Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 258 (2025) 1062–1080

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering

EEG VMAC Toolbox: A User-Friendly Open-Source Toolbox for EEG Signals Visualization Manipulation Analysis and Classification

Gopal Chandra Jana[a,]*, Abhishek Karmakar[b,], Anupam Agrawal[a]

[a]*Interactive Technologies & Multimedia Research (ITMR) Lab, Department of Information Technology, Indian Institute of Information Technology Allahabad, Prayagraj – 211015, Uttar Pradesh, India*

[b]*Indian Institute of Information Technology Kalyani, Department of Computer Science and Engineering, Kalyani – 741235, West Bengal, India*

Abstract

Electroencephalography (EEG) is a well-known, non-invasive method for monitoring and recording electrical activities of the human brain. *Problem:* EEG signal visualization, manipulation, analysis, and classification are essential for clinical experts, doctors, and researchers to make certain decisions. So, there is a need for a toolbox that can provide such functionalities with a user-friendly graphical user interface (GUI); availability may be commercial or open-source. *Aim:* This paper describes the proposed and developed open-source EEG VMAC Toolbox, which provides an interface with features including a series of state-of-the-art methods for EEG signal analysis. *Method:* The main menu options of EEG VMAC Toolbox are - 1) File, 2) Signal Visualization, 3) Filtering, 4) Signal Decomposition, 5) Feature Reduction, 6) Feature Extraction, 7) Label, 8) Classification Models, and 9) Help. Each menu of the toolbox contains several functionalities. In addition to these nine menus, a file conversion option is available at the bottom of the toolbox. EEG VMAC Toolbox integrates all major state-of-the-art functionalities for EEG signal visualization, manipulation, analysis, and classification, which would be a valuable addition to the current literature. *Results and Findings:* The EEG VMAC Toolbox has been developed using Python programming language and tested over the CHB-MIT EEG Scalp EEG dataset, a benchmark dataset for seizure detection. So, this toolbox has a provision to bring psychologists, neuroscientists, clinical experts, and EEG researchers on the same platform to pursue extensive investigation and research for better reach.

Keywords: Electroencephalogram (EEG), EEG toolbox, EEG signal visualization toolbox, EEG signal analysis tool, Open-source toolbox, Python EEG toolbox;

* Corresponding author. Tel.: +91-993-367-2004.
E-mail address: rsi2018508@iiita.ac.in

10.1016/j.procs.2025.04.342

1. Introduction

Prior suitable time-frequency EEG signal analysis is very much essential before procced to train a machine learning model and come up with an investigative decision over human brain electrical activities. EEG recordings for longer duration (like CHB-MIT EEG data [1], consisting of seizure and non-seizure [2]) result in huge memory storage, which makes the analysis by doctors and researchers difficult, and those with less knowledge of code or do not intend to code.

There have been several packages and works in progress in this field. Still, no one has come up with a single interface that gives the dynamics to the user itself to perform different signal processing techniques to train any Supervised Machine Learning model on time series data. So far, existing tools have limited functionalities, such as converting data into a different format, filtering, sampling, and others, that are only related to EEG data preprocessing. Considering the high usability of EEG signals, we have proposed and developed an open-source EEG VMAC Toolbox.

The EEG VMAC Toolbox has been developed with the aim of performing different standard techniques to visualize, manipulate, analyze, and classify EEG signals dynamically, either confirming the entire dataset or a particular selected EEG channel. The noise removal for EEG Data resembles – BandPass filtering and Butterworth low pass filtering in this EEG VMAC Toolbox. For the need of computation to analyze EEG data, flagged signal Decomposition functions/menus, spectrograms, spectrums, feature reduction, feature extraction, and many more have been consolidated. The proposed Toolbox has been equipped to enable the user to obtain models affably made by themselves and classify the test data on those saved models, such as Logistic Regression, SGD Classifier, Support Vector Machine Classifier, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting, K-Nearest Neighbors and Artificial Neural Network. Any aggregated EEG signals could be processed and visualized, according to the agenda of this Toolbox, provided that the dataset must have a time column (Time Series Data is mandatory). Major functionalities of EEG VMAC Toolbox are mentioned in section-3 and also, shown in Fig. 1. On a definite note, we had all our operations being thoroughly checked with a vivid functional analysis and application on CHB-MIT Scalp EEG Dataset [1]. EEG VMAC Toolbox is developed using python programming language with the aim of analyzing EEG recordings whose format is in edf or in csv only. This Toolbox could view EEG data in csv and edf format/files and even the conversions of the data are also provided i) from edf to csv, ii) from edf to excel, iii) from csv to edf. For all manipulation pandastable [3] is being integrated with proper configuration. The library pandastable [3] provides the standard GUI table widget framework in tkinter for data analysis functionality.

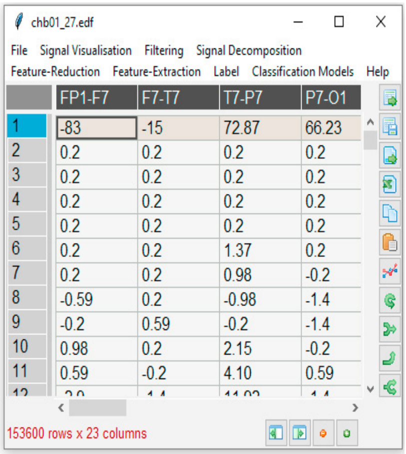


Fig. 1. Major functionalities of EEG VMAC Toolbox.

This work is in the context of developing, an open-source toolbox with user friendly Graphical User Interface (GUI) which totally works for the incorporating EEG signals operations and visualization. The GUI of this toolbox contains specific click-based menu for signals loading, preparation (using the functionality of pandastable [3]) analysis, manipulation, visualization and classification. The analysis outcome can be obtained easily and quickly. The main objective of EEG VMAC Toolbox is that various standard packages (related to few signals processing and classification) and visualizations are summed up in a single GUI. We have used the most widely used seizure detection dataset namely CHB-MIT Scalp EEG dataset [1] for testing the developed toolbox.

More specifically, our plan was to design a genteel open-source toolbox which could connect the pipeline from the data visualization, analysis and manipulation to classification of EEG data and saving necessities on the way. We have added the feature of saving the data in completion of each and every step.

Key contributions in the development of the EEG VMAC Toolbox are:

- **Open-Source User-Friendly GUI:** Features an intuitive graphical user interface that simplifies access to functionalities, making it accessible to users with varying technical expertise.

- **Comprehensive EEG Processing:** Integrates various techniques for EEG signal visualization, manipulation, analysis, and classification, enhancing workflow efficiency for researchers and clinicians.
- **Visualization Capabilities:** Offers diverse visualization options, including channel plots, local peaks, power spectral density, and spectrograms, enabling detailed insights into EEG data.
- **Filtering and Decomposition:** Provides advanced filtering (Band Pass, Butterworth) and decomposition methods (DWT, EMD, FFT, HHT) for effective EEG signal cleaning and analysis.
- **Classification Support:** Facilitates the training and validation of multiple classification algorithms (e.g., Logistic Regression, SVM, Random Forest), empowering users to build tailored models for EEG data classification.

The organization of this paper has been arranged into the following parts: Section 2 describes the summary of the problem Statement, background, experimental data, and contributions. Section 3 contains the framework of the EEG VMAC Toolbox and its features. Section 4 demonstrates GUI screenshots and empirical outputs of the EEG VMAC Toolbox. Section 5 proclaims the additional functionalities. Section 6 shows the comparison table to differentiate EEG VMAC from the existing Toolboxes and includes the future scope of our developed toolbox, and section 7 concludes this paper with a few limitations of this Toolbox.

2. Summary of Problems Statement, Background and Contributions

This section provides insight into our proposed tool's problem statement, background, and contribution. Processing EEG signals has always been a challenging task to perform due to its sensitivity and measuring techniques. One needs to do much research, grill many questions, or surf the internet to find what various kinds of analysis and visualization techniques could be applied to an EEG Signal Visualization, Manipulation, Analysis, and Classification. In order to provide the best-organized schemes, we have designed this EEG VMAC Toolbox GUI that covers various methods of Signal visualization, Manipulation, Analysis, and Classification, which includes data manipulation, pre-processing, data analysis, filtering, signal decomposition, and most unique part – it has provision to train eight supervised machine learning models for classification. More particularly, input data pre-processing, filtering, decomposition, feature reduction, extraction, labeling the class, concatenating two or more files, and then preparing the models - Logistic Regression, Stochastic Gradient Descent Classifier, Support Vector Machine Classifier, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting, K-Nearest Neighbors and Artificial Neural Network can be done easily with the help of our proposed and developed EEG VMAC Toolbox.

The Python script of this toolbox consisting of the algorithms and functions are arranged in a consistent manner with a well demonstration using the Graphical User Interface. EEG VMAC Toolbox is integrated using a python GUI package called "tkinter" in such a way that it makes the operations to execute tightly. The Python libraries being used for the visualization are – pandastable [3], matplotlib [4], mne [5], PIL[6], cv2[7], scipy [8] and for operations – pandas[9], sklearn [10], pickle [11], pyeeg [12], numpy [13], os [14], pyedflib [15], Pylab [16], PyEMD [17], pywt [18], future [19], xlrd [20], tkinter [21] and keras[22].

