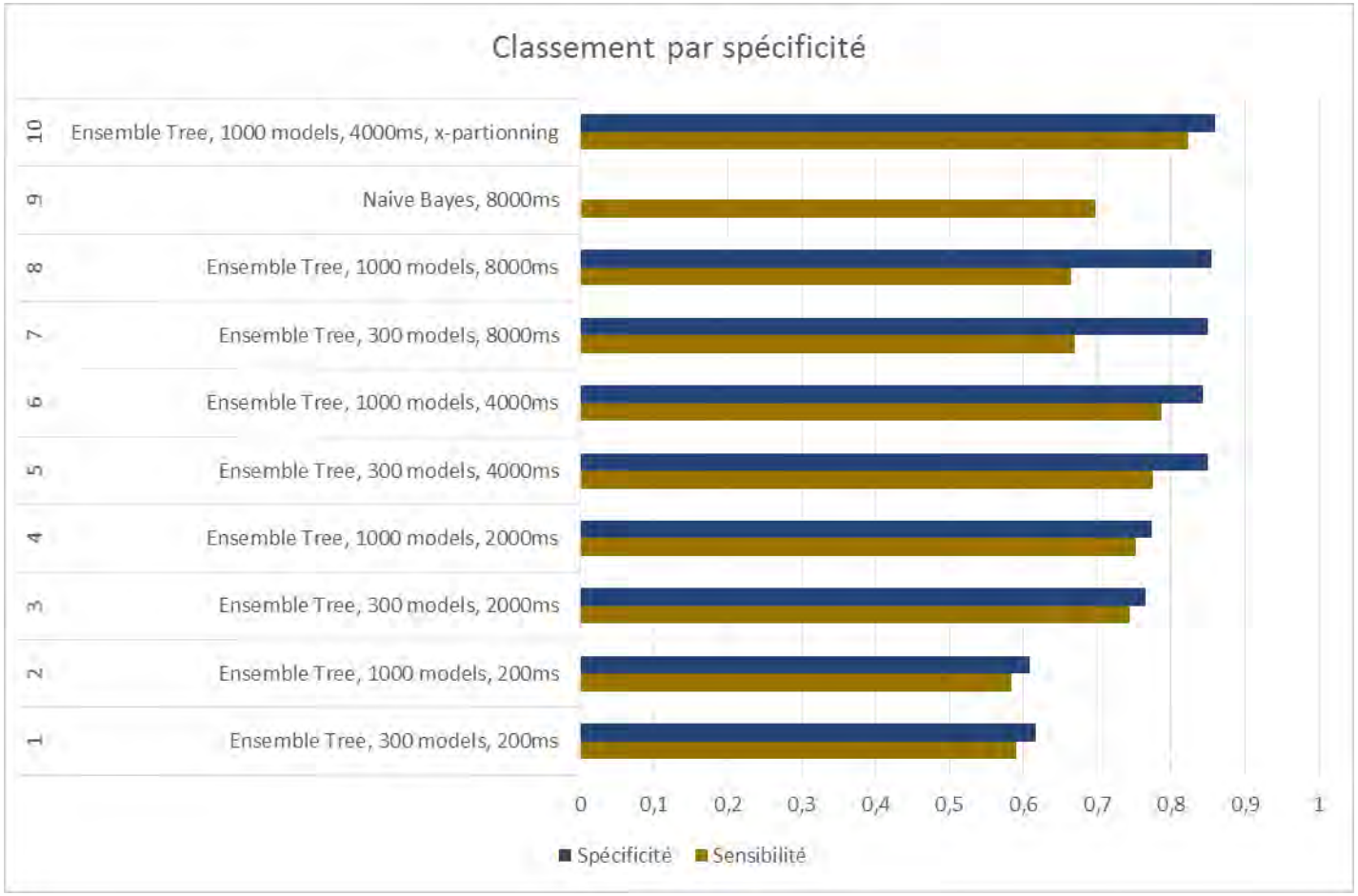
L'augmentation du nombre de neurones se reflète dans les scores obtenus avec MLP. A droite 3 neurones, à gauche dix fois plus. La progression est de 7.68%.

