Computer Science
&
Data Science

Groupe 71

Machine Learning for Detection of Early-Stage Parkinson using Gait Analysis

Ulysse Aubin,
Clément Limanton,
Tom Salembien,
Yann Vastel

supervisé par
Mounim El Yacoubi & Dijana Petrovska

Contents

1 Introduction

1.1 Présentation de la maladie de Parkinson

En 1817, le docteur anglais James Parkinson publie sa fameuse monographie, "An essay on the shaking palsy", dans laquelle il étudie le comportement de 6 individus (dont trois inconnus trouvés dans la rue) qui présentent des symptômes moteurs que l'on associe aujourd'hui à la maladie qui porte son nom. Parkinson meurt sept ans plus tard, mais ses travaux sont repris par plusieurs scientifiques au cours du XIXè siècle, dont notamment une jeune français Jean-Martin Charcot, qui réussit à dissocier et catégoriser les différents symptômes de la maladie et lui donne le nom de maladie de Parkinson. Depuis le XXe siècle, de nombreuses avancées ont été faites sur la recherche de ses causes et les traitements à disposition.

La maladie de Parkinson est une maladie neurologique chronique dégénérative du système nerveux central. Aujourd'hui, elle est le second trouble neurodégénératif le plus fréquent, comptant plus de 200,000 malades en France. Les symptômes principaux atteignent la motricité du malade et se révèlent progressivement, puis lors des phases les plus avancées de la maladie il est commun de retrouver des symptômes cognitifs. Les symptômes les plus courants sont les tremblements, la bradykinésie (lenteur des mouvements), ou encore des difficultés pour marcher. Parmi les symptômes d'ordre cognitif on retrouve la dépression, l'anxiété ou encore l'apathie. La cause principale est le vieillissement, l'âge moyen au moment du diagnostic étant de 58 ans. Cependant, certains cas peuvent se révéler dès la jeunesse. La maladie est notamment connue pour avoir attaqué le boxeur Muhammad Ali dès ses 38 ans, causant sa mort des années plus tard. Les hommes ont 1,5 fois plus de chances d'avoir la maladie que les femmes, et la durée de vie moyenne suivant le diagnostic est de 7 à 15 ans.

Les causes exactes de la maladie restent à découvrir. Cependant, on soupçonne à la fois des facteurs génétiques et environnementaux. Une étude menée en 1994 [1] a trouvé qu'environ 15,8% des malades ont au moins un parent, un frère ou une sœur également atteint de la maladie, contre 4,4% pour la population saine, ce qui confirme la très forte probabilité de facteurs génétiques. Depuis, les scientifiques ont trouvé des liens entre la maladie de Parkinson et la mutation d'au moins 20 gènes différents. En ce qui concerne les facteurs environnementaux, les pesticides sont souvent mentionnés ainsi que les traumatismes crâniens. Parmi les autres causes potentielles on peut trouver les infections ou certains produits chimiques. De façon surprenante, les facteurs identifiés réduisant le risque de Parkinson sont la consommation de tabac et de caféine.

Plusieurs approches ont été adoptées pour essayer de définir des sous-classes de Parkinson, en utilisant par exemple l'âge à l'apparition des symptômes, la vitesse de progression de la maladie, ou les types de symptômes prédominants. L'article [2], datant de 2012, fait un état des lieux sur ces différentes approches, qui sont résumées dans le tableau suivant (Figure 1). Cependant, aucune de ces méthodes n'a été démocratisée ou retenue par la communauté scientifique.

Table 1 Parkinson's disease subtypes identified by data driven studies

Author, year	Subtypes identified
Graham 1999[3]	Short duration (mean 5 years): 1. Good motor control without cognitive impairment 2. Good motor control, executive cognitive deficits 3. Older age at onset, poor motor control + complications, mild cognitive impairment Longer duration (mean 14 years): 1. Poor motor control, no cognitive impairment 2. Poor motor control, moderately severe cognitive impairment
Gasparoli 2002[4]	1. Rapid progression 2. Slow progression
Dujardin 2004[5]	1. Mild motor impairment, relatively preserved cognition 2. 'Reduced overall cognitive efficiency', subcorticofrontal syndrome and more severe motor dysfunction
Lewis 2005[6]	1. Young onset 2. Non-tremor dominant, cognitive impairment and depression 3. Rapid progression without cognitive impairment 4. Tremor dominant
Schrag 2006[7]	1. Young onset 2. Older onset, more rapid progression, less dyskinesias and fluctuations
Post 2008[8]	1. Young onset with slow progression 2. Intermediate age onset with anxiety and depression 3. Oldest onset
Reijnders 2009[9]	1. Rapid progression 2. Young onset with motor complications 3. Non-tremor dominant and psychopathology 4. Tremor dominant
Van Rooden 2011[10]	1. Mild all domains, young 2. Severe motor complications, sleep and depressive symptoms, youngest 3. Medium severity, older 4. Most severe, except mild tremor, prominent motor complications, older
Liu 2011[11]	1. Non-tremor dominant 2. Rapid disease progression 3. Young onset 4. Tremor dominant

Figure 1: Sous-classes proposées

Néanmoins, il existe une échelle de référence pour décrire le stade d'avancement de la maladie chez un patient, nommée la UPDRS. Cette dernière a été mise à jour en 2007 pour donner la MDS-UPDRS (Figure 2). Le terme prodromal est utilisé à travers les différentes études et bases de données pour désigner les phases prématurées de la maladie, ou les patients montrent certains signes mais n'ont pas encore pleinement développé les symptômes.

Actuellement, la maladie de Parkinson ne se guérit pas mais se traite afin de ralentir et limiter au maximum ses symptômes. La seule mesure préventive fiable est l'exercice physique régulier, notamment à partir de 35 ans. Les différents traitements entrent dans trois catégories : les médicaments, les traitements physiques, et les opérations dans certains cas avancés. Les médicaments principaux utilisés sont de la L-DOPA couplé avec certains inhibiteurs pour optimiser

MDS-UPDRS	Method			Triangulation cut-off values
	Centile 90	ROC	OLR	
Part 1				
Mild/moderate	11/12	8/9	11.4/11.5	10/11
Moderate/severe	20/21	23/24	20.5/20.6	21/22
Part 2				
Mild/moderate	12/13	12/13	13.5/13.6	12/13
Moderate/severe	28/29	30/31	29.2/29.3	29/30
Part 3				
Mild/moderate	35/36	27/28	33.4/33.5	32/33
Moderate/severe	57/58	57/58	60.5/60.6	58/59
Part 4				
Mild/moderate	3/4	4/5	4.5/4.6	4/5
Moderate/severe	11/12	13/14	11.4/11.5	12/13

ROC: receiver operating characteristic analysis.
OLR: ordinal logistic regression.
MDS-UPDRS: Movement Disorder Society-Unified Parkinson's Disease Rating Scale.

Figure 2: MDS-UPDRS

son efficacité (souvent la COMT). Chez les patients jeunes, il est parfois préférable d'utiliser des agonistes dopaminergiques pour retarder les effets secondaires de la L-DOPA (mouvements anormaux). Les traitements physiques consistent essentiellement en des séances de kinésithérapie avec des exercices physiques à pratiquer pour préserver la souplesse chez les patients rigides, ainsi que les capacités motrices. Dans certains cas, nous pouvons faire recours à une opération, en utilisant une technique nommée la stimulation cérébrale profonde. Cette technique consiste à installer un neurostimulateur dans le cerveau, qui va envoyer des signaux électriques à certaines parties ciblées du cerveau. L'efficacité du traitement global dépend de manière cruciale du stade auquel la maladie est diagnostiquée.

1.2 Le diagnostic : là où s'inscrit notre projet

Le diagnostic de la maladie de Parkinson se fait principalement par un médecin à partir de l'analyse de ses symptômes, en suivant plusieurs tests développés et avec un analyse de l'historique médical du patient. Cependant, les diagnostiques de la maladie restent assez inefficaces. En effet, le diagnostic peut uniquement être confirmé à posteriori avec la pratique d'une autopsie, qui révèle la présence ou non de corps de Lewy dans le cerveau. Une étude menée en 2013 [3] à conclu que l'efficacité du diagnostic est autour de 80% et ne s'améliore pas sur les 25 dernières années. Le diagnostic devient d'autant plus compliqué chez les patients âgés, où le nombre de faux positifs augmente car on peut confondre certains symptômes avec d'autres maladies, et les

cas de cumul de maladies sont plus courants.

De plus, des examens médicaux complémentaires sont souvent effectués, notamment pour la départager de maladies similaires, comme le tremblement essentiel, la maladie à corps de Lewy (MCL), ou encore la paralysie supranucléaire progressive (PSP). On peut par exemple effectuer une IRM cérébrale, un DATscan, ou encore un scanner cérébral.

Le coût du diagnostic devient donc assez élevé. Il demande des heures auprès d'un spécialiste, et les examens d'imagerie cités environnent les 300 euros en France en plus d'être intrusifs. Il serait donc souhaitable de développer une méthode premièrement non intrusive et plus accessible à tous, et deuxièmement qui permette un diagnostic le plus prématuré possible pour optimiser l'efficacité du traitement.

Ce projet vise ainsi à travailler sur des algorithmes de Machine Learning capable de détecter les phases prématurées de la maladie de Parkinson, à partir de données sur la marche des patients. En effet, des dispositifs peuvent être développés pour récolter des donnés sur la marche de façon peu coûteuse et non intrusive pour un utilisateur, grâce notamment aux accéléromètres des smartphones. Nous pouvons également espérer stratifier les patients (les catégoriser en sous-types), pour différencier les malades des patients en état prodromal par exemple, ou estimer leur état sur l'échelle MDS-UPDRS.

Un autre objectif de ce projet est de développer des algorithmes pour prédire le Freezing of Gait (FoG) : une absence brève et épisodique ou une réduction marquée de la progression vers l'avant des pieds malgré l'intention de marcher.

2 Analyse de l'état de l'art

2.1 Collection des données

Avant de pouvoir faire tourner des algorithmes de Machine Learning performants, il faut avoir des bases de données pertinentes. Les données sur la démarche ont d'abord été identifiées comme des mesures prometteuses en biométrie pour identifier des individus. La démarche à l'avantage de pouvoir être reconnue de loin, par rapport à d'autres méthodes telles que la reconnaissance faciale. Ces données sont prélevées à l'aide de trois types de capteurs : les capteurs de couleur, les capteurs de profondeur, et les capteurs inertiels, chacune présentant ses avantages et ses inconvéniants.

Les capteurs de couleur, tels qu'un capteur photographique CCD ou une caméra classique,

génèrent des imagent RGB en deux dimensions de la personne sur un cadre. L'analyse de la démarche est souvent faite depuis la silhouette de l'individu que l'on construit à partir des images. Le manque de troisième dimension de cette méthode limite de manière considérable les données, et distinguer la silhouette peut devenir compliqué en fonction du fond de l'image.

Les capteurs de profondeur, eux, ont l'avantage de pouvoir analyser en 3 dimensions et identifier plus facilement un individu qui marche, surtout si rien ne bouge dans le fond. Parmis ces capteurs on utilise souvent la Windows Kinect, un produit Microsoft développé à l'origine pour les jeux vidéos en 2010, mais qui depuis est devenu une référence pour d'autres utilisations, notamment grâce à son software development kit. Malheureusement, ces données contiennent en pratique beaucoup de bruit et de fausses données notamment sur des surfaces réfléchissantes dont certaines matières d'habits.

Enfin, comme capteurs inertiels on retrouve des accéléromètres et des tapis sensoriels. L'avantage de ces capteurs est qu'ils sont facilement accessibles, car intégrés à tous les smartphones modernes. Cependant, les différents placements des téléphones dans les poches entraînent des variations d'orientation, ce qui nuit à la qualité des données enregistrées.

En somme, chaque type de capteur enregistre une partie des données sur la démarche et présentent ses propres incomplétudes. On remarque cependant que les données ont l'aire d'être complémentaires, entre des données en 2 et en 3 dimensions, à distance et sur le patient, etc. Plusieurs études [4] ont été menées qui montrent que les approches qui mélangent les trois types de capteurs obtiennent des résultats nettement plus performants.

2.2 Bases de données existantes

Plusieurs bases de données ont été constituées en utilisant ces différents capteurs pour les données sur la démarche des patients atteints de Parkinson et pour identifier le FoG. Voici un résumé des quelques unes qui sont souvent référencées dans les articles des travaux déjà effectués à ce sujet.

PhysioNet : Gait in Parkinson's Disease [5]. Cette base de données utilise 8 capteurs de force sous chaque pied des sujets qui prélèvent des données à une fréquence de 100 Hz. Elle contient 93 sujets atteints de la maladie et 73 sujets sains. On demande aux sujets de marcher à leur rythme pendant 2 minutes pour prélever les données de force verticale.

PhysioNet : Gait in Neurodegenerative Disease Database [6]. Cette base de données à été créée pour distinguer plusieurs maladies neurodégénératives similaires. Elle contient 15 sujets avec la maladie de Parkinson, 20 avec la maladie de Huntington, 13 avec une sclérose latérale

amyotrophique, et 16 sujets sains. Elle mesure 13 données différentes sur la marche des différents sujets.

Multimodal Dataset of Freezing of Gait in Parkinson's Disease [7]. Dans cette base de données, 3 heures et 42 minutes de données ont été enregistrées à partir de 12 patients atteints de Parkinson, où des docteurs ont labellisé les occurrences de FoG. Parmi les données relevées on compte des électroencéphalogrammes (EEG), électromyogrammes (EMG), electrocardiograms (ECG), la conduction cutanée (SC), et l'accélération (ACC).

Daphnet Freezing of Gait Data Set [8] : Cette base de données labélise également des instance de FoG à partir uniquement des données relevées par des accéléromètres à la cheville, la cuisse et dans le dos. Elle compte seulement 9 sujets parkinsoniens, mais a été reprise dans le cadre de plusieurs études.

2.3 Travaux réalisés à ce sujet

Plusieurs travaux de Machine Learning pour classification de Parkinson ont déjà été menés. De telles études ont pu se baser sur des types de données très différentes, notamment sur la voix, ou encore des données de graphisme obtenues avec des tablettes. Certaines ont déjà été menées à partir de données sur la marche, mais la majorité de ces travaux n'utilisent pas encore un mélange des différents types de capteurs.

L'amélioration des techniques pour relever des données gait à été accompagnée des études utilisant ces données pour classifier les maladies neurodégénératives. Par exemple, l'étude [9] est réalisée à partir de 15 sujets atteints de la maladie Parkinson, et 16 sujets sains, et réussit à classifier avec une précision d'environ 90% pourcent les malades, avec la méthode leave-one-out.

En 2021, [10] parviennent à comparer l'éfficacité de nombreuses méthodes différentes pour classer des sujets ayant Parkinson ainsi que d'autres maladies neurodégénératives et des sujets sains, et propose sa propre méthode qui utilise plusieurs types de capteurs (figure 3). Elle parvient ainsi à un classifier 99.86% des sujets correctement. Elle réussit aussi dans un second temps à départager des patients ayant 3 degrés de gravité différents (2, 2.5 et 3 sur la MDS-UPDRS) (figure 4). Encore une fois, la méthode proposée dans l'article classe 98.93% des sujets correctement. On voit clairement que les approches "multi-modal" qui mélangent les types de donnés exploités (capteurs de couleur, de profondeur et inertiels) sont plus performants dans ces deux domaines.