The sole objective of this toolbox is to provide functionalities that could be performed on the EEG time series dataset. Someone without experience with machine learning terms could also train a model and save it for future use. The models could be on epileptic seizure detection, motor imagery task classification, and so on and so forth. In this proposed toolbox, we considered a few machine-learning approaches to construct classifiers for epileptic seizure [2] detection through analysis of the scalp EEG signals.

The metadata for this EEG VMAC Toolbox, including quick guides, manuals, support info, and downloadable links, have been mentioned in the Data availability statement section.

Experimental Data: The proposed and developed EEG VMAC Toolbox has been tested on the CHB-MIT Scalp EEG dataset which is a benchmark dataset for Seizure detection, which is publicly available at [1]. It has mainly 23 cases, mentioned as chb01, chb02 and so on. Each case has many EEG recordings arranged as chb01_01.edf, ch01_02, and so on. We have considered chb01_27 for the purpose of functionalities (except label and classification models functionality) testing of EEG VMAC Toolbox. User may use any of the samples from the CHB-MIT dataset as per their convenience. The experimental dataset follows the European Data format (edf), which supports the BIDS [23] standard.

It is always important to draw a line between our work proposed/developed and other concerned publications. Considering this, we have also attached a comparison table in section 6 (Table 3).

3. EEG VMAC Toolbox Features and Framework

Our proposed and developed open-source EEG VMAC Toolbox has nine clicked-based menus; each menu has different functionalities with different operational features. The detailed functionalities of each menu have been described as follows.

EEG VMAC Toolbox functionalities:

A. File

i) Clear: It clears the table and closes the file which is already opened,

ii) Open EDF file: Open/Views the EDF format files.

iii) Open CSV file: Open/Views the CSV files.

iv) Exit: this one exits the whole GUI (Graphical User Interface) currently running.

B. Signal Visualization

i) Visualize all EEG Channels: Plotting of all channels with different default colors to provide a better visualization to users for distinct EEG signals of selected EEG data files.

ii) Visualize individual EEG Channel: This functionality facilitates the display of the particular channel from the entire dataset.

iii) Visualize local Peaks: This functionality facilitates the display of all the local maximum points and the local minimum points [24] till that point of the channel timing entered by the user.

iv) Visualize with MNE: This functionality facilitates the plotting of the EEG Signals in a detailed and more systematic way, allowing the operations provided under this package [5].

v) Visualize Power Spectral Density: This functionality facilitates the plot of the Power Spectral Density (PSD) [25] [5] of an input signal and provides the provision to save PSD plots.

vi) Visualize Magnitude Spectrum: This functionality facilitates the visualization and saves the produced magnitude spectrum [26] [4] of the input EEG Signals.

vii) Visualize Phase Spectrum: This functionality facilitates the visualization and saves the produced phase spectrum [26] [4] of the input EEG Signals.

viii) Visualize Angle Spectrum: This functionality facilitates the visualization and saves the produced angle spectrum [27] [4] of the input EEG Signals.

ix) Visualize Spectrogram: This functionality facilitates plotting and saving the spectrogram [28] [29] [4] image of the channel selected by the user.

x) Crop file: This functionality facilitates cropping of an input file just by specifying the exact timing (time duration: start time and end time); the data will be cropped out, which lies within the range of the starting time and ending time.

C. Filtering

i) Band Pass Filtering: This functionality facilitates the users to save the performed Band Pass filtering [30] [5] over the selected channel.

ii) Butterworth Filtering: This functionality facilitates the users to save the performed Butterworth [31] [8] filtering over the selected channel.

D. Signal Decomposition

i) Discrete Wavelet Transform (DWT): This functionality facilitates the user to perform three types of DWT [32][33]. The first one is Single Level DWT [34], the second one is Multilevel DWT [35], and the third one is Partial Level DWT [36]. After performing the DWT, the user can save the output.

ii) Empirical Mode Decomposition (EMD): This functionality facilitates users to perform EMD [37] [38] [39] [16] [17] [40] with saving the output option as well.

iii) Fourier Transform: This functionality facilitates users to perform FFT [41] [42] [43]. It takes the channel number as an input and then plots the fast Fourier transform output. Also, the user can save the output.

iv) Hilbert-Huang Transform: This functionality facilitates to user to calculate and plot the HHT transform [44] [8] [16] [40] and save it.

E. Feature Reduction: This functionality facilitates the user to perform Principal Component Analysis (PCA) [45][46] and Independent Component Analysis (ICA) [47][48]. Users can choose one of them to reduce the dimension of the input EEG Data as per the requirement.

F. Feature Extraction: This functionality facilitates the user to extract features from the input EEG signals. Users can extract a list of features which are Max Peak Values [24], Min Peak Values [24], Maximum, Minimum, Mean, Range, Root Mean Square, Variance, Standard Deviation, Kurtosis, Skewness; all can be referred from [49][50], DFA (Detrended Fluctuation Analysis), Hurst (Hurst Exponent), PFD (Petrosian Fractal Dimension), HFD (Higuchi Fractal Dimension), Hjorth (Hjorth mobility and complexity) [49][51], SVD Entropy, BIN Power (Power Spectral Intensity (PSI) and Relative Intensity Ratio (RIR)) [12].

G. Label

i) Add Labels in Data: This functionality allows the user to annotate or label the classes belonging to the input data files, which is useful when performing classification.

ii) Concat Labeled Data: This functionality facilitates to user for concatenating the annotated or labeled files for the classification model training.

H. Classification Models: This functionality facilitates the user to perform the classification task. The classification Model menu option has two functionalities – i) Training and saving models and ii) Checking predictions on saved models. In total, eight classification algorithms have been included in the Training and saving models functionality. Eight classification algorithms are Logistic Regression [52][53], Stochastic Gradient Descent Classifier [54][55], Support Vector Machine [56][57], Decision Tree Classifier [58][59], Random Forest [60][61], Gradient Boosting [62][63], K-Nearest Neighbors [64][65] and Artificial Neural Network [66][67]. The user can build and train a classification model with the help of any desired algorithm they wish to and if they have the pre-trained model ready, they can perform testing using Checking predictions on saved models' functionality.

I. Help: This functionality facilitates the user to get the user manual and recent updates about the EEG VMAC Toolbox.

These are the main functionalities available in the EEG VMAC Toolbox version 1.0.0. To utilize this toolbox's features, users need to install its pre-required Python packages listed in Table 1. In this toolbox, the ninth menu has the functionality for selecting classification models. To use the classification features of this toolbox, users need to select one of the existing models from the ninth menu. To train a particular model, the user should have a training dataset beforehand with a labeled class, as this functionality is a must for training a supervised learning model. After training, the user can save the trained model for the use of testing over the test data. Hyperparameters of the eight classification models are mentioned in the Table 2.

Table 1. Python packages are required to install before running the EEG VMAC Toolbox script.

Package Name	Version	Package Name	Version
tkinter [21]	0.1.0	Sklearn-extensions [68]	0.0.2
mne [5]	0.20.7	pydot [69]	0.0.2
pyeeg [12]	0.4.4	future [19]	0.18.2
matplotlib [4]	3.1.1	pandastable [3]	0.12.2.post1
numpy [13]	1.16.5	pandas [9]	0.25.1
Imageio [70]	2.5.0	xlrd [20]	1.2.0
PIL [6]	7.2.0	Pyedflib [15]	1.2.0
Pillow [71]	6.2.0	Pylab [16]	3.1.1
Opencv-python [72]	4.1.0.25	PyEMD [17]	0.0.10
Os [14]	any	pywt [18]	1.2.0
scipy [8]	1.4.1	pickle [11]	4.0
sklearn [10]	0.23.1	keras [22]	2.3.0
tensorflow [73]	2.0.0a0		

Table 2: Hyperparameter configuration of the Classification models.

Classification models	Hyperparameter configuration
Logistic Regression	penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state= user_input, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None
Stochastic Gradient Descent Classifier	loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1, n_jobs=None, random_state= user_input, learning_rate='optimal', eta0=0.0, power_t=0.5, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, class_weight=None, warm_start=False, average=False
Support Vector Machine	C=user_input, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=user_input
Decision Tree	criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state= user_input, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0
Random Forest	n_estimators= user_input, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state= user_input, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None
Gradient Boosting	loss='deviance', learning_rate=0.1, n_estimators= user_input, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, init=None, random_state= user_input, max_features=None, verbose=0, max_leaf_nodes=None, warm_start=False, validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0
K-Nearest Neighbors	n_neighbors=user_input, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None
Artificial Neural Network	units=user_input, activation=user_input, use_bias=True, kernel_initializer="glorot_uniform", bias_initializer="zeros", kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, bias_constraint=None, random_state= user_input

4. Empirical outputs of the functionalities of EEG VMAC Toolbox

In this section, we have shown and described the major things of all functionalities of our developed toolbox in a demonstrative manner, including empirical outputs of each functionality. Demonstrative outputs of each menu are mentioned as follows.

4.1. Empirical outputs of the functionalities of File menu

The file menu has four functionalities are Open EDF file, Open CSV file, Clear and Exit. Where, Open EDF file allows user to open EDF files, Open CSV file allows user to open CSV files, Clear allows user to clear loaded data and Exit allows user to exit from the GUI. An empirical output of the file menu has been shown using Fig. 2.

