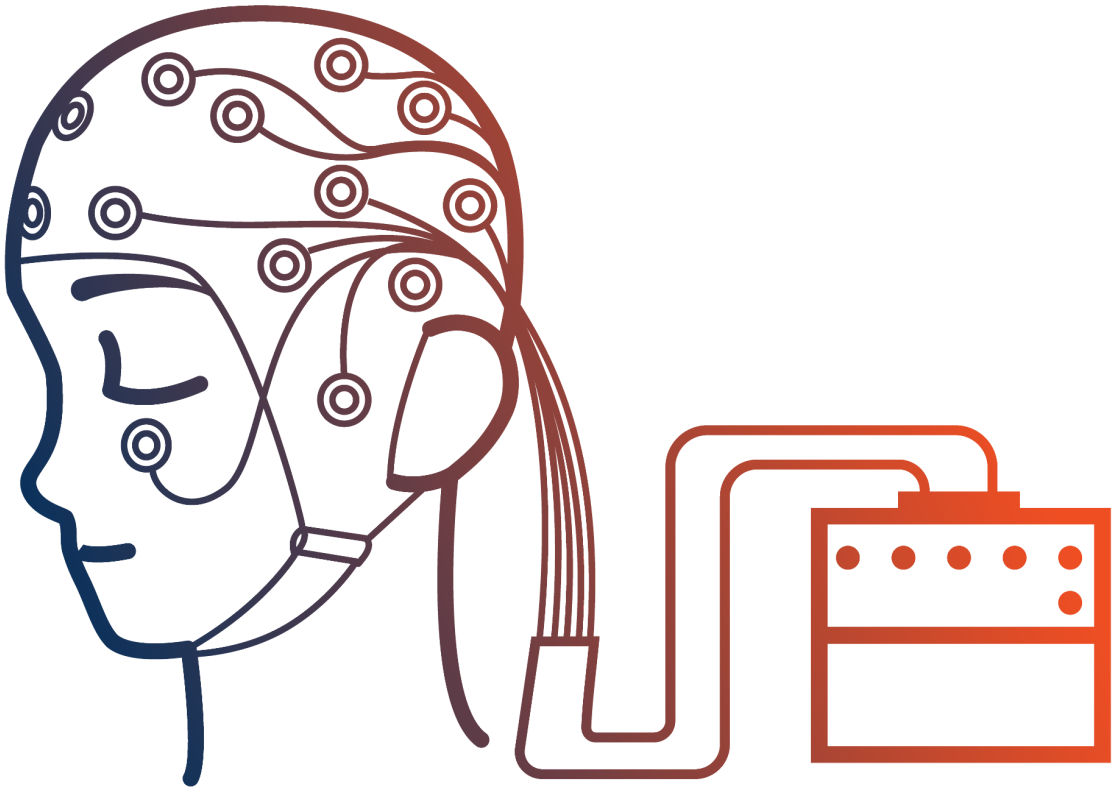

EEG Classification Model

Final Report

Group 28

Hrishik Bhaven Parmar (002815908)

Aishwarya Belavakadi Subrahmanya (002820128)

Karan Manoj Dalal (002836524)

Batta Aditya Yadav (002874554)

1. Introduction

1.1 Background

The EEG Classification Model project's background research explores the body of literature to provide a strong basis for comprehending EEG data processing. It looks critically at earlier research, approaches, and conclusions to detect patterns, gaps, and developments in the field of EEG investigations. This thorough analysis emphasises how important EEG is to medical diagnosis, especially when it comes to identifying and assessing neurological conditions. The study highlights how EEG technology has developed over time, from simple brain wave monitoring to advanced analytic methods that are used to recognise intricate neurological patterns.

