UE Electifs S8

Machine Learning et Applications

Projet de ML: Prédiction du sexe à partir des rythmes cérébraux

Etudiants
Anass EL HOUD
Nouha LIMOURI
Fayza EL GHAZAL

Enseignant
Pr. Yohann DE CASTRO

Ecole Centrale de Lyon
Ecully, France

5 avril 2020

Table des matières

1 Introduction

1.1 Définitions du Machine Learning

Le Machine Learning, aussi appelé apprentissage automatique en français, est une méthode d'analyse de données qui permet à la machine d'apprendre en développant un modèle analytique. Ce modèle est obtenu par le biais d'algorithmes capables, après plusieurs itérations, de découvrir les corrélations préexixtantes entre les données. Anisi,le Machine Learning permet aux ordinateurs de découvrir des insights cachées sans être programmés pour savoir où les chercher[1].
le Machine Learning est un sous domaine de l'intelligence artificielle qui s'interesse principalement à la reconnaissance de pattern à partir des données. Aujourd'hui, il constitue la base indispensable pour la majorité des carrières les plus pertinentes en analyses de données [2].
Il est également défini comme étant l'étude de différents types de problèmes : prise de décision, apprentissage supervisé et non supervisé, clustrering, classification, forecasting, deep-learning, l'apprentissage renforcé, algorithmes génetiques, etc.

Notre projet s'interesse à un challenge de résolution d'un problème qui s'inscrit dans le cadre de l'apprentissage supervisé, la classification plus précisement.

1.2 Domaines d'application du Machine Learning

Le Machine Learning est utilisé dans les différents secteurs de l'industrie aussi bien que dans les domaine créatifs comme la peinture ou le cinéma.
Cette technologie permet de gagner en termes de concurrence puisqu'elle consiste à extraire les informations en temps réel à partir de données. C'est la raison pour laquelle plusieurs industries l'adopte aujourd'hui pour collecter le maximum d'informations et de corrélations entre les différentes données.

La finance est parmi les premiers secteurs à s'intéresser au domaine de l'apprentissage automatique afin de découvrir les informations importantes qui se cachent derrière les données. L'application des différents algorithmes du Machine dans ce domaine permet d'identifier les opportunités d'investissement et les clients à haut risque. Dans ce cardre, s'inscrit également la cybersurveillance qui permet aux banques, par exemple, de détecter les opérations frauduleuses[1].

Le e-marketing est également un domaine où le Machine learning est utilisé de plus en plus pour analyser l'historique d'achat et les préférences des clients pour suggérer les produits qui pourraient les intéresser[1].

Le Machine Learning s'intéresse aussi au secteur de la santé. Des différents types de capteurs sont utilisés dans la collecte de données des patients en temps réel. Une fois la base données

établie, les algorithmes du Machine Learning sont capable d'analyser les corrélations entre les données pour pouvoir prédire sur de nouveaux cas sans avoir à les étudier. Cette technologie peut aussi aider les experts médicaux à analyser les données pour identifier des tendances alarmantes afin d'améliorer les diagnostics et les traitements[1].
Dans ce cadre vient s'inscrire notre projet. En effet, notre challenge s'interesse au secteur de la santé,plus précisemment la physiologie, et consiste à développer un algorithme capable de prédir au mieux le sexe d'une personne en se basant sur les rythmes cérébraux.

1.3 L'apprentissage supervisé comme tâche du Maching Learning

Les problèmes de l'apprentissage supervisé sont les plus populaires en Machine Learning comme en Deep Learning. Avec ce type d'apprentissage, la machine est amenée à apprendre une tâche en étudiant des exemples de cette tâche. Ces exemples sont appelés jeu de données et sont partagés en training set et test set.
Les données de l'entrainement, dans ce cas, portent des 'labels', c'est à dire que la sortie pour chaque donnée est connue. La machine apprend, à travers des entrées et sorties qu'on lui fournit, et génère un modèle (une fonction) avec une erreur minimale.
Ce modèle est testé sur les données de test en essayant de prédire la sortie pour ces nouvelles données. L'objectif est de trouver le modèle optimal qui prédit au mieux les sorties avec un minimum d'erreur, un modèle qui peut être généralisé. Pour ce faire, il est nécessaire de faire des adaptations et beaucoup d'efforts pour comprendre le problème et proposer plusieurs voies de résolution.