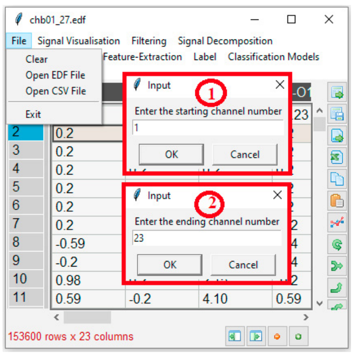


Fig. 2. Functionalities of file menu of EEG VMAC Toolbox.

4.2. Empirical Outputs of Signal Visualization Functionality

Under this menu we have a total ten functionalities. Primarily we have used scipy [8], matplotlib [4], numpy [13] and mne [5] packages. Ten functionalities of this menu bar are (1) Visualize All EEG Signals, (2) Visualize Individual

EEG Channel, (3) Visualize Local Peaks, (4) Visualize with MNE, (5) Visualize Power Spectral Density, (6) Visualize Magnitude Spectrum, (7) Visualize Phase Spectrum, (8) Visualize Angle Spectrum, (9) Visualize Spectrogram, (10) Crop file.

(1) Visualize All EEG Signals, functionality helps to visualize the entire signals (contain in the selected file) values in a table as well as in color plot pop-up window. The signals data were plotted using the python package matplotlib [4] and adjusted it in a pop-up window. A visual representation of all EEG channels color plot has been shown using Fig. 3.

(2) Visualize Individual EEG Channel, functionality helps the user to observe the individual channel in depth. The user just needs to enter the particular channel number in pop-up prompt then the corresponding signals values and color plot will display. A visual representation of the output of this functionality has been shown in Fig. 4.

(3) Visualize Local Peaks, functionality helps user to visualize local maximums and local minimums peaks of a signal of an individual channel. A visual representation of the output of this functionality has been shown using Fig. 5.

(4) Visualize with MNE, functionality helps user to visualize EEG data with MNE [5]. A visual representation of the output of this functionality has been shown using Fig. 6.

(5) Visualize Power Spectral Density, functionality helps user to visualize and analyze the power spectral density (PSD) [25] [5] of an input EEG signals. A visual representation of the output of this functionality has been shown using Fig. 7.

(6) Visualize Magnitude Spectrum, produces a magnitude Spectrum [26] [4] for any selected channel of an input EEG data. The magnitude_spectrum() function in the pyplot module of matplotlib [4] python package plots this spectrum of a periodic signal. A visual representation of the output of this functionality has been shown using Fig. 8.

(7) Visualize Phase Spectrum, functionality produces a Phase Spectrum [26] [4] of an input EEG signal. In the background, the phase_spectrum() function in the pyplot module of matplotlib [4] python package plots the phase spectrum of the input signal. A visual representation of the output of this functionality has been shown using Fig. 9.

(8) Visualize Angle Spectrum, functionality produces an Angle Spectrum [27] [4] of an input EEG signal. In the background The angle_spectrum() funtion in the pyplot module of matplotlib [4] python package plots the angle spectrum of an input EEG signal. A visual representation of the output of this functionality has been shown using Fig. 10.

(9) Visualize Spectrogram, functionality provides user to visualize the spectrogram [28] [29] [4] of a time series data (e.g EEG). A visual representation of the output of this functionality has been shown using Fig. 11.

(10) Crop file, functionality allows the user to take away and store some part of the data which is an essential for some particular cases, this feature is domain specific. We have added this facility in this EEG VMAC Toolbox to separate the data accordingly as per classes or labels.

Suggesting to refer the user manual for the proposed toolbox for the step-by-step process in detail to perform all functionality.

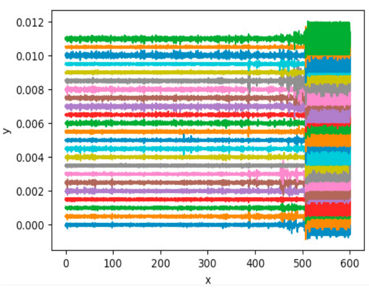


Fig. 3. Output of the visualize all EEG signals functionality.

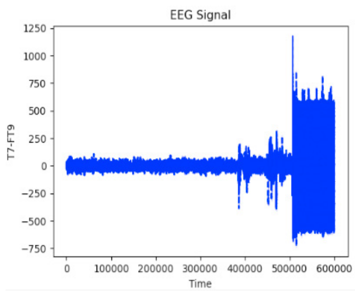


Fig. 4. Output of the visualize individual EEG channel functionality.

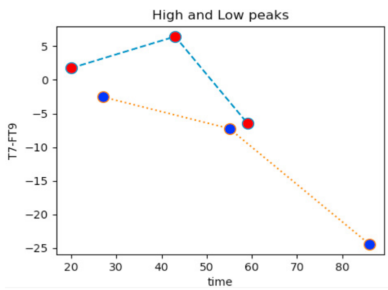


Fig. 5. Output of the visualize local peaks functionality.

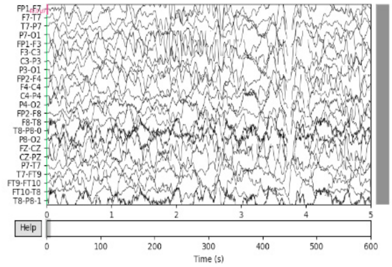


Fig. 6. Output of the visualize with MNE functionality.

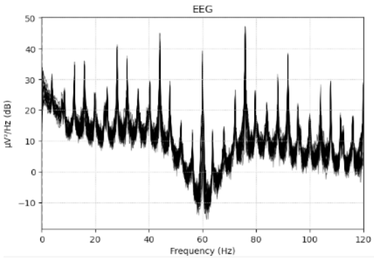


Fig. 7. Output of the visualize power spectral density functionality.

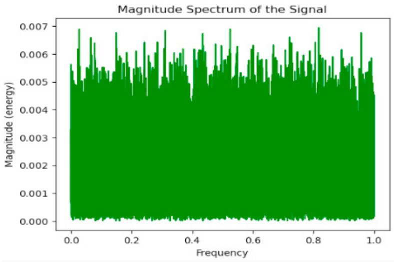


Fig. 8. Output of the visualize magnitude spectrum functionality.

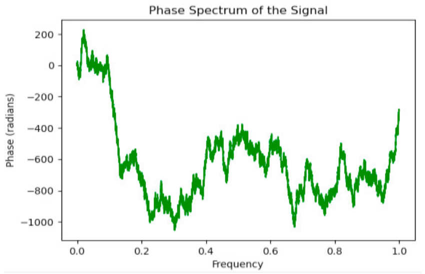


Fig. 9. Output of the visualize phase spectrum functionality.

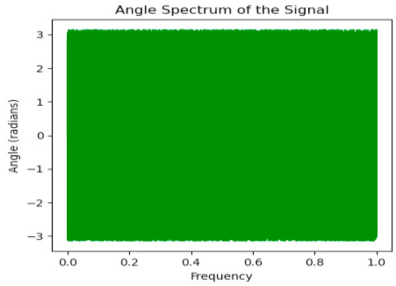


Fig. 10. Output of the visualize angle spectrum functionality.

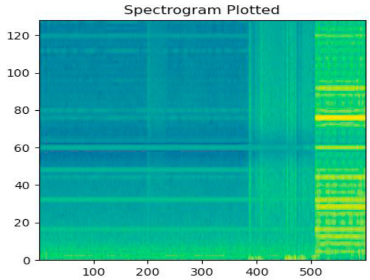


Fig. 11. Output of the visualize spectrogram functionality.

4.3. Empirical Outputs of filtering Functionality

EEG VMAC Toolbox has two main functionalities under filtering menu. The main two functionalities are Band Pass Filtering [30] [5] and Butterworth Filtering [31] [8]. Where, Band Pass Filtering, functionality helps the user to perform the filtering operation over the input data with a low band pass filtering [30] [5] function. Under this functionality, full-fledged band pass filtering of MNE [5] package has been used for computing and plotting this filter. In addition to this the lower frequency used is 7 and higher frequency as 30. The rest of the default parameters are same as in MNE package whose details could be referred from [74]. A visual representation of the output of this functionality has been shown using Fig. 12. On the other hand, Butterworth Filtering, functionality helps the user to perform the filtering operation over the input data with a butterworth filtering [31] [8] function. Under this functionality, Butterworth low pass filtering functionality in Scipy [8] package has been used for the computing and plotting this filter. The rest of the default parameters are same as in Scipy package whose details could be referred from [75]. A visual representation of the output of this functionality has been shown using Fig. 13.

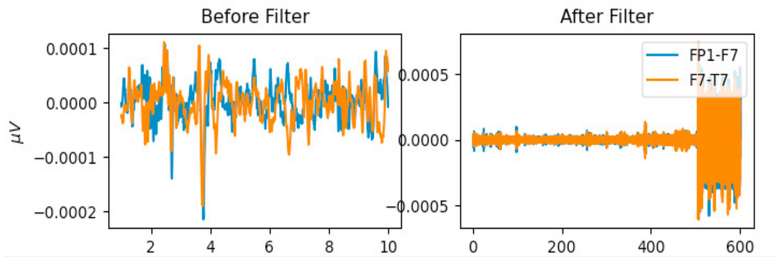


Fig. 12. Output of the Band Pass filtering functionality.

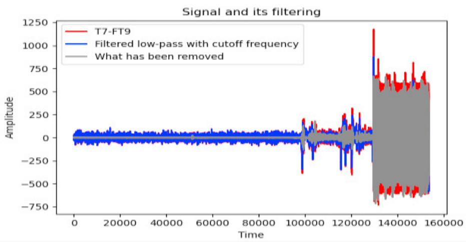


Fig. 13. Output of the ButterWorth filtering functionality.

