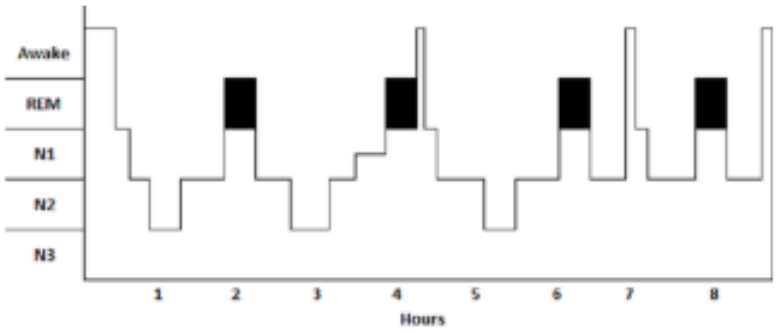
Machine Learning Challenge - Dreem - Sleep Stage Classification

January 4, 2021

1 Objectif et description des données

L'objectif de ce challenge est de réussir à prédire la phase de sommeil dans laquelle se trouve un sujet porteur du casque de mesure développé par Dreem. Les phases de sommeil sont définies comme suit :

- 0 Wake
- 1 NREM1 Sleep (light sleep 1)
- 2 NREM2 Sleep (light sleep 2)
- 3 NREM3 sleep (deep sleep)
- 4 REM sleep (paradoxical sleep)



Les données recueillies par le casque et mises à notre disposition sont les suivantes :

- 7 enregistrements EGG (3 mesures frontales, 4 mesures fronto-occipitales)
- Un enregistrement de pulsation cardiaque
- 3 enregistrements de mouvements des yeux, un sur chaque axe de l'espace

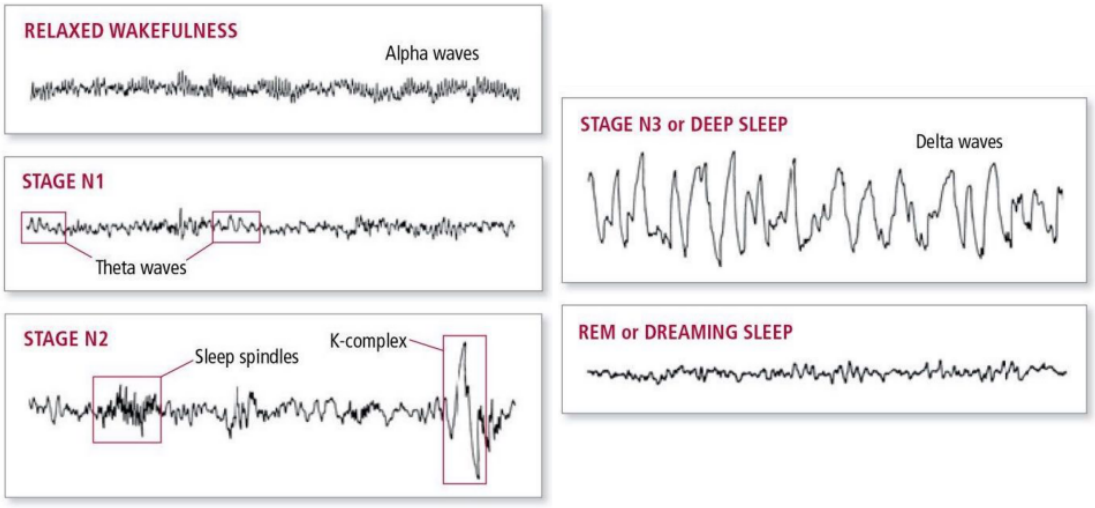
Dans la suite, nous allons proposer un modèle de classificateur permettant de prédire la phase de sommeil à l'aide de ces données, et définir des caractéristiques interessantes quant à l'étude de ces phases.