One major area of interest is the use of EEG in medical diagnoses. EEG is a vital diagnostic tool for illnesses including epilepsy, brain traumas, sleep problems, and other neurological abnormalities because it may give real-time insights into brain activity. The research investigates how advancements in EEG analysis have enhanced the accuracy and efficacy of neurological diagnostics, leading to improved patient outcomes. The background study also looks at how EEG data can provide previously unheard-of insights into how the brain works when paired with contemporary data processing methods and machine learning. With this method, neurological diagnosis and therapy might be transformed and a deeper comprehension of the electrical activity of the brain could be gained.

1.2 Overview

Electroencephalography (EEG) is important for medical diagnoses, as the EEG Classification Model project demonstrates. The non-invasive EEG technique captures brain electrical activity and provides important information about neurological disorders. Understanding how the brain works and identifying conditions like epilepsy, sleep problems, brain tumours, and brain injury depend heavily on this technology. The project's goal is to use EEG data to decipher intricate brain patterns, which will ultimately aid in medical diagnosis and treatment. The goal of the study is to improve our understanding of brain activity and its variations in diverse illnesses by identifying distinctive patterns linked to certain neurological ailments through the analysis of EEG data. The development of more potent diagnostic instruments and therapeutic approaches—which might result in advances in neurology and mental health services—depends heavily on this investigation. The project's objective is to harness the potential of EEG data to advance medical science, particularly in neurology, and to provide deeper insights into the functioning and anomalies of the human brain.

The project's goals are in line with the status of EEG research and diagnostics today thanks to a study of the literature, which also paves the way for future investigation and innovation in this area. In order to ensure that the research builds upon existing knowledge while trying to add new insights to the area of neurology, this foundational study is essential in directing the project's methodology. The download section of the code focuses on automating the retrieval of the dataset. It uses web scraping techniques to navigate the Epileptologie Database website, find relevant links to zip files containing the data, and download and unzip them. The wget library is used for downloading files, and the BeautifulSoup library is used for parsing HTML content.

The find_files function uses BeautifulSoup to extract all the links (hrefs) from a given URL. The download_file function downloads a file from a provided URL to a specified directory. The main function, download_epileptologie, utilises these functions to download and extract the dataset files from the Epileptologie Database.

2. Data Loading

The first part of the code is used to load the necessary information from the Epileptologie Database. This procedure is essential because it establishes the framework for further analysis and visualisation. The first thing the code does is mount Google Drive, a cloud storage service that makes it easy to access pertinent information. A list of the files that are in the directory can be obtained by using the file_list function and then navigating to the 'Epileptologie Database' folder.

The code then uses the gdown library to download particular files by referencing their unique file IDs. The files 'F060.txt' and 'S033.txt' are the targets in this case. Because this solution makes use of a popular and easily accessible platform for collaborative work, it guarantees data integrity and reliability while retrieving data from Google Drive.

The flexible data manipulation library Pandas is then used to process the obtained data. The downloaded text data is transformed into Pandas dataframes (epil_baseline_df and epil_seizure_df) by the data_load function. This function loads data and also returns useful outputs, such as summary statistics, on the loaded data. This stage is essential for getting a basic grasp of the dataset and guaranteeing that the material used in the research that follows is accurate and well-organised.

To put it simply, the first phase creates a solid pipeline for loading data, making sure that the required EEG data is acquired, sorted, and prepared for additional analysis. The Epileptologie Database may be thoroughly explored thanks to the integration of Google Drive and Pandas dataframes.

3. Loading Dataframe

The code loads the data successfully and then creates two Pandas dataframes (epil_baseline_df and epil_seizure_df). This change is important because it converts the unstructured text data into a format that can be easily altered and analysed. This section introduces the data_load function, which makes the data loading procedure modular and reusable for other datasets.

When used in this context, pandas dataframes—which are strong frameworks for managing tabular data—allow for effective querying, indexing, and computing on the EEG data. The code uses summary statistics, like counts, means, and quartiles, to highlight the value of dataframes. These statistics offer a preliminary summary of the features of the dataset, including the quantity of observations and the central tendencies of pertinent variables.