4.4. Empirical outputs of signal decomposition functionality

Signal decomposition is one of the common techniques used in EEG signal analysis. This functionality provides users to perform signal decomposition operations using Discrete Wavelength Transform [32][33] (Single [34], multi-level [35] and partial [36]), Empirical Mode Decomposition [37] [38] [39] [16] [17] [40], Fast Fourier Transform [41] [42] [43] and Hilbert-Huang Transform [44] [8] [16] [40]. EEG VMAC provides a simple click-based GUI to utilize the functionalities of these three major decomposition techniques. Under the Discrete Wavelet Transform functionality, user can use single level, multilevel, and partial DWT as per the need. Where, Single Level DWT [34] functionality takes the EEG channel number and the mode (i.e name of the dwt family) as an input and then compute approximate coefficients and details coefficients and accordingly plots it. In this functionality, pywt [18] package has been used for the Single Discrete Wavelength Transform. A visual representation of the output of this functionality has been shown using Fig. 14 and Fig. 15. Similarly, Multilevel DWT [35] functionality takes the channel number, mode (i.e the name of the dwt family) and number of levels as an input to compute the output of the decomposition. In this functionality, pywt [18], and wavedec [18] has been used for the Multilevel Discrete Wavelength Transform. A visual representation of the output of this functionality has been shown using Fig. 16 and Fig. 17. Similarly, Partial Level DWT [36] functionality computes just a set of coefficients and is then useful when you need only the approximations or only details at the given level. In this functionality, pywt [18] package has been used for the Partial Discrete Wavelength Transform. A visual representation of the output of this functionality has been shown using Fig. 18 and Fig. 19. Similarly, the Empirical Mode Decomposition functionality helps the user to compute and visualize the IMFs of particular input EEG signals. To perform Empirical Mode Decomposition functionality, pylab [18] and PyEMD [17] packages have been integrated with this toolbox. It calculates the IMFs by using the emd() function. A visual representation of the output of this functionality has been shown using Fig. 20 and Fig. 21. Similarly, Fourier Transform functionality facilitates users to compute and visualize the fast Fourier transform [41] [42] [43] of an input signal. In this functionality, scipy.fftpack has been used in the background to calculate the FFT. A visual representation of the output of this functionality has been shown using Fig. 22 and Fig. 23. Similarly, Hilbert-Huang Transform functionality facilitates users to compute and visualize the Hilbert-Huang Transform (HHT) [44] [8] [16] [40] of an input EEG signal. In this functionality, Pylab [16], scipy.signal [8] and PyEMD [17] have been used in the background to compute the HHT. A visual representation of the output of this functionality has been shown using Fig. 24 and Fig. 25.

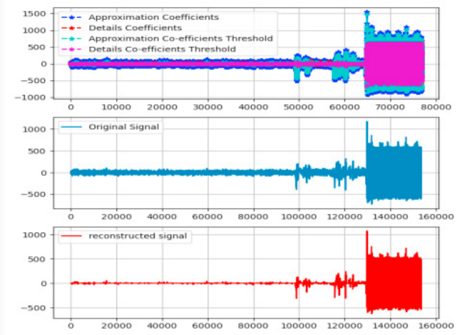
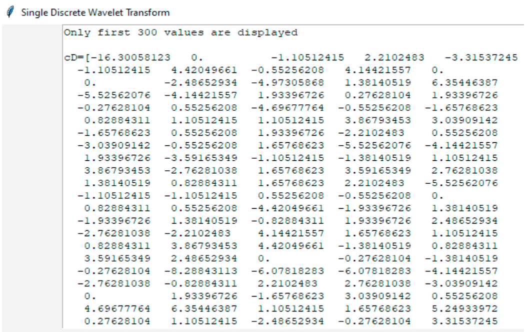


Fig. 14. Output of the single level DWT functionality.



Single Discrete Wavelet Transform

```
Only first 300 values are displayed

cD=[-16.30058123   0.          -1.10512415   2.2102483   -3.31537245
  -1.10512415   4.42049661  -0.55256208   4.14421557   0.
   0.          -2.48652934  -4.97305868   1.38140519   6.35446387
  -5.52562076  -4.14421557   1.93396726   0.27628104   1.93396726
  -0.27628104   0.55256208  -4.69677764  -0.55256208  -1.65768623
   0.82884311   1.10512415   1.10512415   3.86793453   3.03909142
  -1.65768623   0.55256208   1.93396726  -2.2102483    0.55256208
  -3.03909142  -0.55256208   1.65768623  -5.52562076  -4.14421557
   1.93396726  -3.59165349  -1.10512415  -1.38140519   1.10512415
   3.86793453  -2.76281038   1.65768623   3.59165349   2.76281038
   1.38140519   0.82884311   1.65768623   2.2102483   -5.52562076
  -1.10512415  -1.10512415   0.55256208  -0.55256208   0.
   0.82884311   0.55256208  -4.42049661  -1.93396726   1.38140519
  -1.93396726   1.38140519  -0.82884311   1.93396726   2.48652934
  -2.76281038  -2.2102483    4.14421557   1.65768623   1.10512415
   0.82884311   3.86793453   4.42049661  -1.38140519   0.82884311
   3.59165349   2.48652934   0.          -0.27628104  -1.38140519
  -0.27628104  -8.28843113  -6.07818283  -6.07818283  -4.14421557
  -2.76281038  -0.82884311   2.2102483    2.76281038  -3.03909142
   0.           1.93396726  -1.65768623   3.03909142   0.55256208
   4.69677764   6.35446387   1.10512415   1.65768623   5.24933972
   0.27628104   1.10512415  -2.48652934  -0.27628104   3.31537245
```

Fig. 15. Shows the provision to save the computed values during performing single level DWT functionality.

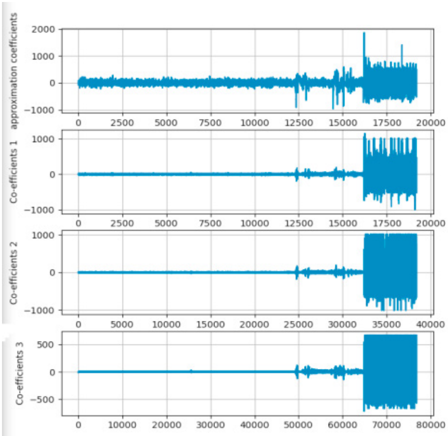


Fig. 16. Output of the multilevel DWT functionality.

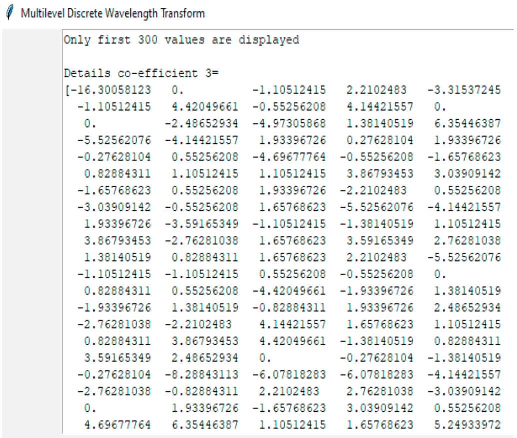


Fig. 17. Shows the provision to save the computed values during performing multilevel DWT functionality.

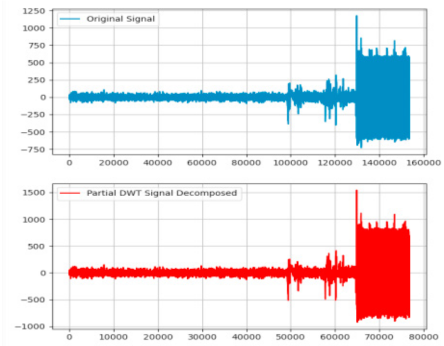


Fig. 18. Output of the partial DWT functionality.

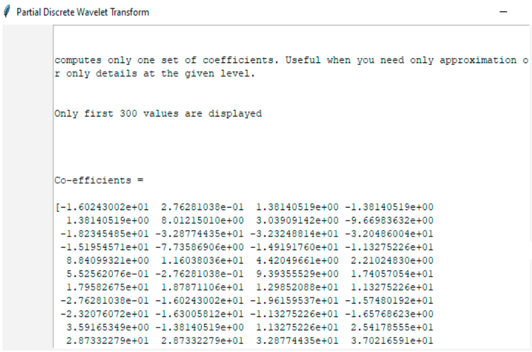


Fig. 19. Shows the provision to save the computed values during performing partial DWT functionality.

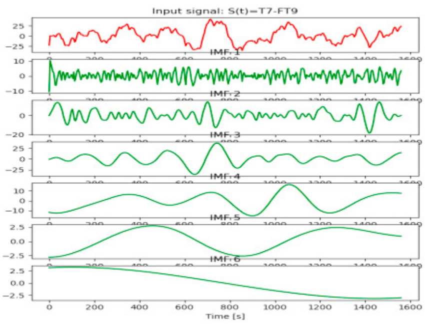


Fig. 20. Output of the EMD functionality.