Single-modal Methods	ALS vs. CO	PD vs. CO	HD vs. CO	NDDs vs. CO
RBF+DL [10]	89.66%	87.10%	83.33%	-
QBC [19]	100%	80.00%	71.43%	-
Meta-classifiers [17]	96.13%	90.36%	88.67%	-
HMM [18]	-	90.32%	-	-
C-FuzzyEn+SVM [21]	-	96.77%	-	-
PE+SVM [22]	92.86%	-	-	-
Multi-modal Methods	ALS vs. CO	PD vs. CO	HD vs. CO	NDDs vs. CO
DCLSTM [24]	97.43%	97.33%	94.96%	96.42%
The proposed method	***100%***	***99.86%***	***99.74%***	***99.47%***

Figure 3: Efficacité des méthodes de classification des maladies neurodégénératives

Single-modal Methods	Accuracy	Multi-modal Methods	Accuracy
CapsNet	95.01%	DCCA	94.30%
HMM	92.20%	KCCA	92.48%
Q-BTDNN[20]	93.10%	DCLSTM[24]	96.71%
Original LSTM	91.13%	The proposed method	***98.93%***
CNN	82.86%		
LSTM+CNN[5]	97.48%		
GRU	92.53%		
BiLSTM	91.58%		

Figure 4: Efficacité des méthodes de classification des niveaux de parkinson

3 Les bases de données

3.1 Description

3.1.1 Le gel de la marche

L'une des conséquences de la maladie de Parkinson est le Freezing of Gait (FoG) ou en français "gel de la marche". Ce symptôme de la maladie ralenti fortement les mouvements voire rend les patients incapables de bouger pendant plusieurs secondes ou minutes. Les crises ne sont pas forcément prévisibles à l'avance. Ces crises peuvent être extrêmement dangereuses car elles stoppent tout mouvement de la personne malade pendant un long moment et cela peu arriver à n'importe quel instant. Il est donc nécessaire d'arriver à détecter ces instants de FoG ou de le des prévoir. La base de données que nous avons utilisé fait parti d'un travail "Multimodal Dataset of Freezing of Gait in Parkinson's Disease" [11] créée par une collaboration de différentes écoles et instituts de recherche en Chine.

La base de données [1]se constitue de 12 patients qui ont réalisé différentes tâches afin de détecter

[1]La base de données

les instants de crises FoG et les instants "normaux". Les patients avaient entre 57 et 81 ans avec une moyenne d'âge de 69.1 ans et étaient atteints de Parkinson depuis 1 à 20 ans avec une moyenne de 9.3 ans. [2] Chaque patient a du réaliser quatre tâches différentes et après étude vidéo de chacun des parcours, des médecins ont pu aider à labelliser les données récoltées en deux états:

- 0 : Non-FoG;
- 1 : FoG.

Les quatres tâches réalisées par chaque patient sont les suivantes:

Figure 5: Tâches 1-2

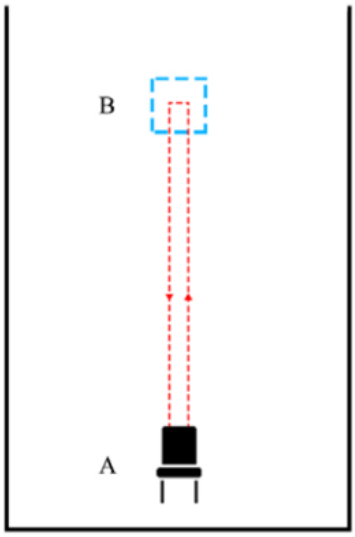

Figure 6: Tâches 3-4

Dans la première tâche, lorsque le patient est prêt il se lève de la chaise au point A puis tourne à droite en direction du point C. Un obstacle se trouve sur son chemin et il doit le contourner. Arrivé au point C, le patient fait un tour complet sur lui même puis traverse les obstacles 1, 2

[2]GitHub - Multimodal Parkinson Data Processing

et 3 jusqu'au point D. Il fait ensuite un dernier tour sur lui même puis retourne au point B (en passant par les obstacles 3 puis 2) et retourne enfin à sa chaise au point A. La seconde tâche consiste à répéter la première.

Lors de la troisième tâche le patient doit se lever de sa chaise au point A puis, arrivé au point B, il doit faire un tour sur lui-même dans un espace restreint délimité par un carré dessiné au sol. Il retourne ensuite au point A et la tâche se termine. La quatrième tâche consiste à répéter la troisième.

Lors de ces épreuves, les patients sont équipés de différents capteurs musculaires et cérébraux qui relèvent les accélérations des bras et jambes du patient ainsi que les réactions et comportements de certaines zones du cerveau. Les données ainsi obtenues sont ensuite filtrées tous les 500Hz, donc 0.02s. Sur la durée totale de l'expérience qui est de 222 minutes et 3 secondes, il y eu 334 crises FoG pour une durée de 88 minutes et 19 secondes. Les données cérébrales sont nommées en fonction de la partie du cerveau qui est étudiée selon la figure ci-dessous.

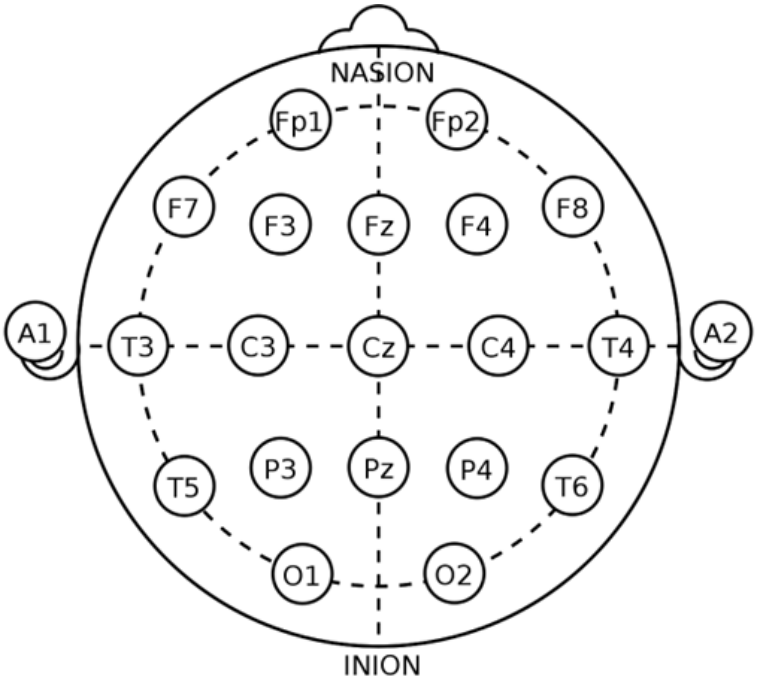

Figure 7: Emplacements des capteurs sur le cerveau

Ainsi, 60 données sont récoltées en tout:

- 1 colonne pour le temps (une unité de temps vaut 0.02s);
- 30 colonnes pour les signaux du cerveau;
- 28 colonnes de données d'accélérations des différents muscles;

- 1 colonne pour les labels (FoG / Non-FoG).

3.1.2 La base PPMI

La base de donnée de la Parkinson's Progression Markers Initiative (PPMI) provient d'une étude clinique observationnelle et longitudinale comprenant des évaluations de personnes atteintes de la maladie de Parkinson (MP), de patients au stade prodromal et de personnes en bonne santé. Lors de notre projet lorsque nous avons utilisé la base de données PPMI nous avons utilisé une sous étude de cette large dernière. La base de données utilisés, Gait ArmSwing comprend uniquement des données de Gait.

Le système de marche utilisé pour réaliser l'étude comprend trois capteurs portables sans fil légers contenant trois accéléromètre, gyroscopes et magnétomètres. Le système mesure l'accélération du mouvement dans trois axes orthogonaux en fonction du temps. Les capteurs sont portés sur les deux poignets et sur le bas du dos des participants pendant toutes les mesures de la marche. Les mesures sont effectués lors des visites annuelles des sujets du groupe de la cohorte génétique. L'évaluation comprend six tests : - Sway 30 secondes yeux ouverts - déplacement du centre de gravité en condition habituelle. - Sway 30 secondes yeux fermés - déplacement du centre de gravité dans des conditions difficiles. - Timed Up and Go (TUG) 1- évaluation de la mobilité, des transferts et des virages. - TUG 2- tâche répétée. - Marche habituelle - 1 minute - vitesse de marche souhaitée

On obtient en tout 56 mesures à partir de cette évaluation.

3.2 Cleaning des données

La base de donnée Gait ArmSwing a déjà subit un nettoyage par les scientifiques récoltant les données brutes, pour plus de précision voir [12]. Danes premier temps il est nécessaire de re-grouper les données de différents documents csv, en effet les labels (aussi appelé cohort) associés à l'ID de chacune des personnes se trouve sur un autre document. De plus le nom associé à une même date de test effectuer est parfois différents selon la personne (par exemple V06 et v06 sont similaires). Après une réorganisation nous obtenons un fichier de 56 mesures, l'identifiant, de la date de test, le nom du label en colonnes et des 192 personnes testées en lignes. Les labels sont appelé Cohort et sont associés à la valeur 1 pour l'état Podromal (stade avant parkinson durant lequel les personnes ne remplissent pas les critères de diagnostic de la MP, c'est-à-dire la

bradykinésie et au moins un autre signe moteur) et 3 pour un état Parkinsonien avancé.

Remarque : En réalité, on a considéré qu'une même personne effectuant deux tests à deux mois différents comme étant deux personnes différentes, les résultats pouvant être complètement différent d'un mois à l'autre (ce qui n'est pas gênant pour un travaille de classification).

Une fois notre base de travail relativement propre, nous réalisons des analyses statistiques pour s'approprier la base.

Dans un premier temps, on analyse la repartition dans le temps des tests selon le label, (rappel: $cohort\_1 \longrightarrow Podromal$ et $cohort\_3 \longrightarrow ParkinsonDisease$)

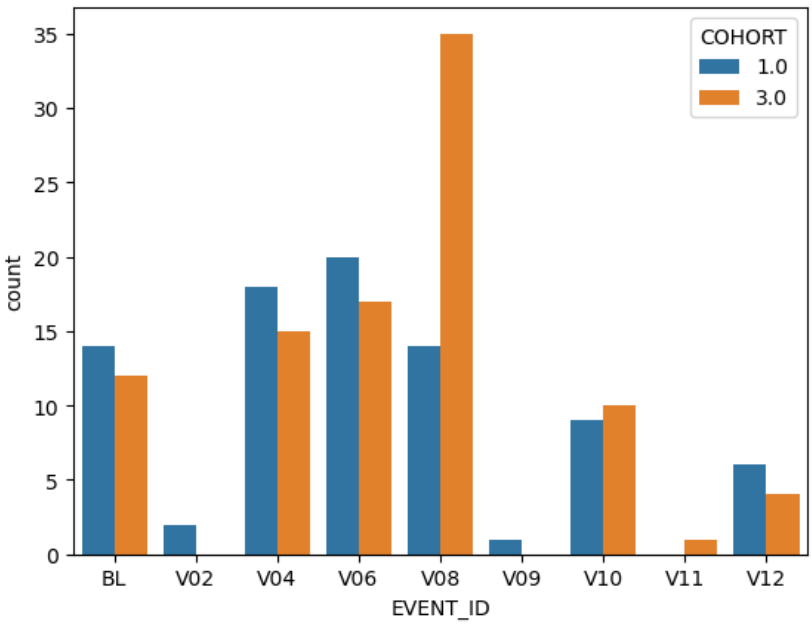

Figure 8: Répartition des test cliniques dans le temps

Ainsi, on compte 192 personnes différentes reparties de façon équilibré entre les deux classes, 96 personnes Podromal et 96 personnes PD.

Après observation des données, on remarque un certains nombre de Nan pour certaines personnes. Les figures suivantes présentes la répartition des Nan selon les 56 mesures et selon les 192 personnes.

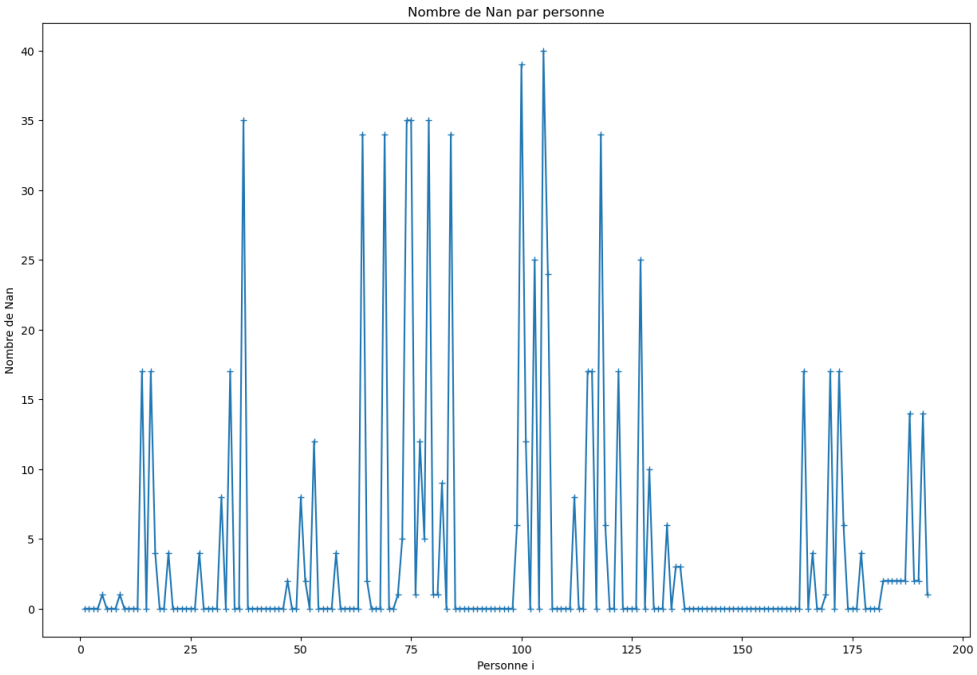

Figure 9: Nombre de Nan par personnes

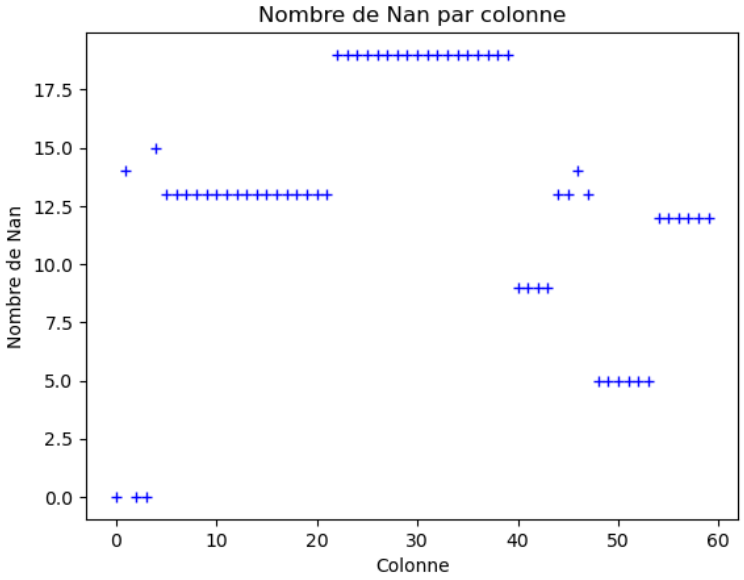

Figure 10: Nombre de Nan par mesure

On remarque, un nombre relativement important de Nan pour certaines personnes, et plus globalement sur l'ensemble des $192 * 56 = 10752$ données de notre base nous avons 783 Nan au total soit environ 7% de Nan.

