EEG Features Extraction:

1. preprocess\_EEG():

* Arguments: raw, features
* Description: This function can be used to preprocess EEG data and extract features for use in machine learning or other data analysis tasks.
* Steps:

1. The function preprocesses EEG data by applying various filtering and spectral analysis techniques and extracts features from the data.
2. The function applies a bandpass filter to the raw data using the (firwin) function from the signal module of the SciPy library.
3. The filtered data is then passed through separate bandpass filters for different frequency bands (theta, alpha, and beta) using the same (firwin) function with different frequency ranges.
4. The (filtfilt) function from the signal module is used to apply zero-phase filtering to the filtered data.
5. The (signal.welch) function is used to compute the power spectral density (PSD)estimates of the filtered data for each frequency band.
6. The resulting PSD estimates are stored in variables (psdtheta), (psdalpha), and (psdbeta).
7. The function calculates the mean of the PSD estimates for each frequency band and appends them to the feature list.
8. The resulting feature list is returned as the output of the function.
9. feat\_extract\_EEG():

* Arguments: raw
* Description: This function can be used to extract features from EEG data for each participant and video in the DREAMER dataset, which can then be used for machine learning or other data analysis tasks.
* Steps:

1. The function creates a 3D numpy array called (EEG\_tmp) with dimensions (23, 18, 42) to store the extracted features.
2. The function loops over each participant, video, and EEG channel and extracts the baseline and stimulus data for that channel using indexing.
3. The preprocess\_EEG() function is called on both the baseline and stimulus data to extract the power spectral density estimates for theta, alpha, and beta frequency bands.
4. The ratio of the PSD estimates for stimulus and baseline data is then computed and stored in the (EEG\_tmp) array at the appropriate location.
5. After all features have been extracted and stored in (EEG\_tmp), the function creates a pandas Data Frame called (data\_EEG) to store the features in a tabular format.
6. The StandardScaler() function from the sklearn module is used to standardize the extracted features, and the resulting standardized features are stored back in the (data\_EEG) Data Frame.
7. The function drops the original unstandardized features from (data\_EEG) and returns the resulting Data Frame as the output of the function.
8. participant\_affective():

* Arguments: raw
* Description: This function can be used to extract participant demographic information and affective labels for each video in the DREAMER dataset, which can then be used for machine learning or other data analysis tasks.
* Steps:

1. The function creates a 3D numpy array called a with dimensions (23, 18, 9) to store the extracted information.
2. The function loops over each participant and video and extracts the relevant information from raw using indexing.
3. The extracted information is stored in the appropriate location in the array.
4. After all information has been extracted and stored in a, the function creates a pandas Data Frame called b to store the information in a tabular format.
5. The reshape function is used to convert the 3D array into a 2D numpy array that can be used to create the b Data Frame.
6. The function renames the columns of the b Data Frame to match the types of information that were extracted.
7. The resulting Data Frame is returned as the output of the function.

ECG Features Extraction:

1. statistics\_caculation():

* Arguments: waves
* Description: This function can be used to compute basic statistics for a given set of numerical values, which can be useful for data analysis or machine learning tasks.
* Steps:

1. The function computes basic statistics (mean, standard deviation, minimum, maximum, and range) for the input values using functions from the NumPy library.
2. The nan in (np.nanmean), (np.nanstd), (np.nanmin), and (np.nanmax) allows the functions to handle missing values by treating them as NaN.
3. The function returns the computed statistics as a tuple in the following order: mean, standard deviation, minimum, maximum, range.
4. calculate\_static\_features():

* Arguments: waves1, waves2, waves3, and waves4.
* Description: This function can be used to compute basic statistics for multiple sets of numerical values and to compute additional statistics for different parts of the brain. This can be useful for analyzing ECG data or for machine learning tasks that involve ECG features.
* Steps:

1. The function calls the calculations function to compute additional statistics for the left and right sides of the brain. The calculations function takes 10 arguments, which correspond to the mean, standard deviation, minimum, maximum, and range values for the baseline and stimulus data of the left and right sides of the brain.
2. The calculations function computes the mean and standard deviation of the baseline and stimulus data for each side of the brain, as well as the difference between the maximum and minimum values (range).
3. The calculate\_static\_features() function then computes the mean, standard deviation, minimum, maximum, and range for both sides of the brain by averaging the corresponding values computed by the calculations function.
4. Finally, the function computes the overall mean, standard deviation, minimum, maximum, and range by averaging the left and right-side values.
5. The function returns the computed statistics as a tuple in the following order: mean, standard deviation, minimum, maximum, and range.
6. get\_beaks():

* Arguments: raw
* Description: Function to extract certain features (peaks and baseline/stimulus data) from EEG signals in the raw data.
* Steps:

1. The function returns four sets of peaks and four sets of baseline/stimulus data, which are assigned to the variables (peaks\_bl), (peaks\_sl), (peaks\_br), (peaks\_sr), (basl\_left), (stim\_left), (basl\_right), and (stim\_right).
2. time\_domain():

* Arguments: raw.
* Description: Function to extract time domain features (P, R, and T wave) from ECG signals in the DREAMER dataset.
* Steps:

1. The function iterates over all participants and videos in the raw data and extracts the baseline and stimuli data for left and right ECG signals.
2. The function uses functions from the NeuroKit library to extract peaks and delineate waves from the ECG signals for each condition and side of the brain.
3. The function calls the calculate\_static\_features() function with the extracted peaks to compute basic statistics (mean, standard deviation, minimum, maximum, and range) for each wave type (P, R, and T) for each side of the brain.
4. The function stores the computed statistics in separate Data Frames (p\_wave\_df, r\_wave\_df, and t\_wave\_df) for each wave type.
5. The function concatenates the three Data Frames horizontally using the pd.concat() function and returns the resulting Data Frame as time\_ECG.
6. hrv\_features():

* Arguments: raw
* Description: Function to extract heart rate variability (HRV) features from ECG signals in the DREAMER dataset.
* Steps:

1. The function iterates over all participants and videos in the raw data and extracts the baseline and stimuli data for left and right ECG signals.
2. The function uses functions from the (NeuroKit) library to extract peaks and compute HRV features for each condition and side of the brain.
3. The function computes three types of HRV features for each side of the brain: time domain features, frequency domain features using the "welch" method, and non-linear features.
4. The function computes the ratio of HRV features between baseline and stimuli conditions for each side of the brain.
5. The function computes the average of HRV features between left and right sides of the brain.
6. The function stores the computed features in separate DataFrames (time\_ECG, frequency\_ECG, and non\_linear) for each HRV feature type.
7. The function concatenates the three DataFrames horizontally using the pd.concat() function and returns the resulting DataFrame as ECG.