SOUTENANCE M1 MIAGE

Par BOUSTIQUE SOUFIANE

Stage du 23 mai 2022 au 09 septembre 2022 chez Apneal en tant que Data Analyst / Engineer

Plan de la présentation

1) Présentation de la start-up et du sujet du stage
2) Environnement technique
3) Missions réalisées et résultats obtenus
    a) Labellisation
    b) Algorithmes pour la manipulation de base de données
    c) Algorithmes pour migration de données RML
    d) Algorithmes pour la manipulation de fichiers de signaux
    e) Entraînement et exploitation de signaux dans un réseau de Deep Learning
    f) Bilan des Missions
4) Bilan Personnel du stage

Présentation de la start-up

- Domaine : Biotechnologie
- Fondateurs : Guillaume Cathelain (CTO) et Séverin Benizri(CEO)
- But : Développer une application mobile à base d'intelligence artificielle pour aider à diagnostiquer l'apnée du sommeil
- Apnée du Sommeil : Interruptions de la respiration durant le sommeil

Environnement technique

Amazon Web Service (AWS)

- Stockage de données sur le cloud avec S3 Bucket
- Machine Virtuel Windows avec Amazon Workspaces
- Amazon SageMaker pour le développement de code sur JupyterLab (Notebook)
- Github pour versionner notre code.

Amazon SageMaker

Amazon S3

Noxturnal

- Noxturnal est un logiciel médical qui permet l'analyse et l'édition de rapports d'examen du sommeil.
- Test PSG sur patients -> Signaux Captées et transférées à Noxturnal -> Analyse des signaux et labellisation -> Diagnostic du patient (qualité du sommeil).
- Un fichier Noxturnal est composé d'un fichier Data.ndb de type SQLite et d'un fichier EDF.

X (Monday, May 9, 2022) - Noxturnal

Fichier Modifier Visualiser Analyse Rapports Dispositifs Outils Aide

bichat

Résultats | Breathing segmentation signals

Informations

ID : 6
Nom : X
Adresse :
Ville :
Téléphone :

Sexe : Homme
Date de naissance :
Âge :
Taille : 17200 cm
Poids : 96 kg
IMC : 0

Modifier

Paramètres de sommeil

Normal | Léger | Modéré | Sévère

IDO 10.8 | IAH 30.4

IAH : Index d'Apnées Hypopnées

0% Pourcentage de ronflement

Index de limitations de débit

7h 56m Est. durée totale de sommeil

Aperçu des signaux

Périodes

Apnées Phase RIP — 148.5s

O^2 Désaturation (100 / 70)

Ronflements

Qualité

Monday, May 9, 2022

30.4 IAH | 11.2 IA | 19.2 IH

10.8 IDO | 85 Min | 96 Moyenne

0% Pourcentage de ronflements

100% Qualité du signal

11:00 PM | 12:00 AM | 01:00 AM | 02:00 AM | 03:00 AM | 04:00 AM | 05:00 AM | 06:00 AM | 07:00 AM | 08:00 AM

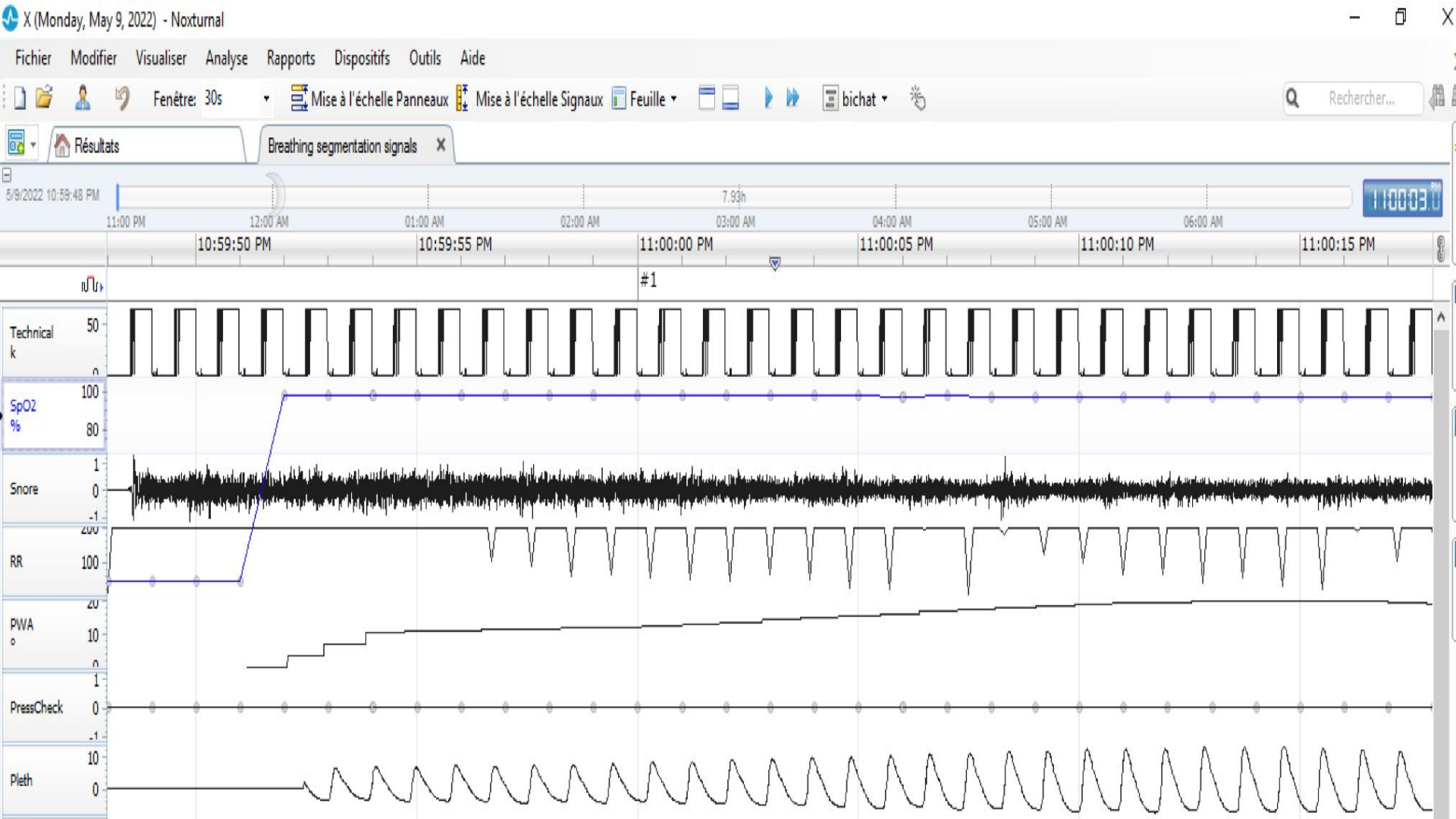
Fermer l'enregistrement | Afficher les signaux | Afficher le rapport | Imprimer le rapport | Nouvel enregistrement

Console d'analyse | Captures d'écran

5/9/2022 10:59:48 PM | 7h 56m 4s | Enregistrement global | Tags

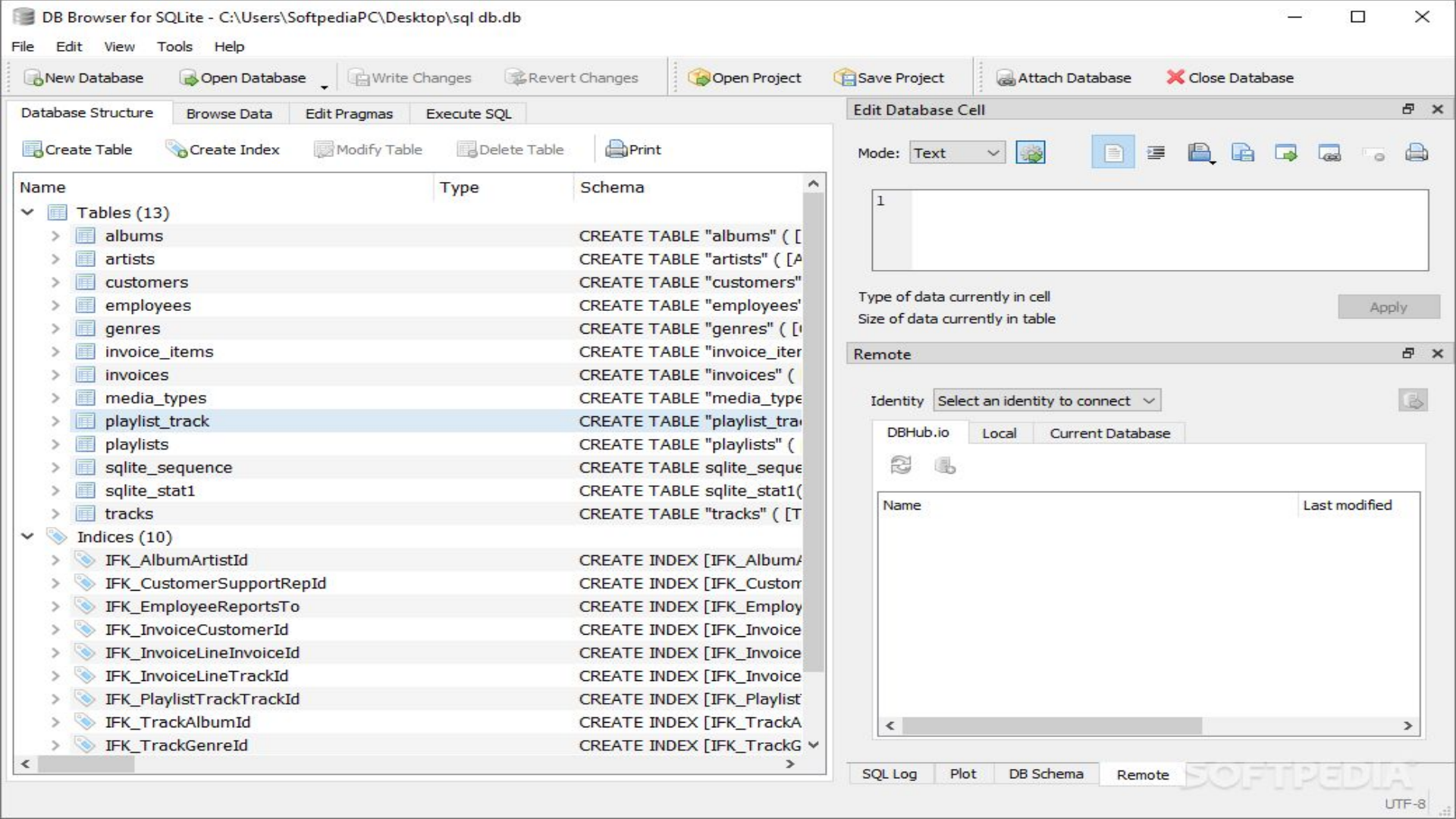
Événements | Périodes | Scorages | Signaux