ITÉRATION 2

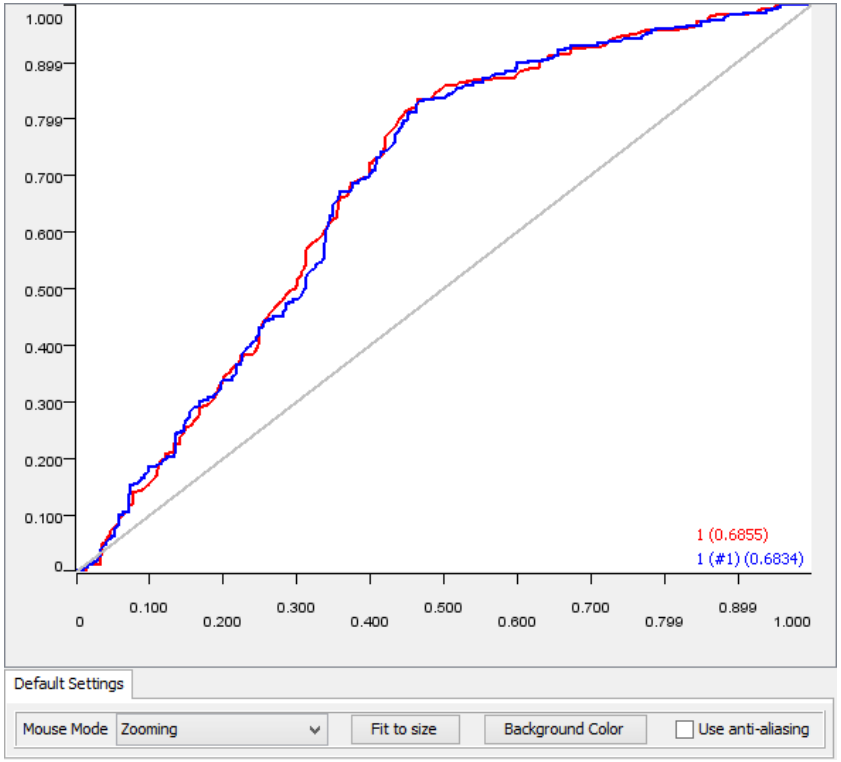
RÉSULTATS DES ESSAIS

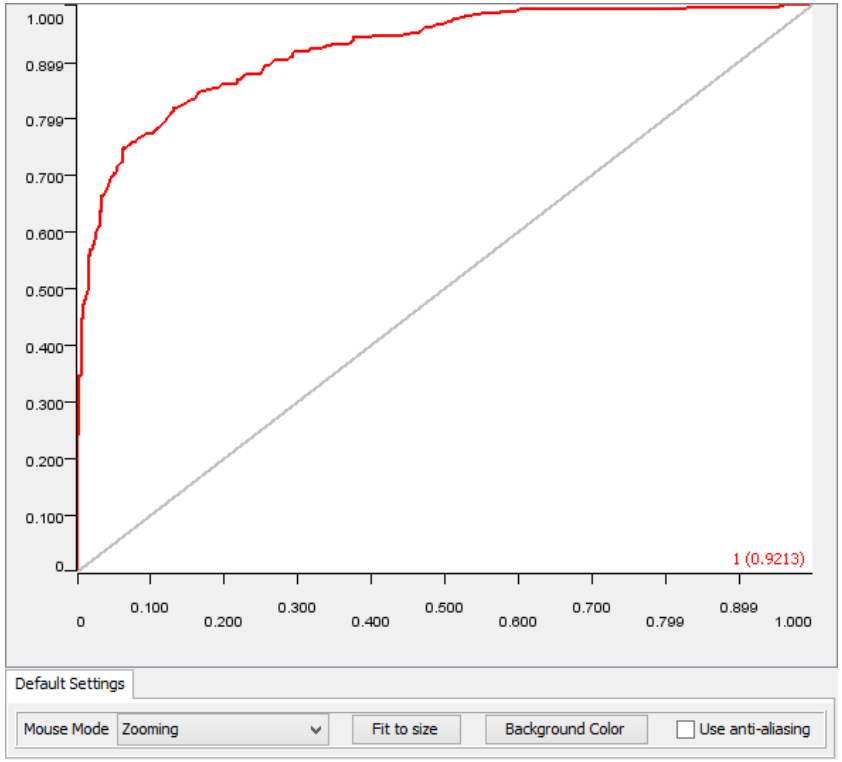
No	Nom	Classés juste	Classés faux	Précision	Erreur	VP	FN	VN	FP	K de Cohen	Sensibilité	Spécificité	AUC
1	Ensemble Tree, 300 models, 200ms	409	265	60.682	39.318	156	253	108	157	0.201	0,590909091	0,617073171	0,6855
2	Ensemble Tree, 1000 models, 200ms	404	270	59.941	40.059	149	255	106	164	0.185	0,584313725	0,608591885	0,6834
3	Ensemble Tree, 300 models, 2000ms	451	146	75.544	24.456	197	254	68	78	0.507	0,743396226	0,765060241	0,851
4	Ensemble Tree, 1000 models, 2000ms	456	141	76.382	23.618	200	256	66	75	0.524	0,751879699	0,773413897	0,852
5	Ensemble Tree, 300 models, 4000ms	432	95	81.973	18.027	166	266	48	47	0.626	0,775700935	0,849840256	0,8993
6	Ensemble Tree, 1000 models, 4000ms	433	94	82.163	17.837	163	270	44	50	0.628	0,787439614	0,84375	0,8996
7	Ensemble Tree, 300 models, 8000ms	346	88	79.724	20.276	85	261	42	46		0,669291339	0,850162866	0,8732
8	Ensemble Tree, 1000 models, 8000ms	346	88	79.724	20.277	87	259	44	44		0,664122137	0,854785479	0,8752
9	Naive Bayes, 8000ms	303	131	68.816	30.184	303	0	131	0	0	0,698156682		
10	Ensemble Tree, 1000 models, 4000ms, x-partionning	742	136	84.51	15.49	279	463	60	76		0,82300885	0,858998145	0,9213

CLASSEMENT DES ESSAIS



COURBES ROC (PARTIEL)