Après analyse de l'état de l'art, il existe de nombreuses méthodes d'imputation (voir [3]) des données pour éviter perdre trop de données à cause des Nan, tels que :

- moyenne ou mode (sensible aux valeurs abbérantes)
- Hot Deck qui correspond à $\frac{\text{valeur tirée au hasard parmi valeurs existantes}}{\text{dernière valeur connue}}$
- Regression locale (LOESS)
- KNN (K-Nearest Neighbors)

Nous utiliserons le méthode KNN (K plus proche voisins) étant efficace en pratique. Nous prendrons 3 voisins pour ajuster notre imputation localement et limiter l'impact global des données importantes plus éloignés, de plus nous utiliserons la distance usuelle euclidienne.

Algorithme KNN

1. Choix d'un entier k : $1 \geq k \geq n$
2. Calculer les distances $d(Y_{i^*,Y_i})$, i = 1, . . . , n
3. Retenir les k observations $Y_{i_1}, ..., Y_{i_k}$ pour lesquelles ces distances sont les plus petites.
4. Affecter aux valeurs manquantes la moyenne des valeurs des k voisins :

$$(y_{ij})_{mis} = y_{i^*j^*} = \frac{1}{k}(Y_{i_1}, ..., Y_{i_k})$$

Ainsi, on obtient un base de données labelisée complète sur laquelle on peut travailler.

[3] Méthodes d'imputations de Nan

4 Différents modèles

4.1 Random Forest

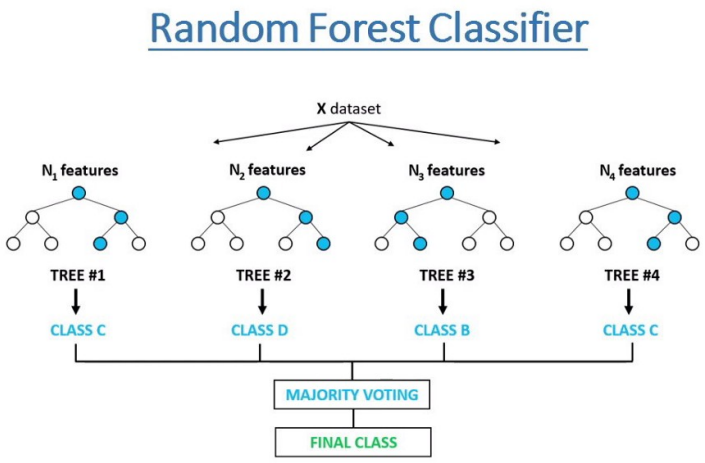

Figure 11: Schéma Random Forest

L'algorithme Random Forest est très utilisé car c'est une méthode non paramétrique qui ne donne pas d'estimateur sous une forme prédéterminée mais s'appuie uniquement sur les données d'entrées. On peut donc avoir des modèles plus variés et riches.

On se place dans un cas où nous avons des données $D_n = \{(X_i, Y_i)\}_{1 \leq i \leq n}$ où les X_i sont les données d'entrée et les Y_i les labels. On souhaite construire une fonction de régression $\widehat{f}_M$ pour prédire les observations associées à de nouveaux $x \in \mathbb{R}^d$. Un estimateur de type forêt aléatoire (Random Forest) est de la forme:

$$\widehat{f}_M : x \mapsto \frac{1}{M} \sum_{j=1}^{M} \widehat{f}_{j,n}(x, \eta_j)$$

où $\widehat{f}_{j,n}$ est "un arbre", n dépend des données $D_n = \{(X_i, Y_i)\}_{1 \leq i \leq n}$ et η_j représente toutes les variables aléatoires nécessaires à la construction de l'arbre. Il reste alors à définir comment on construit un arbre $\widehat{f}_{j,n}$ pour $1 \leq j \leq n$.

Pour mieux représenter graphiquement la construction de l'arbre, on supposera que les $(X_i)_{1 \leq i \leq n}$ sont dans $[0,1]^d$.

Dans les schémas ci-dessous, on décrit étape par étape la construction d'un arbre $\widehat{f}_{j,n}$:

Dans l'étape 0, on possède juste les X_i pour $1 \leq i \leq n$. A l'étape 1, on choisit aléatoirement

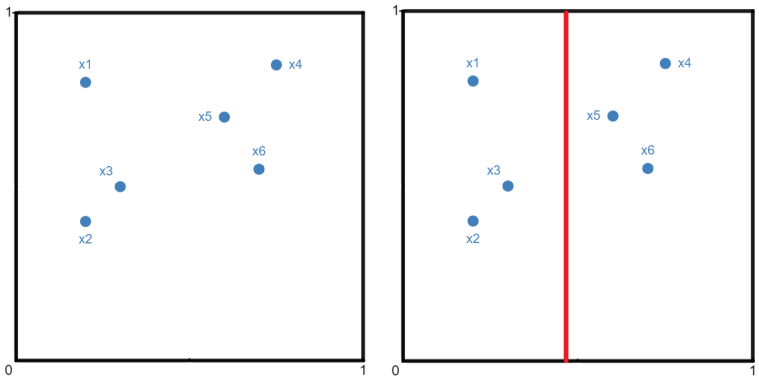

Figure 12: Etapes 0-1 du Random Forest

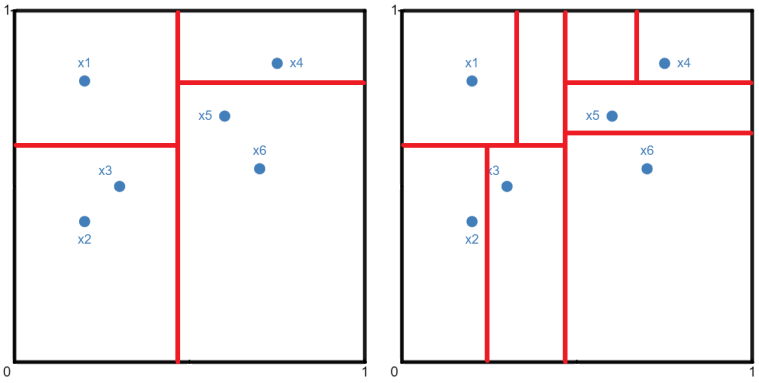

Figure 13: Etapes 2-finale du Random Forest

une dimension entre 1 et d et une position dans $[0,1]$. On partitionne alors l'espace en deux sous-espaces complémentaires.

On répète ensuite le processus dans chacun des sous-espaces ainsi obtenu comme montré en figure. A l'itération finale, on peut alors définir l'arbre

$$\widehat{f}_{j,n}(.,\eta_j) : x \mapsto \frac{1}{N_n(x)} \sum_{i=1}^{n} \mathbb{1}_{X_i \in A_n(x)} Y_i$$

où $A_n(x)$ est la cellule à laquelle appartient x et $N_n(x)$ compte le nombre de X_i dans $A_n(x)$.

La fonction $\mathcal{L}_A$ est appelée "critère CART" pour Classification And Regression Trees est définit de la façon suivante:

$$\mathcal{L}_A : (j,z) \mapsto \frac{1}{N_n(A)} \sum_{i=1}^{n} \left(Y_i - \overline{Y_A}\right)^2 \mathbb{1}_{X_i \in A} - \frac{1}{N_n(A)} \sum_{i=1}^{n} \left(Y_i - \overline{Y_{A_L}} \mathbb{1}_{X_i \in A} - \overline{Y_{A_R}} \mathbb{1}_{X_i \in A}\right)^2$$

où $A_L = \{x \in A | x_j < z\}$ et $A_R = \{x \in A | x_j > z\}$ et $\overline{Y_A}$ représente la moyenne des Y_i qui sont associés à des $X_i \in A$. Cette fonction est utilisée dans l'algorithme de Random Forest et en la maximisant on peut trouver la "meilleure" séparation de l'espace selon une erreur quadratique.

En pratique, chaque arbre est construit selon le pseudo-code suivant:

Algorithm 1 Pseudo Code Random Forest

Require: Un set de données $D_n = \{(X_i, Y_i)\}_{1 \leq i \leq n}$.

Initialisation:

On initialise $\mathcal{P} = \left\{[0,1]^d\right\}$

for $A \in \mathcal{P}$ **do**

On tire $n_{try} \in \{2, ..., d\}$ dimension au hasard : $\mathcal{B}_d$ dans η_j;

On tire au hasard a_n couple dans D_n;

On obtient alors $(j_\star, x_\star)$ qui maximise la fonction $(j, z) \mapsto \mathcal{L}_A(j, z)$ sur $j \in \mathcal{B}_d, z \in [0, 1]$ pour définir le découpage de A. On obtient alors A_1 et A_2;

On peut ensuite redéfinir $\mathcal{P} = (\mathcal{P} \backslash A) \sqcup A_1 \sqcup A_2$;

end for après l'atteinte d'un certain critère (nombre d'itération du découpage, précision du découpage, tous les X_i séparés...).

4.2 XGBoost

XGBoost (eXtreme Gradient Boosting) est une méthode de machine learning qui repose sur les arbres de décision et l'apprentissage d'ensemble séquentiel. Le modèle d'ensemble d'arbres consiste en un ensemble d'arbres de classification et de régression (CART : Classification and Regression Trees).

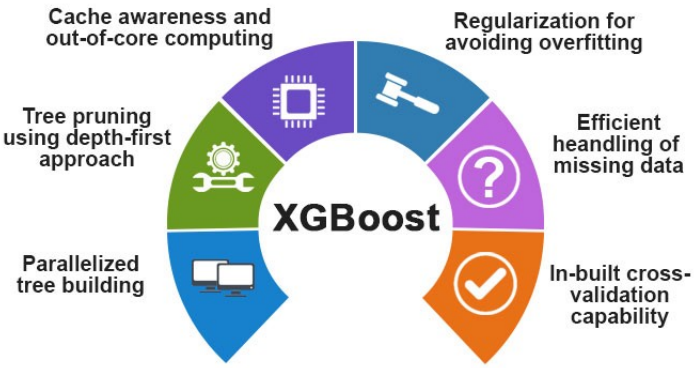

Figure 14: First Approach Random Forest

Pour comprendre le XGBoost, il est important de comprendre les principes du bagging et du boosting. Le bagging, ou bootstrap aggregating est une méthode ensembliste parallèle. Cela signifie que l'on va créer plusieurs sous bases de données depuis la base de données principale et entraîner chaque sous base indépendemment les unes des autres (en parallèle) et ensuite prendre la prédiction qui apparaît le plus souvent. Cette technique fonctionne en 3 étapes:

- 1. On crée un nombre B de sous-échantillons de l'ensemble de données avec remise (on peut avoir la même valeur dans différents échantillons mais pas deux fois la même valeur dans le même sous-échantillon);

- 2. On entraîne avec notre méthode choisie chacun des modèles pour en ressortir une prédiction par modèle;

- 3. On compte toutes les prédictions qui sont apparues et la plus fréquente sera le résultat final de notre modèle complet.

Le boosting fonctionne sur le même schéma que le bagging mais fait interagir les sous-échantillons entre eux. L'algorithme fonctionne de manière séquentielle, contrairement au random forest, et s'améliore ainsi à chaque agrégation.

- 1. On crée un premier modèle avec notre algorithme choisi puis on entraîne sur les données prises en compte. On attribue un poids égal à chaque observation. A partir de ces premiers résultats, on trouve les observations mal classifiées et on augmente le poids de ces dernières. Ceci permettra d'y faire plus attention lors des étapes suivantes (les erreurs seront remarquées par leur poids différent des valeurs bien classifiées);
- 2. On construit alors un second modèle pour corriger les erreurs précédentes. On entraîne ce nouveau modèle à l'aide des données pondérées que l'on a tiré de la première étape de l'algorithme. On répète alors cette étape jusqu'à ce que l'ensemble des données d'entraînement soit prédit correctement ou qu'un autre critère soit atteint.
- 3. Le modèle final sera alors le modèle obtenu avec les pondérations par les anciens modèles d'arbres.

En résumé, si on note $(e_i)_{1\leq i\leq B}$ les estimations après entraînement de chaque sous-échantillon et $(p_i)_{1\leq i\leq N}$ les poids associés à chaque estimation, l'estimation du modèle final sera donné par:
- $\frac{1}{B}\sum_{i=1}^{B} e_i$ en bagging;
- $\frac{1}{B}\sum_{i=1}^{B} p_i e_i$ en boosting.

L'algorithme de XGBoost est une version amélioré du boosting de gradient qui est cas particulier du boosting avec une minimisation des erreurs par descente de gradient. Pour plus de précision sur le fonctionnement de xgboost, voir [13].

Algorithm 2 Pseudo Code XGBoost

Require: On initialise les hyperparamètres du modèles (expliqués par la suite).

Initialisation:

On initialise le modèle avec une valeur constante $f_0(x)$

for $1 \leq m \leq M$ **do**

Calcul du pseudo résidu: $r_{im} = -\frac{\partial L(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)}$

Création d'un classifieur faible : h_m calibré sur les données (x_i, r_{im})

Poids de ce classifieur faible :

$$\gamma_m = argmin_\gamma \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \gamma * h_m(x_i))$$

On met à jour le modèle : $f_m(x) = f_{m-1}(x) + \eta\gamma_m h_m(x)$

Avec, $\begin{cases} \eta \text{ le taux d'apprentissage défini en entrée} \\ \gamma_m \text{ le poids associé au classifieur faible} h_m \end{cases}$

end for

4.3 Feed-Forward Neural Network (FFNN)

Nous travaillerons ici avec un FFNN pour une classification binaire : Podromal / PD. Notre réseau est composé d'un couche d'entrée (input layer), d'un ensemble de couches cachées (hidden layers) et enfin d'un couche de sortie (output layer sur la figure suivante).