ENG 12:24 AM 8/29/2022


X (Monday, May 9, 2022) - Noxturnal
Fichier
Modifier
Visualiser
Analyse
Rapports
Dispositifs
Outils
Aide
Fenêtre: 30s
Mise à l'échelle Panneaux
Mise à l'échelle Signaux
Feuille
bichat
Rechercher...
Résultats
Breathing segmentation signals
5/9/2022 10:59:48 PM
7.93h
11:00 PM
12:00 AM
01:00 AM
02:00 AM
03:00 AM
04:00 AM
05:00 AM
06:00 AM
10:59:50 PM
10:59:55 PM
11:00:00 PM
11:00:05 PM
11:00:10 PM
11:00:15 PM
#1
Technical k
SpO2 %
Snore
RR
PWA °
PressCheck
Pleth

DB Browser for SQLite

- DB Browser for SQLite (DB4S) est un outil de haute qualité, visuel et open source pour créer, concevoir et éditer des fichiers de base de données compatibles avec SQLite.
- Permet de lire le fichier Data.ndb généré par Noxturnal.


DB Browser for SQLite - C:\Users\SoftpediaPC\Desktop\sql db.db
File
Edit
View
Tools
Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Close Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Create Table
Create Index
Modify Table
Delete Table
Print
Name
Type
Schema
Tables (13)
albums
artists
customers
employees
genres
invoice_items
invoices
media_types
playlist_track
playlists
sqlite_sequence
sqlite_stat1
tracks
Indices (10)
IFK_AlbumArtistId
IFK_CustomerSupportRepId
IFK_EmployeeReportsTo
IFK_InvoiceCustomerId
IFK_InvoiceLineInvoiceId
IFK_InvoiceLineTrackId
IFK_PlaylistTrackTrackId
IFK_TrackAlbumId
IFK_TrackGenreId
Edit Database Cell
Mode:
Text
Type of data currently in cell
Size of data currently in table
Apply
Remote
Identity
Select an identity to connect
DBHub.io
Local
Current Database
Name
Last modified
SQL Log
Plot
DB Schema
Remote
UTF-8

Edf Browser

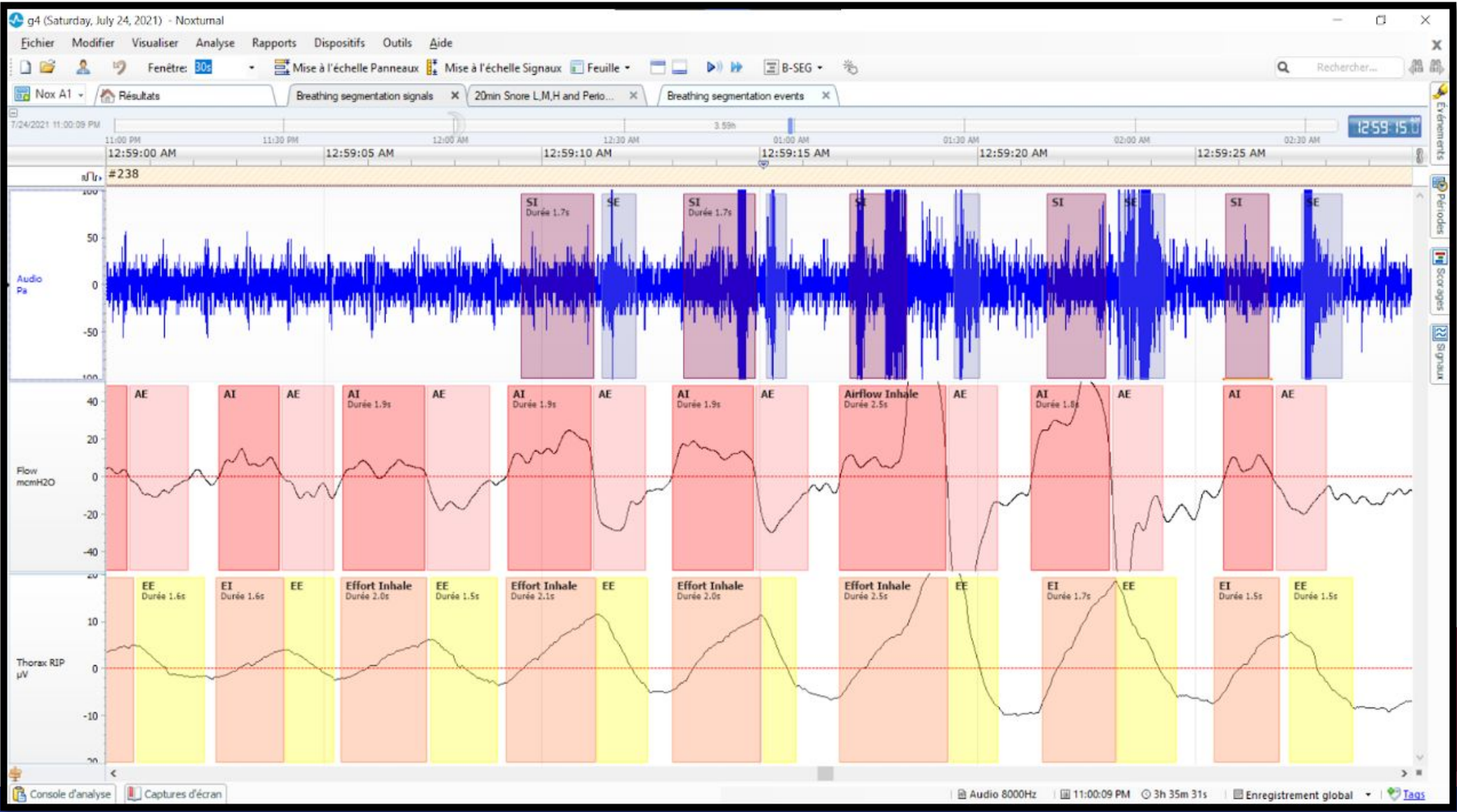
- Un visualiseur, un annotateur et une boîte à outils gratuits, opensource, multiplateforme et universels destinés à des fichiers de stockage de séries temporelles (signaux).
- Permet de lire le fichier Edf généré par Noxturnal.


ST7191J0-PSG_with annotations.edf - EDFbrowser
File Signals Timescale Amplitude Filter Montage Tools Settings Timesync Window Help
Color Density Spectral Array Fpz-Cz
Annotations
Relative Filter: Inv. More
Hz / Time 0h 1h 2h 3h 4h 5h 6h 7h 8h 9h
dBuV
+100 uV
Fpz-Cz
-100 uV
+100 uV
Pz-Oz
-100 uV
+100 uV
horizontal
-100 uV
Sleep stage 4
2:47:30 Duration: 30 sec
2-SEP 1:54:55 (2:47:25)
10 sec
W
R
N1
N2
N3
N4
Stage / Time
Description Sleep stage 4 Onset 02:47:30.000 Duration 30.000 sec Modify Delete Create

Missions réalisées et résultats obtenus

1) Labellisation de signaux

- Permet de constituer une base de données pour pouvoir entraîner un modèle de Deep -Learning.
- Test de PSG -> Récupération et Exportation des signaux sur Noxturnal -> Labellisation des signaux sur Noxturnal (signal audio, signal flux nasal, signal mouvement du thorax).
- La labellisation est constituée d'un ensemble de règles pour chaque signal qui permettent d'annoter ce signal.


g4 (Saturday, July 24, 2021) - Noxturnal
Fichier Modifier Visualiser Analyse Rapports Dispositifs Outils Aide
Fenêtre: 30s
Mise à l'échelle Panneaux
Mise à l'échelle Signaux
Feuille
B-SEG
Nox A1
Résultats
Breathing segmentation signals
20min Snore L,M,H and Perio...
Breathing segmentation events
12:59:00 AM
12:59:05 AM
12:59:10 AM
12:59:15 AM
12:59:20 AM
12:59:25 AM
#238
Audio
Pa
SI
Durée 1.7s
SE
Flow
mcmH2O
AE
AI
Durée 1.9s
Airflow Inhale
Durée 2.5s
Thorax RIP
µV
EE
EI
Durée 1.6s
Effort Inhale
Durée 2.0s
Durée 2.1s
Durée 1.5s
Durée 1.7s
Console d'analyse
Captures d'écran
Audio 8000Hz
11:00:09 PM
3h 35m 31s
Enregistrement global
Tags

g3 (Friday, July 23, 2021) - Noxturnal

Fichier Modifier Visualiser Analyse Rapports Dispositifs Outils Aide

GC-seg

Nox A1 | Résultats | Breathing and snoring segme... | Grille d'événements

Ajouter une colonne | Retirer | Propriétés | Réorganisation des colonnes | Exporter...