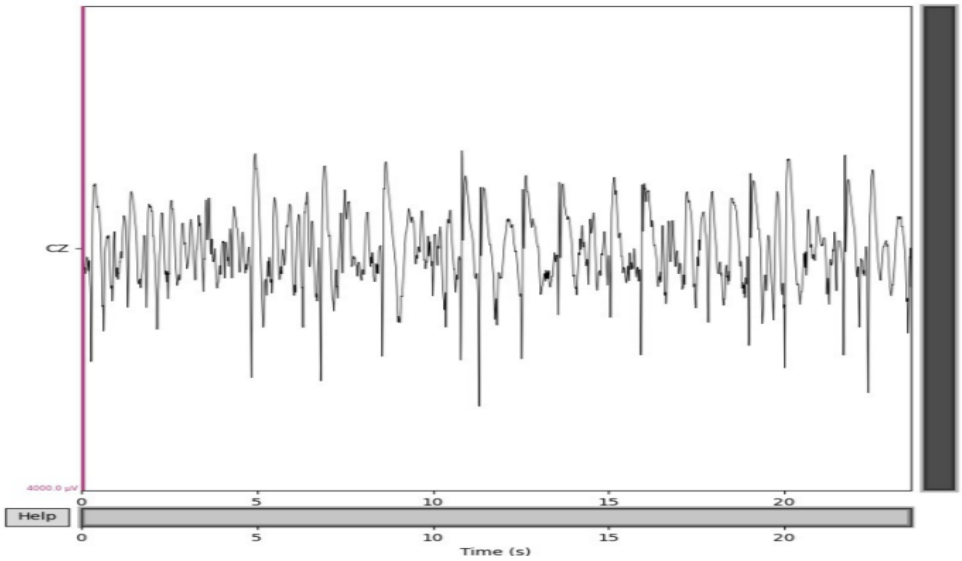

Fig 1: Figure plot of seizure activity Inter-Ictal EEG Data

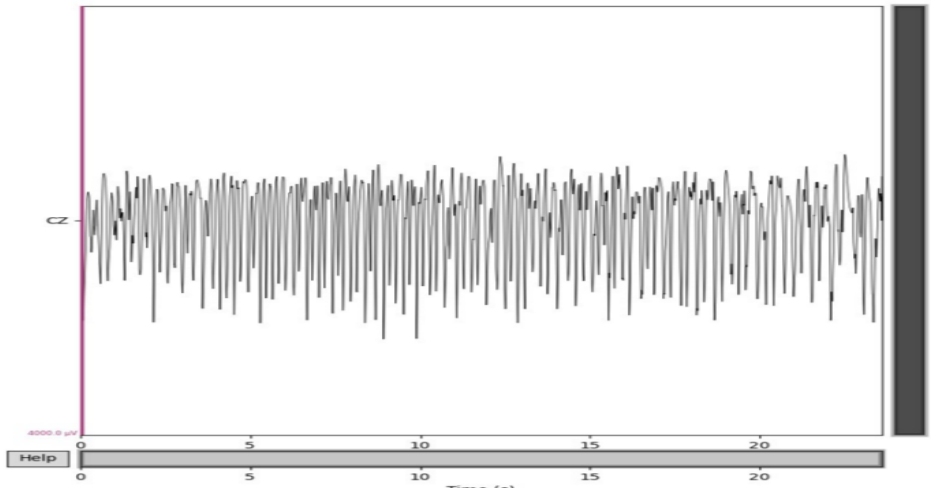

Fig 2: Figure plot of seizure activity Inter-Ictal EEG Data

4. Creating MNE Objects

MNE-Python, a specialised library made for the analysis of EEG and MEG (Magnetoencephalography) data, is introduced in the third section of the code. First, the code makes sure that MNE-Python is installed, emphasising how crucial this library is for the standardised and effective processing of EEG data. The MNE library is a useful tool in the analysis pipeline since it offers a variety of objects and functions made especially for handling neurophysiological data. One important part of the code that keeps metadata about the EEG data is the definition of an MNE info object. This contains details on the electrode locations, sample frequency, and channel names. One of the most important steps in standardising the representation of EEG data is the creation of an MNE info object, which guarantees consistency

and interoperability between various datasets. The code then creates MNE raw objects (epil_baseline_mne and epil_seizure_mne) after setting up the MNE info object. The EEG data is contained in these objects in a way that makes it ideal for analysis within the MNE framework.