Plusieurs algorithmes d'apprentissage supervisé sont disponibles : Réseaux de Neurones, arbres de décisions, Random Forest, ect. Cependant, chaque algorithme possède des plus et des moins. Il est donc nécessaire de bien définir le problème pour choisir l'algorithme le plus adéquat.

2 Présentation du contexte

2.1 L'électroencephalographie pour prédire le sexe

Le challenge sujet de notre projet est un challenge proposé par une starup de neurotechnologie Dream basée à Paris et consiste à prédire le sexe d'un individu en fonction de son activité cérébrale.
Mesurer l'activité cérébrale ou autrement dit l'électroencephalographie est un examen médical qui consiste à placer des capteurs sur le crâne d'une personne pour mesurer l'activité électrique des neurones localisés aux alentours de ces capteurs. Ainsi, on va pouvoir mesurer un champs magnétique qui sera traduit en time series : le temps selon l'axe des abscisses et l'amplitude de l'activité mesurée à l'endroit du capteur en ordonnée.

FIGURE 1 – Technologie de l'électroencephalographie

Lorsqu'une personne est endormie, et en fonction de son état de sommeil, on va observer des patterns différents qui vont permettre de caractériser un peu mieux ce qui se passe au niveau du cerveau pendant la nuit.

2.2 Objectif et données du problème

Ce challenge se base sur une ancienne publication de Scientific Reports et a comme objectif de reproduire les résultats de cette publication en utilisant des données différentes.

2.2.1 Version publiée sur Scientific Reports

Les idées clées de cette publication sont :

- 80% de précision pour prédire le sexe d'un individu à partir de son électroencephalogramme en utilisant le Deep Learning.
- la bande 20-25 Hz de l'activité cérébrale est déterminante pour caractériser le sexe d'un individu.
- la capacité des réseaux norones d'extraire un grand nombre d'informations de l'electroencephalogramme comparé à une évaluation visuelle par un médecin.

Le dataset utilisé dans cette publication comprend :

- 1308 sujets répartis en Train et Test.
- 40 segments de 2 secondes de l'électroencephalogramme mesurés sur 26 endroits différents de la tête pour chaque sujet.
- patients testés avec les yeux fermés et dans un état relaxé.

2.2.2 Cahier des charges du projet

Pour notre challenge, on utilise des données issues du bandeau Dream porté pendant la nuit par le sujet et qui va permettre de mesurer l'activité cérébrale et donner l'électroencephalogramme. Pour cela, on prend les mesures sur 7 endroits différents de la tête en position frontale et en position occipitale. En faisant les mesures pendant plusieurs nuits d'affilée on va pouvoir constituer notre base de données.

Finalment on obtient le dataset :

- 946 individus différents pour le Train et 946 individus différents pour le Test, 78% hommes.
- 40 segments de 2 secondes de l'électroencephalogramme sur 7 endroits différents échantillonés à 250 Hz.
- sujets testés pendant leur sommeil (yeux fermés, état relaxé).

Résumé :

- **Dataset :** Pour chaque individu donné Xi (avec i = 1, ... , 946), on a 40 x 7 x 500 informations.
  40 segments, 7 électroencephalogramme (endroits différents), 500 correspond à 2 secondes d'onde échantillonné à 250Hz = 500 points.
  La variable à prédire Yi (avec i = 1, ... , 946) ; Yi=0 si homme et Yi=1 si femme.
- **Risque :** Précision(accuracy) = ratio des bonnes prédictions.
- **Benchmark :** Réimplémenter l'approche proposée dans la publication d'origine de Scientific Reports , accuracy = 0.77, sachant que la précision d'une prédiction random sur le dataset de test est aux alentours de 0.68.