2 Extraction de caractéristiques

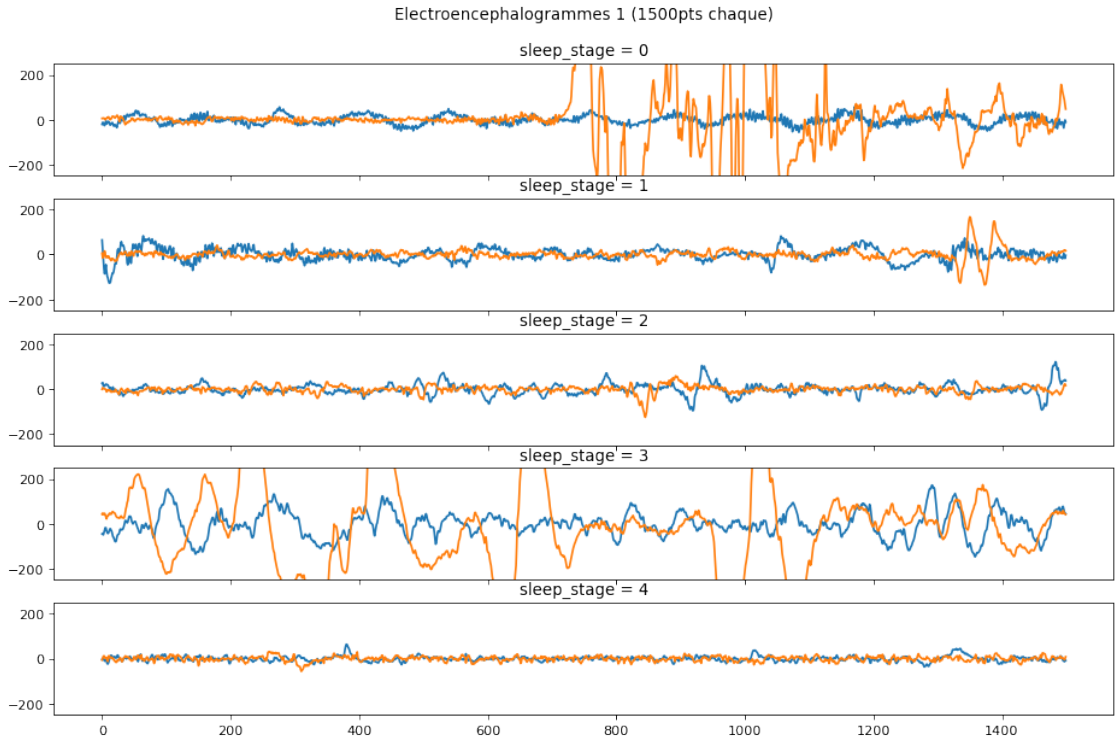
2.1 Features extraites des EEGs

2.1.1 Prétraitement

En observant un échantillon de signaux EEGs, on constate que certains motifs sont caractéristiques de chacunes des phases de sommeil. Ceux-ci sont résumés sur la figure ci-dessous.



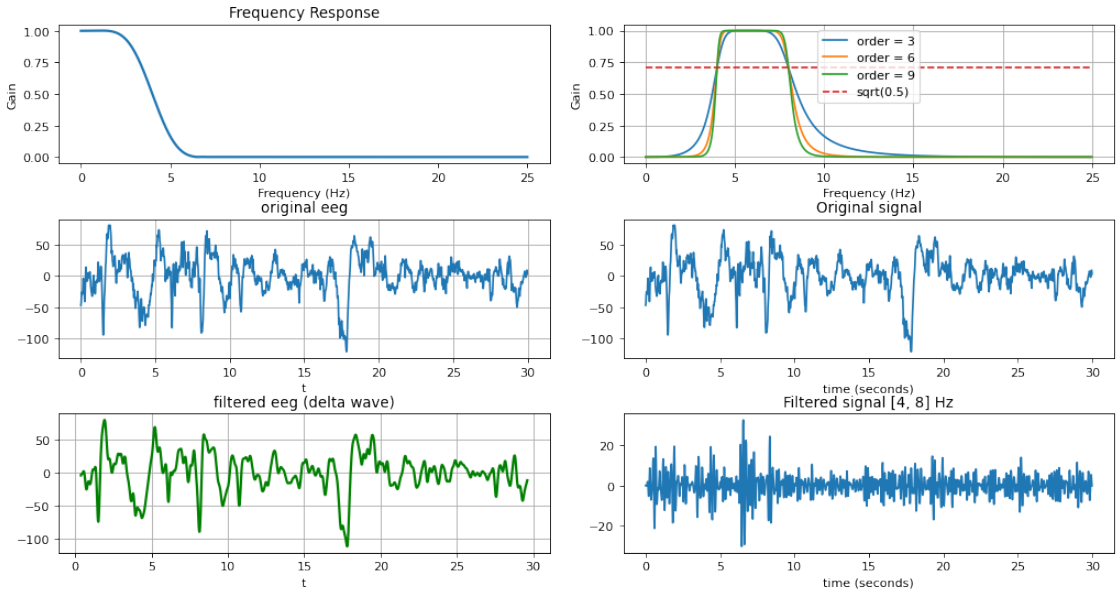
Dans les enregistrements EEG fournis, on retrouve les caractéristiques décrites précedement. Cependant, on constate que les enregistrements sont également relativement bruités, et presentent des sauts qui pourraient être la résultante de mouvement du sujet lors de l'enregistrement.



