# Scripts:

**Functions.py**

Contains all functions

**Pre\_processing.py**

Removes patient-identifying info from file

Selects leads and relevant signals/info from files

* Input: to be processed files (file\_name, file\_path), new file path (new\_file\_path)
* Output: pre-processed files (.edf)

**CreateFeatureDataSheet.py**

Calculate features and create feature data sheet per recording

Feature datasheet contains:

- Sleep stage labels

- Artifact labels

- Features for all EOG, EMG and EEG leads

* Input: to be processed files (file\_name, file\_path), new file path (xlsx\_file\_path), epoch size in seconds (epoch\_size), nr signal ()
* Output: Feature data sheet (.xlsx)

**CombineFeatureDataSheet.py**

**CompareFeaturesBetweenGroups.py**

**PowerSpectralDensityPlots.py**

**FeatureEvaluation.py**

# Functions:

**edf\_preprocessing(file\_name, file\_path, new\_file\_path)**

""" Preprocess an EDF file by:  
 - Anonymization (i.e. removing all header information that is patient specific)  
 - Removal of unnecessary signals  
 - Addition of EEG leads  
 (- Bandpass filtering of EEG signals) --> gebeurt in volgende stap  
  
 Parameters  
 ----------  
 file\_name: str  
 Filename of the to be preprocessed EDF file  
 file\_path: str  
 File path of the to be preprocessed EDF file  
 new\_file\_path: str  
 File path of the preprocessed EDF file  
  
 Returns  
 -------  
 New anonymous and preprocessed EDF file is created in [new\_file\_path].  
 The new EDF file has the same extension as the input file and is named: file\_name + p  
 (for example: PSG001 --> PSG001p).  
 If succesful, the following statement is printed: 'Preprocessing of [new\_file\_name] completed')  
  
 """

**stage\_labelling(annotations, signal\_length, fs, epoch\_size)**

"""

Adds sleep stage labels from visually scored hypnogram to epochs

:param annotations:

list - annotations from EDF file header (= header["annotations"]) with

[time, duration, annotations]

:param signal\_length:

int - length of signal in EDF file (= len(signals[0]))

:param fs:

int - sample frequency (= signal.headers[0].get("sample\_rate"))

:param epoch\_size:

int - epoch\_size in samples (=epoch\_size\_s\*fs)

:return: sleep\_stage\_labels

list - sleep stage labels per epoch of visually scored hypnogram

"""

**artifact\_labelling(signal\_headers, signals, fs, epoch\_size)**

"""

Detects artifacts in the signals and adds labels to the epochs in which an artifact

is detected as 'movement artifact', 'impedance artifact' or 'general artifact'.

:param signal\_headers:

list - containing dictionaries with signal header (information)

(output from pyedflib.highlevel.read\_edf)

:param signals:

ndarray - containing raw signals

(output from pyedflib.highlevel.read\_edf)

:param fs:

int - sample frequency (= signal.headers[0].get("sample\_rate"))

:param epoch\_size:

int - epoch\_size in samples (=epoch\_size\_s\*fs)

:return: impArtifact\_labels

ndarray - for each epoch/index 1 (impedance artifact) or 0 (no impedance artifact)

:return: movArtifact\_labels

ndarray - for each epoch/index 1 (movement artifact) or 0 (no movement artifact)

:return: artifact\_labels

ndarray - for each epoch/index 1 (impedance and/or movement artifact) or 0

(no impedance and/or movement artifact)

"""

**windowing(signal, epoch\_size)**

"""

Cuts signal [y] into epochs/windows with size [epoch\_size]

:param signal: array - one raw signal

:param epoch\_size: int - size of epoch in samples (= seconds \* sample frequency)

:return: epochs: list with epochs of the signal as array

"""

**bandpower(y, fs, fmin, fmax)**

"""

Calculation of the signal bandpower

:param y: signal

:param fs: sample frequency

:param fmin: lower cutoff frequency of bandpower range

:param fmax: upper cutoff frequency of bandpower range

:return: bandpower

"""

**spectralFeatures(y, fs):**

"""

Calculation of spectral edge frequency, median frequency, mean frequency, spectral kurtosis, spectral skewness and

spectral entropy

:param y: signal

:param fs: sample frequency

:return: spectral\_edge, median\_freq, mean\_freq, spectral\_kurtosis, spectral\_skewness, spectral\_entropy

"""

**hjorth\_features(y)**

"""

Calculation of the Hjorth features (activity, mobility and complexity)

:param y: signal

:return: activity, mobility, complexity

"""

**FeatureCalculation(signals, signal\_headers, fs, epoch\_size, dfFeatureData)**

"""

Calculation of features for all EEG leads, EMG and EOG (after bandpass filtering)

:param signal\_headers:

list - containing dictionaries with signal header (information)

(output from pyedflib.highlevel.read\_edf)

:param signals:

ndarray - containing raw signals

(output from pyedflib.highlevel.read\_edf)

:param fs:

int - sample frequency (= signal.headers[0].get("sample\_rate"))

:param epoch\_size:

int - epoch\_size in samples (=epoch\_size\_s\*fs)

:param dfFeatureData:

dataframe - containing sleep stage labels + artifact labels

:return: dfFeatureData

dataframe - containing sleep stage labels, artifact labels and features

"""

**CombineFeatureDataSheets(age\_category, patientData\_file\_name, patientData\_file\_path, featureData\_file\_path, combinedFeatureData\_file\_path)**

**ExtractEpochsPerSleepStage(age\_category, nr\_signal, signal\_label, remove\_artifacts, patientData, edfFiles\_file\_path)**