3 Exploration et préparation des données

Notre base de données est composée de 946 échantillions. Chaque échantillon représente 40 segments indépendants de 2 secondes de 7 parties de canaux (sondes) EEG à 250 Hz. Autrement dit, les données sont enregistrées à une fréquence de 250 Hz (500 échantillons de 2 secondes). Ces données peuvent être représentées sous forme de matrices avec les dimensions :

$$N_{train} = 40 \times 7 \times 500$$

$$N_{test} = 40 \times 7 \times 500$$

Les fichiers étant donné sous format HDF5, il nous fallait faire un peu de recherche pour pouvoir les manipuler. L'idée est d'importer les données sous leur format d'origine (HDF5) et de les convertir à des objects facile à manipuler sous Python.

```
import h5py
import pandas as pd

#Loading X
print('Started loading...')
file = h5py.File('X_train.h5', 'r')
data = file['features'][()]
print('Finished!')
data = np.array(data)

#Loading Y
y = pd.read_csv('y_train.csv')
data_y = np.array(data)
```

Listing 1 – Code pour l'importation des données

3.1 Labels ou classes

Dans ce problème, les deux classes sont homme (0) ou femme (1). En visualisant les données, on peut distinguer entre le nombre d'apparition ou d'observations des deux classes.
Le code suivant permet de calculer le nombre d'observations de chaque classes

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb

#Le nombre d'observations de chaque classe
y['label'].value_counts()
```

Le résultat du code est consigné de le tableau ci-dessous

Classes	Homme (0)	Femme (1)
Nombre	737	209

Pour une visualisation plus claire, on trace l'histogramme des observations des deux classes.

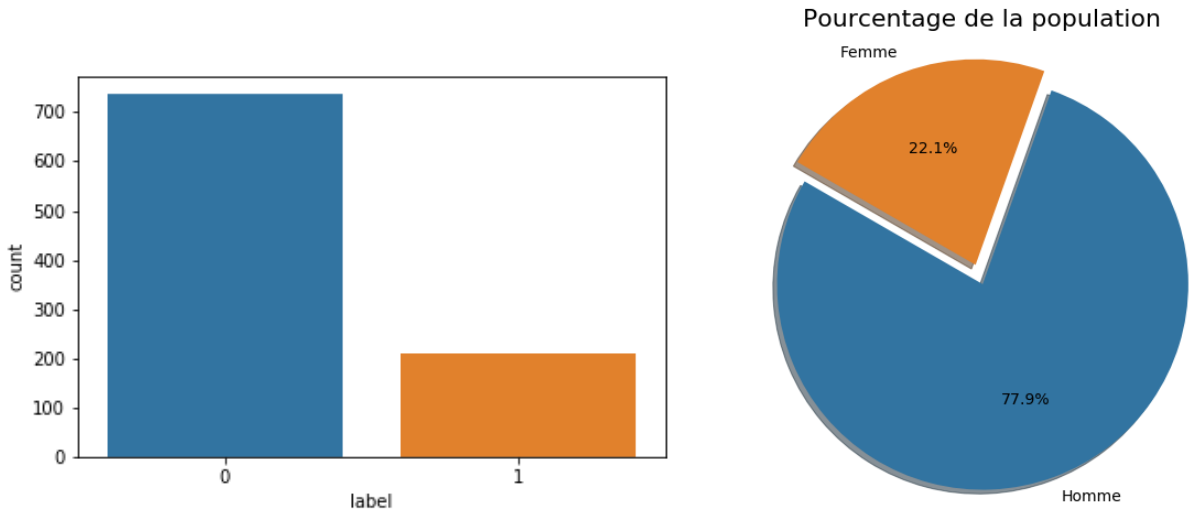


FIGURE 2 – Histograme et Pie cercle des observations pour les deux classes (Homme (0) et Femme (1))

On remarque que le nombre des observations de la classe "Homme" est très grand par rapport à la classe ou label "Femme". Dans ce cas, on risque de biaiser le classifieur d'un premier coup. C'est un problème de **Classification Binaire de Classes Déséquilibrées** ou en anglais, **Binary Label-Imbalanced Classification**. La classe "Femme" dans notre problème de classification est sous-représentée par rapport à l'autre classe. Le modèle de machine learning risque d'être suffisamment pénalisé s'il n'en tient pas compte. C'est ce que l'on remarquera lorsque l'on utilisera une méthode de classification directe sans tenir cet aspect de déséquilibre des classes.
Le calcul des poids pour chaque classe devient crucial dans ce cas, ceci peut se faire par le code :

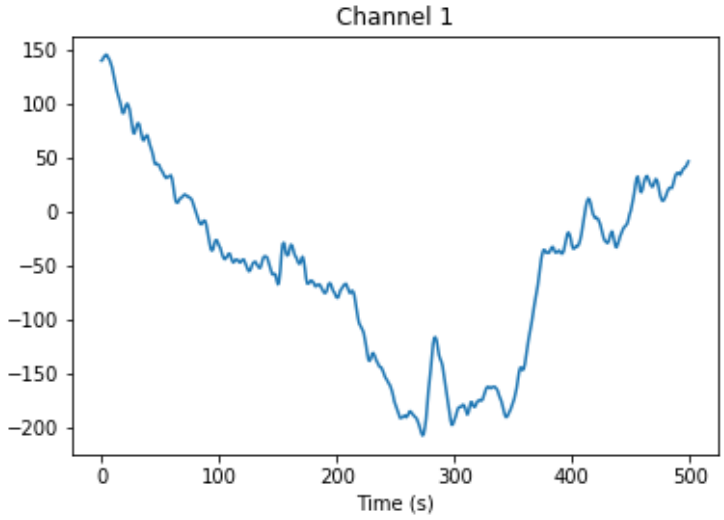
```
import numpy as np
import pandas as pd
from sklearn.utils.class_weight import compute_class_weight