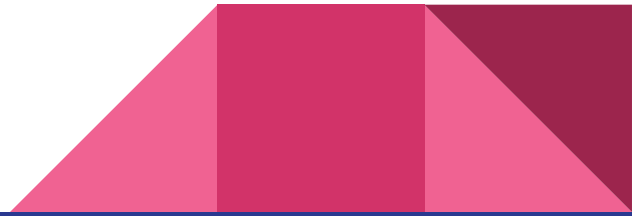
	Événement	Durée s	Heure de début s	Heure de fin s	Heure de début	Heure de fin	Début époque #	Fin époque #
318	Hypopnée	12.36	31,193.80	31,206.17	7:40:02 AM	7:40:14 AM	1040	1040
319	Hypopnée	20.33	31,234.12	31,254.45	7:40:42 AM	7:41:03 AM	1041	1042
320	Hypopnée	17.18	31,376.60	31,393.78	7:43:05 AM	7:43:22 AM	1046	1046
321	Hypopnée	25.53	31,428.22	31,453.75	7:43:57 AM	7:44:22 AM	1047	1048
322	Hypopnée	13.99	31,541.37	31,555.36	7:45:50 AM	7:46:04 AM	1051	1052
323	Hypopnée	17.13	31,566.06	31,583.19	7:46:14 AM	7:46:31 AM	1052	1053
324	Airflow exhale	1.21	31,670.90	31,672.11	7:47:59 AM	7:48:00 AM	1055	1056
325	Exhale sound	0.71	31,670.95	31,671.66	7:47:59 AM	7:48:00 AM	1055	1056
326	Effort exhale	0.99	31,671.10	31,672.10	7:47:59 AM	7:48:00 AM	1055	1056
327	Airflow inhale	1.99	31,673.23	31,675.22	7:48:02 AM	7:48:03 AM	1056	1056
328	Effort inhale	2.14	31,673.28	31,675.41	7:48:02 AM	7:48:04 AM	1056	1056
329	Inhale sound	1.64	31,673.49	31,675.13	7:48:02 AM	7:48:03 AM	1056	1056
330	Airflow exhale	1.75	31,675.23	31,676.97	7:48:04 AM	7:48:05 AM	1056	1056
331	Exhale sound	0.73	31,675.26	31,675.99	7:48:04 AM	7:48:04 AM	1056	1056
332	Effort exhale	1.62	31,675.44	31,677.06	7:48:04 AM	7:48:05 AM	1056	1056
333	Airflow inhale	1.98	31,677.92	31,679.90	7:48:06 AM	7:48:08 AM	1056	1056
334	Effort inhale	2.03	31,678.09	31,680.12	7:48:06 AM	7:48:08 AM	1056	1056
335	Inhale sound	1.60	31,678.19	31,679.79	7:48:06 AM	7:48:08 AM	1056	1056
336	Airflow exhale	1.68	31,679.90	31,681.59	7:48:08 AM	7:48:10 AM	1056	1056
337	Exhale sound	0.84	31,679.98	31,680.82	7:48:08 AM	7:48:09 AM	1056	1056
338	Effort exhale	1.66	31,680.19	31,681.85	7:48:08 AM	7:48:10 AM	1056	1056
339	Airflow inhale	1.86	31,682.13	31,683.99	7:48:10 AM	7:48:12 AM	1056	1056
340	Effort inhale	1.88	31,682.42	31,684.29	7:48:11 AM	7:48:13 AM	1056	1056
341	Inhale sound	1.58	31,682.43	31,684.01	7:48:11 AM	7:48:12 AM	1056	1056
342	Airflow exhale	1.55	31,684.01	31,685.56	7:48:12 AM	7:48:14 AM	1056	1056
343	Exhale sound	0.95	31,684.16	31,685.11	7:48:12 AM	7:48:13 AM	1056	1056
344	Effort exhale	1.66	31,684.29	31,685.95	7:48:13 AM	7:48:14 AM	1056	1056
345	Airflow inhale	1.73	31,686.49	31,688.22	7:48:15 AM	7:48:16 AM	1056	1056
346	Effort inhale	1.55	31,686.92	31,688.48	7:48:15 AM	7:48:17 AM	1056	1056
347	Inhale sound	0.65	31,687.25	31,687.89	7:48:16 AM	7:48:16 AM	1056	1056
348	Exhale sound	0.73	31,688.26	31,688.99	7:48:17 AM	7:48:17 AM	1056	1056
349	Airflow exhale	1.36	31,688.28	31,689.64	7:48:17 AM	7:48:18 AM	1056	1056
350	Effort exhale	1.36	31,688.52	31,689.88	7:48:17 AM	7:48:18 AM	1056	1056
351	Airflow inhale	1.73	31,690.72	31,692.45	7:48:19 AM	7:48:21 AM	1056	1056
352	Effort inhale	1.58	31,690.94	31,692.51	7:48:19 AM	7:48:21 AM	1056	1056
353	Airflow exhale	1.25	31,692.45	31,693.70	7:48:21 AM	7:48:22 AM	1056	1056
354	Exhale sound	0.58	31,692.55	31,693.14	7:48:21 AM	7:48:21 AM	1056	1056
355	Effort exhale	1.06	31,692.73	31,693.78	7:48:21 AM	7:48:22 AM	1056	1056
356	Airflow inhale	1.81	31,694.52	31,696.33	7:48:23 AM	7:48:25 AM	1056	1056

Événements | Périodes | Scorages | Signaux

Console d'analyse | Captures d'écran | 7/23/2021 11:00:08 PM | 9h 59m 51s | Enregistrement global | Tags

2) Algorithmes pour la manipulation de base de données

- Base de données de type SQLite
- Problème : fichier mal enregistré ou scorage effectué dans la mauvaise feuille de scorage ou fichier ndb.
- Éviter de refaire le scorage à la main -> Gain de temps.
- But : Fusionner deux fichiers SQLite :
  - Fusionner deux feuilles de scorages.
  - Importer une feuille de scorage d'un fichier à un autre.
- Algorithme très utile pour une manipulation rapide de ses fichiers.



Présentation de la base de donnée

Tables principales pour la fusion :

- Scoring_key
- Scoring_marker
- Scoring_revision

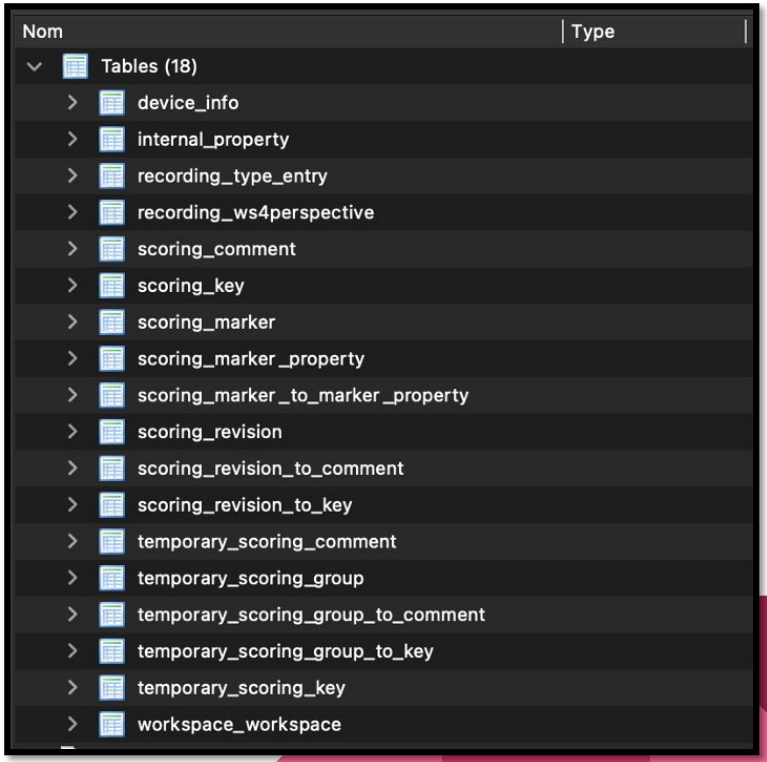


Table : scoring_marker

	id	starts_at	ends_at	notes	type	location	is_deleted	key_id
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
17822	17...	637639911099420000	637639911400270000	NULL	respi-eosd_type1	Resp.Flow-Cannula.Nasal	0	24
17823	17...	637639911527980000	637639911665610000	NULL	apnea-central	Resp.Flow-Cannula.Nasal	0	24
17824	17...	637639911726460000	637639911895480000	NULL	respi-eosd_type1	Resp.Flow-Cannula.Nasal	0	24
17825	17...	637639912762990000	637639912920090000	NULL	apnea-obstructive	Resp.Flow-Cannula.Nasal	0	24
17826	17...	637639913054910000	637639913382800000	NULL	respi-eosd_type1	Resp.Flow-Cannula.Nasal	0	24
17827	17...	637639913460550000	637639913737730000	NULL	respi-eosd_type1	Resp.Flow-Cannula.Nasal	0	24
17828	17...	637639918304480000	637639918520820000	NULL	respi-eosd_type1	Resp.Flow-Cannula.Nasal	0	24
17829	17...	637639924733780000	637639924855460000	NULL	respi-eosd_type1	Resp.Flow-Cannula.Nasal	0	24