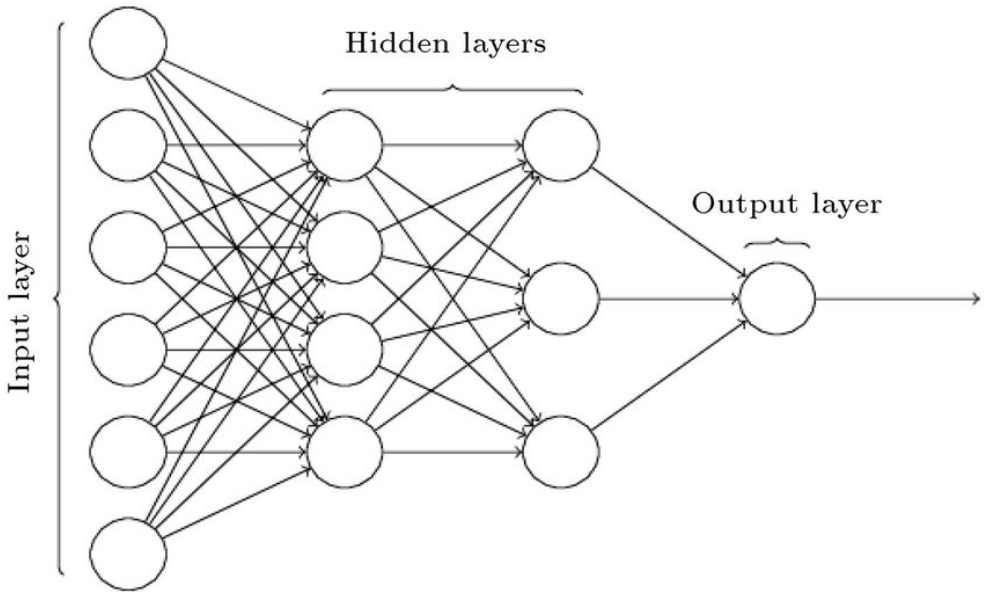

Figure 15: Simple Neural Network

Feedforward

On note $(X_i)_{1\leq i\leq n}$ les données d'entrée dans $\mathbb{R}^d$ et $(Y_i)_{1\leq i\leq n}$ les labels associés à chaque X_i, avec $Y_i \in \{0, 1\}$.

Le FFNN est un modèle paramétrique, que l'on peut décrire comme étant une superposition de régressions logistiques (couches linéaires et couches non linéaires), qui nous permet de modéliser $\mathbb{P}(Y = k|X)$ pour toutes les classes k possibles (0 ou 1 dans notre cas).

Un modèle élémentaire à été établi pour des valeurs d'entrée réelle par Franck Rosenblatt en 1957 avec :

$$f : \mapsto \mathbb{1}_{\sum_{j=1}^{d} \omega_j x_j + b \geq 0}$$

Ainsi, dans notre cas avec une sortie binaire, avec $\sigma : \mapsto \mathbb{1}_{x\geq 0}$. On peut écrire f :

$$f : \mapsto \sigma(\omega^T x + b)$$

Historiquement, on utilise de nombreuses fonction non-linéaires tel que la fonction sigmoïde, ReLU, tanh... Nous utiliserons la fonction sigmoïde dans cet exemple par commodité.

Il s'agit désormais d'estimer et d'optimiser les paramètres (poids et biais) de notre réseau en comparant la sortie prédite du réseau $\hat{\mathbf{y}}^{(\mathbf{i})}$ avec l'étiquette correspondante $\mathbf{y}^{(\mathbf{i})}$.

Optimisation

Pour l'estimation des paramètre nous minimisons l'opposé de la log-vraisemblance ou encore la cross entropy.

Nous utiliserons dans notre modèle la fonction de perte appelée Binary Cross Entropy comme fonction de perte à minimiser, qui se calcule de la façon suivante pour une classification binaire avec j$\in \{1, 2\}$:

$$\mathcal{L}(\hat{\mathbf{y}}^{(\mathbf{j})}, \mathbf{y}^{(\mathbf{j})}) = -\mathbf{y}_{\mathbf{1}}^{(\mathbf{j})} ln(\hat{\mathbf{y}}_{\mathbf{1}}^{(\mathbf{j})}) - \mathbf{y}_{\mathbf{2}}^{(\mathbf{j})} ln(\hat{\mathbf{y}}_{\mathbf{2}}^{(\mathbf{j})})$$

Il s'agit désormais de mettre à jour les paramètres du modèle par rétropropagation de l'erreur.

Backpropagation

Désormais il est nécessaire de pouvoir mettre à jour les poids lors de l'apprentissage, on utilise pour cela la méthode de **Backpropagation** qui se base sur l'algorithme de descente de gradient afin de trouver le minimum local de la fonction de perte itérativement.

Algorithm 3 Pseudo Code SGD

Require: On initialise les hyperparamètres du modèles : pas d'apprentissage (η,nombre de neurones dans couches cachées, nombre de couches cachées..)

Initialisation:
On initialise θ (les paramètres) aléatoirement.
for E epochs **do**
- Selectionne un batch d'échantillons noté B de façon randomisée
- Répéter les 2 étapes suivantes jusqu'à ce que l'epoch soit complétée (nombre de mini-batch suffisamment important pour pouvoir couvrir training set):
  - Calcul des gradients :$\Delta = \nabla_\theta l_n(\theta)$
  - Mise à jour des paramètres : $\theta \longleftarrow \theta - \eta\Delta$
- S'arrêter à un certain critère : lorsque la fonction de perte cesse diminuer

end for

Remarque : l'initialisation des hyperparamètres est décisive pour éviter l'explosion ou l'évanescence du gradient

Pour plus de précision sur les calculs de dérivées partiels de la fonction de perte, je conseil cette visualisation faites par Google [4]

Sortie du réseau

La couche de sortie d'un réseau neuronal pour la classification binaire comporte généralement un seul neurone avec une fonction d'activation sigmoïde $\sigma(x) = \frac{1}{1+e^{-x}}$. Si la sortie du neurone est supérieure à 0.5, nous supposons que la sortie est égale à 1, et sinon, nous supposons que la sortie est égale à 0.

[4] Backpropagation vizualisation

4.4 Variational autoencoders

Pour stratifier les patients atteints de Parkinson, on choisit d'utiliser des algorithmes de clustering. Cependant au vu de l'état de l'art on ne peut utiliser simplement un algorithme GMM ou KMeans directement sur les données. On décide donc d'utiliser un Variational auto encodeur (VAE).

Le principe du VAE est que l'on a accès à des observations x_i de même loi que x qui dépendent d'une variable non observée z (variable latente). On note $p_\theta(x, z)$ la densité jointe du couple (x, z) où θ est un paramètre à estimer. On ne peut ni calculer $p_\theta(x)$ ni l'estimer avec un algorithme EM car on ne peut calculer $p_\theta(z|x)$. On décide donc d'approximer $p_\theta(z|x)$ par une autre densité $q_\phi(z|x)$.

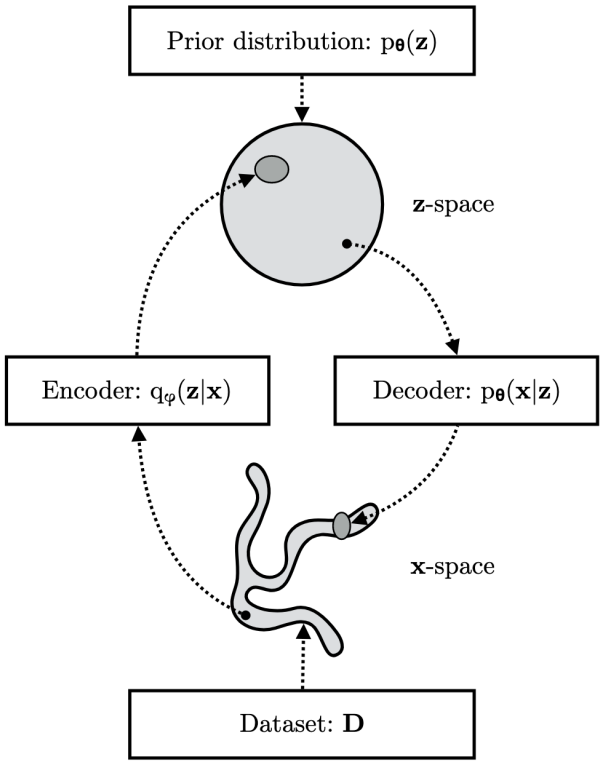

Figure 16: Variational autoencodeur (VAE) architecture

L'espace z est appelé espace latent et on réalisera notre clustering sur cet espace plus régulier que notre dataset du fait de la prior que l'on a imposé dessus. De plus on a l'égalité suivante:

$$logp_\theta(x) = \mathbb{E}_{q_\phi(x|.)}[log\frac{p_\theta(x,z)}{q_\phi(z|x)}] + \mathbb{E}_{q_\phi(x|.)}[log\frac{q_\phi(z|x)}{p_\theta(z|x)}]$$

On appelle ELBO (Evidence Lower Bound) le terme de gauche de l'égalité précédente:

$$ELBO = \mathbb{E}_{q_\phi(x|.)}[log\frac{p_\theta(x,z)}{q_\phi(z|x)}] = \mathbb{E}_{q_\phi(x|.)}[logp_\theta(x|z)] - \mathbb{E}_{q_\phi(x|.)}[log\frac{q_\phi(z|x)}{p_\theta(z)}]$$

On a $ELBO \leq logp_\theta(x)$ donc en maximisant la ELBO on maximise aussi $logp_\theta(x)$.

On choisit $q_\phi(z|x)$, l'encodeur, une densité gaussienne de paramètre $\mu_\phi(x)$ et $\sigma_\phi(x)$ avec μ_ϕ et σ_ϕ la sortie d'un FFNN. On choisit pour $p_\theta(z)$ une densité gaussienne centrée réduite. On peut calculer explicitement $\mathbb{E}_{q_\phi(x|.)}[log\frac{q_\phi(z|x)}{p_\theta(z)}]$ et on approxime $\mathbb{E}_{q_\phi(x|.)}[logp_\theta(x|z)]$ avec un estimateur de Monte Carlo. On utilise aussi un FFNN pour modéliser $p_\theta(x|z)$, le décodeur. Ainsi on entraîne notre réseau en maximisant la ELBO.

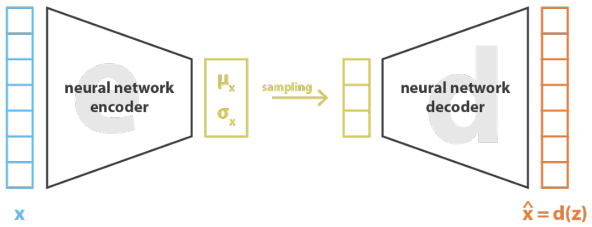

Figure 17: Architecture VAE avec FFNN

5 Discussion et comparaisons

5.1 Freezing of Gait

Pour débuter l'étude de cette base de données, il était impératif de mieux comprendre les données et de les visualiser. Voici ci-dessous une partie du tableau de données pour la tâche 1 réalisée par le patient 1:

	FP1	FP2	F3	F4	C3	C4	P3	P4	O1	O2	...	WaistGYROY
0	7.064	7.3853	5.3534	6.9239999999999995	3.7559	1.7674	4.4223	6.7464	10.777999999999999	3.7583	...	495.0
1	7.8537	7.6998	6.1821	6.9615	3.8015	1.2309	4.1826	6.7128	10.5566	2.9936	...	508.3143625619463
2	9.8257	9.1855	8.1727	7.9598	5.3095	2.0016	5.3646	7.9092	12.0235	4.0203	...	524.387974965697
3	12.2397	11.2825	10.5597	9.531	7.6492	3.5774	7.4899	9.7218	14.5659	6.4164	...	541.4044060888043
4	14.2806	13.2718	12.5526	11.1866	9.783999999999999	5.2566	9.5088	11.3648	16.6226	9.1463	...	557.547224807996
...	...	...	...	...	...	...	...	...	...	...	...	...
180496	13.4574	12.9637	10.9016	10.6666	10.1241	7.7911	7.6474	7.9377	4.942	-5.5287	...	31.347194351283434
180497	15.0615	15.5597	12.3495	12.8891	11.3221	10.5797	9.5361	9.2444	7.5713	-1.4217	...	34.00084515321725
180498	16.3636	17.3752	13.4653	15.0994	12.4483	13.6728	11.3722	10.5227	10.8216	3.9685	...	36.38089877956757
180499	16.667	17.778	13.6059	16.616	13.0492	16.4723	12.6855	11.2524	14.1313	9.9318	...	37.907301603954956
180500	15.9352	16.9962	12.8189	16.9944	13.0086	18.3942	13.2824	11.2792	16.7962	15.054	...	37.99999999999999

180501 rows × 57 columns

ArmACCX	ArmACCY	ArmACCZ	ArmGYROX	ArmGYROY	ArmGYROZ	SC	FoG
7864.999999999997	4712.0	-1087.0000000000002	661.0	-688.0000000000001	509.0	1830.0	0
8055.034672936841	4661.843221132418	-1105.551849700487	657.591356318603	-723.7358033380314	533.3733042627478	1830.000024975919	0
8224.98366769971	4607.356201127454	-1114.1575083679693	662.330623225968	-762.9319993019711	563.7483589721879	1830.000018712776	0
8379.215326000023	4552.647570554921	-1094.5872421857157	672.8742119737909	-801.6602935978467	595.4367615508307	1829.9999999616748	0
8522.097989534535	4501.825959988033	-1028.6113173357255	686.8785338143692	-835.9923919293071	623.7501094197453	1829.9999874737161	0
...	...	...	...	...	...	...	...
2334.056456643045	7361.136505364567	3317.5079412699047	98.71579934601047	76.21574182358978	-7.0829270380630325	1640.0001087110377	0
2328.9333098958064	7358.299736967412	3316.9002675321417	99.6066855628172	77.93152651511956	-7.1795711207090696	1640.000135101448	0
2324.3819348269103	7351.694715888174	3316.438623159417	100.53967210663562	79.23944029492714	-7.234751684325269	1640.00010713634	0
2321.153706505321	7343.5264632059925	3316.3846525244603	101.38177243363886	80.23156938316856	-7.193288165294595	1640.0000527808213	0
2320.0	7336.000000000001	3316.9999999999995	102.0	81.0	-7.0	1639.9999999999998	0

Figure 18: Tableau patient 1 / tâche 1

On voit bien dans la première colonne les unités de temps puis d'autres colonnes qui correspondent à des données cérébrales et enfin des données sur les mouvements du patient. La dernière colonne "FoG" contient des 0 et des 1 qui sont les labels.

Nous avons ensuite voulu voir les différents données des patients en regardant les proportions de crises de FoG pour quelques patients et quelques tâches. Nous avons donc tracé graphiquement la proportion de crises FoG pour les patients 1, 2 et 3 (tâche 1 pour tous).

On remarque alors tout de suite des différences notables entre les patients. Le patient 2 n'a subi aucune crise de FoG durant toute la durée de la tâche 1 alors que pour le patient 3, il a été en état de crise pendant 90% du temps. Ne connaissant par l'origine de chaque patient (âge, durée de la maladie, ...) nous ne pouvons pas déduire de conclusions sur les différences physiques entre les patients avec simplement l'information de la proportion de crise FoG.