On peut donc sentir qu'une information sur la phase de sommeil se trouve dans la composition fréquentielle du signal ainsi que dans son énergie. Afin d'extraire des caractéristiques des signaux EEG qui pourraient ainsi être utiles, nous cherchons à calculer la puissance des signaux dans différentes bandes de fréquence. En nous appuyant sur les travaux de Khald Ali I. Aboalayon, Miad Faezipour, Wafaa S. Almuhammadi et Saeid Moslehpour, nous allons décomposer le signal selon les bandes de fréquences suivantes :

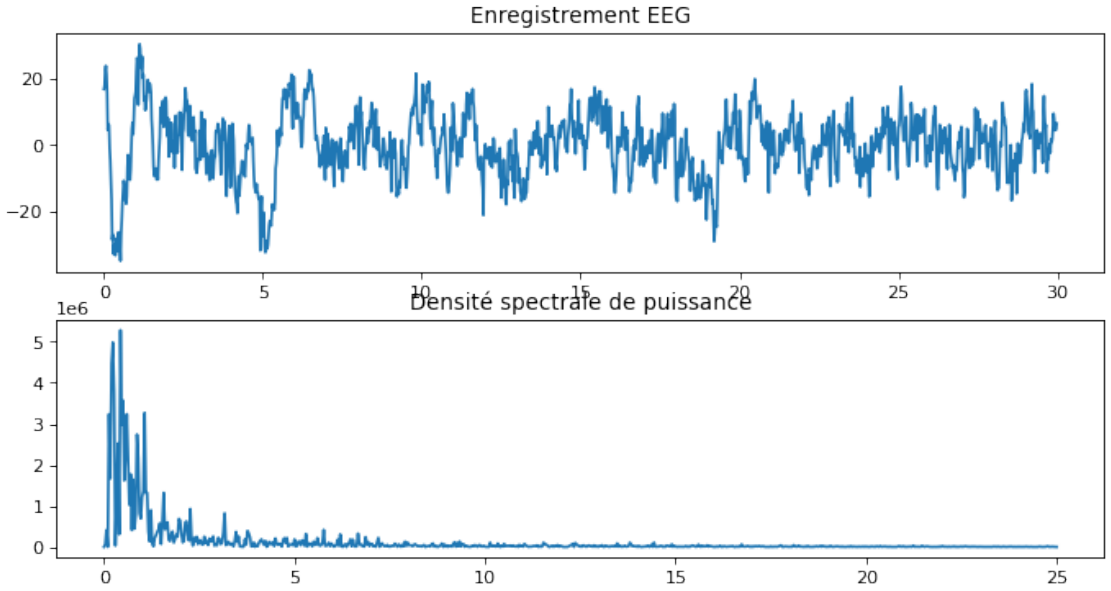
- Delta (δ) : 0-4 Hz
- Theta (θ) : 4-8 Hz
- Alpha (α) : 8-13 Hz
- Beta (β) : 13-30 Hz
- Gamma (γ) : >30 Hz

Puis nous calculerons la puissance dans chacune de ces bandes. On utilise des filtres passe bas/haut/bande pour obtenir la réstriction du signal à ces bandes de fréquence. Les filtres utilisés sont des filtres Butterworth qui présentent l'avantage d'avoir un gain aussi constant que possible sur la bande passante. Sur la figure ci-dessous se trouvent deux exemples de EEG filtrés.