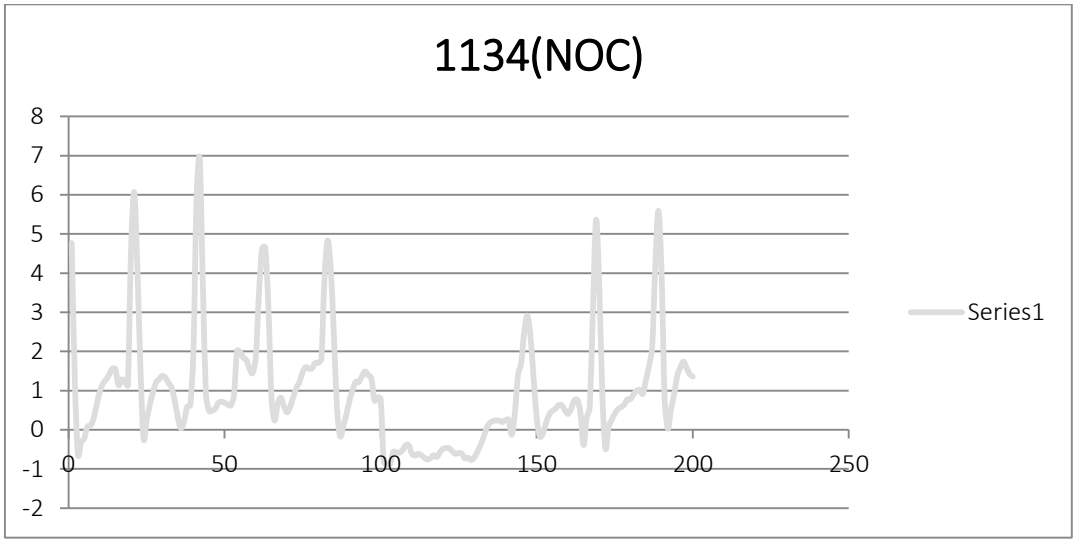


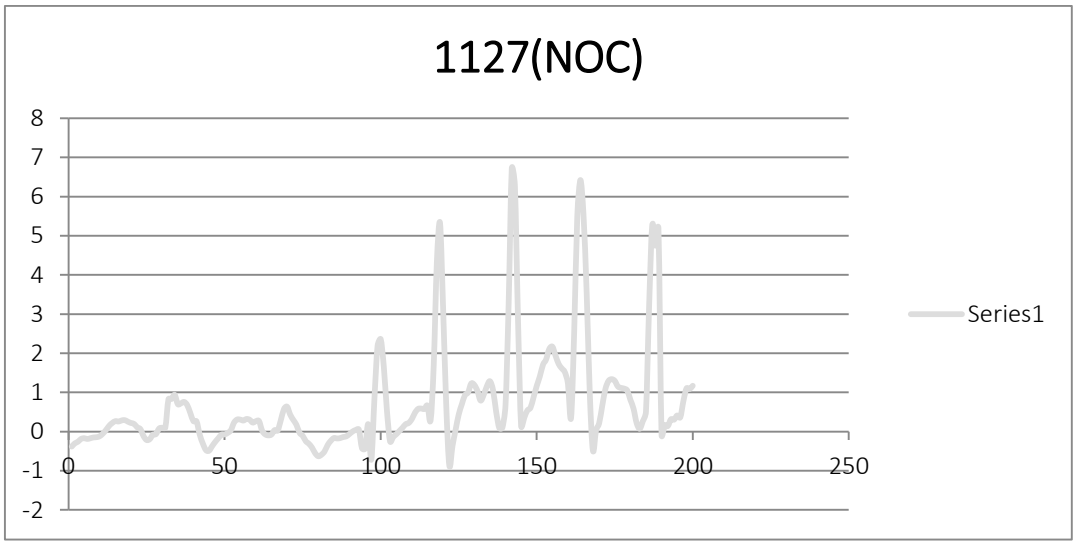


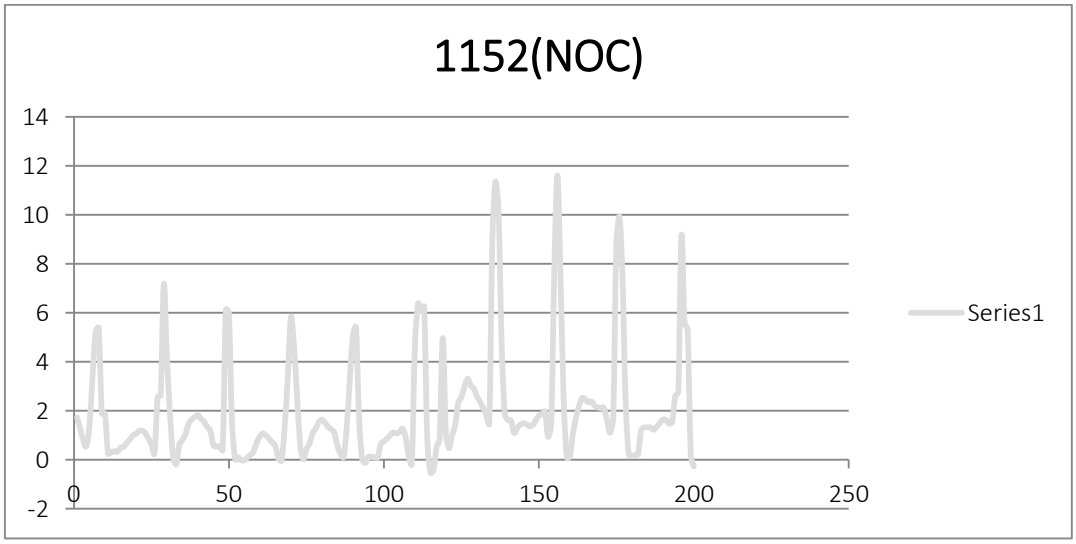
Progression des capacités de détection en fonction du nombre d'enregistrements conservés ; dans l'image de droite, le modèle est entraîné avec 10 enregistrements – 200 ms - seulement. A gauche, la courbe ROC est obtenue avec 200 enregistrements – soit une durée de 4 secondes -. La progression est de 23.58%.