Table : scoring_key

	id	date_created	name	owner	type
	Fi...	Filtre	Filtre	Filtre	Filtre
1	1	637640108531631689	Position	Respiratoire Flux Nasal	Automat
2	2	637640108531928744	Activité	Respiratoire Flux Nasal	Automat
3	3	637640108531968747	Apnée/Hypopnée	Respiratoire Flux Nasal	Automat
4	4	637640108587263198	Respiration paradoxale	Respiratoire Flux Nasal	Automat
5	5	637640108587312926	Ronflements	Respiratoire Flux Nasal	Automat

Table : scoring_revision

	id	name	is_deleted	tags	version	owner	date_created
	Fi...	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	RH	0		0		637879696372589637
2	2	B-SEG	0		0		637895265284921157
3	3	B-SEG	0		1		637908217064407640
4	4	B-SEG	0		2		637908246244187820
5	5	B-SEG	0		3		637908252700533057
6	6	BSEG-2	0		0		637919481719483513

Partie Algorithmique

- Python Notebook.
- Bibliothèque Pandas, Numpy, Sqlite3, datetime.
- Pas de documentation, ni de ressources sur internet expliquant les bases de données de noxturnal.
- Versionnage grâce à Github dans une branche spécifique (develop).

Algorithme Principal

```
def MergeDataFiles1(name_file,name_file_to_be_modified,name_scorage,name_scorage_to_be_modified):

    '''
    Usefull to merge Annotations , it take for example the scorage in name_scorage  of file name_file and add it all in the scorage name_scorage_to_be_modified
    of name_file_to_be_modified , usefull for BSEG - BSEG2

    Des doublons peuvent exister après merge

    ATTENTION : toujours bien enregistrer les scorages des data avant de merge !
    '''
    conn = sq.connect(name_file_to_be_modified)
    conn1 = sq.connect(name_file)

    requet = " AND (location='Audio-Respiratory' OR location='Resp.Movement-Inductive.Thorax' OR location='Resp.Flow-Cannula.Nasal') "
    requet = requet+ " AND "
    requet =requet + " ( type='cycles respiratoires-airflowinhale'"
    requet =requet + " OR type='cycles respiratoires-airflowinhale'"
    requet =requet + " OR type='cycles respiratoires-airflowexhale'"
    requet =requet + " OR type='cycles respiratoires-soundinhale'"
    requet =requet + " OR type='cycles respiratoires-soundexhale'"
    requet =requet + " OR type='cycles respiratoires-effortinhale'"
    requet =requet + " OR type='cycles respiratoires-effortexhale'  "
    requet =requet + " OR type='snorebreath' ) "
    requet1= " (SELECT max(id) FROM scoring_revision WHERE name='" + name_scorage + "') )"
    requet1="  SELECT key_id FROM scoring_revision_to_key WHERE revision_id = "+requet1
    requet1= " SELECT * FROM scoring_marker WHERE key_id IN ( "+requet1
    requet1 = requet1 + requet

    df1 = pd.read_sql_query(requet1,conn1)
    df_r=real_annot_id(df1)

    key_id=addNewIdToScoringKey(conn)
    revisionId=addNewScoringRevision(conn,name_scorage_to_be_modified)
    addNewScoringRevisionToKey(revisionId,key_id,conn,name_scorage_to_be_modified)
    addNewDataToScoringMarker(df_r,key_id,conn)

    conn.commit()
    conn.close()
    conn.close
    conn1.close()
    conn1.close

    ## ON SAUVEGARDE
    conn = sq.connect(name_file_to_be_modified)
    cur = conn.cursor()
    requet1=" DELETE FROM temporary_scoring_group WHERE id IN (SELECT id FROM temporary_scoring_group)"
    cur.execute(requet1)
    conn.commit()
    cur.close()
```

Sous-fonctions

```python
def addNewIdToScoringKey(conn):
    '''
        Add new Id to the table ScoringKey located in db file of conn

    '''
    requet = " SELECT * FROM scoring_key "
    df = pd.read_sql_query(requet,conn)
    key_id = (df[['id']].iloc[-1][0])+1
    Date = encodeDateNow()
    row = pd.DataFrame({  'id' : [key_id],'date_created':[Date],'name': [''],'owner':[''],'type':['Manual']   })
    addRowsToData(row,'scoring_key',conn)

    return key_id
```

```python
def addNewScoringRevision(conn,name_scorage):
    requetMaxId = " SELECT max(id) FROM scoring_revision "
    requetMaxVersion = " SELECT max(version) FROM scoring_revision WHERE name = '"+name_scorage+"' "
    df = pd.read_sql_query(requetMaxId,conn)
    maxRevisionId = int(df.loc[0]) + 1
    df = pd.read_sql_query(requetMaxVersion,conn)
    if (list(df.loc[0])[0] == None ) :
        # Scorage n'exite pas déjà , donc on le crée
        MaxVersion = 0
    else :
        MaxVersion = int(df.loc[0]) + 1
    Date = encodeDateNow()
    row = pd.DataFrame({  'id' : [maxRevisionId],'name':[name_scorage],'is_deleted': [0],'tags':[''],'version':[MaxVersion],'owner':[''],'date_created':[Date]   })
    addRowsToData(row , 'scoring_revision' , conn)

    return maxRevisionId
```

```python
def addNewDataToScoringMarker(data,key_id,conn):
    '''
        Add a DataFrame to the table scoring_marker located in db file of conn

    '''
    requetMaxId = " SELECT max(id) FROM scoring_marker "
    df = pd.read_sql_query(requetMaxId,conn)
    maxId = int(df.loc[0]) + 1
    L=[]
    for i in range(data['id'].size):
            L.append(maxId+i)
    Di=pd.DataFrame({  'id' : L})
    data['id']=Di['id'].values

    data['key_id']=data[['key_id']].apply(lambda x: key_id, axis = 1)
    addRowsToData(data , 'scoring_marker' , conn)
```

Sous-fonctions

```python
def real_annot_id(dataF):

    '''
    dataF is a DataFrame

    take a data Frame of scoring_marker an return the a dataFram with annotations that still exist in the file

    '''

    L_id_not_deleted =  list(dataF[dataF['is_deleted']==0]['id'])
    out = L_id_not_deleted.copy()
    for iD in L_id_not_deleted:
        tmp = dataF[dataF['id']==iD]
        if(len(tmp) != 1 ): # id est unique
            return None
        starts_at = int(tmp['starts_at'])
        ends_at = int(tmp['ends_at'])
        typee = list(tmp['type'])[0]
        location = list(tmp['location'])[0]
        is_deleted = 1
        # We check if this sample has been deleted ( ie if the :  id whith is_deleted=1 > id whith is_deleted=0)
        condition = (dataF['starts_at'] == starts_at) &  (dataF['ends_at'] == ends_at) & ( dataF['type']==typee ) & ( dataF['location']==location )
        max_id = max(list(dataF[condition]['id']))
        deleted = int ( dataF[dataF['id']==max_id]['is_deleted']  )
        if( deleted == 1):
            out.remove(iD)

    col=['starts_at', 'type', 'location', 'is_deleted']
    data_out = dataF[dataF['id'].isin(out)]

    return data_out.drop_duplicates(subset=col, keep='last')
```

Sous-fonctions

```python
def coder_date(anne, mois,j,h,mn,s,ms):
    dt = datetime(anne, mois,j,h,mn,s,ms)
    seconds = dt.timestamp()
    S=abs(datetime.fromisocalendar(2, 1, 1).timestamp())+seconds+(365*24*3600)-(24*3600)
    return int(S*1000*10000)


def decoder_date(nb):
    nox_datetime =  nb
    seconds_since_JC = nox_datetime//10000000
    milliseconds_since_JC = (nox_datetime - seconds_since_JC*10000000)//10000
    micro_seconds_since_JC = (nox_datetime - seconds_since_JC*10000000 - milliseconds_since_JC*10000)//10
    timedelta_since_JC = timedelta(seconds=seconds_since_JC, milliseconds=milliseconds_since_JC, microseconds=micro_seconds_since_JC)
    JC_datetime = datetime.fromisocalendar(1, 1, 1)
    converted_datetime = JC_datetime + timedelta_since_JC
    return converted_datetime
```

```python
def coder_date(ms, sec, mn, h, j, mois, annee, precision=None):
    s=0
    for i in range (1,annee):
        if (i % 4 ==0 and i%100 !=0) or (i%400==0):
            s+=366*24*3600
        else:
            s+=365*24*3600


    if (annee % 4 ==0 and annee%100 !=0) or (annee%400==0):
        m =[ 31 , 29 , 31 , 30 , 31 , 30 , 31 , 31 , 30 ,31 , 30 , 31 ]
        M=0
        for i in range(mois-1):
            s+=m[i]*3600*24
    else :
        m =[ 31 , 28 , 31 , 30 , 31 , 30 , 31 , 31 , 30 ,31 , 30 , 31 ]
        for i in range(mois-1):
            s+=m[i]*3600*24

    s+=h*3600+mn*60+sec+ms/1000+(j-1)*3600*24

    return int(s*1000*10000)
```

3) Algorithmes pour migration de données RML

Partie 1 :

➢ But : Importation d'annotations depuis fichier RML pour les mettres dans fichier data.ndb (SQLite)

Partie 2 :

➢ Téléchargement et importation de fichiers de fichiers annoté depuis le bucket.