2.1.2 Calcul de l'énergie du signal dans les bandes spectrales

L'approche la plus simple est de calculer la densité spectrale de puissance du signal EEG en utilisant la méthode Fast Fourier Transform. Puis, pour chacune des bandes de fréquence $\delta, \theta, \alpha, \beta$ (γ étant trop haute vis à vis de la fréquence d'échantillonnage 50Hz) on calcule l'énergie sur la bande de fréquence. Ci-dessous se trouve un EGG1 ainsi que sa transformée de densité de puissance. On obtient, avec les 7 EEGs, 28 features. Ci-dessous se trouve la densité spectrale de puissance d'un signal non filtré.



Plusieurs méthodes existent pour calculer la densitée de puissance et l'énergie, à commencer par appliquer la formule suivante à la transformée de Fourier FFT :

$$E(f_{min}, f_{max}) = \sqrt{\sum_{f=f_{min}}^{f_{max}} FFT(f)^2}$$

D'autre méthodes existent, telles que celles proposées dans la librairie `bandpower` et notamment les methodes 'weltch' et 'multitaper'. Celles-ci propose une variance plus faible dans le domaine fréquencielle. Après plusieurs essais, les meilleurs résultats ont été donnés avec la méthode 'multitaper'.

2.1.3 Calcul de l'entropie des EEGs

En observant les EEG affichés plus tôt, on constate le caractère stochastique de ceux-ci, caractère qui se manifèste d'autant plus dans certaines phases du sommeil. L'entropie, qui s'exprime sous une forme de type

$$H(s) = -\sum_i ln(P(s_i))\mathbb{P}(s_i)$$

permet de donner une indication sur le caractère irrégulier du signal. On utilise ici les fonctions de la library `EntroPy` pour calculer l'entropie sous plusieurs méthodes (ApEnt, LizEnt, SamEnt, DisEnt), ainsi que les mesures fractales Detrended fluctuation et Higuchi fractal dimension. On calcule ces caractéristiques sur les 7 EEG ce qui donne 42 features supplémentaires.

2.1.4 Calcul de l'Esis et du MMD

L'idée de cette caractéritique est que l'on considère que le signal une vitesse et une énergie. La vitesse du signal est calculée à partir de la fréquence du signal et de son longueur d'onde $v = f \times \lambda$. Pour la fréquence, f, on prende le milieu de la bande de fréquence considérée (ex: pour les ondes delta f = 2Hz, pour les theta f = 6Hz). Pour la longueur d'onde on prend $\lambda = 100$ (longueur des fenêtres définies pour MMD).

Puis,

$$Esis = \sum_{i=1}^{N} |X_i|^2 \times v$$

X_i : valeur i du signal et N = longueur du signal

Pour chaque bande de fréquence, on divise le signal en 15 fenêtres de 100 points et on calcule sur chaque fenêtre la valeur d entre le maximum et le minimum du sous-échantillion. Puis on calcule le MMD du signal (filtré).