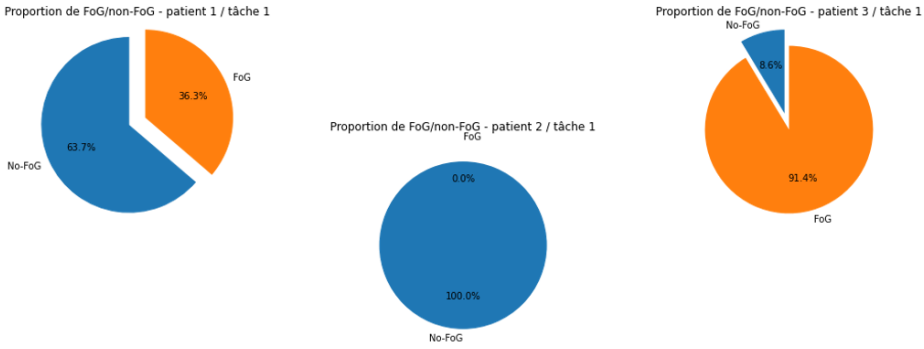

Figure 19: Proportions de crises FoG pour différents patients

Pour faire une première étude de corrélation entre toutes les données des patients et les labels de FoG, nous avons d'abord voulu tester la corrélation avec la matrice de corrélation qui mesure des degrés de relation linéaire entre deux variables (ici chaque donnée prise à part sera comparée linéairement aux labels FoG 0/1). Nous avons alors tracé la matrice de corrélation pour le patient 1 tâche 1 afin de nous faire une première idée des relations possible. Voici le résultat:

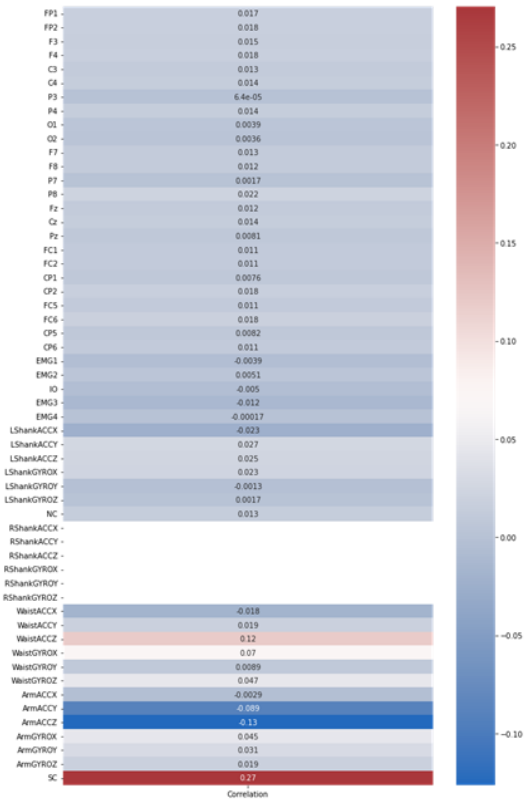

Figure 20: Matrice de corrélation patient 1 / tâche 1

D'après cette matrice, la donnée qui aurait la meilleure corrélation linéaire serait "SC" qui correspond aux mouvements de l'index avec une corrélation de 0.27. Le résultat est sensiblement le même pour les différents patients et tâches. Cependant, cette valeur de corrélation maximum parait assez faible et il semble compliqué de pouvoir conclure à une relation linéaire entre ces deux variables de façon sûre.

Pour trouver des corrélations plus visuelles, nous avons alors tracé différentes données et les valeurs des labels sur le même graphique pour pouvoir les comparer à l'oeil. Deux résultats en ressortent:

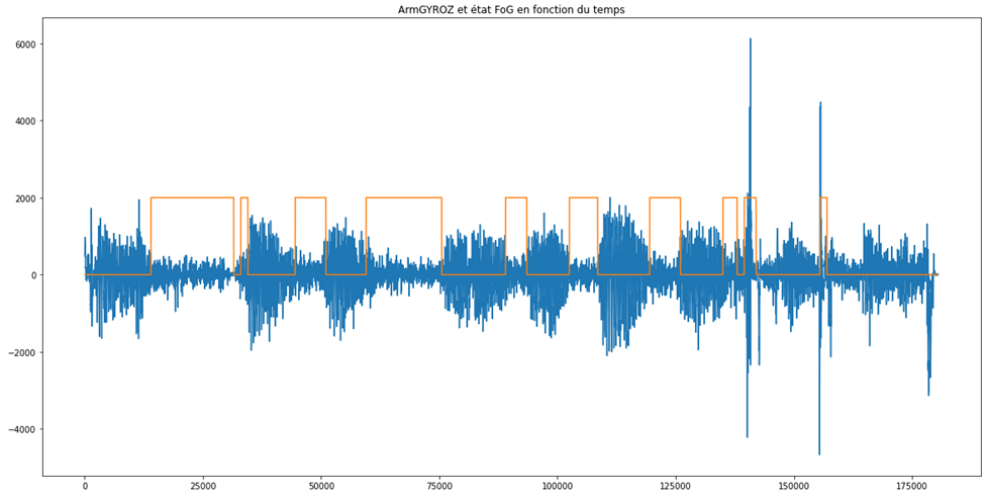

Figure 21: ArmGyroZ et label FoG en fonction du temps pour le patient 1 tâche 1

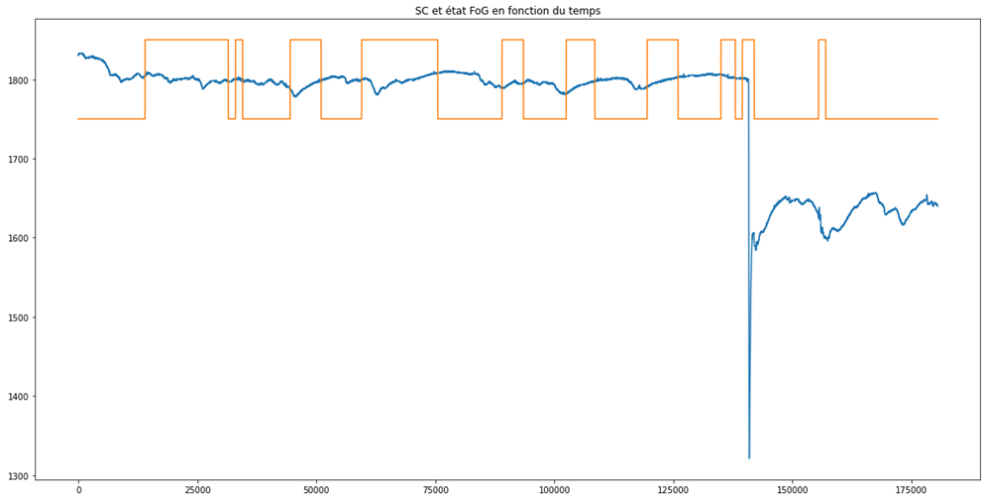

Figure 22: SC et label FoG en fonction du temps pour le patient 1 tâche 1

On remarque que SC ne semble pas vraiment lié visuellement aux valeurs du label FoG. Cela devient encore plus flagrant lorsque l'on compare avec la donnée ARMGYROZ (qui correspond à l'accélération du bras selon l'axe Z). Sur ce graphique, on voit tout de suite que lorsque l'accélération diminue, le label est de 1 qui indique une crise de FoG et lorsque le patient accélère le label devient 0 qui indique que la personne n'est plus en état de crise. On peut tout de même voir quelques impuretés vers la fin du graphique avec des pics d'accélération en pleine crise FoG. Pour les études suivantes, nous allons nous débarasser de ces données aberrantes en utilisant un filtre.

Nous avons alors voulu tenter de réaliser un clustering selon la donnée ARMGYROZ. Nous avons alors utilisé un algorithme proche du k-means, en calculant la moyenne de chaque cluster pour ensuite associer les points suivants au cluster (FoG ou Non-FoG) qui est le plus proche:

$$Label : t \mapsto \begin{cases} 1 & \text{si } |ARMGYROZ\,(t) - \overline{m_{FoG}}| \leq |ARMGYROZ\,(t) - \overline{m_{Non-FoG}}| \\ 0 & \text{sinon.} \end{cases}$$

où ARMGYROZ est la fonction qui retourne la valeur de l'accélération du bras selon l'axe z à un instant t donné et $\overline{m_{FoG}}$ (respectivement $\overline{m_{Non-FoG}}$) correspond à la moyenne des valeurs d'accélération du cluster FoG (respectivement Non-FoG).
Après plusieurs essais pour réussir à faire un clustering sur d'autres patients, la méthode n'a pas pu aboutir à cause de certaines limites. Premièrement, les moyennes des différents clusters dépendent de chaque personne et donc on ne peut pas utiliser une moyenne d'un cluster FoG pour le patient 1 pour une autre personne qui n'aurait pas la même accélération moyenne en état de crise ou en état normal. De plus, certaines personnes ont été si souvent en état de crise qu'il est trop dur de distinguer les deux moyennes.

Par exemple, pour le patient 3, on remarque que même lorsque le patient 3 ne se trouve plus en état de crise FoG, son accélération ne remonte pas de façon durable (seuls quelques pics subsistent). Il est alors compliqué de dégager deux clusters bien séparés par leur moyenne respective.
Nous nous sommes alors tourné ensuite vers une méthode de machine learning supervisée car nous possédions les labels attribués par des médecins. Nous avons alors décidé d'utiliser l'algorithme de Random Forest qui permet de ne faire aucune paramétrisation sur le système et aucune supposition sur la forme du modèle.

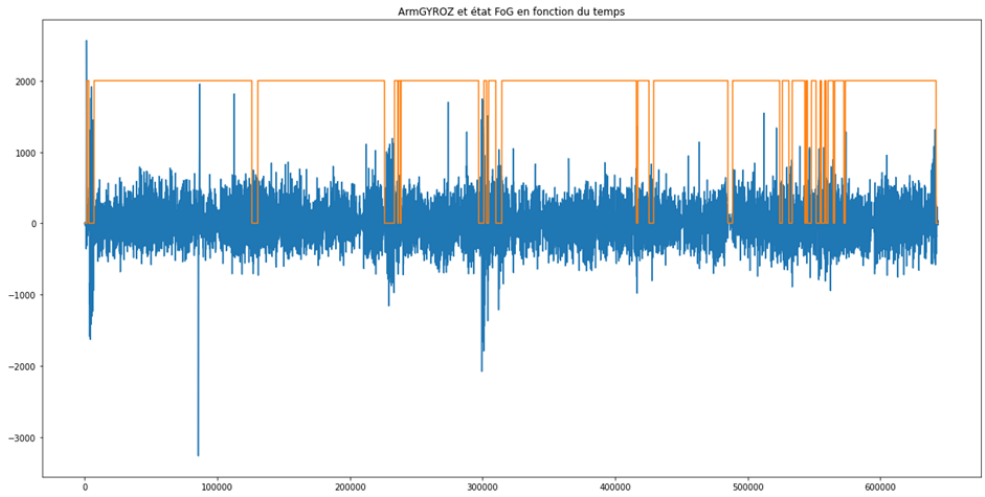

Figure 23: ArmGyroZ et label FoG en fonction du temps pour le patient 3 tâche 1

En voulant utiliser les données des 12 personnes de la base de données, nous nous sommes rendus compte qu'il était impossible de le faire car cela utilisait trop de RAM que ce soit sur nos ordinateurs personnels ou sur Google Colab. Nous avons donc du réduire la base de données et ne considérer que 5 personnes sur les 12. Pour l'algorithme Random Forest, nous avons alors utilisé seulement les personnes 1 à 5 et pour chacun nous avons considéré la tâche 1 (car elle est souvent la plus longue et cela permet d'avoir plus de données pour un même nombre de personne considéré). Nous avons alors obtenu un tableau de 1,266,505 relevés tous labellisés. On va donc choisir d'entrainer le modèle sur les 4 premières personnes pour ensuite tester sur la dernière personne le modèle ainsi construit. On teste alors avec différentes valeurs de profondeur d'arbres (dans le code il s'agit de la valeur "N estimators"). On obtient alors le tableau de précision suivant:

N_estimators	20	22	24	26	28	30	32	34
Accuracy	0,593	0,674	0,644	0,669	0,641	0,667	0,620	0,661

Figure 24: Précision en fonction du nombre d'estimateurs/arbres

La plupart des précisions sont contenues entre 60 et 68%. On décide donc de continuer avec un nombre d'arbres égal à 22 pour optimiser la précision. Les données étant labellisées, on peut tracer une matrice de confusion pour N = 22 arbres:

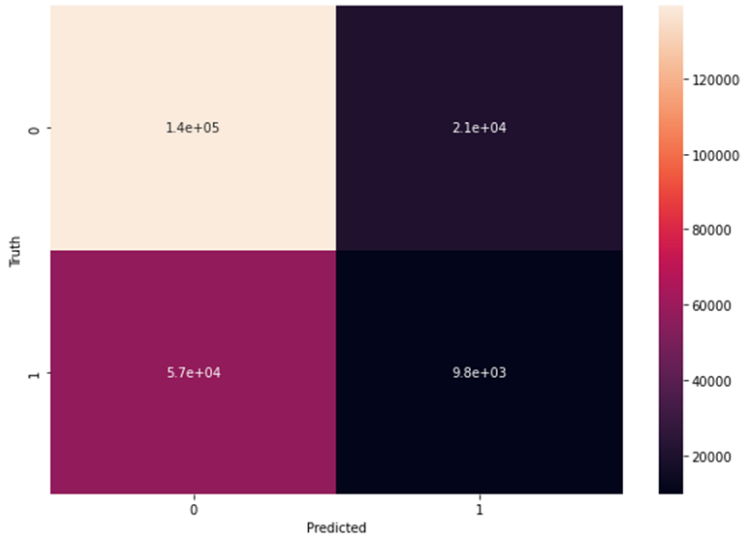

Figure 25: Matrice de confusion pour N = 22 arbres

5.2 PPMI

5.2.1 Première approche

Après avoir réalisé l'analyse et le cleaning des données partie 3.2. Il est nécessaire de faire une première approche, pour voir la corrélation entre les différentes mesures et le label. Pour cela, j'ai dans un premier temps réalisé une matrice de corrélation avec la méthode de Pearson. Le coefficient de corrélation de Pearson peut être utilisé pour résumer la force de la relation linéaire entre deux échantillons de données, elle se calcule de la façon suivante :

$$r_p = \frac{cov(X,Y)}{\sigma_X \sigma Y}$$

Avec avec X représentant une mesure et Y les labels, ainsi pour n = 192 personnes,

$$cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x_i})(y_i - \overline{y_i})$$

.

Ainsi, on observe sur la figure 5.2.1une certaine corrélation entre les mesures (0.48) et les labels notamment pour la durée moyenne des pas pendant le virage de la personne : $TUG2\_TURN\_DUR$

Remarque : TUG est un test de rendement physique général utilisé pour évaluer la mobilité, l'équilibre et le rendement locomoteur.

De plus on observe une relation inversement linéaire (-0.43) des labels avec le jerk(dérivé de la

vitesse : secousse) de l'accélération du mouvement des jambes en dual task : $JERK\_T\_DT$.