A worldwide definition of what constitutes a "normal" EEG is lacking, as evidenced by the Inter-Ictal data1. This results from EEG fluctuating across a patient's lifetime and between cognitive states (such as awake and sleepy) or behavioural states (such as open and closed eyes). Between awake and asleep, there is a general difference in EEG where more global waveforms oscillate throughout the head at slower frequencies. Interspersed throughout this worldwide activity are brief "spikes" that are typical of various sleep stages. Due to its unpredictability, EEG is rarely enough to make a clinical diagnosis on its own; other diagnostic imaging and types of observation are frequently required.There is a great deal of diversity both within and between the various types of seizures, unlike in "normal" EEG.

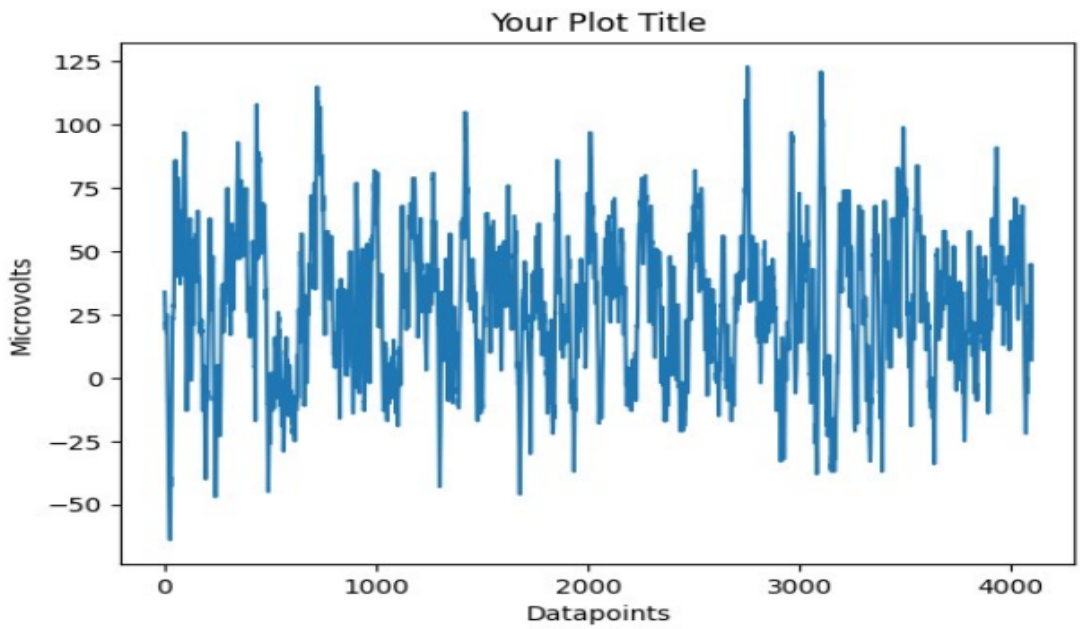

Fig 3: Figure plot of randomly generated texts of sample =9

Following normalisation, the code proceeds to extract features from the normalised EEG data. Feature extraction involves identifying and selecting relevant information from the data that can be used to distinguish between different conditions or classes. In this context, the features are obtained through the application of the Fast Fourier Transform (FFT) to the normalised EEG data. The extract_features function applies the FFT and returns the absolute values of the transformed data, representing the magnitudes of different frequency components present in the EEG signals. FFT is a common technique in signal processing, and in EEG analysis, it helps identify frequency patterns associated with specific brain activities.