w=compute_class_weight('balanced', np.unique(y), y)
```

Les deux poids résultants des deux classes : w= weights : [0.65178342 , 2.23843117].

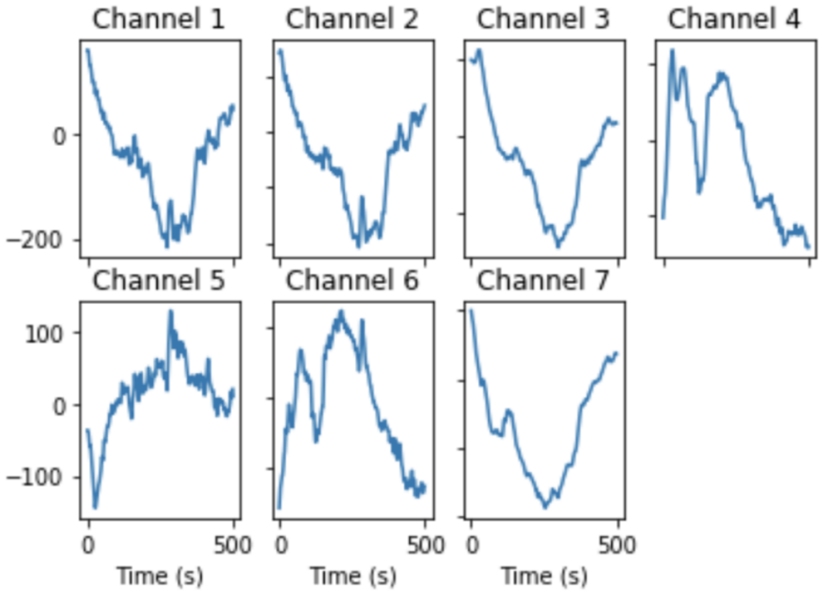
3.2 Visualisation des ondes cérébrales

Dans son livre "Why we sleep ?", le professeur de neurosciences Matthew Walker explique que l'activité cérébrale ne présente pas un schéma régulier vu son comportement rapide et asynchrone. Dans ce cas, on parle des activités cérébrales en sommeil profond NREM ou sommeil REM. Ce caractère supposé chaotique nous mène à visualiser les ondes cérébrales comme un point de départ vers la résolution du problème.

On visualise dans un premier temps, le signal de la première sonde d'un échantillon. La figure prend la forme d'une onde répartie sur 500 secondes.



On peut également visualiser les 7 canaux d'un échantillon en même temps en bouclant sur chaque sous-plot



La première remarque intéressante à faire est que la concaténation de l'onde cérébral enregistré par chaque sonde a une forme similaire à une onde sinusoïdale. Cela peut être intéressant afin d'extraire les features. C'est effectivement l'étape suivante.

4 Résolution du problème

4.1 Extraction des caractéristiques/features

Dans cette partie, on cite les trois propositions à lesquelles on a pensé et qu'on a implémenté par la suite. L'extraction des features est une phase cruciale pour l'implantation du modèle.

1. Premier reflexe est de faire appel aux estimateurs trivials. On applique dans ce cas une réduction de dimension pour passer d'une matrice de dimensions (946 , 40, 7, 500) à une matrice de dimensions (946, 7, 20000) ou (946, 140000). La réduction se fera soit par combinaison des axes ou par les estimateurs triviales comme la moyenne. Cette proposition sera notée par la suite **RDT** (Réduction - Dimension - Triviale).