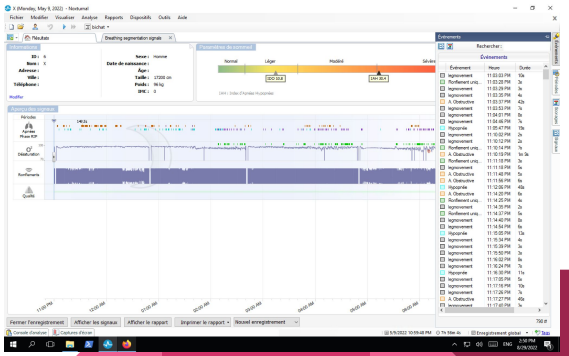
Partie 1

- ➢ Entrée : Fichiers Rml avec annotations.
- ➢ Sortie : Fichier Noxturnal avec annotations
- ➢ En ayant ces annotations on a des indications sur IAH et IDO qu'on avait pas avant -> utilisé pour faire d'autres missions

->

	id	starts_at	ends_at	notes	type	location	is_deleted	key_id
0	2	637877341830000000	637877341935000064	None	LegMovement	Leg 2	0	1
1	3	637877342084999936	637877342120000000	None	snorebreath	None	0	1
2	4	637877342095000064	637877342125000064	None	LegMovement	Leg 2	0	1
3	5	637877342150000000	637877342190000000	None	LegMovement	Leg 2	0	1
4	6	637877342175000064	637877342595000064	None	apnea-obstructive	Flow Patient	0	1
...	...	...	...	...	...	...	...	...
785	787	637877624820000000	637877624920000000	None	hypopnea-obstructive	Flow Patient	0	1
786	788	637877624875000064	637877625195000064	None	oxygensaturation-drop	SpO2.Averaged-Probe	0	1
787	789	637877625164999936	637877625184999936	None	LegMovement	Leg 2	0	1
788	790	637877625195000064	637877625245000064	None	LegMovement	Leg 2	0	1
789	791	637877625455000064	637877625525000064	None	LegMovement	Leg 2	0	1

->



Partie Algorithmique

Bibliothèques :

- lxml et tqdm pour parcourir le fichier rml
- pandas et sqlite3 pour la manipulation de la base de données
- datetime et timedelta pour la conversion de date

Fonction Principal

```
def ajouterAnnotation(fichierRml,fichierData,iD):

    #Initialisation
    df=pd.DataFrame(columns=['Type','Start','Duration','Location'])
    conn = sq.connect(fichierData)
    cur = conn.cursor()

    #Setting information about the patient
    mytree = ET.parse(fichierRml)
    myroot = mytree.getroot()
    cur.execute("UPDATE internal_property SET value = "+str(iD)+" WHERE key = 'ID' ")
    cur.execute("UPDATE internal_property SET value = ' "+myroot[0][3].text+" ' WHERE key = 'Gender' ")
    cur.execute("UPDATE internal_property SET value = ' "+myroot[0][4].text+" ' WHERE key = 'Height' ")
    cur.execute("UPDATE internal_property SET value = ' "+myroot[0][5].text+" ' WHERE key = 'Weight' ")
    conn.commit()

    ## Add Annotations from tags Event of Rml file to df table
    a=0
    doc = minidom.parse(fichierRml)
    elements = doc.getElementsByTagName("Event")
    print(f"Nous avons {elements.length} éléments")
    for i in tqdm(elements):
        df.loc[a]=[i.getAttribute("Type"),i.getAttribute("Start"),i.getAttribute("Duration"),i.getAttribute("Family")]
        a+=1

    ## Removing non usefull tags from df table A MODIFIER POUR IDENTIFIER TOUS LES SUPERFLU
    index=df[df['Type']=='Gain'].index
    df.drop(index,inplace=True)
    index=df[df['Location']=='User'].index
    df.drop(index,inplace=True)
    index2=df[df['Type']=='PttDrop'].index
    df.drop(index2,inplace=True)

    ## Making df in DATA ndb shape
    df.rename(columns = {'Start': 'starts_at','Type':'type','Location':'location'},inplace=True)
    df.reset_index(drop = True, inplace = True)

    ## Taking the startRecording Data from RML file
    mytree = ET.parse(fichierRml)
    myroot = mytree.getroot()
    T=myroot[2][2][0][0].text
    print(T)
    T=decomposerDate(T)

    ## Processing the rows of df
    df['starts_at']=df['starts_at'].astype('float64')
    df['Duration']=df['Duration'].astype('float64')

    df['starts_at']=heure1(df['starts_at'],T)

    df=df.assign(ends_at=0)
    df['ends_at']=df['starts_at']+df['Duration']*1000*10000
    df['ends_at']=df['ends_at'].astype('int64')
    df.drop(columns=["Duration"],inplace=True)
    df['type']=df['type'].apply(conversionType)
    df['location']=df['location'].apply(conversion_Location)
    df=df.assign(notes=None,id=0)
    df['key_id']=1
    df['id']=2+np.arange(df.shape[0])
    df['is_deleted']=0
    df = df.reindex(columns=['id','starts_at','ends_at','notes','type','location','is_deleted','key_id'])

    ## Delete All from table scoring_marker of fichierData
    requet=" DELETE FROM scoring_marker WHERE id IN (SELECT id FROM scoring_marker)"
    cur.execute(requet)
    conn.commit()

    ## Add df to fichierData
    addRowsToData(df,'scoring_marker',conn)

    ## Saving fichierData
    requet=" DELETE FROM temporary_scoring_group WHERE id IN (SELECT id FROM temporary_scoring_group)"
    cur.execute(requet)
    conn.commit()
    cur.close()
    conn.close()
```

Sous fonctions

```python
def coder_date(ms, sec, mn, h, j, mois, annee, precision=None):
    s=0
    for i in range (1,annee):
        if (i % 4 ==0 and i%100 !=0) or (i%400==0):
            s+=366*24*3600
        else:
            s+=365*24*3600


    if (annee % 4 ==0 and annee%100 !=0) or (annee%400==0):
        m =[ 31 , 29 , 31 , 30 , 31 , 30 , 31 , 31 , 30 ,31 , 30 , 31 ]
        M=0
        for i in range(mois-1):
            s+=m[i]*3600*24
    else :
        m =[ 31 , 28 , 31 , 30 , 31 , 30 , 31 , 31 , 30 ,31 , 30 , 31 ]
        for i in range(mois-1):
            s+=m[i]*3600*24

    s+=h*3600+mn*60+sec+ms/1000+(j-1)*3600*24

    return int(s*1000*10000)
```

```python
# ['User', 'Limb', 'SpO2', 'Neuro', 'Respiratory', 'Cardiac', 'Nasal','Snore']
def conversion_Location(l):
    if(l=='Limb'):
        return 'Leg 2'
    if(l=='SpO2'):
        return 'SpO2.Averaged-Probe'
    if(l=='Respiratory' or l=='Nasal'):
        return 'Flow Patient'
    if(l=='Neuro'):
        return 'EEG A1-A2'
    if(l=='Snore'):
        return 'Audio-Respiratory'


def addRowsToData(row , nameOfTable , conn):
    row.to_sql(name=nameOfTable,con=conn,if_exists='append', index=False)
    conn.commit()

def decomposerDate(date):
    a = int(date[0:4])
    m = int(date[5:7])
    j = int(date[8:10])
    h = int(date[11:13])
    mn = int(date[14:16])
    s = int(date[17:19])
    L=[a,m,j,h,mn,s]
    return L

def heure(nb):
    return coder_date(0,T[5]+nb,T[4], T[3], T[2],T[1],T[0])

def heure1(row,T):
    for i in range(row.size):
        row[i]= coder_date(0,T[5]+row[i],T[4], T[3], T[2],T[1],T[0])
    return row
```

Partie 2

- Dans cette partie, il faut utiliser l’algorithme précédent pour l’appliquer à plusieurs fichiers qui se trouvent dans le Bucket.
- Code pour importer ses fichiers depuis le bucket.
- Code pour exporter ses fichiers dans le bucket.
- Bibliothèques : sagemaker.s3 , boto3, os

Connexion au bucket

```
session = sagemaker.Session()
sts_client = session.boto_session.client("sts")
role = session.get_caller_identity_arn()
if "role" in role:
    credentials = session.boto_session.get_credentials()
    credentials = dict(SessionToken=credentials.token, AccessKeyId=credentials.access_key, SecretAccessKey=credentials.secret_key)
else:
    sts_client = session.boto_session.client("sts")
    assumed_role_object = sts_client.assume_role(
        RoleArn="arn:aws:iam::789873108456:role/DataScientist",
        RoleSessionName="LabelStudioUploadSession",
        DurationSeconds=3600*12
    )
    credentials = assumed_role_object['Credentials']
s3 = boto3.resource(
    's3',
    aws_access_key_id=credentials['AccessKeyId'],
    aws_secret_access_key=credentials['SecretAccessKey'],
    aws_session_token=credentials['SessionToken'],
)
```