Cependant l'observation des relations linéaires entre les données est très restrictive, c'est pourquoi on calcul la corrélation de Spearman sur les figures 5.2.1. La corrélation de Spearman analyse la relation entre le rang des observations pour une mesure X et les labels Y, ce qui permet de détecter l'existence de relations monotones (croissante ou décroissante) quel que soit leur forme (linéaire ou non).

$$r_s = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X} \sigma rg_Y}$$

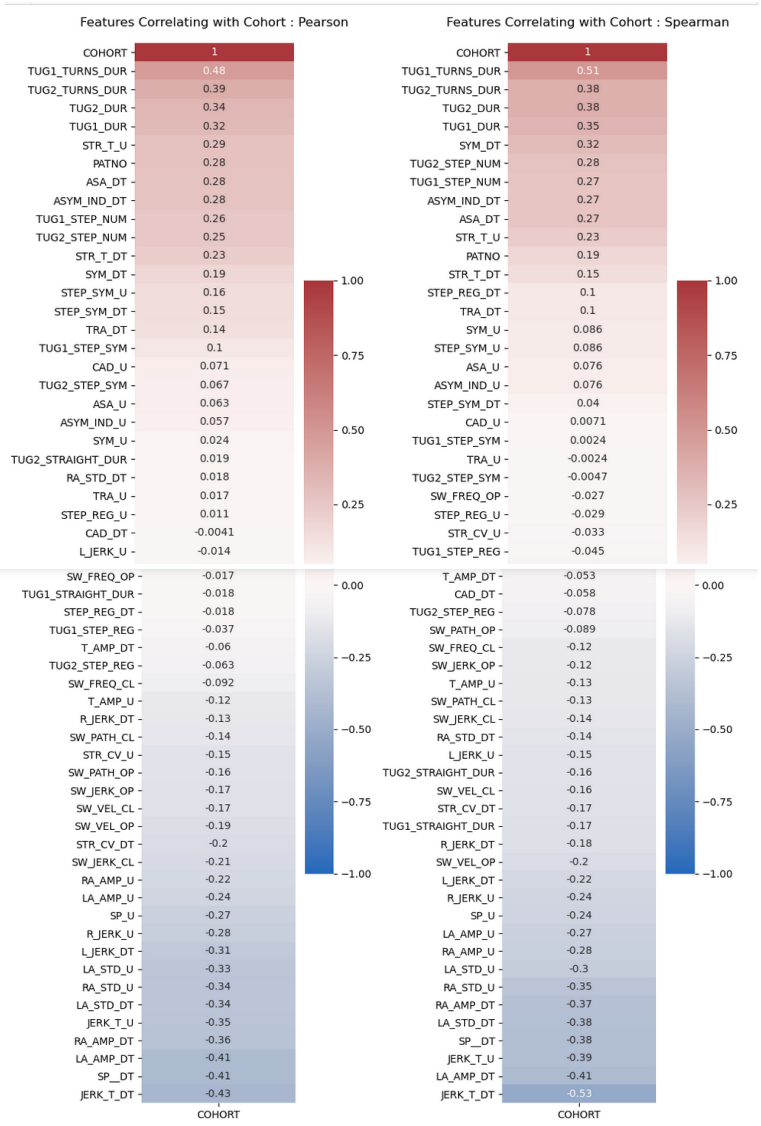

Figure 26: Matrice de corrélation entre les 56 mesures et les labels

Ainsi d'après la figure précédente les corrélations semblent relativement similaire, on trace 5.2.1représentant la différence entre les corrélations pour chaque mesure.

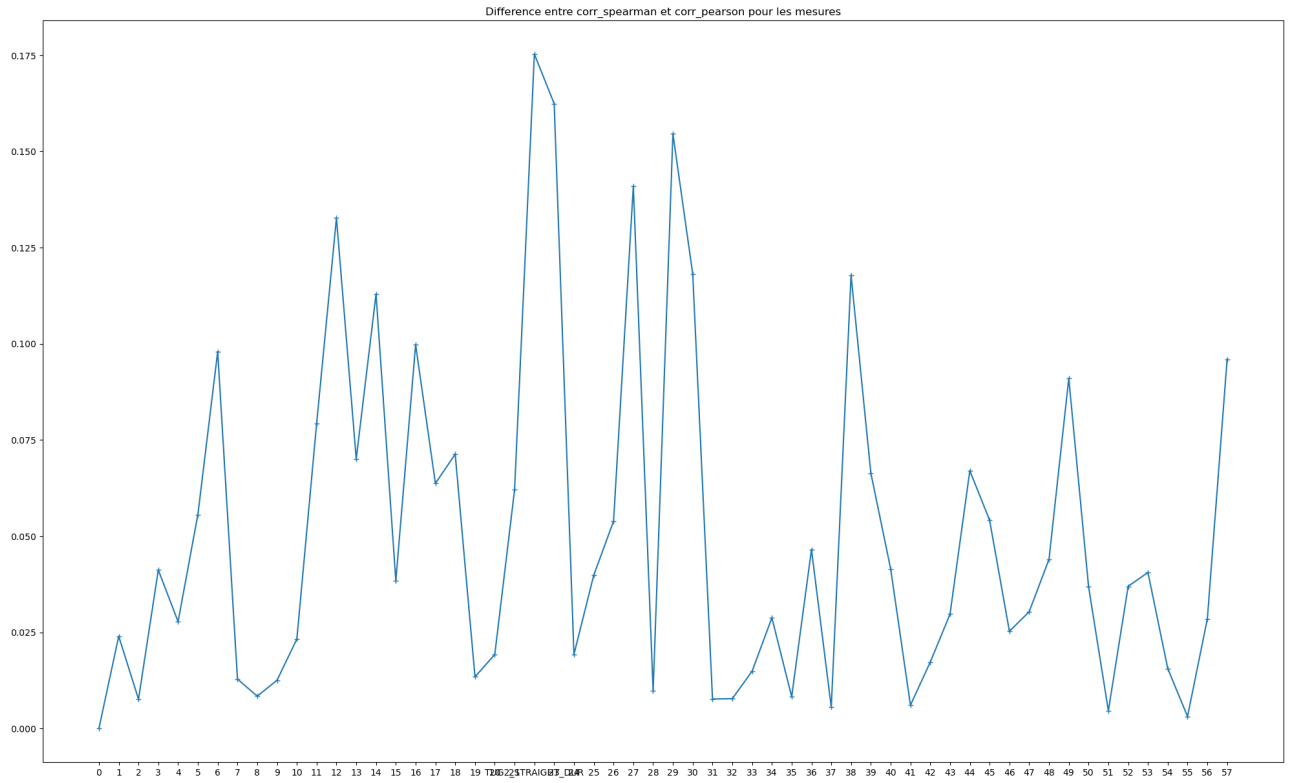

Figure 27: Différences entre les 2 types de corrélation

Ainsi, on observe que les différences de corrélations sont infimes, en effet la plus grande différence se fait pour $TUG2\_STRAIGHT\_DUR$ (durée des pas en ligne droite du test TUG) avec 0.175. Ainsi on en conclut que nos données on des corrélations linéaire ou monotones sensiblement similaire, mais tout de même trop faible pour pouvoir envisager un possible classification linéaire des 192 personnes.

Par la suite, il est nécessaire de séparer notre base de données pour l'entrainement et le test. Pour savoir comment split notre base de donnée nous utiliserons le principe de Pareto, qui dans la littérature est très souvent utilisé. Ce principe empirique suggère que environ 80% des effets sont le produit de seulement 20% des causes. C'est pourquoi notre séparons aléatoirement en un training set (80% de données de la base) et un test set (20% restant).

5.2.2 Algorithme 1 : Random Forest

Désormais on souhaite classifier nos données avec une méthode plus élaboré étant donnée les corrélations observées dans la partie précédente. Ainsi, l'algorithme de random forest expliqué

partie 4.1, sera notre modèle de référence à comparer avec d'autre modèle élaborés.

Comme nous l'avons vu lors de la présentation, différents hyperparamètres sont à fixer pour notre modèle (profondeur de l'arbre, nombre minimum de feuilles...)

Ainsi, pour fixer le meilleur set d'hyperparamètre tout en évitant l'overfitting de notre modèle. Nous allons travailler sur des tests de grilles de paramètres avec évaluation de chaque set par crossvalidation (K-fold avec K=5 suffisant sachant que nous avons 192 personnes, la base est assez petite).

- Premier test du modèle avec les paramètres par défaults
- Ensuite, balayage en établissant un grille large des hyperparamètres, puis test de 500 combinaisons aléatoires d'hyperparamètres parmi ceux de la grille. Pour chaque test un réalise une cross-validation (5-folds). Enfin on obtient le set d'hyperparamètre ayant le moyen score le plus élevé des 500 combinaisons aléatoires.
- Enfin, on créer une grille avec une paramétrisation des hyperparamètres proche de ceux trouvés précédemment avec la grille aléatoire. On réalise autant de cross validation que de combinaison d'hyperparamètres possibles avec notre grille. Ainsi, on obtient une des meilleures paramétrisation de notre modèle pour notre set de données.

En procédant ainsi, on obtient une Accuracy = 0.7 pour le modèle sans paramétrisation spécifique, puis une Accuracy = 0.79 après le recherche aléatoire et enfin **Accuracy = 0.85** avec le set d'hyperparamètres obtenu avec précision: $\{'bootstrap' : True, 'max\_depth' : 60, 'max\_features' : 'sqrt', 'min\_samples\_leaf' : 2, 'min\_samples\_split' : 10, 'n\_estimators' : 160\}$

5.2.3 Algorithme 2 : XGBoost

Afin d'avoir une procédure homogène entre les modèles, on réalise une étude similaire au modèle de random forest, c'est à dire on procède à une recherche du meilleur set d'hyperparamètres par des grilles.

En réalisant cet algorithme on obtient une Accuracy = 0.77 quelque soit la paramétrisation du modèle sur un même ensemble d'entrainement, ce qui semble étonnant. Mais plusieurs problèmes ont eu lieu. La première grille est très large car il faut balayer tout les combinaisons d'hyperparamètres possible. Parmi toutes les combinaisons possibles (plus de 100000), nous réalisons des crossvalidations sur seulement 500 combinaisons prises aléatoirement parmi les 100000,

car au dessus de 500 entrainements puis crossvalidations les nombre de calculs à effectuer est immense et donc le temps de calcul devient très long. Mais en balayant seulement 500 combinaisons sur 100000 il y a peu de chance de tomber sur un paramétrisation proche de celle optimale. Ainsi, le seconde grille plus restrictive sera elle aussi peu optimale.

Nous avons tout de même des pistes (que nous n'avons pas pu mettre en place) pour une initialisation plus ou moins prometteuses, certaines pratiques fonctionnent de manière à paramétrer successivement les paramètres ayant la plus forte influence sur le modèle:

1 - Fixer un taux d'apprentissage assez élevé (entre 0.1 et 1)

2 - Paramétrer la structure des arbres, rechercher avec une petite grille (environ 20 points) de la profondeur max d'un arbre et du seuil minimal du nombre d'individu présent dans un noeud

3 - Le coefficient γ, qui correspond à la pénalisation de profondeur, en effet il empêche la construction d'un arbre trop profond s'il n'est pas performant. Paramétrisation possible avec une petite grille.

4 - Colsample_bytree (ratio d'échantillonnage aléatoire sur les variables à chaque construction d'un arbre) et subsample(ration de sélection aléatoire des échantillons pour entraîner un nouvel arbre), paramétrer avec un grille pour des valeurs supérieur entre 0.5 et 1. 5 - Le taux d'apprentissage (0.01 puis 0.001) tout en augmentant n_estimators (le nombre d'arbres) avec possibilité d'early-stopping pour éviter tout surajustement.

La force de ce protocole réside dans le fait que l'on fixe au fur et à mesure les paramètres optimisés ayant le plus d'impact en sortie, ce qui réduit de façon conséquence le nombre de combinaisons à chaque étape.

5.2.4 Algorithme 3 : FFNN

Enfin, comme décrit partie 4.3, un réseau de neurone permet de trouver des relations très complexes entre les données. Cependant, il existe un certain nombre d'hyperparamètre à fixer notamment sur les dimensions du réseau. Il existe de nombreuses méthodes empiriques pour déterminer le nombre correct de neurones à utiliser dans les couches cachées, telles que les suivantes :

- Le nombre de neurones cachés doit être compris entre la taille de la couche d'entrée et la taille de la couche de sortie.

- Le nombre de neurones cachés doit être égal à 2/3 de la taille de la couche d'entrée, plus la taille de la couche de sortie. nombre de neurones cachés doit être inférieur à deux fois la taille de la couche d'entrée.

La plupart des problèmes peuvent être résolus en utilisant une seule couche cachée dont le nombre de neurones est égal à la moyenne des couches d'entrée et de sortie. Si l'on paramétrise avec un nombre de neurones trop faibles, cela entraînera un sous-ajustement et un biais statistique élevé. En revanche, notre modèle posséde un nombre de neurones trop élevé, cela peut entraîner un surajustement, une variance élevée et un temps de processing assez long.

Étant donnée que notre base de données est relativement petite, il est peu pertinent de prendre un nombre de couches important (par risque de surapprentissage sur notre petite base de données). C'est pourquoi j'ai opté après differentes paramétrisation (comparaison par cross validation) pour un modèle avec 40 neurones dans la première couche, une couche de dropout pour éviter le surajustement, 20 neurones dans la deuxième, à nouveau un dropout et enfin une couche de sortie avec un seul neurone pour une classification binaire. Nombre de paramètre total : 3121, voir 5.2.4.

Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 40)	2280
dropout_4 (Dropout)	(None, 40)	0
dense_7 (Dense)	(None, 20)	820
dropout_5 (Dropout)	(None, 20)	0
dense_8 (Dense)	(None, 1)	21

Total params: 3,121
Trainable params: 3,121

Figure 28: Paramétrisation de notre modèle FFNN

5.3 Comparaison

Pour effectuer la comparaison de performance entre les différents algorithmes de travail sur PPMI, il a été nécessaire "d'homogénéiser" les algorithmes. Nous avons donc entrainer tous les modèles avec le même training set, puis l'évaluation sur le même test set. De plus chaque modèle a été paramétré avec leur meilleur set d'hyperparamètre (selon cross-validation, voir l'algorithme de recherche5.2.2).

Différents critères existent pour pouvoir évaluer un modèle, nous allons utiliser les suivants étant très utilisés dans la littérature .

L'Accuracy est le rapport entre le nombre d'observations correctement prédites et le nombre total d'observations. On utilise l'accuracy pour un rapport général de la performance du modèle avec des ensembles de données équilibrés. Etant donnée que notre base de donnée est équirépartie entre Podromal et PD, ce critère semble pertinent.