GRAPHIQUES D'ACTIVITÉS NORMALES

Les trois graphiques suivants sont représentatifs des activités de type « course » avec des pics réguliers et de grande amplitude.







SCRIPT DE MISE EN ŒUVRE DU PIVOT DES DONNÉES

Le langage utilisé est Python.

```
import sys

## Rows iterator
iterator = inData0.iterator()

## Variables
values = ''
rowValues = []
counter = 1
fall = 0

## Get columns
tableSpec = inData0.getDataTableSpec()
svmIndex = tableSpec.findColumnIndex('svm')
fallIndex = tableSpec.findColumnIndex('fall')
recordsetIndex = tableSpec.findColumnIndex('recordset')
sampleSizeIndex = tableSpec.findColumnIndex('local.sampleSize')
activityIndex = tableSpec.findColumnIndex('activity')

## Iterate over inputed rows
while iterator.hasNext():
    counter = counter + 1
    row = iterator.next()
    sampleSize = row.getCell(sampleSizeIndex).getIntValue() + 1
    values = values + str(row.getCell(svmIndex).getDoubleValue()) + ','

    ## Get MAX value for the fall column
    rowFall = row.getCell(fallIndex).getIntValue()
    if rowFall > fall:
        fall = 1

    ## N records treated => new row
    if counter % sampleSize == 0:
        values = values[0:-1]
        rowValues.append(IntCell(fall))
        rowValues.append(StringCell(row.getCell(activityIndex).getStringValue()))
        rowValues.append(StringCell(values))
        rowValues.append(IntCell(row.getCell(recordsetIndex).getIntValue()))
        rowKey = row.getKey()
        newRow = DefaultRow(rowKey, rowValues)
        outContainer.addRowToTable(newRow)
        values = ""
        rowValues = []
        counter = 1
        fall = 0
```

SCRIPT DE MISE EN ŒUVRE DE LA FFT

Le langage utilisé est Java.

```
import math.transform.jwave.*;
import math.transform.jwave.handlers.*;

Transform t = new Transform(new DiscreteFourierTransform());

Double[] my_liste = new Double[2 * c_aggregatedSVM.length];
double[] sliste = new double[2 * c_aggregatedSVM.length];

int j = 0;
for (int i = 0; i < c_aggregatedSVM.length; i++)
{
	sliste[j] = (double) (c_aggregatedSVM[i]);
	j = j + 1;
	sliste[j] = (double) 0;
	j = j + 1;
}

sliste = t.forward(sliste); // 1-D DFT forward

for (int i = 0; i < sliste.length; i++)
{
	my_liste[i] = (Double) sliste[i];
}

out_aggregatedSVM = my_liste;
```