Importation de fichiers Ndb et RML depuis Bucket

```python
## download path name of all files of RML AND DATA FROM BICHAT
s3 = boto3.resource('s3')
my_bucket = s3.Bucket('tmp-datascience-apneal')
i=0
L=[]
for object_summary in my_bucket.objects.filter(Prefix="bichat/scored-psg"):
    L.append(object_summary.key)
L.pop(0)

## Process to make it well in a list
T=[]
for l in L:
    tmp = l.split('/')
    folder_name1= tmp[2]
    folder_name2=tmp[3]
    flatT= [item for l in T for item in l]
    if( (folder_name1 not in flatT) and (folder_name2 not in flatT) ):
        T.append([folder_name1,folder_name2])
    if('rml' in l):
        T[len(T)-1].append(l)
    if('ndb' in l):
        T[len(T)-1].append(l)

## Second Process to make it well in a list version 2

T_v2 =[]
for t in T:
    if(len(t)==4):
        Id = int(re.findall(r'-?\d+\.?\d*',t[0].split('-')[0])[0])
        #Id = re.findall(r'-?\d+\.?\d*',t[0].split('-')[0])[0]
        t.append(Id)
        T_v2.append(t)
```

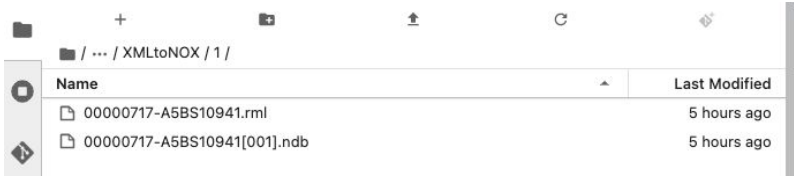
```
['bichat/scored-psg/APNEAL001-LF- 22.04.2022-CH02 Scoré/00000717-A5BS10941/00000717-A5BS10941-AT.xml',
 'bichat/scored-psg/APNEAL001-LF- 22.04.2022-CH02 Scoré/00000717-A5BS10941/00000717-A5BS10941.rml',
 'bichat/scored-psg/APNEAL001-LF- 22.04.2022-CH02 Scoré/00000717-A5BS10941/00000717-A5BS10941[001]-T.edf',
 'bichat/scored-psg/APNEAL001-LF- 22.04.2022-CH02 Scoré/00000717-A5BS10941/00000717-A5BS10941[001].edf',
 'bichat/scored-psg/APNEAL001-LF- 22.04.2022-CH02 Scoré/00000717-A5BS10941/00000717-A5BS10941[001].ndb',
 'bichat/scored-psg/APNEAL003-NS- 25.04.2022-CH07 Scoré/00001953-A5BS10401/00001953-A5BS10401.rml',
 'bichat/scored-psg/APNEAL003-NS- 25.04.2022-CH07 Scoré/00001953-A5BS10401/00001953-A5BS10401[001]-T.edf',
 'bichat/scored-psg/APNEAL003-NS- 25.04.2022-CH07 Scoré/00001953-A5BS10401/00001953-A5BS10401[001].edf',
 'bichat/scored-psg/APNEAL003-NS- 25.04.2022-CH07 Scoré/00001953-A5BS10401/00001953-A5BS10401[001].ndb',
 'bichat/scored-psg/APNEAL004-TA- 25.04.2022-CH02 Scoré/00000719-A5BS10941/00000719-A5BS10941-AT.xml',
 'bichat/scored-psg/APNEAL004-TA- 25.04.2022-CH02 Scoré/00000719-A5BS10941/00000719-A5BS10941.rml',
 'bichat/scored-psg/APNEAL004-TA- 25.04.2022-CH02 Scoré/00000719-A5BS10941/00000719-A5BS10941[001]-T.edf',
 'bichat/scored-psg/APNEAL004-TA- 25.04.2022-CH02 Scoré/00000719-A5BS10941/00000719-A5BS10941[001].edf',
 'bichat/scored-psg/APNEAL004-TA- 25.04.2022-CH02 Scoré/00000719-A5BS10941/00000719-A5BS10941[001].ndb',
```

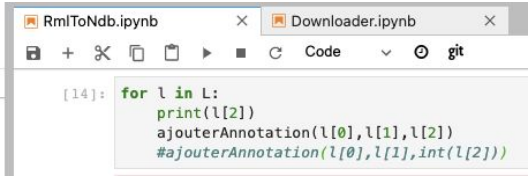


```
[['APNEAL001-LF- 22.04.2022-CH02 Scoré',
  '00000717-A5BS10941',
  'bichat/scored-psg/APNEAL001-LF- 22.04.2022-CH02 Scoré/00000717-A5BS10941/00000717-A5BS10941.rml',
  'bichat/scored-psg/APNEAL001-LF- 22.04.2022-CH02 Scoré/00000717-A5BS10941/00000717-A5BS10941[001].ndb',
  1],
 ['APNEAL003-NS- 25.04.2022-CH07 Scoré',
  '00001953-A5BS10401',
  'bichat/scored-psg/APNEAL003-NS- 25.04.2022-CH07 Scoré/00001953-A5BS10401/00001953-A5BS10401.rml',
  'bichat/scored-psg/APNEAL003-NS- 25.04.2022-CH07 Scoré/00001953-A5BS10401/00001953-A5BS10401[001].ndb',
  3],
 ['APNEAL004-TA- 25.04.2022-CH02 Scoré',
  '00000719-A5BS10941',
  'bichat/scored-psg/APNEAL004-TA- 25.04.2022-CH02 Scoré/00000719-A5BS10941/00000719-A5BS10941.rml',
  'bichat/scored-psg/APNEAL004-TA- 25.04.2022-CH02 Scoré/00000719-A5BS10941/00000719-A5BS10941[001].ndb',
  4],
```

Téléchargement dans le Notebook

```
bucket_name='tmp-datascience-apneal'
for t in T_v2:
    os.mkdir('./'+str(t[4]))
    s3.meta.client.download_file(bucket_name,t[2],str(t[4])+'/'+t[1]+'.rml')
    s3.meta.client.download_file(bucket_name,t[3],str(t[4])+'/'+t[1]+'[001].ndb')# ici renomer en B-001.ndb
    #shutil.rmtree('./'+str(t[4]))
```





Autres Fonctions utiles pour la manipulation du Bucket

```python
def createFolder(bucket_nam,directory_nam):
    s3 = boto3.client('s3')
    bucket_name = bucket_nam
    directory_name = directory_nam #it's name of your folders
    s3.put_object(Bucket=bucket_name, Key=(directory_name+'/'))
```

```python
def moveFileInSameBucket(source,target):
    '''
    source : S3 URI
    target : S3 URI
    '''
    cmd = 'aws s3 mv '+source+' '+target
    proc=subprocess.Popen(cmd, shell=True, stdout = subprocess.PIPE)
```

```python
'''
for l in Lp:
    source = 's3://tmp-datascience-apneal/raf21db/edf_input_nox+apneal/'+l+'_nox+apneal.edf'
    target = 's3://tmp-datascience-apneal/raf21db/edf_input_nox+apneal/'+l+'/'+l+'_nox+apneal.edf'
    moveFileInSameBucket(source,target)
    #print(source)
    #print(target)



'''
```

```
"\nfor l in Lp:\n    source = 's3://tmp-datascience-apneal/raf21db/edf_input_nox+apneal/'+l+'_nox+apneal.edf'\n    target = 's3://tmp-datascience-apneal/raf21db/edf_input_nox+apneal/'+l+'
```

```python
s3_resource = boto3.resource('s3')
# Copy object A as object B
s3_resource.Object('tmp-datascience-apneal', 'tmp-datascience-apneal/raf21db/edf_input_nox+apneal/g1_nox+apneal/g1_nox+apneal.edf').copy_from(CopySource='tmp-datascience-apneal/raf21db/e
```

```python
def moveFileInSameBucket(source,target):
    '''
    source : S3 URI
    target : S3 URI

    '''
    cmd = 'aws s3 mv '+source+' '+target
    proc=subprocess.Popen(cmd, shell=True, stdout = subprocess.PIPE)

#cmd='aws s3 mv s3://tmp-datascience-apneal/raf21db/edf_input_nox+apneal/g1_nox+apneal.edf s3://tmp-datascience-apneal/raf21db/edf_input_nox+apneal/g1_nox+apneal/g1_nox+apneal.edf '
```

```python
def downloadDirectoryFroms3(bucketName, remoteDirectoryName):
    s3_resource = boto3.resource(
    's3',
    aws_access_key_id=credentials['AccessKeyId'],
    aws_secret_access_key=credentials['SecretAccessKey'],
    aws_session_token=credentials['SessionToken'],
)
    bucket = s3_resource.Bucket(bucketName)
    for obj in bucket.objects.filter(Prefix = remoteDirectoryName):
        if not os.path.exists(os.path.dirname(obj.key)):
            os.makedirs(os.path.dirname(obj.key))
        bucket.download_file(obj.key, obj.key) # save to same path
```

4) Algorithmes pour la manipulation de fichiers Edf

- Un fichier de type Edf regroupe les signaux captées par le test PSG ainsi que des informations personnelles sur le patient.
- Lecture sur EdfBrowser ou Noxturnal.
- But de la Mission : rééchantillonner et augmenter le volume sonore du signal Audio pour le rendre audible et compatible avec le logiciel Noxturnal. (nécessaire pour la labellisation et donc pour l'entraînement du réseau de neurones). Et aussi, fusionner deux fichiers Edf.
- Bibliothèque très importante : “pyedflid” sur python, pour la manipulation de ses fichiers.