On note TP(True Positive), TN(True Negative), FP(False Positive), FN(False Negative).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision est le rapport entre les observations positives correctement prédites et le total des observations positives prédites.

$$Prcision = \frac{TP}{TP + FP}$$

Recall (sensibility : Fraction des positifs qui ont été correctement identifiés.

$$Sensibility = \frac{TP}{(TP + FN)}$$

Score F1 (alias F-Score) : le score F1 prend en compte la précision et le rappel. Il est créé en trouvant la moyenne harmonique de la précision et du rappel.

$$F1 = \frac{2 * precision * rappel}{prcision + rappel}$$

Il semble intéressant dans un premier temps de s'intéresser à l'ensemble de ces critères :

Modèle	Rapport			Accuracy
	precision	recall	f1-score	
	1.0 0.80	0.89	0.84	
Random Forest	3.0 0.89	0.81	0.85	0.85
	precision	recall	f1-score	
	1.0 0.84	0.73	0.78	
XGBoost	3.0 0.70	0.82	0.76	0.77
	precision	recall	f1-score	
	0.0 0.83	0.91	0.87	
FFNN	1.0 0.87	0.76	0.81	0.83

Figure 29: *Tableau résumant les performances*

Remarque : les labels 1 et 3 ont été remplacés par des 0 et 1 pour une meilleure lisibilitées cependant le rapport ne les a pas modifiés (1->0 : Podromal et 3->1 : Parkinson)

Le modèle de random forest et le FFNN sont tous les deux performant étant donné que la tâche est de distinguer deux états de la maladie étant relativement proches (Podromal et Parkinson). Cependant, notre modèle de XGBoost est moins performant les 2 autres modèles, ce qui semble étonnant comparé à l'état de l'art, essayons d'expliquer cela.

Étant donné que chacune des classes est représenté de façon équilibré dans notre data set, il semble intéressant de comparer les accuracies. Cependant, nous avons observé que l'accuracy des modèles pouvait varier de façon importante en fonction du training set. En effet, lorsque nous lançons faisons un *train_test_split* grâce à sklearn, 80 pourcents de la base de donnée est choisi aléatoirement pour former la base d'entrainement. Ainsi, l'accuracy peut varier d'un entrainement à un autre, c'est pourquoi on a réaliser un boxplot avec 100 executions pour chacun des modèles à chaque fois sur des bases d'entrainement choisi aléatoirement (80/20), on obtient la figure 30.

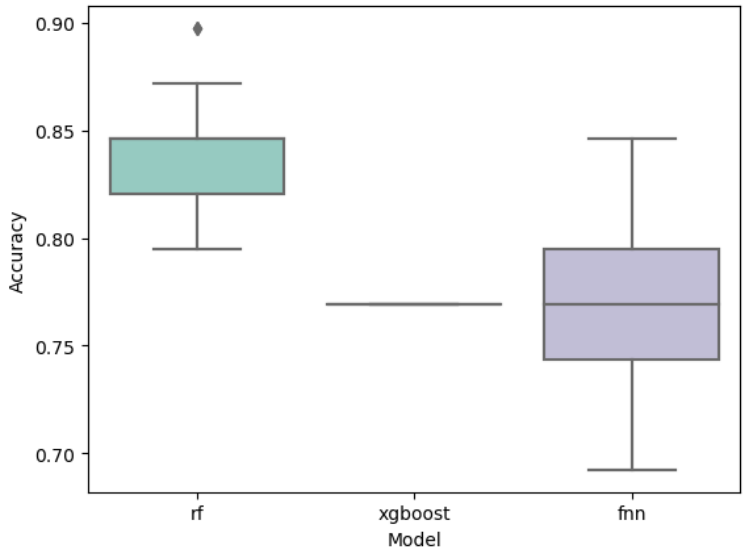

Figure 30: Accuracy des modèles

Ainsi, les résultats ici sont plutôt explicites, en effet le modèle de random forest semble dépasser assez largement les 2 autres modèles en accuracy sur les différents lancements. Cependant, le modèle de xgboost semble nous donner la même accuracy quelque soit la base d'entraînement et de test, ce qui parait étonnant. De plus, dans la littérature, pour des données tabulaires de petite à moyenne taille comme notre cas, l'algorithme XGBoost est considéré comme le meilleur de sa catégorie pour des problèmes de régression/classification grâce à une auto-amélioration séquentielle, voir 4.2. De plus, la paramétrisation d'un modèle de xgboost est assez complexe, en effet en plus du nombre important d'hyperparamètres, ceux-ci ne se calibrent pas tous en même temps (execution asynchrone de l'algortihme). Ainsi, la paramétrisation de notre modèle avec la grille de recherche aléatoire parmi différentes combinaisons d'hyperparamètres ne nous à pas permis de trouver les paramètres optimaux pour notre base de donnée car la grille que nous avons établi devait être trop importantes (recherche trop large).

5.4 Stratification

Comme précedemment expliqué nous utiliserons un VAE avant le clustering des patients.

On choisit un espace latent de dimension deux pour avoir une meilleure appréciation graphique de nos résultats. Pour réaliser l'entrainement de ce VAE on utilise les patients atteints de Parkinson et ceux au stade prodromal. On choisit de retenir les paramètres de notre VAE qui permettent à un algorithme de clustering (avec deux clusters) de discerner avec le plus de précision un pa-

tient avec Parkinson et un au stade prodromal. Une fois les paramètres obtenus on utilise le VAE uniquement sur les patients atteints de Parkinson et on génère un espace latent. On essaye deux méthodes KMeans et GMM. On obtient une meilleure précision avec Kmeans (70%) qu'avec un GMM (55%). On décide donc d'utiliser Kmeans.

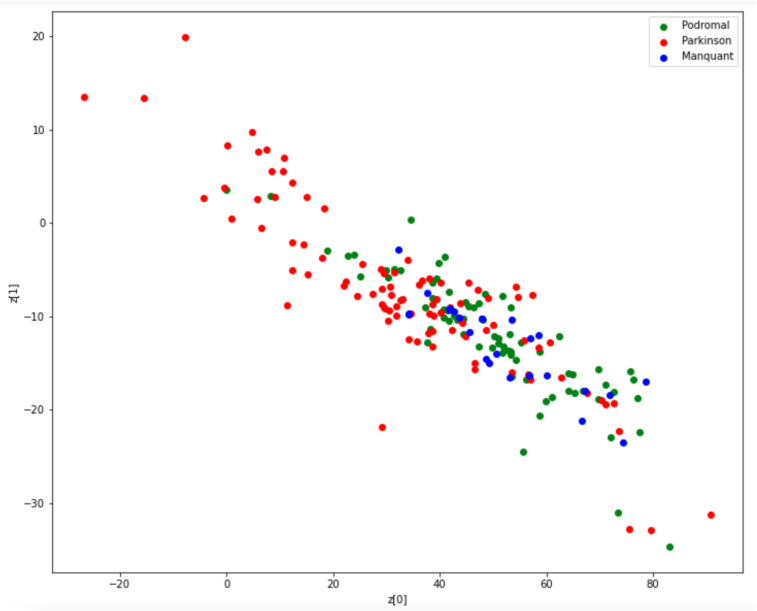

Figure 31: Espace latent après entraînement de notre VAE

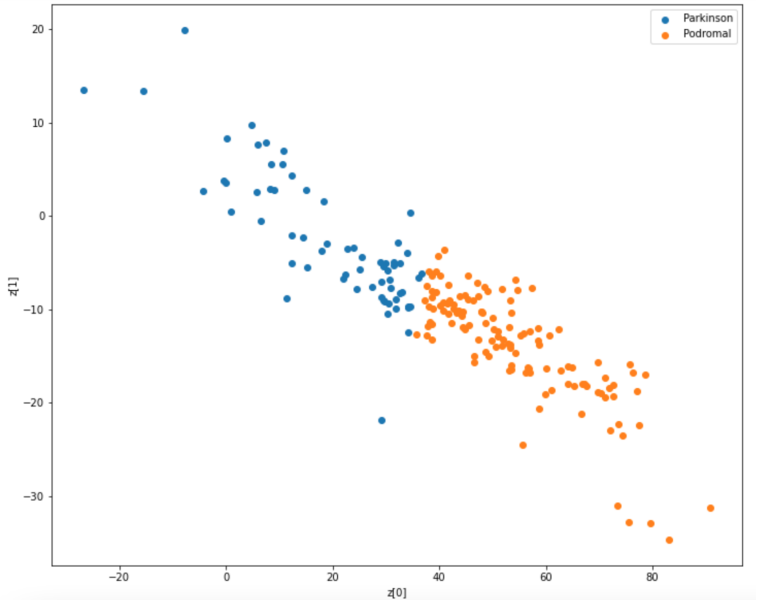

Figure 32: Clustering avec KMeans sur l'espace latent

Une fois avoir obtenu les paramètres de notre modèle on l'applique uniquement aux patients

atteints de Parkinson et on réalise un clustering avec Kmeans. Il est donc nécessaire de connaître le nombre de clusters à réaliser au sein des patients atteints de Parkinson. On utilise la elbow method avec l'inertie des clusters. On obtient le graphique suivant :

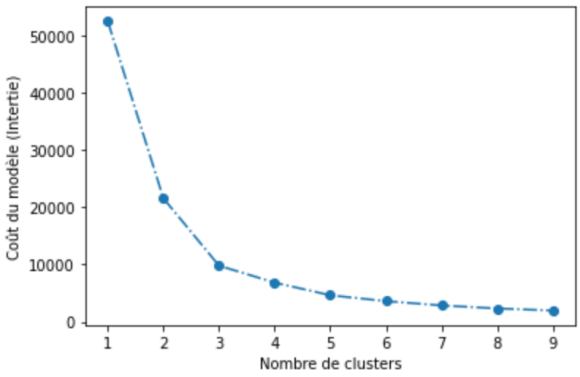

Figure 33: Elbow Method

On voit donc que le nombre de cluster optimal est 3. On utilise donc Kmeans avec 3 clusters sur l'espace latent créé à partir du datset contenant seulement les patients atteints de Parkinson.

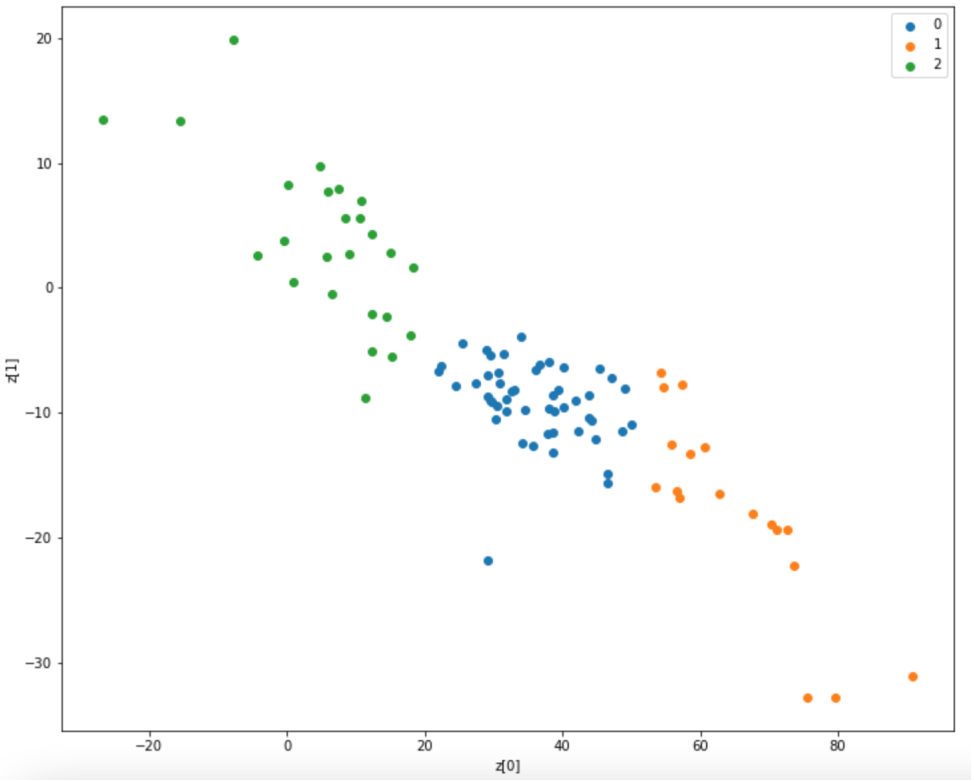

Figure 34: Clustering des patients atteints de Parkinson

6 XAI pour notre modèle de Random Forest

Les modèles d'apprentissage automatique prédictifs utilisés deviennent encore plus puissants lorsqu'ils sont associés à des outils d'interprétabilité comme SHAP. Ces outils identifient les mesures d'entrée les plus décisives pour un résultat prédit, ce qui est utile pour expliquer ce que fait le modèle et diagnostiquer les problèmes potentiels.

Nous avons utilisé SHAP pour notre interprétation. SHAP (SHapley Additive exPlanations) est une approche de la théorie des jeux pour expliquer la sortie de tout modèle d'apprentissage automatique. Elle relie l'allocation optimale de crédits aux explications locales/globales en utilisant les valeurs de Shapley classiques de la théorie des jeux et leurs extensions connexes.

L'idée centrale derrière les explications des modèles d'apprentissage automatique basées sur la valeur de Shapley est d'utiliser les résultats d'allocation équitable de la théorie des jeux coopératifs pour répartir le crédit de la sortie d'un modèle entre ses caractéristiques d'entrée.

Nous utiliserons également le terme plus spécifique de valeurs SHAP pour désigner les valeurs de Shapley appliquées à une fonction d'espérance conditionnelle d'un modèle d'apprentissage automatique.

Les valeurs SHAP peuvent être très compliquées à calculer (elles sont NP-hard en général), mais les modèles linéaires sont si simples que nous pouvons lire les valeurs SHAP directement sur un graphique de dépendance partielle. Lorsque nous expliquons une prédiction la valeur SHAP pour une caractéristique spécifique est juste la différence entre la sortie attendue du modèle et le graphe de dépendance partielle à la valeur de la caractéristique.

Le théorème de Shapley nous dit qu'il existe une unique répartition satisfaisant quatre propriétés (efficience, symetrie, joueur nul et additivité), assurant que la répartition du gain entre les joueurs est équitable.