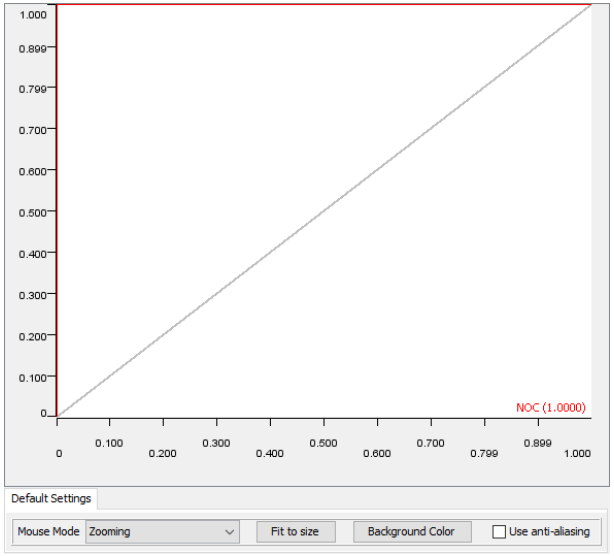
ITÉRATION 3

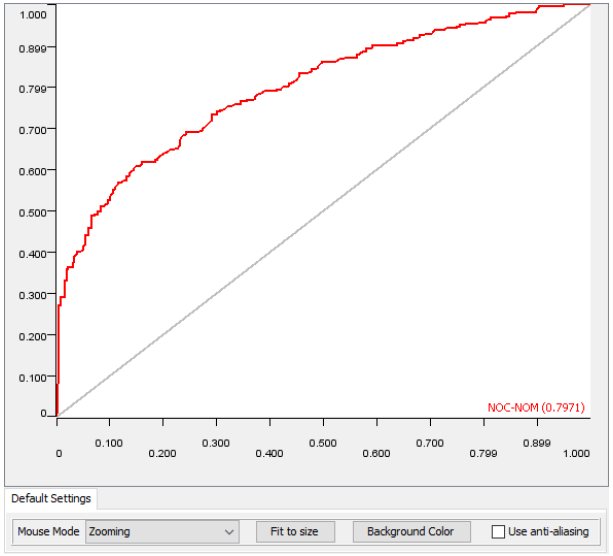
RÉSULTATS DES ESSAIS

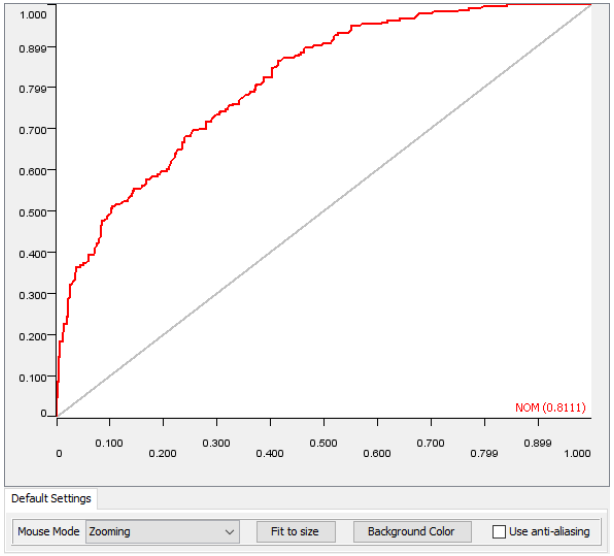
No	Nom	Classés juste	Classés faux	Précision	Erreur	VP	FN	VN	FP	K de Cohen	AUC	Sensibilité	Spécificité	Modèle 1 AUC
23	Double modèle, 200 enregistrements. Course; tree ensemble, 1000 modèles, FFT 10 coefficients. Chute; tree ensemble, 1000 modèles	557	80	87,441	12,559	461	96	55	25	0,627	0,9102			0,9985
24	Double modèle, 200 enregistrements. Course; tree ensemble, 1000 modèles, FFT 25 coefficients. Chute; tree ensemble, 1000 modèles	557	80	87,441	12,559	460	97	54	26	0,629	0,9161			0,9996
25	Double modèle, 200 enregistrements. Course; tree ensemble, 1000 modèles, FFT 75 coefficients. Chute; tree ensemble, 1000 modèles	557	80	87,441	12,559	459	98	53	27	0,631	0,9104			1
26	Double modèle, 200 enregistrements. Course; tree ensemble, 1000 modèles, FFT 50 coefficients. Chute; tree ensemble, 1000 modèles	554	83	86,97	13,03	459	95	56	27	0,614	0,9146			1
31	Tree ensemble, 300 modèles, 200 enregistrements, 5 coefficients FFT	560	77	87,912	12,088	466	94	57	20	0,635	0,9237			N/A
32	Tree ensemble, 300 modèles, 200 enregistrements, 50 coefficients FFT	558	79	87,598	12,402	464	94	57	22	0,627	0,9225			N/A
33	Tree ensemble, 300 modèles, 200 enregistrements, 7 coefficients FFT	562	75	88,226	11,774	466	96	55	20	0,646	0,92			N/A
34	Tree ensemble, 300 modèles, 200 enregistrements, 4 coefficients FFT	562	75	88,226	11,774	463	99	52	23	0,651	0,9216			N/A
35	Tree ensemble, 1000 modèles, 200 enregistrements, 7 coefficients FFT	561	76	88,069	11,931	465	96	55	21	0,642	0,9218			N/A