2. Puisqu'il s'agit des données sous forme d'ondes électriques, l'exploitation des fréquences pourrait également être intéressante. On peut passer au domaine des fréquences par une transformation de fourier et faire une analyse des séries temporelles par le module PyEEG qui est capable d'extraire directement les features EEG par l'échantillon d'entropie [3]. Le modèle sera par la suite construit sur ce domaine au lieu de l'espace temporel. Cette proposition sera notée par la suite **FT** (Fréquence - Transformation).

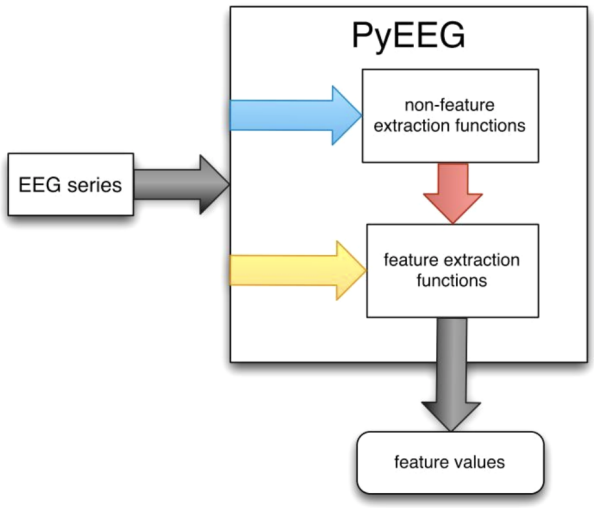


FIGURE 3 – Extraction des features EEG par le module PyEEG

3. Une troisième proposition, qui semble plus sophistiquée, consiste à calculer une caractéristique qui permet de différencier plus profondément entre les ondes enregistrées. Il s'agit de la densité spectrale de puissance (Power spectral density) qui sera estimée par la méthode de Welch. Cette proposition sera notée par la suite **Welch**.

En effet, la méthode de Welch fournit un bon estimateur de la densité spectrale de puissance en suivant un algorithme qui découpe le signal et fait une moyenne de la transformée de Fourier sur chaque segment découpé. Par conséquent, La densité spectrale de puissance estimée s'écrit :

$$\hat{\mathrm{S}}_{\mathrm{Welch}}\left(\omega\right) = \frac{1}{\mathrm{KNU}} \sum_{k=1}^{\mathrm{K}} \left| \sum_{n=1}^{\mathrm{N}} h(n) x(n + (k-1)\mathrm{D}) \exp(-j\omega n) \right|^2$$

Avec K nombre de segments, N est la longueur du segment, D le nombre de points de décalage entre les segments, et U est la constante de normalisation des pondérations.

NB. On note que pour chacune des trois propositions, on appliquera des méthodes adéquates pour pourvoir construire et optimiser notre modèle de classification. d'où la nécessité de leur affecter des noms.

4.2 Prétraitement des données

Le prétraitement des données effectué dans le Jupyter notebook "Prepare Data" comprend les étapes suivantes :

1. Réordoner les données. Ce qui permet de passer d'une forme (946 , 40, 7, 500) à (946, 7, 500, 40) afin de manipuler les 7 sondes en premier.