Fonction pour rééchantillonner

```python
def resample_singal_audio_edf(name_file_source,new_name_file,reduce_volume=20):
    '''
    pour les fichers de psg -> resampling à 8000 hz
    '''

    fichier_r = name_file_source
    freal = pyedflib.EdfReader(fichier_r)
    if( list(g.iloc[[18]].label)[0] != 'Mic'):
        print('error in acces of singal Mic Number ')
        return False
    signal_audio1=freal.readSignal(18)
    resampled_signal=scipy.signal.decimate(signal_audio1, 6)
    digital_min = -2 ** (16 - 1)
    digital_max = 2 ** (16 - 1) - 1
    nb = 368078/reduce_volume
    resampled_signal1 = (resampled_signal-np.mean(resampled_signal))
    resampled_signal2 = resampled_signal1*((nb)/max(abs(resampled_signal1)))
    physical_min = - 1.2 * max(np.max(np.abs(resampled_signal2)), 0.000001) # -1.2 pour pas atteindre le max ou min
    physical_max = physical_min * digital_max / digital_min
    signals = resampled_signal2
    freal.close()

    T=pyedflib.highlevel.read_edf(fichier_r)
    headers={'label': 'Audio',
      'dimension': 'Pa',
      'sample_rate': 8000.0,
      'sample_frequency': 8000.0,
      'physical_max': physical_max,
      'physical_min': physical_min,
      'digital_max': digital_max,
      'digital_min': digital_min,
      'prefilter': T[1][18]['prefilter'],
      'transducer': T[1][18]['transducer']}


    #freal.close()
    T=pyedflib.highlevel.read_edf(fichier_r)
    T[0][18]=signals#18
    T[1][18] = headers#18

    '''
    for i in range(len(T[1])):
        if( T[1][i]['physical_min'] > min(T[0][i]) ):
            T[1][i]['physical_min'] = T[1][i]['physical_min'] - 1
        if( T[1][i]['physical_max'] < max(T[0][i]) ):
            T[1][i]['physical_max'] = T[1][i]['physical_max'] + 1
    '''

    # On corrige physical_min et physical_max
    digital_min = -2 ** (16 - 1)
    digital_max = 2 ** (16 - 1) - 1
    for i in range(1,len(T[0])):
        signal = T[0][i]
        physical_min = - 1.2 * max(np.max(np.abs(signal)), 0.000001)
        physical_max = physical_min * digital_max / digital_min
        dicT = T[1][i]
        dicT['physical_max']=round(physical_max,1)
        dicT['physical_min']=round(physical_min,1)
        dicT['digital_max']=digital_max
        dicT['digital_min']=digital_min

    pyedflib.highlevel.write_edf(new_name_file,T[0],T[1],header=T[2], file_type=-1)
    return T[2]
```

```
[[array([9.81185626e-04, 9.81185626e-04, 9.81185626e-04, ...,
         7.62601371e+01, 5.41033735e+01, 4.34089811e+01]),
 array([ 9.81185626e-04,  9.81185626e-04,  9.81185626e-04, ...,
         7.94440446e+00, -1.63328978e+01, -2.00123190e+01]),
 array([9.81185626e-04, 9.81185626e-04, 9.81185626e-04, ...,
        3.10296202e+01, 1.25293530e+01, 8.86139416e+00])],
 [{'label': 'EOG LOC-A2',
   'dimension': 'uV',
   'sample_rate': 200.0,
   'sample_frequency': 200.0,
   'physical_max': 375.5885,
   'physical_min': -375.598,
   'digital_max': 32767,
   'digital_min': -32768,
   'prefilter': '',
   'transducer': ''},
  {'label': 'EOG ROC-A2',
   'dimension': 'uV',
   'sample_rate': 200.0,
   'sample_frequency': 200.0,
   'physical_max': 375.5885,
   'physical_min': -375.598,
   'digital_max': 32767,
   'digital_min': -32768,
   'prefilter': '',
   'transducer': ''},
  {'label': 'EEG C3-A2',
   'dimension': 'uV',
   'sample_rate': 200.0,
   'sample_frequency': 200.0,
   'physical_max': 375.5885,
   'physical_min': -375.598,
   'digital_max': 32767,
   'digital_min': -32768,
   'prefilter': '',
   'transducer': ''}],
 {'technician': '',
  'recording_additional': '',
  'patientname': 'X',
  'patient_additional': '',
  'patientcode': 'B-001',
  'equipment': '',
  'admincode': '',
  'gender': '',
  'startdate': datetime.datetime(2022, 4, 22, 22, 44, 54),
  'birthdate': '',
  'annotations': []}]
```

Fonction pour fusionner

```python
def merge(fichier_nox,fichier_apneal,fichier_sortie,patient_name='X'):
    T_nox = pyedflib.highlevel.read_edf(fichier_nox)
    T_apneal = pyedflib.highlevel.read_edf(fichier_apneal)
    if(  ( T_apneal[2]['startdate']==T_nox[2]['startdate'] ) == False  ):
        print('le startdate du fichier_nox est différent du fichier_apneal ')
        return None

    # Avant de merge on va corriger le physical/digital min et max du fichier nox
    for i in range(len(T_nox[1])):
        if( T_nox[1][i]['physical_min'] > min(T_nox[0][i]) ):
            T_nox[1][i]['physical_min'] = T_nox[1][i]['physical_min'] * 1.2
        if( T_nox[1][i]['physical_max'] < max(T_nox[0][i]) ):
            T_nox[1][i]['physical_max'] = T_nox[1][i]['physical_max'] * 1.2

    # Avant de merge on va corriger le physical/digital min et max du fichier apneal
    digital_min = -2 ** (16 - 1)
    digital_max = 2 ** (16 - 1) - 1
    for i in range(len(T_apneal[0])):
        signal = T_apneal[0][i]
        physical_min = - 1.2 * max(np.max(np.abs(signal)), 0.000001)
        physical_max = physical_min * digital_max / digital_min
        dicT = T_apneal[1][i]
        dicT['physical_max']=round(physical_max,1)
        dicT['physical_min']=round(physical_min,1)
        dicT['digital_max']=digital_max
        dicT['digital_min']=digital_min

    # Fin

    T_apneal[1][0]['label']='Audio'
    T_merge = [[],[],[]]
    T_merge[0] = T_nox[0] + T_apneal[0]
    T_merge[1] = T_nox[1] + T_apneal[1]
    T_merge[2] = T_nox[2]
    T_merge[2]['patientname']=patient_name


    pyedflib.highlevel.write_edf(fichier_sortie, signals = T_merge[0] , signal_headers = T_merge[1] , header=T_merge[2], file_type=-1)
```

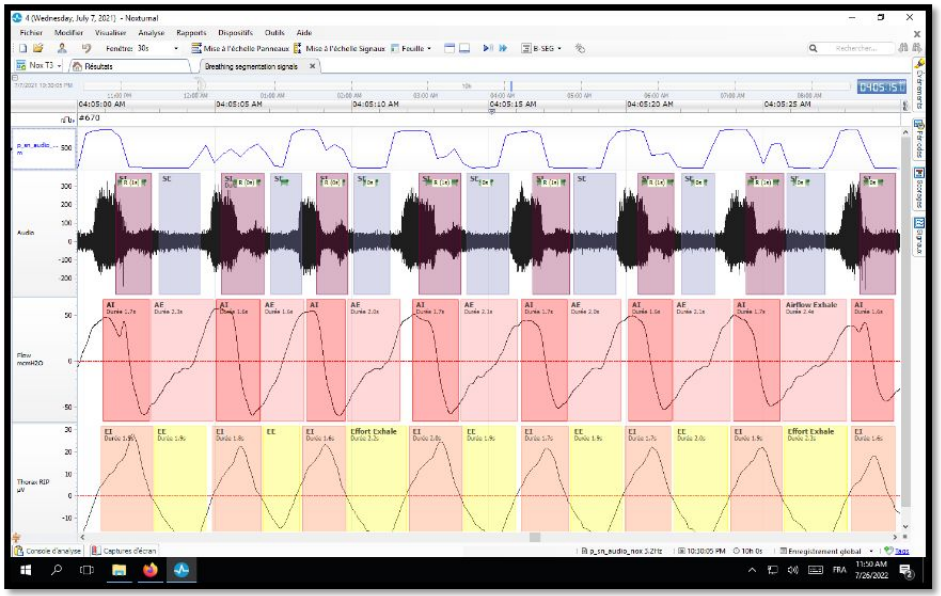
Importation de probabilités en signaux sur Noxturnal