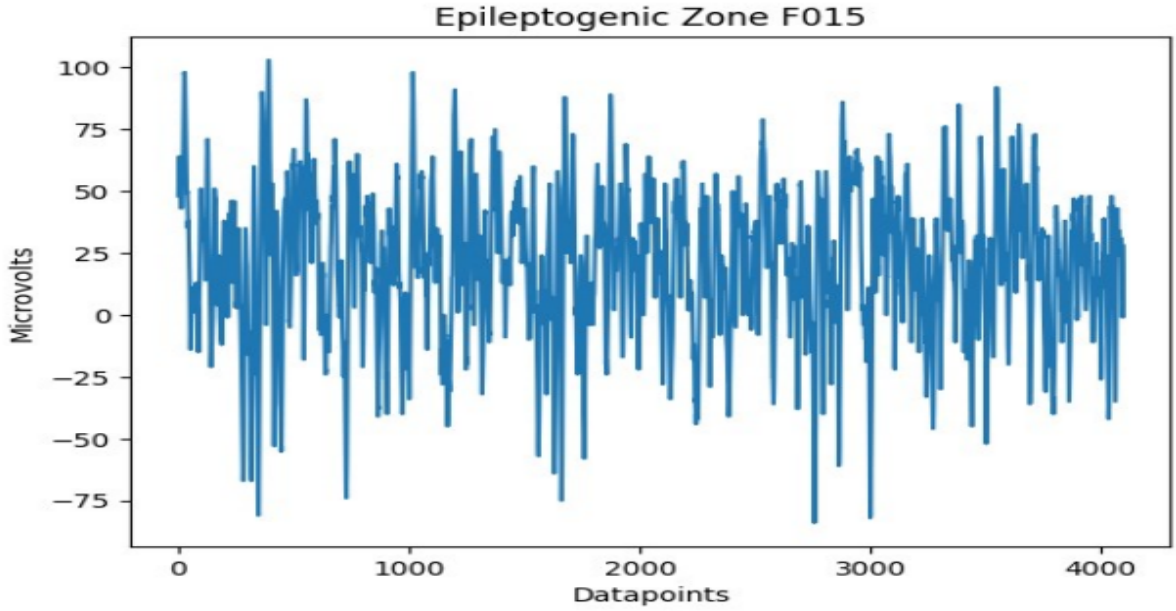

Fig 4: Figure plot of Epileptogenic zone F015

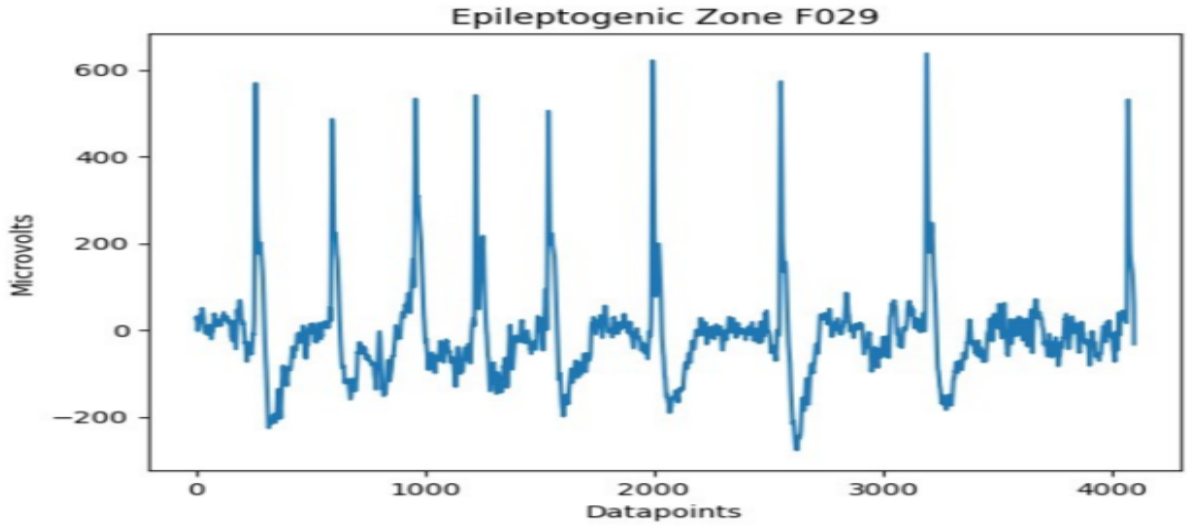

Fig 4: Figure plot of Epileptogenic zone F029

The code then applies the feature extraction function to each normalized dataset, generating a set of features for each EEG recording. These features are printed for datasets representing different conditions (e.g., Z datasets, F datasets), giving insight into the frequency