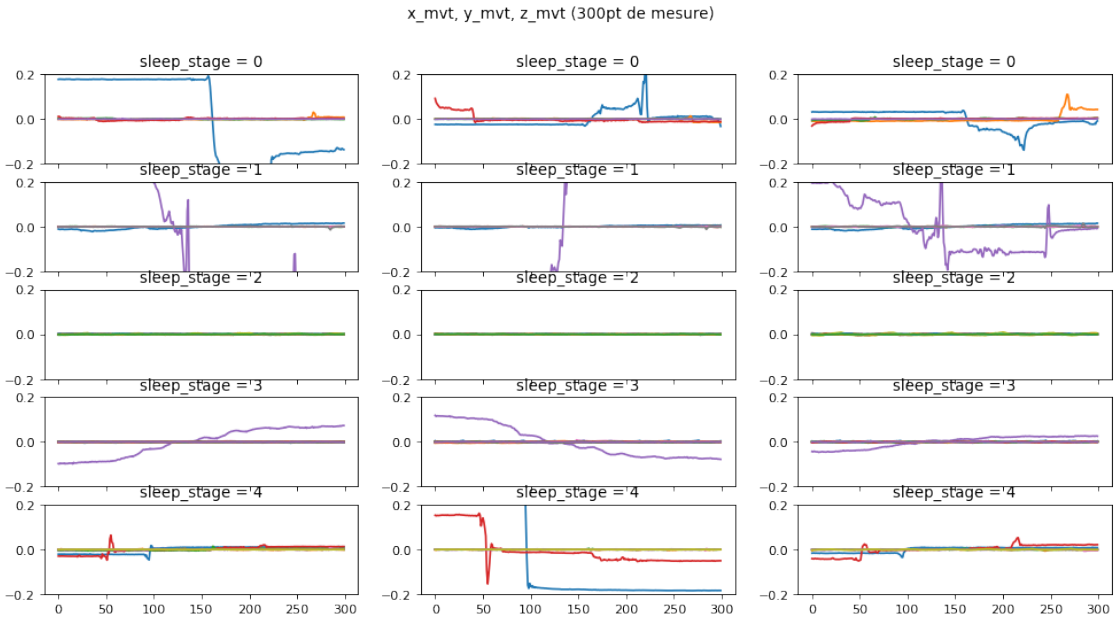
$$d = \sqrt{\Delta x^2 + \Delta y^2}$$

$$MMD = \sum_{i=1}^{w} |d_i|$$

w : total number of sliding (sub) windows in an epoch.

2.2 Features pour le mouvement des yeux

En observant les signaux du mouvement des yeux, nous remarquons que les mouvements rapides et amples dans les classes basses de sommeil (0,1,2). On constate aussi des oscillations courtes et rapides dans le stade de sommeil paradoxal REM. Sur la figure ci-dessous se trouvent quelques exemples de mouvements des yeux en fonction de la phase de sommeil.



On peut donc supposer que l'amplitude, la vitesse, ansi que l'aléatoire des mouvements sont significatifs de la classe de sommeil dans laquelle le témoin se trouve. Face à ces constats, nous proposons donc analyse temporelle, fréquencielle et énergétique des signaux, ce qui permettra de quantifier l'amplitude des mouvements ainsi que la vitesse de ceux-ci. Nous calculons pour chacunne des variables de position :

Temporel:

1. Écart type de la position
2. Écart type de la variation de position
3. Amplitude moyenne de la position
4. Amplitude moyenne de la variation de position
5. Énergie du signal
6. Entropie du signal ApEnt et mesure fractale du signal

Fréquenciel:

8. Valeur maximale de la transformée de Fourier
9. Fréquence à laquelle ce maximum est atteint
10. Energie du signal par la densité spectrale de puissance
11. Puissance moyenne du signal

Cela représente 33 features au total.

2.3 Features extraites des pouls

Pour le pouls, nous avons choisis différentes features tirées de l'analyse fréquencielle du signal.

Temporel :

1. Fréquence cardiaque. Pour extraire la fréquence cardiaque, nous utilisons la bibliotheque heartpy dont la fonction de calcul du BPM donne de bons résultats malgré le bruit présent sur les signaux. Dans le cas ou cette fonction ne peut pas donner de résultat, nous utilisons la médiane de la série comme valeure (utilisation de SimpleImputer)
2. Variation de la fréquence cardiaque RMSSD. Cette mesure, également issue de la bibliothèque heartpy, donne une information sur la variation de la pulsation cardiaque. Pour se faire, la distance entre deux pics est prise en moyenne quadratique.