```
def reorder_data(x):
    a, b, c, d = x.shape
    Xcopy = np.copy(x)
    Xcopy.shape = (a, c, d, b)
    return Xcopy
```

2. Applatir les données en effectuant une réduction de dimension. C'est une étape nécessaire si on cherche à faire une extraction des features par la proposition **RDT**.

```
def flat_x(x):
    a, b, c, d = x.shape
    return x.reshape((a*b, c, d), order='C')
```

3. Les données sont divisées en deux parties. Une pour l'entrainement du modèle, et l'autre pour le test de validation.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

4. Normaliser les caractéristiques en supprimant la moyenne et en mettant à l'échelle la variance des unités par StandardScaler.

```
from sklearn.preprocessing import StandardScaler

sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

4.3 Méthodes et Techniques

Nous proposons 4 modèles ou méthodes pour résoudre le problème. On adaptera ses méthodes pour les trois propositions de l'extraction des features (**RDT**, **FT** et **Welch**). On présente dans un premier temps les quatres modèles de classification proposés.

4.3.1 Random Forest Classifier

Random Forest est un premier choix utilisé pour la classification. Il s'agit de combiner des arbres de décision multiples. En faisant la moyenne de l'impact de plusieurs arbres de décision, d'où la tendance à améliorer la prédiction et le score F1. C'est ce qui nous intéresse.

4.3.2 SVM Classifier

Comme il s'agit d'une classification binaire dont le label de classe est soit la personne est homme ou femme (sexe), et comme le nombre d'échantillons est différent largement du nombre des caractéristiques. Nous pensons également que SVM (Machine à vecteurs de support) serait un bon classificateur dans ce cas.

4.3.3 XGBoost Classifier

XGBoost est une bibliothèque open source fournissant une implémentation haute performance d'arbres de décision renforcés par Gradient Boost. Les performances de XGBoost ne sont pas une blague, elle est devenue la bibliothèque de référence pour gagner de nombreux concours Kaggle.

4.3.4 Convolutional Neural Network

On peut également penser aux réseaux de neuronnes Un réseau de neuronnes convolutifs qui est un algorithme d'apprentissage profond qui peut prendre une donnée d'entrée, attribuer une importance (poids et biais apprenables) à divers features et être capable de retrouver les corrélations et les différences.Le principe de produire des combinaisons non linéaires via des combinaisons non linéaires des entrées pondérées.

5 Implémentation et Expériences

5.1 Première expérience

Comme tout projet numérique, il est intéréssant d'effectuer une première tentative avec une méthode triviale et basique. Nous allons utiliser une forêt aléatoire sur la transformation de Fourier avec des données aplaties. (le choix des hyperparamètres est consigné ci-dessous) :

```
from sklearn.metrics import confusion_matrix, classification_report,
    accuracy_score
from sklearn.ensemble import RandomForestClassifier
import cmath

#Calcul de la TF pour les donnees de depart
X_1=[]
for i in range(0,len(dataset)):
    X_1.append(np.abs(np.fft.fftn(dataset[i])))

nsamples,nx,ny,nz = X_1.shape
X_1 = X_1.reshape(nsamples,nx*ny*nz)

X_train, X_test, y_train, y_test = train_test_split(X_1, y, test_size=0.3)

rf_balanced = RandomForestClassifier(n_estimators=100, class_weight='
    balanced')
rf_balanced.fit(X_train,y_train)
pred_y=rf_balanced.predict(X_test)
print(classification_report(y_test,pred_y))
print(confusion_matrix(y_test,pred_y))
```

La matrice de confusion et le rapport de classification sont présentés dans le tableau suivant

	Precision	Recall	F1-score	Support
0	0.79	0.99	0.88	225
1	0	0	0	59
Accuracy			0.79	284

$$\text{ConvMat}= \begin{bmatrix} 223 & 2 \\ 59 & 0 \end{bmatrix}$$

Avec le même raisonnement, on a implémenté un modèle similaire avec SVM. On constate toujours que le modèle trouve des difficultés énormes à labelliser la deuxième classe (Femme). Le F1-score nulle pour la deuxième classe est un résultat déroutant sachant qu'on a utilisé des poids pour équilibrer l'effet de chaque classe (afin d'éviter les pénalisations dues au déséquilibre des classes dont on a parlé auparavant).

5.2 Deuxième expérience : Réseaux de neurones-CNN

L'architecture du réseau s'est inspirée des réseaux convolutifs profonds qui ont été conçus pour la classification des images. La matrice d'entrée du réseau est une matrice de 7 (canaux EEG) × 256. Le nombre de filtres est 100 dans la première et la deuxième couche à 50 dans les autres couches. L'activation a été effectuée à l'aide de la fonction ReLu. Une fonction de regroupement a été appliquée avant d'utiliser une fonction d'abandon (Dropout). La classification finale a été obtenue en appliquant une couche dense avec une activation softmax, résultant en une probabilité p pour le sexe masculin ou féminin. Les différentes paramètres sont résumés dans la figure qui suit.