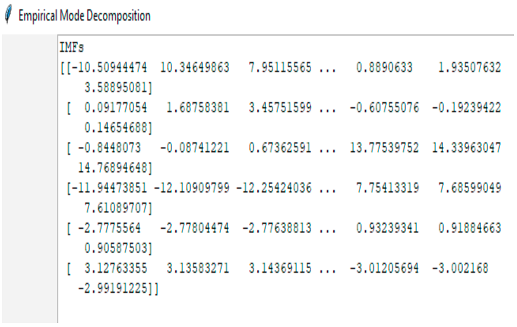


Fig. 21. Shows the provision to save the computed values during performing EMD functionality.

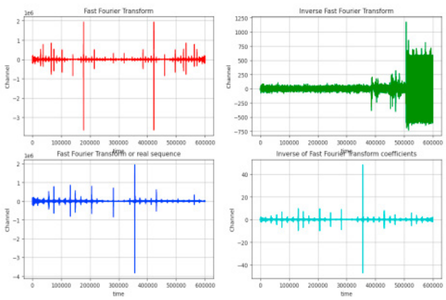


Fig. 22. Output of the Fourier Transform functionality.

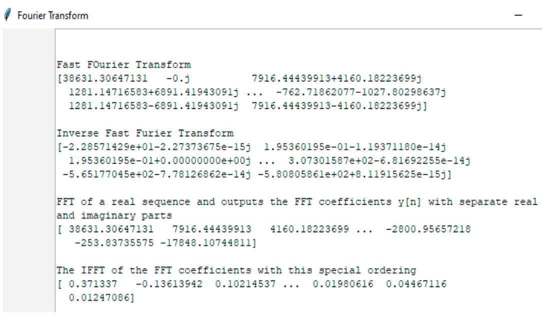


Fig. 23. Shows the provision to save the computed values

during performing Fourier Transform functionality.

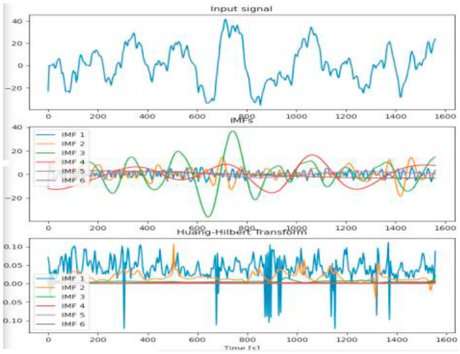


Fig. 24. Output of the Hilbert-Huang Transform functionality.

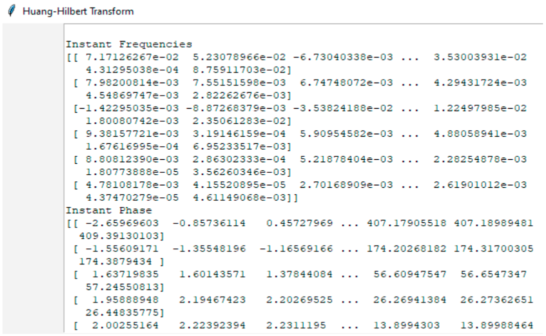


Fig. 25. Shows the provision to save the computed values during performing Hilbert-Huang Transform functionality.

4.5. Empirical outputs of feature reduction functionality

Feature Reduction functionality allows users to reduce features as well as consider only the essential EEG channels. This functionality of the EEG VMAC Toolbox includes – Principal Component Analysis (PCA) [45][46], and Independent Component Analysis (ICA) [47][48]. Both of these techniques have been included in the Toolbox for the use of dimension reduction of the EEG Data in terms of channel and feature extraction from input EEG signals. The toolbox allows users to specify the number of components they wish to retain. A visual representation of the output of this functionality has been shown using Fig. 26 and Fig. 27.

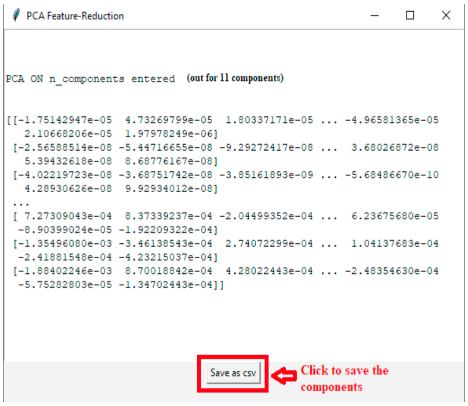


Fig. 26. Outputs of the PCA functionality.

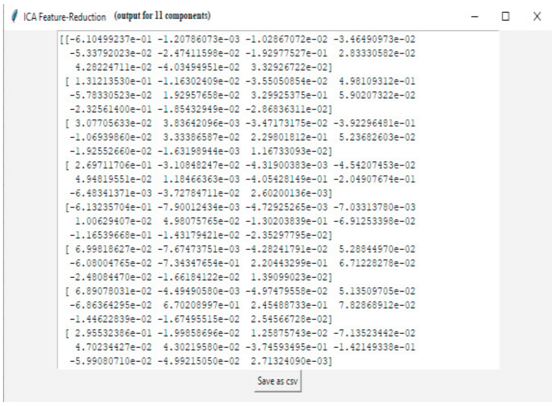


Fig. 27. Outputs of the ICA functionality.

4.6. Empirical outputs of feature extraction functionality

Feature extraction functionality facilitates users to extract features from an input EEG signals. User can extract list of features which are time domain features such as Max Peak Values [24], Min Peak Values [24], Maximum, Minimum, Mean, Range, Root Mean Square, Variance, Standard Deviation, Kurtosis, Skewness, all can be referred from [49] [50], Time-frequency domain features such as DFA (Detrended Fluctuation Analysis), Hurst (Hurst Exponent), PFD (Petrosian Fractal Dimension), HFD (Higuchi Fractal Dimension), Hjorth (Hjorth mobility and complexity) [49]

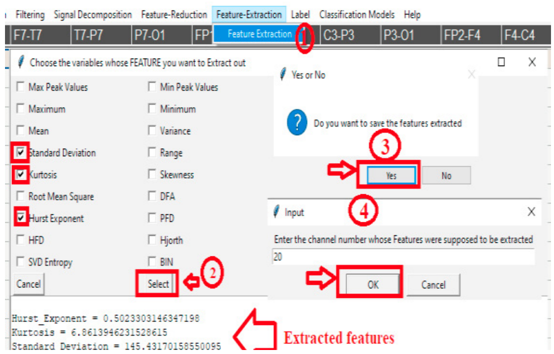


Fig. 28. Output of the feature extraction functionality.

[51], SVD (Singular Value Decomposition) Entropy, and Frequency-domain features such as BIN Power (Power Spectral Intensity (PSI) and Relative Intensity Ratio (RIR)) [12]. A visual representation of the output of this functionality has been shown in Fig 28.

4.7. Empirical outputs of Label functionality

Label menu of EEG VMAC Toolbox has two main functionalities which are Add label, better known as annotation [80] in the data (or for a class of data samples) and concatenation of all data after adding the label. After selecting the Add label from Label menu, user can load a data file to add class label in it and has a provision to save the data after adding the class label. For multiple data with classes annotated, the concatenation of those files with the same number of attributes could be performed through concat labelled data functionality. The functionalities of Label menu helps the user to add a class label properly before feeding the data into supervised machine learning models. A visual representation of this functionality has been shown using Fig. 29.

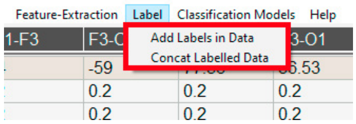


Fig. 29. Shows the functionalities of the Label menu.

4.8. Empirical outputs of classification models functionality

Classification Models menu of EEG VMAC Toolbox has two main functionalities, which are training and saving models, and checking predictions over test data using the saved model. Training and Saving Models functionality facilitates users to perform training and save the trained pre-existed supervised machine learning models without doing any coding. Before proceeding to train an existing model, the user needs to load the labeled data (the user can prefer the functionalities of the Label menu of this toolbox to prepare data and add its class labels). After loading the labeled training data, the user needs to click on the Training and Saving Models functionality. This allows a pop-up window with all the names of pre-existed models and has provision to select anyone from them to perform training. An empirical training result of the Decision Tree and Support Vector Machine classifier has been shown using Fig. 30 and Fig. 31, respectively. Checking predictions on the saved model functionality, facilitates users to use the saved model for testing/prediction. The user needs to select the saved trained model and the test data to get predicted results. An empirical prediction/testing result of the Support Vector Machine and Random Forest classifier has been shown using Fig. 32 and Fig. 33, respectively, where 70% of data have been considered for training and 30% from testing, suggesting to refer the user manual of this toolbox to understand the process in detail.

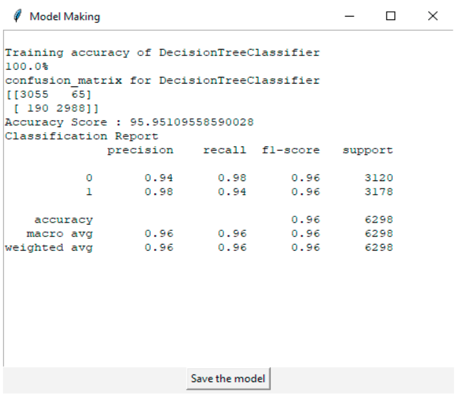


Fig. 30. Shows the performance of the Decision Tree classifier training.