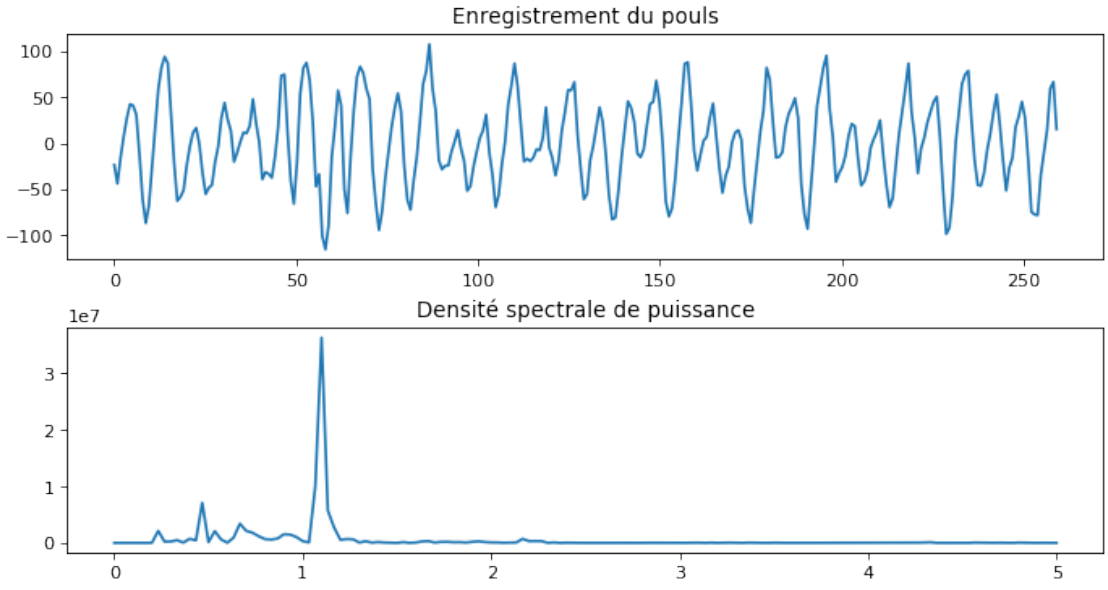
$$RMSSD = \sqrt{\frac{1}{n-2}\sum_{0}^{n-2}(R_{n+1} - R_n)^2}$$

3. Mesures d'entropies ApEn SamEn DisEn et complexité de Lempel-Ziv (définies plus tôt). Ces mesures permettent de caractériser l'irrégularité dans les signaux, irrégularité qui présente une correlation interessante avec les phases de sommeil.
4. Amplitude moyenne, variance absolue moyenne et énergie

Fréquentiel :

1. Valeure moyenne de la densité spectrale de puissance
2. Énergie du signal en passant par la densite spectrale de puissance

Au total, cela représente donc 6 features.



3 Utilisation de Random Forest pour classifier

Nous avons testé les modèles suivants en effectuant un entrainement simple sans optimisation particulière. Les modèles testés sont les suivants, ainsi que les F1-score recolté

- SVC, 0.208
- KNN 5 clases, 0.636
- MLPClassifier, 0.323
- DecisionTree, 0.650
- RandomForest, 0.739
- Naive Bayes (GaussianNB), 0.093

Les résultats nous ont conduit à selectionner le RandomForest

3.1 Description de l'algorithme Random Forest

RandomForest est une technique d'apprentissage basée sur la construction d'arbres de décision. Un grand nombre d'arbre décisionnel est construit, puis la meilleure prédiction est sélectionnée.

Chaque arbre est construit en utilisant des techniques connues de construction d'arbres décisionnels et à l'aide - d'un échantillon d'observation de taille donnée tiré aléatoirement avec remise parmi les observations d'entrainement - d'un échantillon de caractéristiques de taille plus petite que la racine du nombre totale de caractéristiques, elles aussi tirées aléatoirement parmi les caractéristiques fournies

Nous utiliserons le modèle RandomForest implémenté dans la bibliothèque `sklearn`. Dans ce module, les paramètres tels que le nombre d'arbre, le nombre minimal d'observation par feuilles ou par noeuds, la profondeur maximale de l'arbre, le mode de construction de chaque abre, peuvent êtres réglés. Aussi nous rechercherons les paramètres idéaux pour notre modèle à l'aide d'une *grid search*.

3.2 Post-processing des features

Lorsque certaines mesures (notamment le pouls) ne peuvent être calculées, un NaN est inséré à la place dans les dataset. Ici nous allons donc utiliser un imputer pour remplacer les valeures NaN par la médiane de la série. En effet, lorsque l'on ne peut pas estimer une mesure, retenir la médiane permet de réduire le risque d'être trop éloigné de la vraie valeur alors inconnue.

Nous avons également tenté de procéder à de l'aglomération de caractéristique par les méthodes PCA et Feature Agglomeration du module sklearn, mais sans grand impacts sur les résultats.

3.3 Sélection des features

Face au grand nombre de features et à la faible significativité de celles-ci, nous avons entrainé un premier modèle avec l'entièreté des caractéristiques, puis avons sélectionné celles qui avaient le plus d'importance dans le modèle. Nous avons également veillé à ce que les features ainsi selectionnées avaient une variance suffisante pour être justement exploitées.