36	Tree ensemble, 1000 modèles, 200 enregistrements, 75 coefficients FFT	564	73	88,54	11,46	465	99	52	21	0,695	0,9226	N/A
37	Tree ensemble, 1000 modèles, 200 enregistrements, 4 coefficients FFT	565	72	88,697	11,303	465	100	51	21	0,665	0,9166	N/A
38	Tree ensemble, 1000 modèles, 200 enregistrements, 5 coefficients FFT	562	75	88,226	11,774	464	98	53	22	0,65	0,9247	N/A
39	Tree ensemble, 1000 modèles, 200 enregistrements, 6 coefficients FFT	563	74	88,383	11,617	466	97	54	20	0,652	0,9239	N/A
40	Tree ensemble, 1000 modèles, 200 enregistrements, 8 coefficients FFT	561	76	88,069	11,931	466	95	56	20	0,641	0,9215	N/A
41	Tree ensemble, 1000 modèles, 200 enregistrements, 9 coefficients FFT	562	75	88,226	11,774	464	98	53	22	0,65	0,9234	N/A
42	Tree ensemble, 1000 modèles, 200 enregistrements, 10 coefficients FFT	555	82	87,127	12,873	465	90	61	21	0,608	0,9231	N/A
43	Tree ensemble, 300 modèles, 200 enregistrements, 5 coefficients FFT	557	80	87,441	12,559	462	95	56	24	0,625	0,9211	N/A

Courbes ROC de détection des activités







Courbes ROC obtenues pour la détection des activités. A gauche, on constate que la détection d'activités de type « course » donne des résultats excellents puisque le score AUC est de 1. Au centre, la combinaison des activités de type « course » et « marche » produit un score de 0.7971. Ces deux activités ne sont donc pas très proches en termes de valeurs enregistrées. Pour finir, à droite, la courbe ROC spécialisée pour la détection des activités de marche montre un taux relativement modeste de détection avec une un score AUC de 0.8111.