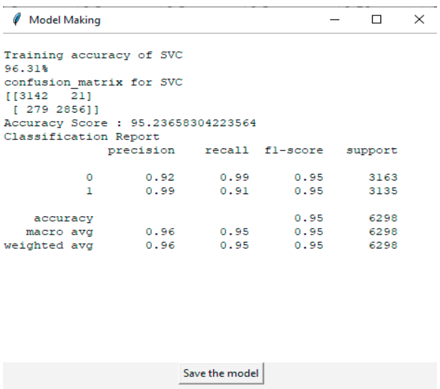


Fig. 31. Shows the performance of the Support Vector Machine classifier training.

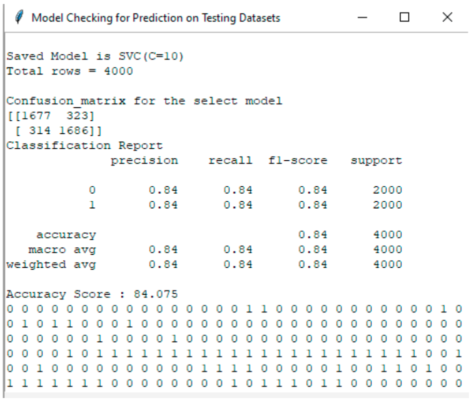


Fig. 32. Shows the prediction performance of the Support Vector Machine classifier over test data.

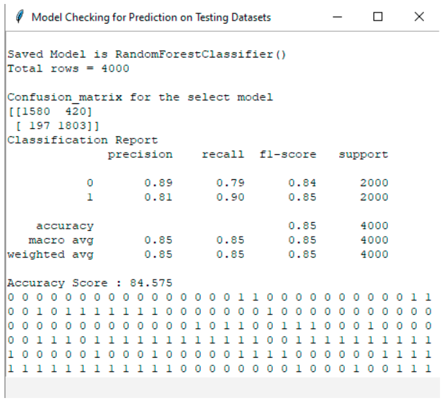


Fig. 33. Shows the prediction performance of the Random Forest classifier over test data.

4.9. Functionalities under Help

The help menu of this toolbox will help users to get the user manual and an updated version of the EEG VMAC Toolbox. After clicking on Help, the user can find the direct link to the EEG VMAC Toolbox website and find more details about this toolbox. Also, the help menu has another functionality, "About Dataset", which shows details (column names or any additional data that the data file has) of the selected data file. A visual representation of this functionality is shown in Fig 34.

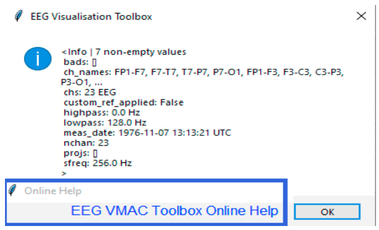


Fig. 34. Shows the help functionality of the toolbox.

5. Additional functionalities of EEG VMAC Toolbox

As mentioned earlier, the data edf or csv file is displayed in a table format using pandastable [3], a major advantage has been taken from that module. The pandastable Graphical User Interface has a showbar and statusbar, we have imported this package due to the feature of data exploration, functionalities could be referenced from [3]. Fig. 35 has been added to show the functionalities.

In the toolbar down most part, we have added the functionality of conversion of files from one format to the other. They are as follows: - i) Convert edf to csv, ii) Convert edf to excel, iii) Convert csv to edf. This EEG VMAC Toolbox operates in only two extensions, i) edf files, and ii) csv files. If the user has any other file format, like suppose excel, that should get converted to any of these two. In case the conversion is done from csv to edf, the researcher/user should specify the sampling rate, i.e. for CHB-MIT its 256Hz. The functionalities of file conversion are highlighted and shown in Fig. 35.

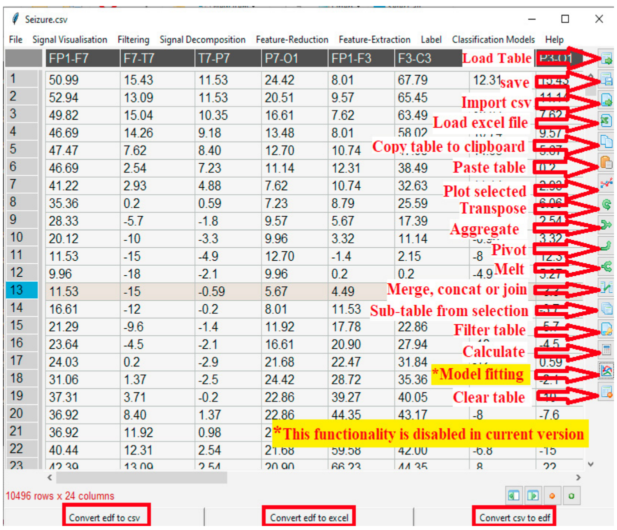


Fig. 35. Shows Pandastable toolbar and statusbar integrated in EEG VMAC Toolbox. The file conversion functionalities highlighted with red rectangular boxes at the bottom.

6. Comparison with existing Tools

In this section, we have included a comparison of our toolbox with other existing tools. We have compared our toolbox with the EDF browser [76] [77], PyEEG [12] and EEGLab [78] [79], AnyWave [81], HAPPE [82], and qEEGt [83]. The EDF browser [76] [77] supports only EDF type files for any operations. This browser offers visualization of EEG signals for a particular selected channel from the loaded EEG data. Moreover, it also provides filtering functionalities for noise removal, the possibility of down-sampling of the working signal, and the power on the frequency bands computation. EEG VMAC Toolbox has more visualization scope compared to the EDF browser. It plots the spectrums, spectrograms, and the Power Spectral Density (PSD). These figures can be stored as an image itself or in csv format. In addition, this EEG VMAC Toolbox can also do signal decomposition, feature extraction and reduction, and filtering. There are a few common features, like the power on frequency bands, the filtering, and spectrograms. However, the EDF browser does not have the facility of doing data operations and saving it in csv, edf, excel, and pickle format files and combining the visualization with model training sections.

EEGLab [78] [79] is an add-on toolbox of MATLAB for processing continuous and event-related EEG signal datasets. Besides, EEGLAB not only processes EEG data but also provides visualization functionalities, both in time-domain signal plotting form and also in a Graphical-Scalp drawing. EEGLab incorporates time-frequency analysis, IRF Filtering, and other interactive process flexibility to the user along with the facility of topological plotting of EEG Montages and many more. On the other hand, our proposed EEG VMAC Toolbox does not have montage functionalities; rather, it has ICA, PCA, feature decomposition, extraction, and classification model training and testing facilities.

PyEEG [12] is a Python open-source module that has been developed to extract the EEG features. Initially, the motive behind the development of such modules was for epilepsy seizure detection, but now PyEEG has been upgraded. The interesting fact for PyEEG is that there are decomposing signal functionalities especially for the re-projection. In the EEG VMAC Toolbox, PyEEG acts as an integral part. For feature extraction more than half of the features demonstrated above were extracted using this package. For our Toolbox, PyEEG plays an integral part in making it more effective.

AnyWave [81] is a cross-platform tool capable of running on all common operating platforms. It provides basic visualizations, marking tools and data export functionality. It also provides external tools as plug-ins, permitted to be developed in any of the common languages such as C++, python and MATLAB to visualize and process EEG data. In addition to various filter and visualization settings, it also provides applying montage and processes data. AnyWave has the feature of independent component analysis (ICA) only. Interim, our EEG VMAC Toolbox cannot plot the scalp data as a montage, but it has the facility of various signals plotting it in time-domain format. In fact, our toolbox facilitates independent component analysis (ICA) and Principal Component Analysis (PCA). also, the classification model training and prediction/testing functionality makes it rare among all.

HAPPE [82] mainly operates on raw files consisting of event-related and resting-state EEG data through a series of filtering and pre-processing of EEG data in a standardized manner in MATLAB. It has multiple stages of semi-automated setting, designing a pipeline. HAPPE implements a wavelet-enhanced ICA (W-ICA) approach followed by ICA. Whereas the EEG VMAC is an open-source Python toolbox, it includes ICA, PCA, and various feature extraction techniques and also has a provision for training and testing the classification models.

qEEGt [83] toolbox is based on the first wave of the Cuban Human Brain Mapping Project (CHBMP) and has marked as the first application in age-corrected normative Statistical Parametric Mapping. The evaluation of z spectra by built-in-age regression is obtained to calculate EEG Scalp spectra. This toolbox solves the inverse problem by utilizing VARETA. The qEEGt is totally based on the Quantitative Analysis of EEG data and its visualization. A different perspective of EEG VMAC Toolbox focuses on visualization, manipulation, analysis, and classification.

In our EEG VMAC Toolbox, there has been no scope for montage or scalp display and operations accordingly, but we have several different functionalities (e.g., training and saving supervised machine learning model, etc.) as mentioned in the above sections. We tried to assemble it in a way for newcomers or for the researchers to Visualize, Manipulate/Modify, Analyze, and Classify the EEG Signals in one place using other already developed open-source packages – mne [5], scipy [8], pyeeg [12], pandastable [3], tensorflow [73], pickle [11], tkinter [21], matplotlib [4], pydot [69], sklearn [10] and many others already discussed above.

Since this toolbox is an open-source and first version, it has several scope to improve the existing functionalities and by adding relevant features specifically with the use and idea of several new visualization approaches/schemes. For instance, EEG VMAC Toolbox can be updated to visualize EEG plot Montage and analyses more of complex networks in EEG Datasets. Specially, the montage plotting, source measures of EEG, and more classification models can be added in future. A tabular version of these comparison has been shown in Table 3.