Nous utilisons pour cela la methode 'feature-importances' du modèle RandomForest entrainé, et sélectionnons les caractéristiques ayant une importance supérieure à 0.005.

```
Train prior to feature removal : 0.9989037180994466
Test prior to feature removal : 0.7729378922075391
[[ 481   10   36    1   14]
 [  62   47  106    3   42]
 [  27   12 1206   86   72]
 [  26    1   88  659    1]
 [  34   11  141   18  520]]
```

```
Features to remove :
Index(['hrv', 'lzi_ent', 'avamp', 'energie', 'fft_avg', 'fft_ene',
       'eye_time_varstd_x', 'eye_time_ampavg_x', 'eye_time_detfluc_x',
       'eye_fft_max_x', 'eye_fft_argmax_x', 'eye_fft_argmax_y',
       'eye_fft_argmax_z', 'eye_fft_ene_x', 'eye_fft_powavg_x', 'eeg3_app_ent',
       'eeg7_app_ent', 'eeg7_delta_power', 'eeg7_theta_power',
       'eeg7_alpha_power', 'eeg1_beta_mmd', 'eeg2_delta_esis',
       'eeg2_theta_mmd', 'eeg2_alpha_mmd', 'eeg2_beta_mmd', 'eeg3_delta_mmd',
       'eeg3_delta_esis', 'eeg3_theta_mmd', 'eeg3_alpha_mmd', 'eeg3_beta_mmd',
       'eeg4_delta_mmd', 'eeg4_delta_esis', 'eeg4_theta_mmd', 'eeg4_alpha_mmd',
       'eeg4_beta_mmd', 'eeg5_beta_mmd', 'eeg6_theta_mmd', 'eeg6_alpha_mmd',
       'eeg6_beta_mmd', 'eeg7_delta_mmd', 'eeg7_delta_esis', 'eeg7_theta_mmd',
       'eeg7_theta_esis', 'eeg7_alpha_mmd', 'eeg7_beta_mmd'],
      dtype='object')
```

```
Train prior to feature removal : 0.9984262647079574
Test prior to feature removal : 0.7920467187731517
[[ 473    8   37    3   14]
 [  53   46   84    5   41]
 [  32   10 1275   52   71]
 [  12    3   86  667    2]
 [  29   11  166    9  515]]
```

3.4 Grid search et cross-validation

Nous commençons par effectuer une grid-search. Nous découpons le dataset en un échantillon d'entrainement et un échentillon de test, et nous effectuons la grid-search avec les paramètres suivants :

```
{'max_samples': [0.3, 0.5, 0.7], 'min_samples_leaf': [1, 3, 5], 'max_features':
[1, 3, 5], 'max_depth': [None, 5, 10], 'criterion': ['gini', 'entropy'],
'n_estimators': [100], 'bootstrap': [True]}
```

```
Train: 0.9983795468225298
Test: 0.7621736894333069
[[ 462   13   39    2   26]
 [  58   53  106    5   40]
 [  47   13 1209   54   60]
 [  23    2  119  626    3]
 [  30   15  165   18  516]]
```

Les paramètres retenus sont les suivants :

```
{'bootstrap': True,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 5,
 'max_samples': 0.7,
 'min_samples_leaf': 1,
 'n_estimators': 100}
```

Enfin nous effectons une cross validation à 10 repliements pour valider les perfomances du modèle. Les résultats sont les suivants :

```
Accuracy using RF with 10 cross validation : 65.19%
Scores of folds:
[0.58930741 0.6618064  0.73390036 0.65937627 0.6581612  0.61846902
 0.69380316 0.67557716 0.56199352 0.66693679]
```

4 Conclusion

Le modèle proposé permet de prédire la phase de sommeil avec une précision moyenne de 65% (F1-score).

Pour améliorer ce score, on pourrait explorer les pistes suivantes :

- Retraiter les enregistrements pour tenter d'en extraire le bruit, ou ignorer la mesure lorsque l'enregistrement est abérant (par exemple si l'EEG présente des sauts, si le pulse est trop bruité, etc). Le modèle ne serait alors pas trompé lors de l'entrainement.
- Affiner les caractéristiques, par exemple pour mieux détecter les motifs spécifiques sur les EEG (les K-complexes par exemple)