Scores FFT

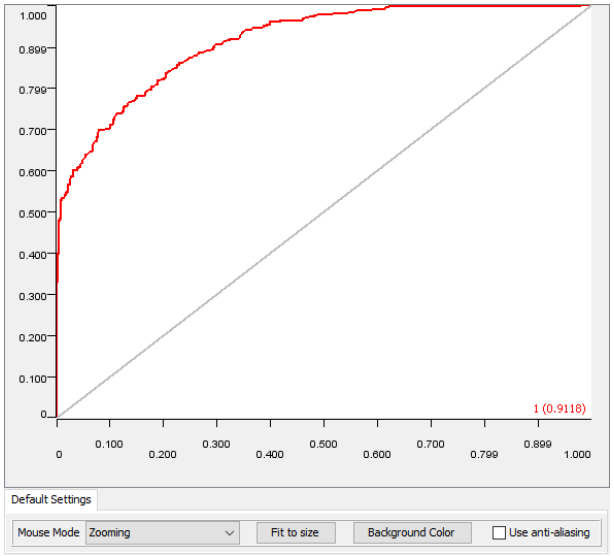
Nb frequencies	Accuracy	Error	Correct	Wrong
4	41,429	58,571	261	369
5	46,349	53,651	292	338
6	46,667	53,333	294	336
7	47,619	52,381	300	330
8	45,397	54,603	286	344
9	48,095	51,905	303	327
10	48,095	51,905	303	327
25	52,381	47,619	330	300
50	52,698	47,302	332	298
75	53,81	46,19	339	291
125	51,746	48,254	326	304

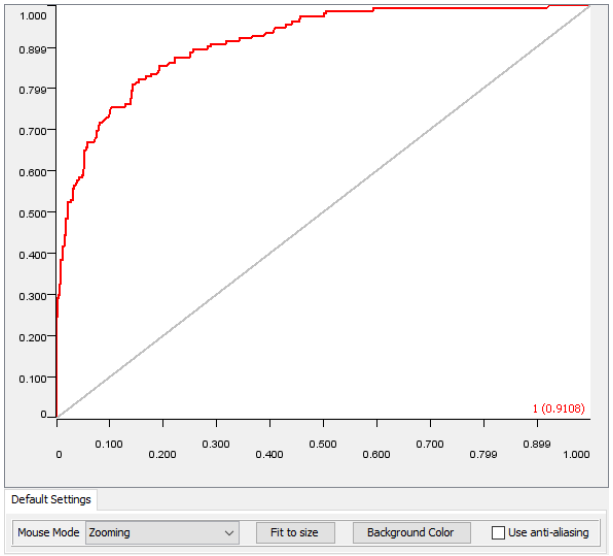
Score FWT

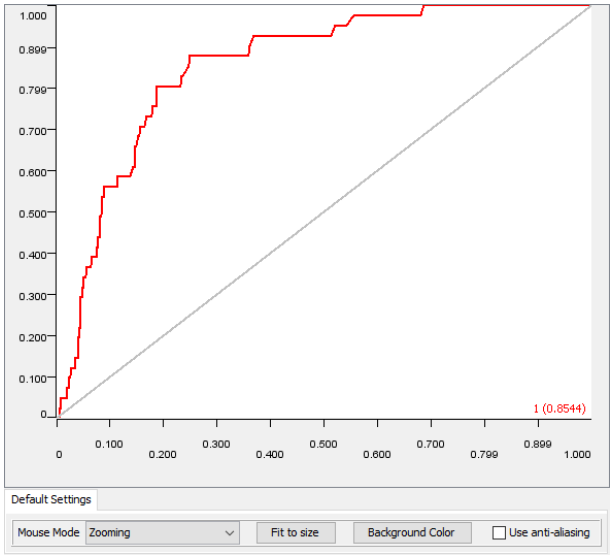
Nb frequencies	Accuracy	Error	Correct	Wrong
4	39,841	60,159	251	379
5	43,492	56,508	274	365
6	43,016	56,984	271	359
7	44,762	55,238	282	348
8	44,921	55,079	283	347
9	46,19	53,81	291	339
10	45,714	54,286	288	342
25	53,958	46,032	340	290
50	54,762	45,238	345	285
75	52,381	47,619	330	300
125	51,905	48,095	327	303

ITÉRATION 4

COURBES ROC







Courbes ROC obtenues pour déterminer le nombre d'enregistrements - soit la durée - le plus représentatif pour le classement des cas de chute. Les durées sont respectivement de 2 secondes à gauche, 4 secondes au milieu et 8 secondes à droite avec des scores de 0.9118, 0.9108 et 0.8544.