Table 3. Comparison with other existing Tools.

Tools Name	Supported File Formats	Dependency Needed	Filtering	Signal Decomposition	Feature Reduction	Feature Extraction	Training and Testing Classification Model
EDF Browser [76] [77] (Open Source)	edf, edf+, bdf and bdf+ files	Works in any operating system. No dependencies	Butterworth, Chebyshev, Bessel or Moving Average	None	None	None	None
PyEEG [12] (Open Source)	Python list or numpy array data structure	Python	No filtering	None	None	DFA, Hurst-Exponent, PFD, HFD, Hjorth, SVD-Entropy, BIN array	None
EEGLab [78][79] (Open Source)	edf and set files	MATLAB	Linear Finite Impulse Response (FIR)	None	ICA	None	None
AnyWave [81]	trc, cnt, vhdr, meg4, edf/bdf	C++, MATLAB or Python	Available (speciation not reviewed)	None	ICA	None	None
HAPPE [82]	EGI-exported (Electrical Geodesics, Inc.) Matlab files	MATLAB	Available (speciation not reviewed)	None	ICA	None	None
qEEGt [83]	Not specified	MATLAB	Available (speciation not reviewed)	None	None	None	None

EEG VMAC Toolbox (Open Source)	edf and csv	Python	Butterwoth, Bandpass	Discrete Wavelength Transform (DWT), Empirical Mode Decomposition (EMD), Fourier Transform (FFT), Haung-Hilbert Transform (HHT).	ICA, PCA	Max-peaks, min-peaks, maximum, minimum, mean, variance, standard deviation, range, kurtosis, skewness, root mean square and all the PyEEG functions mentioned in the above section.	In current version models available for training are: Logistic Regression, SGD Classifier, SVC, Random Forest, Gradient Boosting, KNN and ANN.

7. Conclusion

The proposed and developed EEG VMAC Toolbox has been mainly designed to develop a user-friendly GUI for users having minor coding skills for the EEG time series data analysis and to use supervised Machine Learning models for classification. This toolbox has integrated with various mainstream EEG signals processing packages to make it easier for users to visualization, Manipulation, Analysis, and Classification with less involvement of codes; even users with good coding knowledge can easily custom and extend this toolbox.

EEG VMAC Toolbox has been successfully tested in all the trials of the EEG time series dataset for Data Visualization, Manipulation, Analysis, and Classification. For our prime workflow, we have demonstrated all our techniques and operations using a time series CHB-MIT Scalp EEG database and proved that the user can proceed for EEG visualization, Manipulation, Analysis, and Classification with effortless use of EEG VMAC Toolbox. Also, Users can use this toolbox for EEG time series (edf or csv) analysis and classification for Alzheimer's, Autism spectrum disorder, Schizophrenia etc.

This toolbox can be executed on Windows, Linux, iOS, or any other operating system having a Python environment with the mentioned packages in Table 1. It is recommended that the system should have a minimum of 8 GB RAM and an Intel i5 core processor. GPU could be an advantage in training a classification model.

However, i) while using this EEG VMAC Toolbox, sometimes, when opening the EDF or CSV files, the table appears blank. In such cases, the user needs to maximize or restore down the window size once. ii) Only edf and csv format files are accepted. The excel format files need to be converted to edf or csv before any kind of operation. iii) The model fitting option in pandastable showtoolbar has been under modification. The limitations of the toolbox are that it has been tested only with the CHB-MIT Scalp EEG dataset, and it relies on standard Python packages for EEG data visualization, analysis, and classification.

Data availability statement

The proposed and developed EEG VMAC Toolbox has been tested on the CHB-MIT scalp EEG dataset, which is a benchmark EEG dataset of seizure detection. This dataset is publicly available at the link https://physionet.org/content/chbmit/1.0.0/. The proposed toolbox source code and metadata will be available at the link https://www.gcjana.in/EEGVMACToolbox/ and will be provided to researchers upon request. The process of sending the request is mentioned on the website of the toolbox.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was carried out at the Interactive Technologies & Multimedia Research Lab (ITMR Lab) supported by the Department of Information Technology, Indian Institute of Information Technology Allahabad (https://www.iiita.ac.in/), UP, India. The authors also acknowledge all the Python packages used in this development. The authors are grateful for this support.

References

[1] Shoeb Ali. (2010) “Chb-mit scalp eeg database.” physionet.org

[2] Caryn Jory, Rohit Shankar, Deborah Coker, Brendan McLean, Jane Hanna, Craig Newman. (2016) “Safe and sound? A systematic literature review of seizure detection methods for personal use.”, Seizure **(36):** 4-15.

[3] “Pandastable Introduction.”, https://pandastable.readthedocs.io/en/latest/description.html., Accessed 16th Sep. 2022.

[4] “Overview - Matplotlib 3.2.2 Documentation.”, https://matplotlib.org/3.2.2/contents.html., Accessed 16th Sep. 2022.

[5] “MNE-MNE 1.0.Dev0 Documentation.”, https://mne.tools/dev/., Accessed 16th Sep. 2022.

[6] “Pillow.”, https://pillow.readthedocs.io/en/stable/index.html., Accessed 16th Sep. 2022.

[7] “OpenCV: OpenCV-Python Tutorials.”, https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html., Accessed 16th Sep. 2022.

[8] “SciPy Documentation - SciPy v1.9.0.Dev0+1138.384db09 Manual.”, https://scipy.github.io/devdocs/index.html., Accessed 16th Sep. 2022.

[9] “Pandas Documentation - Pandas 1.3.5 Documentation.”, https://pandas.pydata.org/docs/., Accessed 16th Sep. 2022.

[10] “DevDocs - Scikit-Learn Documentation.”, https://devdocs.io/scikit_learn/. Accessed 16th Sep. 2022.

[11] “Pickle - Python Object Serialization - Python 2.7.18 Documentation.” https://docs.python.org/2/library/pickle.html., Accessed 16th Sep. 2022.

[12] Forrest Sheng Bao, Xin Liu, Christina Zhang. (2011) “PyEEG: An Open Source Python Module for EEG/MEG Feature Extraction.”, Computational Intelligence and Neuroscience **2011:** 406391.

[13] “NumPy Documentation.”, https://numpy.org/doc/., Accessed 16th Sep. 2022.

[14] “Os - iscellaneous Operating System Interfaces - Python 3.10.1 Documentation.”, https://docs.python.org/3/library/os.html., Accessed 16th Sep. 2022.

[15] “PyEDFlib -EDF/BDF Toolbox in Python.” https://pyedflib.readthedocs.io/en/latest/., Accessed 16th Sep. 2022.

[16] “Basic Plotting with Pylab - Mpl-Tutorial 0.1 Documentation.” https://jakevdp.github.io/mpl_tutorial/tutorial_pages/tut1.html., Accessed 16th Sep. 2022.

[17] “PyEMD’s Documentation — PyEMD 0.2.13 Documentation.” https://pyemd.readthedocs.io/en/latest/., Accessed 16th Sep. 2022.

[18] Gregory R. Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O’Leary. (2019) “PyWavelets: A Python Package for Wavelet Analysis.”, Journal of Open Source Software **4 (36):** 1237.

[19] “Future Statement Definitions — Python 3.10.1 Documentation.” https://docs.python.org/3/library/__future__.html., Accessed 16th Sep. 2022.

[20] “Xlrd - Xlrd 2.0.1 Documentation.” https://xlrd.readthedocs.io/en/latest/., Accessed 16th Sep. 2022.

[21] “Graphical User Interfaces with Tk - Python 3.10.1 Documentation.”, https://docs.python.org/3/library/tk.html., Accessed 16th Sep. 2022.

[22] “Keras Documentation.” https://faroit.com/keras-docs/1.2.0/., Accessed 16th Sep. 2022.

[23] “Electroencephalography-BIDS Extension Proposal.” https://bids-specification.readthedocs.io/en/stable/04-modality-specific-files/03-electroencephalography.html., Accessed 16th Sep. 2022.

[24] “Scipy.Signal.Find_peaks - SciPy v1.7.1 Manual.”, https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html., Accessed 16th Sep. 2022.

[25] Miller, S. L., & Childers, D. (2012). “Power Spectral Density.” In Probability and Random Processes (pp. 429–471). Elsevier.

[26] Robinson, E. A., & Treitel, S. (2008) “Magnitude spectrum and phase spectrum” Digital imaging and deconvolution: The abcs of seismic exploration and processing. Society of Exploration Geophysicists.

[27] Goodman, Joseph W. (1996) “Introduction to Fourier Optics.”, 2nd ed., McGraw-Hill, New York.

[28] Oppenheim, Alan V., Ronald W. Schafer, and John R. Buck. (1999) “Discrete-Time Signal Processing.” 2nd Ed. Upper Saddle River, NJ: Prentice Hall.

[29] Chassande-Motin, Éric, François Auger, and Patrick Flandrin. (2008) “Reassignment.” in Time-Frequency Analysis: Concepts and Methods. Edited by Franz Hlawatsch and François Auger. London: ISTE/John Wiley and Sons.

[30] Belle A. Shenoi. (2006) “Introduction to digital signal processing and filter design.” (p. 120), John Wiley and Sons.

[31] Butterworth, S. (1930) “On the Theory of Filter Amplifiers." Experimental Wireless and the Wireless Engineer. **7:** 536–541.