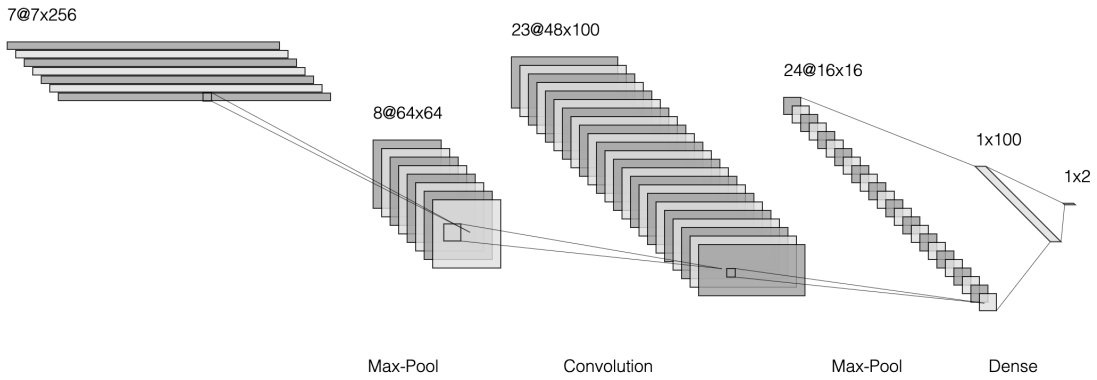


FIGURE 4 – L'architecture de notre réseau proposé

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling1D, Conv1D
from keras import backend as K

num_classes = 2
epochs = 30
```

```
model = Sequential()
model.add(Conv1D(filters=100,kernel_size=3, activation='relu', input_shape=(
    inp1, inp2), padding='same'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=50, kernel_size=1, activation='relu'))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy, optimizer=kera.
    optimizers.Adadelta(), metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1
    ,validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Nous avons lancé l'entrainement du réseau en utilisant les données après réordonner leur dimensions. Chaque segment a reçu une labélisation one-hot (avec la fonction to-categorical qui convertit un vecteur de classe (nombres entiers) en matrice de classe binaire, indiquant un homme ou une femme. L'entraînement a été effectué dans un premier temps avec 30 epochs. Les résultats du modèle sur les données du test (la partie dédiée au test et non pas du challenge) :

Précision	F1-Score
0.80	0.18

5.3 Troisième expérience : Vote majoritaire

En vue d'optimiser notre première tentative, nous avons fait appel à la méthode **Welch**. Tout d'abord, on appliquer cette méthode qui accor estimer la densité spectrale de puissance pour chaque sonde. Comme expliqué auparavant, la méthode de Welch accorde une estimation de la densité spectrale de puissance d'un signal. On l'applique sur chaque élèment de X (chaque sonde) comme montré dans le code suivant

```
from time import time
import scipy.signal as signal

def welch_calcul(x, X):
    sf = 250
    f = signal.welch(x[0], sf)[0]
    trans = np.array([ signal.welch(x[i], sf)[1] for i in range(len(x))])
    a, b, c, _ = X.shape
    return trans.reshape((a, b, c, len(f)))

#Calcul
trainX = welch_calcul(x_train, X_train)
testX  = welch_calcul(x_test, X_test)
wls_train = np.mean(trainX, axis=1)
wls_test =  np.mean(testX, axis=1)
```