Ainsi il existe une et une seul fonction $\phi_i(c)$ vérifiant les contraintes suivantes :

Nous sommes intéressés par la façon dont chaque caractéristique affecte la prédiction d'un point de données. Dans un modèle linéaire, il est facile de calculer les effets individuels. Ainsi à

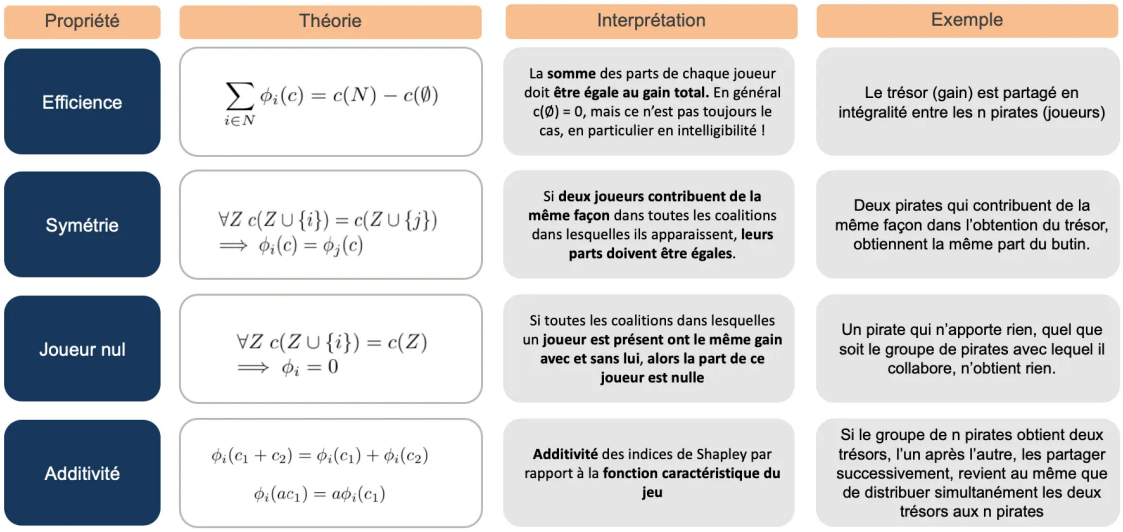

Figure 35: Propriété Shapley Value

l'aide du livre [14] Voici à quoi ressemble la prédiction d'un modèle linéaire pour une instance de données :

$$\hat{f}(x) = \beta_0 + \beta_1 x_1 + ... + \beta_p xp$$

avec pour tout j $\in [1 : p]$, $\beta_j$est le poids associé à la jème feature x_j

Ainsi, la contribution ϕ_j de la jème feature sur la prédiction $\hat{f}(x)$ s'écrit:

$$\phi_j(f, x) = \sum_{S \subseteq (1,.,p) \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} (\mathbb{E}[f(x)|X_{S \cup \{i\}} = x_{S \cup \{i\}}] - (\mathbb{E}[f(x)|X_S = x_S])$$

Donc,la somme des valeurs de Shapley d'une observation x est égale à l'écart entre la prédiction f(x) et la moyenne des prévisions ($\mathbb{E}[\mathbb{f}(\mathbb{x})]$),

$$\sum_{j=1}^{p} \phi_j(\hat{f}) = f(x) - \hat{E}(f(\hat{x}))$$

.

Ainsi, une figure représentative des ϕ_i, (provenant de [5]).

On s'intéresse dans un premier temps à l'intelligibilité globale de notre modèle random forest car c'est celui-ci qui possède l'accuracy la plus élevée.

Un graphique d'importance des variables liste les variables les plus significatives par ordre

[5]SHAP : Mieux comprendre l'interprétation de modèles

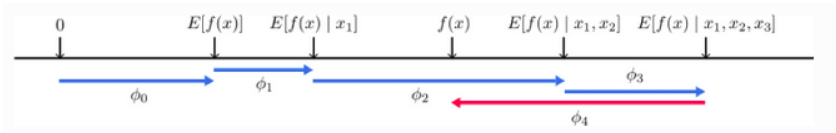

Figure 36: Prédiction de f(x) expliquée par la somme des effets de ϕ_i

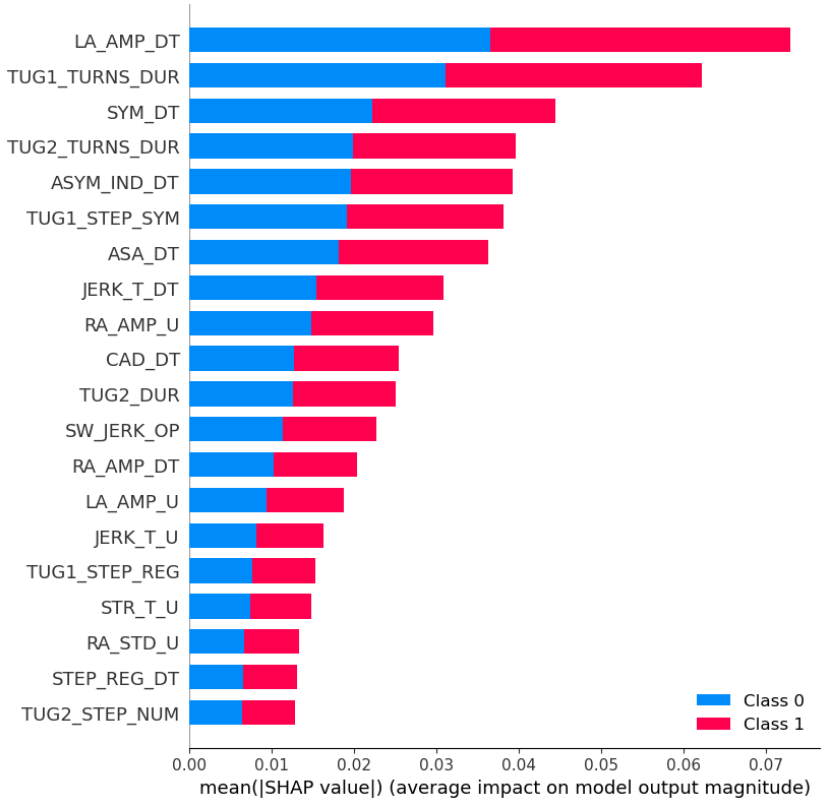

Figure 37: Intelligibilité Globale de notre modèle Random Forest

décroissant. Les variables du haut contribuent plus au modèle que celles du bas et ont donc un pouvoir prédictif élevé.

Toutes les caractéristiques semblent contribuer de la même manière à ce que les deux classes soient diagnostiquées avec PD (étiquette=1) ou podromal (étiquette = 0) car les couleurs occupent 50% des rectangles.

De plus, l'amplitude moyenne du bras gauche (LA_AMP_DT) est la mesure qui a le plus grand pouvoir prédictif selon le modèle de random forest. La deuxième mesure ayant le plus d'impact en moyenne sur les prédictions et $TUG1\_TURNS\_DUR$, la durée moyenne des pas pendant le virage du test TUG (présenté précédemment). Comme attendu nous retrouvons ces résulatas dans l'analyse précédente des corrélations entre les labels et les mesures, avec

$TUG1\_TURNS\_DUR$ qui était la mesure la plus corrélés aux labels (0.48 pour corr de pearson et 0.51 pour corr de spearman).

Enfin, on s'intéresse à une intelligibilité plus locale de notre modèle de Random Forest.

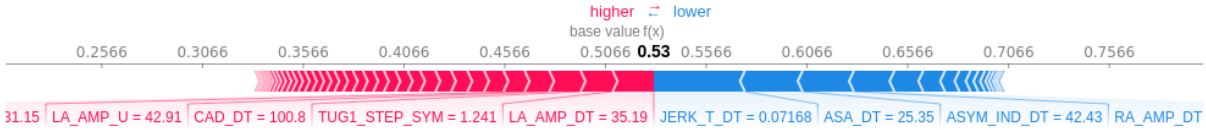

Figure 38: Intelligibilité Locale sur la première valeur (Random Forest)

Ainsi, pour la prédiction de la première personne de la base de test notre modèle de Random Forest nous donne en sortie la valeur 0.53, la classification est peu précise, en effet plus l'on se rapproche de 1 plus la personne risque d'être dans la phase avancé de Parkinson et inversement vers 0 la personne à plus de risque d'être dans la phase Podromale.

La valeur de base est à 0.5060 qui correspond à la moyenne des prédictions du modèle sur l'ensemble des données d'entraînement. Ainsi, cela correspond à la valeur qui serait prédite si nous ne connaissions aucune caractéristique pour la personne observé. En effet il est tout à fait cohérent d'avoir une valeur de base proche de 0.5 étant donné sur 192 personnes, la moitié sont en phase prodromal et l'autre moitié en phase avancée de Parkinson. Ainsi, le modèle nous prédit pratiquement autant de 1 (PD) que de 0 (Podromal), et donc une moyenne proche de 0.5.
Les chiffres sur les flèches du graphique sont la valeur de la feature pour cette instance.

Le fait que JERK-T-DT (Jerk from Left arm in Dual task walking (deg/sec^3)) = 0.072, fait tendre la classification de la personne vers l'état prodromal de façon conséquente (décalage de 0.05 vers la gauche donc vers Podromal). C'est cette mesure qui à le plus d'impact sur la classification de la personne.

De la même façon, LA-AMP-DT (Average Amplitude of the Left arm degree during Dual Task) = 35.19 pour le modèle à classer la personne comme étant plutôt dans une phase avancée de parkinson (décalage d'environ 0.03 vers la droite donc vers PD)

Le rouge représente les caractéristiques qui ont fait tendre la classification de la personne vers la phase Parkinsonienne, et le bleu les caractéristiques qui l'ont fait tendre vers Podromal. In-

tuitivement, plus la flèche est grande, plus l'impact de la caractéristique sur la sortie est important.

Si nous voulons une représentation plus globale des prédictions précédentes, nous pouvons utiliser une variante du diagramme de force. Dans la figure ci-dessous, nous pouvons voir les prédictions sur l'ensemble de test.

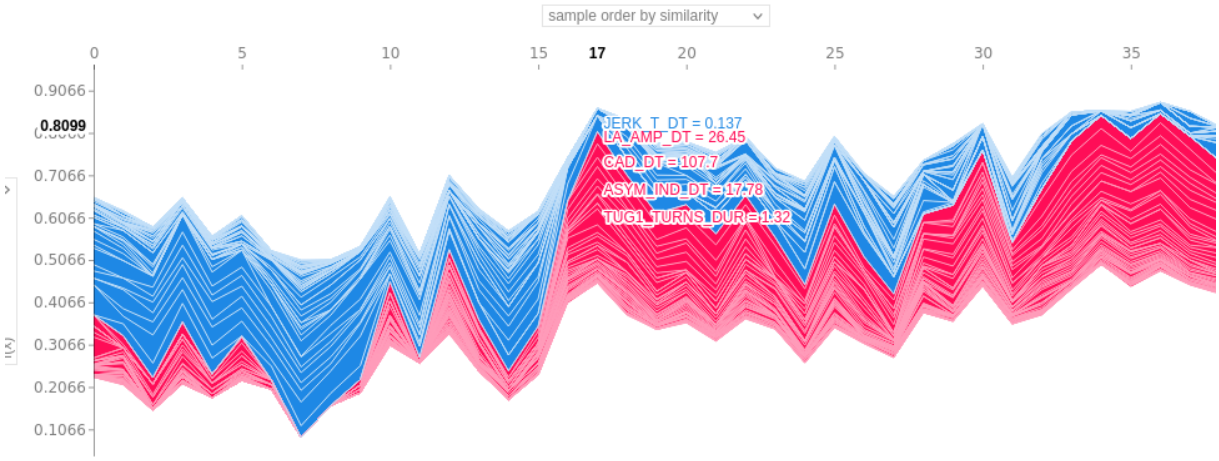

Figure 39: Intelligibilité Locale sur l'ensemble des valeurs de test (Random Forest)

L'axe des ordonnées nous donne la valeur obtenu en sortie, ainsi de façon interactive on peut observer sur chaque donnée de test l'impact de chacune des mesures sur la valeur de prédiction de la phase de la maladie chez la personne (PD ou Prodromal).
On observe par exemple que pour la 17 ème personne du test sur la figure, la probabilité pour que la personne soit en phase parkinsonienne est de 0.81 (personne en réalité PD donc classification correcte) avec encore une fois JERK-T-DT et LA-AMP-DT étant les mesures ayant l'impact le plus important sur la classification de cette 17ème personne, ce qui est globalement en accord avec la figure 8 d'intelligibilité globale.

7 Conclusion

À travers ces travaux, nous arrivons à notre échelle à parvenir à des résultats pour la classification des personnes atteintes de la maladie, ainsi que pour la prédiction du freezing of gait. À terme, ces études de machine learning, couplé à des méthodes pour faciliter la collection de données pourront aboutir à des dispositifs accessible à tous pour évaluer soi même son risque de parkinson, ou anticiper les situations dangereuses que peuvent causer le freezing of gait.

De plus, nous rappelons que le taux de mauvais diagnostiques de la maladie reste très élévé

(environ 20%) et que ce diagnostique se fait par constatation des symptômes, de façon similaire aux données que nous analysons avec ces alorithmes. Nous pouvons donc également espérer qu'un jour les données rigoureuses analysés par des systèmes d'apprentissage profond pourront dépasser la capacités des humains en ce qui concerne le diagnostique.

Les enjeux médicaux et financiers sont immense, pour une maladie qui coûte 10 mille euros par an par patient. Nous avons pu constater le potentiel de ce domaine et espérons que des progrès continueront dans un futur proche !

References

[1] B. S. N. J. Payami H, Larsen K, "Increased risk of parkinson's disease in parents and siblings of patients," 1994. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/7605419/

[2] A. L. Connie Marras, "Parkinson's disease subtypes: lost in translation?" 2012. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/22952329/

[3] A. S. M. D. F. A. L. G. Rizzo G, Copetti M, "Accuracy of clinical diagnosis of parkinson disease: A systematic review and meta-analysis," February 2016. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/26764028/

[4] Q. W. Q. L. Q. Zou, L. Ni and S. Wang, "Robust gait recognition by integrating inertial and rgbd sensors," April 2018. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/28368842/

[5] A. L. G. L. H. J. I. P. C. M. R. . . S. H. E. P. P. Goldberger, A. and PhysioNet, "Gait in parkinson's disease," 2008. [Online]. Available: https://physionet.org/content/gaitpdb/1.0.0/

[6] A. L. G. L. H. J. I. P. C. M. R. . . S. H. E. . P. P. Goldberger, A. and PhysioNet, "Gait in neurodegenerative disease database," 2015. [Online]. Available: https://www.physionet.org/content/gaitndd/1.0.0/

[7] H. Li, "Multimodal dataset of freezing of gait in parkinson's disease," 2021. [Online]. Available: https://data.mendeley.com/datasets/r8gmbtv7w2/3

[8] J. H. Daniel Roggen, Meir Plotnik, "Daphnet freezing of gait data set," 2013. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Daphnet+Freezing+of+Gait

[9] W. Zeng and C. Wang, "Classification of neurodegenerative diseases using gait dynamics via deterministic learning," 2015. [Online]. Available: https://dl.acm.org/doi/abs/10.1016/j.ins.2015.04.047

[10] H. Z. Aite Zhao; Jianbo Li; Junyu Dong; Lin Qi; Qianni Zhang; Ning Li; Xin Wang, "Multimodal gait recognition for neurodegenerative diseases," 11 March 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9376702

[11] W. Zhang, D. Huang, H. Li, L. Wang, Y. Wei, K. Pan, L. Ma, H. Feng, J. Pan, and Y. Guo, "Sensoring and application of multimodal data for the detection of freezing of gait in parkinson's disease," 2021. [Online]. Available: https://arxiv.org/abs/2110.04444

[12] T. A. M. C. T. A. U. Anat Mirelman, PhD, "Gait assessment – methods document," 19 March 2018. [Online]. Available: https://github.com/Ohm-T/Parkinson-Classification/blob/main/Tom/PPMI_Methods_Gait_AM0180319.pdf

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016. [Online]. Available: http://arxiv.org/abs/1603.02754

[14] C. Molnar, *Interpretable Machine Learning*, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book