[32] Daubechies, I. (1992) “Ten Lectures on Wavelets.” CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics.

[33] Mallat, S. G. (1989) “A theory for multiresolution signal decomposition: The wavelet representation.” IEEE Transactions on Pattern Analysis and Machine Intelligence, **11(7):** 674–693.

[34] “Discrete Wavelet Transform (DWT) — PyWavelets Documentation.” https://pywavelets.readthedocs.io/en/latest/ref/dwt-discrete-wavelet-transform.html#single-level-dwt., Accessed 16th Sep. 2022.

[35] "Multilevel decomposition using wavedec — PyWavelets Documentation." https://pywavelets.readthedocs.io/en/latest/ref/dwt-discrete-wavelet-transform.html#multilevel-decomposition-using-wavedec., Accessed 16th Sep. 2022.

[36] "Partial Discrete Wavelet Transform data decomposition downcoef — PyWavelets Documentation." https://pywavelets.readthedocs.io/en/latest/ref/dwt-discrete-wavelet-transform.html#partial-discrete-wavelet-transform-data-decomposition-downcoef., Accessed 16th Sep. 2022.

[37] Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., Yen, N.-C., Tung, C. C., & Liu, H. H. (1998) "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis." Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454(1971):** 903–995.

[38] Rilling, G., Flandrin, P., & Goncalves, P. (2003) "On empirical mode decomposition and its algorithms." IEEE-EURASIP workshop on nonlinear signal and image processing **3(3):** 8-11.

[39] "EMD — PyEMD 0.2.13 Documentation." https://pyemd.readthedocs.io/en/latest/emd.html., Accessed 16th Sep. 2022.

[40] "PyLab - SciPy Wiki Dump." https://scipy.github.io/old-wiki/pages/PyLab., Accessed 16th Sep. 2022.

[41] Brigham, E. O., & Morrow, R. E. (1967) "The fast Fourier transform." IEEE Spectrum **4(12):** 63–70.

[42] Griffin, D. & Jae Lim. (1984) "Signal estimation from modified short-time Fourier transform." IEEE Transactions on Acoustics, Speech, and Signal Processing **32(2):** 236–243.

[43] "Fourier Transforms." https://docs.scipy.org/doc/scipy/reference/tutorial/fft.html., Accessed 16th Sep. 2022.

[44] Huang, N. E., & Shen, S. S. P. (2014) "Hilbert–huang transform and its applications" (2nd ed., Vol. 16). WORLD SCIENTIFIC.

[45] Lingjun Li, Shigang Liu, Yali Peng, Zengguo Sun, (2016) "Overview of principal component analysis algorithm", Optik **127(9):** 3935-3944.

[46] "Principal component analysis (PCA)." https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html., Accessed 16th Sep. 2022.

[47] "Overview of independent component analysis." (2004). In J. V. Stone, Independent Component Analysis. The MIT Press.

[48] "Sklearn.Decomposition.FastICA." Scikit-Learn, https://scikit-learn/stable/modules/generated/sklearn.decomposition.FastICA.html., Accessed 16th Sep. 2022.

[49] Stancin, I., Cifrek, M., & Jovic, A. (2021) "A review of eeg signal features and their application in driver drowsiness detection systems." Sensors **21(11):** 3786.

[50] Jiang, D., Lu, Y., Ma, Y., & Wang, Y. (2019) "Robust sleep stage classification with single-channel EEG signals using multimodal decomposition and HMM-based refinement." Expert Systems with Applications **121:** 188–203.

[51] Hjorth, B. (1970) "EEG analysis based on time domain properties." Electroencephalography and Clinical Neurophysiology, **29(3):** 306–310.

[52] Maalouf, M. (2011) "Logistic regression in data analysis: An overview." International Journal of Data Analysis Techniques and Strategies **3(3):** 281.

[53] "Sklearn.Linear_model.LogisticRegression." Scikit-Learn, https://scikit-learn/stable/modules/generated/sklearn.linear_model.LogisticRegression.html., Accessed 16th Sep. 2022.

[54] Bao, Feng, and Thomas Maier. (2020) "Stochastic Gradient Descent Algorithm for Stochastic Optimization in Solving Analytic Continuation Problems." Foundations of Data Science **2(1):** 1.

[55] "Sklearn.Linear_model.SGDClassifier." Scikit-Learn, https://scikit-learn/stable/modules/generated/sklearn.linear_model.SGDClassifier.html., Accessed 16th Sep. 2022.

[56] Zhang, Y. (2012) "Support vector machine classification algorithm and its application." In C. Liu, L. Wang, & A. Yang (Eds.), Information Computing and Applications **308:** 179–186. Springer Berlin Heidelberg.

[57] "Sklearn.Svm.SVC." Scikit-Learn, https://scikit-learn/stable/modules/generated/sklearn.svm.SVC.html., Accessed 16th Sep. 2022.

[58] Kamiński, B., Jakubczyk, M., & Szufel, P. (2018) "A framework for sensitivity analysis of decision trees." Central European Journal of Operations Research **26(1):** 135–159.

[59] "Decision Trees." Scikit-Learn, https://scikit-learn/stable/modules/tree.html., Accessed 16th Sep. 2022.

[60] Breiman, L. (2001) "Random Forests." Machine Learning **45(1):** 5–32.

[61] "Sklearn.Ensemble.RandomForestClassifier." Scikit-Learn, https://scikit-learn/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html., Accessed 16th Sep. 2022.

[62] Natekin, A., & Knoll, A. (2013) "Gradient boosting machines, a tutorial." Frontiers in Neurorobotics 7.

[63] "Sklearn.Ensemble.GradientBoostingClassifier." Scikit-Learn, https://scikit-learn/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html., Accessed 16th Sep. 2022.

[64] Wang, L. (2019) "Research and implementation of machine learning classifier based on knn." IOP Conference Series: Materials Science and Engineering **677(5):** 052038.

[65] "Sklearn.Neighbors.KNeighborsClassifier." Scikit-Learn, https://scikit-learn/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html., Accessed 16th Sep. 2022.

[66] Jain, A. K., Jianchang Mao, & Mohiuddin, K. M. (1996) "Artificial neural networks: A tutorial." Computer **29(3):** 31–44.

[67] "Neural Network Models (Supervised)." Scikit-Learn, https://scikit-learn/stable/modules/neural_networks_supervised.html., Accessed 16th Sep. 2022.

[68] McGinnis, Will. "Sklearn-Extensions: A Bundle of 3rd Party Extensions to Scikit-Learn." PyPI, https://github.com/wdm0006/sklearn-extensions., Accessed 16th Sep. 2022.

[69] Carrera, Ero. "Pydot: Python Interface to Graphviz's Dot." PyPI, https://github.com/pydot/pydot., Accessed 16th Sep. 2022.

[70] "Imageio 2.13.5 Documentation." https://imageio.readthedocs.io/en/stable/., Accessed 16th Sep. 2022.

[71] Alex Clark. (2019) "Python Imaging Library (Fork)." PyPI, https://python-pillow.org., Accessed 16th Sep. 2022.

[72] "Opencv-Python: Wrapper Package for OpenCV Python Bindings." PyPI, https://github.com/skvark/opencv-python., Accessed 16th Sep. 2022.

[73] "API Documentation: TensorFlow Core v2.7.0." TensorFlow, https://www.tensorflow.org/api_docs., Accessed 16th Sep. 2022.

[74] "Mne Filter MNE 0.24.1 Documentation." https://mne.tools/stable/generated/mne.filter.filter_data.html., Accessed 16th Sep. 2022.

[75] "Filtering signal with a butterworth low-pass filter and plotting the STFT of it with a Hamming window and then the Laplace transform." https://github.com/guillaume-chevalier/filtering-stft-and-laplace-transform., Accessed 16th Sep. 2022.

[76] "EDF browser Manual." https://www.teuniz.net/edfbrowser/EDFbrowser%20manual.html., Accessed 16th Sep. 2022.

[77] "EDF browser." https://www.teuniz.net/edfbrowser/., Accessed 16th Sep. 2022.

[78] Delorme, A., & Makeig, S. (2004) "EEGLAB: An open source toolbox for analysis of single-trial EEG dynamics including independent component analysis." Journal of Neuroscience Methods **134(1):** 9–21.

[79] "EEGLAB documentation." https://sccn.ucsd.edu/githubwiki/files/eeglab_plugins.pdf., Accessed 16th Sep. 2022.

[80] Sándor Beniczky et al. (2017) "Standardized computer-based organized reporting of EEG: SCORE – Second version," Clin. Neurophysiol **128(11):** 2334–2346.

[81] B. Colombet, M. Woodman, J.M. Badier, and C.G. Bénara. (2015) "AnyWave: A Cross-Platform and Modular Software for Visualizing and Processing Electrophysiological Signals." Journal of Neuroscience Methods **242:** 118–26.

[82] Laurel J. Gabard-Durnam, Adriana S. Mendez Leal, Carol L. Wilkinson and April R. Levin. (2018) "The Harvard Automated Processing Pipeline for Electroencephalography (HAPPE): Standardized Processing Software for Developmental and High-Artifact Data." Frontiers in Neuroscience **12:** 97.

[83] Jorge Bosch-Bayard, et al. (2020) "A Quantitative EEG Toolbox for the MNI Neuroinformatics Ecosystem: Normative SPM of EEG Source Spectra." Frontiers in Neuroinformatics **14:** 33.