Ensuite, on construit 7 classifieurs différents sur chaque sonde avec un apprentissage par vote de majorité. Autrement dit, la prédiction finale sera agrégée par un vote majoritaire. Le choix d'un nombre impair de classifieurs est fait, comme expliqué en cours, pour éviter d'avoir des votes égaux (nombre pair). On a pris comme classifieur "XGBClassifier" qui est un algorithme optimisé de boosting de gradient dont le principe est de combiner les résultats d'un ensemble de modèles plus simples et plus faibles afin de fournir une meilleur prédiction. XGBoost est paramétrable comme tout autre modèle de Machine Learning. Dans le code suivant, on donne un exemple de réglage des hyper-paramètres à l'instanciation du classifier :

```
import xgboost as xgb

params = {'max_depth':2, 'eta':1., 'objective':'binary:logistic' }

clf7 = []

train_pred = np.zeros((len(y_train0), 7))

# 7 classifieurs differents sur chaque sonde
for i in range(7):
    clf7.append(xgb.XGBClassifier(**params))
    bst = clf7[i].fit(psd_train[:, i, :], y_train0)
    predictions = clf7[i].predict(psd_train[:, i])

    y_pred = np.where(predictions >= 0.5, 1., 0.)
    train_pred[:, i] = y_pred
    print('Precision du capteur ' + str(i+1) + ' :' , accuracy_score(
   y_train0, y_pred))
```

```
mean_predictions = np.mean(train_pred, axis=1)
y_train_pred = np.where(mean_predictions >= 0.5, 1., 0.)
print('Precision sur training avec mean voting : ', accuracy_score(y_train0,
    y_train_pred))

clf_voting = xgb.XGBClassifier(**params)
clf_voting.fit(train_pred, y_train0)

predict = clf_voting.predict(train_pred)
y_train_pred2 = np.where(predict >= 0.5, 1., 0.)
print('Precision sur training avec XGB voting : ', accuracy_score(y_train0,
   y_train_pred2))
```

Après le training, les précisions des 7 classifieurs :

```
Précision du capteur 1 : 0.8672985781990521
Précision du capteur 2 : 0.8609794628751974
Précision du capteur 3 : 0.8593996840442338
Précision du capteur 4 : 0.8846761453396524
Précision du capteur 5 : 0.8562401263823065
Précision du capteur 6 : 0.8404423380726699
Précision du capteur 7 : 0.8562401263823065
Précision sur training avec mean voting :  0.8530805687203792
Précision sur training avec XGB voting :  0.9620853080568721
```

FIGURE 5 – Les précisions des classieurs à vote majoritaire

Les performances et les résultats du modèle sur les données de test :

Précision	F1-Score
0.792	0.247

6 Conclusion

A partir de ce projet, on a pu montrer comment l'apprentissage automatique peut être appliqué dans un domaine comme la neuroscience et la médecine. Le problème à traiter était un problème de classification binaire. Nous avons commencé par une visualisation et analyse des données qui permet de bien comprendre le problème et élargir notre espace de solutions à proposer. Après le pré-traitement des données, nous avons implementé trois solutions : La premiere est une classification à l'aide de forêt aléatoire et SVM sur la transformée de Fourier des données aplaties. La deuxième utilisait des réseaux de neuronnes convolutionnels dont l'architecture est inspirée des celles utilisées pour la classification des images. Et la troisième solution se basait sur l'algorithme de XGBoost en utilisant la méthode de Welch pour l'estimation de la puissance

spectrale. Ces deux dernières solutions furent les plus satisfaisantes d'après les résultats (F1-Score et précision). On peut également penser pour optimiser davantage nos solutions à proposer d'autres idées pour extraire les features. Dans ce contexte, on aurait pensé à la méthode qui consiste à détecter les fuseaux de sommeil (qui sont des ondes qui apparaissent soudainement parmi les ondes plus lentes) [5] et faire une classification sur le nombre, la largeur et la durée de ces fuseaux. Cette méthode s'avère également intéressante et peut faire l'objet d'une autre implémentation de ce projet permettant d'atteindre des erreurs de classification plus faibles.

7 Références

[1] "Intro to Machine Learning | Udacity." Intro to Machine Learning | Udacity. Accessed April 27, 2016.
[2] BASTIEN L - Machine Learning : Définition, fonctionnement et secteurs d'application http ://www.artificiel.net/machine-learning-definition
[3] Forrest S. Bao, Xin Liu and Christina Zhang, "PyEEG : An Open Source Python Module for EEG/MEG Feature Extraction," Computational Intelligence and Neuroscience, March, 2011
[4] Solomon, Jr, O M. PSD computations using Welch's method. [Power Spectral Density (PSD)]. United States : N. p., 1991. Web. doi :10.2172/5688766.
[5] Raphael Vallat, A simple and efficient sleep spindles detector. Walker Lab, U.C Berkeley. https ://raphaelvallat.com/spindles.html