- Tableau de probabilité (probabilité qu'un ronflement se situe sur une portion du signal audio) -> Transformation en fichier Edf -> Fusion avec les autres signaux grâce à "Merge" -> Visualisation sur Noxturnal

```
def df_to_edf(file_name, fichier_sortie, stratDate):
    df= pd.read_hdf(file_name, 'df')
    digital_min = -2 ** (16 - 1)
    digital_max = 2 ** (16 - 1) - 1
    T=[[],[],[]]
    for c in list(df.columns):
        tmp = np.asarray(list(df[c]))
        T[0].append(tmp)
        physical_min = - 1.2 * max(np.max(np.abs(list(df[c]))), 0.000001)
        physical_max = physical_min * digital_max / digital_min
        header={'label': c,
                'dimension': '',
                'sample_rate': 3.2,
                'sample_frequency': 3.2,
                'physical_max': round(physical_max,1),
                'physical_min': round(physical_min,1),
                'digital_max': digital_max,
                'digital_min': digital_min,
                'prefilter': '',
                'transducer': ''}
        T[1].append(header)
    T[2] = pyedflib.highlevel.make_header(startdate=stratDate)

    return pyedflib.highlevel.write_edf(fichier_sortie, signals = T[0] , signal_headers = T[1] , header=T[2], file_type=-1)
```

	p_sn_audio_nox	p_b_audio_nox	p_sn_audio_apneal	p_b_audio_apneal
0	0.055688	0.214077	0.080478	0.276491
1	0.095979	0.140081	0.245545	0.257857
2	0.427723	0.665469	0.132731	0.248296
3	0.068899	0.193732	0.020767	0.147418
4	0.015564	0.156872	0.006070	0.083248
...	...	...	...	...
115195	0.002822	0.066048	0.002608	0.062188
115196	0.004713	0.050301	0.004512	0.048870
115197	0.003263	0.061367	0.003248	0.060146
115198	0.012163	0.063966	0.011540	0.062785
115199	0.062005	0.110832	0.058848	0.108724



5) Entraînement et exploitation de signaux dans un réseau de Deep Learning

- ➢ But de la Mission : Entraîner sur réseau de deep learning nommée « U-sleep » sur des signaux de types EEG (signaux qui mesurent l'activité électrique cérébrale) puis d'analyser et de comparer la segmentation de ces événements fournis par l'Algorithme.
- ➢ Fichier Edf avec signaux -> Process sur U-sleep -> Import d'annotations sur Noxturnal

2 Upload a polysomnography file to stage

Select a polysomnography (PSG) file from your computer to upload for sleep staging.

Once uploaded you will be presented with basic information about the file. You should ensure the validity of this information before proceeding.

CAUTION

Your file will be uploaded to our compute servers.
The file must be fully anonymized or public domain.
DO NOT UPLOAD SENSITIVE MEDICAL FILES.

Upload | Choose file | Browse

The file must be one of type(s): ***edf, h5, hdf5*** *- Need Help?*
The file must be at most ***1250.0 MiB*** *large.*

If you are just testing and don't have a file at hand:

Test With Public File

From the Sleep-EDF Database
https://doi.org/10.13026/C2X676

Signaux pris en argument par le réseau

EEG : L'électroencéphalographie est une méthode d'exploration cérébrale qui mesure l'activité électrique du cerveau par des électrodes placées sur le cuir chevelu souvent représentées sous la forme d'un tracé appelé électro encéphalogramme.

EOG : Une électro oculographie (EOG) est un examen médical dont le but est d'enregistrer le potentiel de repos de l'œil

Your PSG File

Filename	g1_usleep.edf
Sample rate (max)	200 Hz
Channels	F4-M1 C3-M2 F3-M2 C4-M1 O1-M2 O2-M1 E1-M2 E2-M1
Number of channels	8
Length	43200.0 seconds
Date recorded (UTC+2)	2021-07-22 01:30
Date uploaded (UTC+2)	2022-07-11 09:31

Delete file from server

Back Forward

3 Configure the prediction task

Finally, before we can start the sleep staging process on file **g1_usleep.edf** using model **U-Sleep v1.0**, we need you to specify a few additional settings.

Channels: You need to select which channels to provide to the model. Please use the dropdown menus below to select one or multiple channels from each group as requested. Please note:

- If you select multiple channels within a group (e.g. multiple EEG channels), then the model will consider all combinations of input channel configurations. Each combination will be used for prediction one at a time. A majority voting is computed across all predictions to produce the final sleep staging scores. Using larger numbers of combinations typically improve performance at the cost of increased computation time.
- You may currently select up to a maximum of **24** combinations of channels.
- We have attempted to pre-select appropriate channels for model **U-Sleep v1.0**. Please manually check if the selected channels match the model's required inputs.

Please select EEG channels: 6 items selected

Please select EOG channels: E1-M2, E2-M1

Channel combinations currently selected: 12

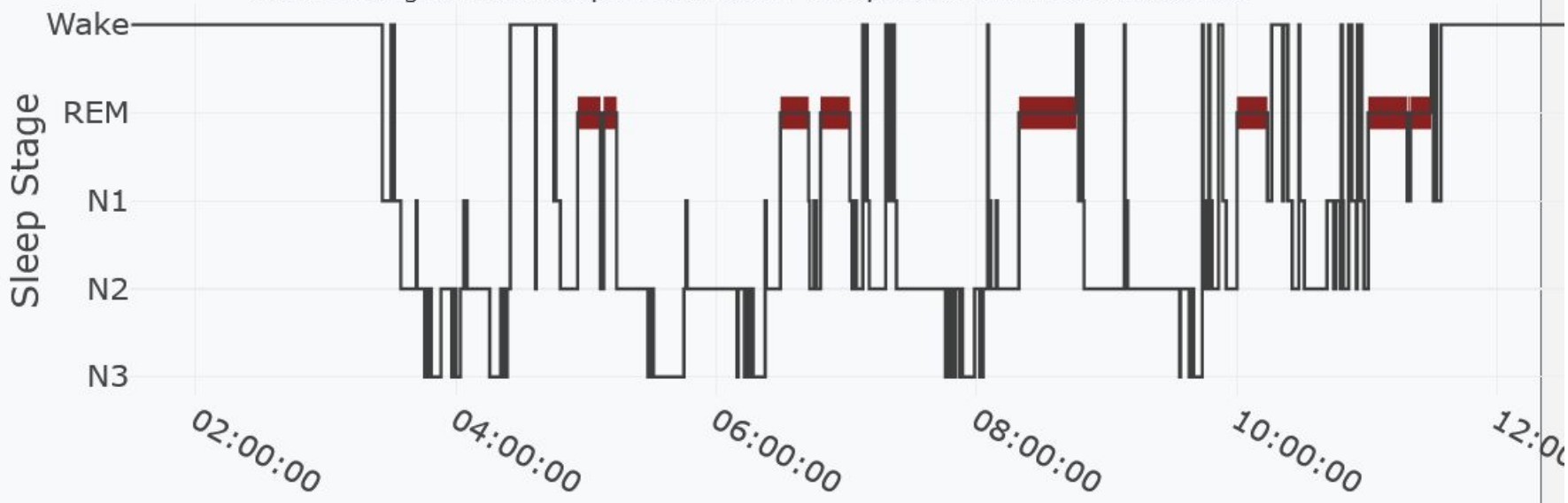
Prediction frequency: Below you may select the number of predictions per second that the model should output. The frequency is at least 1/30 Hz (i.e., one stage scored per 30 seconds; the typical value), but may be set to a higher frequency.

1/30 Hz — 1 Hz

≈0.03 Hz (1 prediction(s) per 30 seconds)

Hypnogram

Click and drag to zoom. Sleep statistics below will update to reflect the current view



Download Sleep Stages

Sleep Statistics

Currently selected range

01:30:09 to 13:29:39

Total awake	274.5 min	38.2 %
Total asleep	445.0 min	61.8 %
N1	48.0 min	6.7 %
N2	232.5 min	32.3 %
N3	53.0 min	7.4 %
REM	111.5 min	15.5 %

Affichage sur Noxturnal puis Analyse

Importation de prédiction sur Noxturnal

- Importation des annotations du fichier Tsv generé par U-sleep sur Data.ndb (procédure habituelle

	id	starts_at	ends_at	notes	type	location	is_deleted	key_id
0	14185	637625070091229952	637625139691229952	None	sleep-wake	NaN	NaN	29
1	14186	637625139691229952	637625142091229952	None	sleep-n1	NaN	NaN	29
2	14187	637625142091229952	637625142991229952	None	sleep-wake	NaN	NaN	29
3	14188	637625142991229952	637625144791229952	None	sleep-n1	NaN	NaN	29
4	14189	637625144791229952	637625148991229952	None	sleep-n2	NaN	NaN	29
...	...	...	...	...	...	...	...	...
153	14338	637625429791229952	637625430391229952	None	sleep-wake	NaN	NaN	29
154	14339	637625430391229952	637625430691229952	None	sleep-n1	NaN	NaN	29
155	14340	637625430691229952	637625431291229952	None	sleep-wake	NaN	NaN	29
156	14341	637625431291229952	637625432491229952	None	sleep-n1	NaN	NaN	29
157	14342	637625432491229952	637625502001229952	None	sleep-wake	NaN	NaN	29

Bilan des Missions

- Partie sur labellisation des données et la familiarisation avec le logiciel Noxturnal.
- Partie sur la manipulation et le développement de fonctionnalités sur des bases de données et des fichiers Edf.
- Partie sur l'entraînement de données sur des algorithmes de deep-learning.
- Résultats très satisfaisants -> Gain de temps pour toute l'équipe -> Automatisation de tâches.
- Développement de mes compétences sur Aws et logiciel médical, très utile dans le monde du travail.
- Plus de tâches en relation avec l'analyse de données souhaité pour élargir mes connaissances.

Bilan Personnel du stage