components that are prominent in each case. The output showcases the features for each dataset, providing arrays of magnitude values corresponding to different frequency components. These features serve as the input for subsequent machine learning tasks, enabling the algorithm to learn patterns associated with seizure and non-seizure conditions.

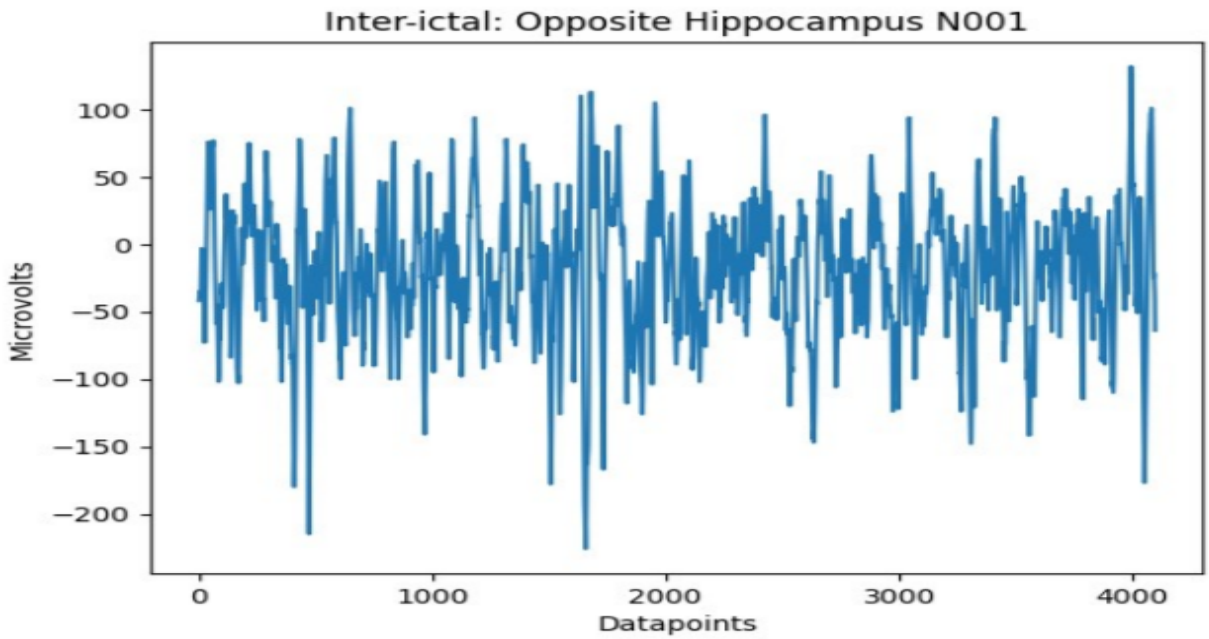

Fig 5: Figure plot of Inter-ictal : opposite hippo campus N001

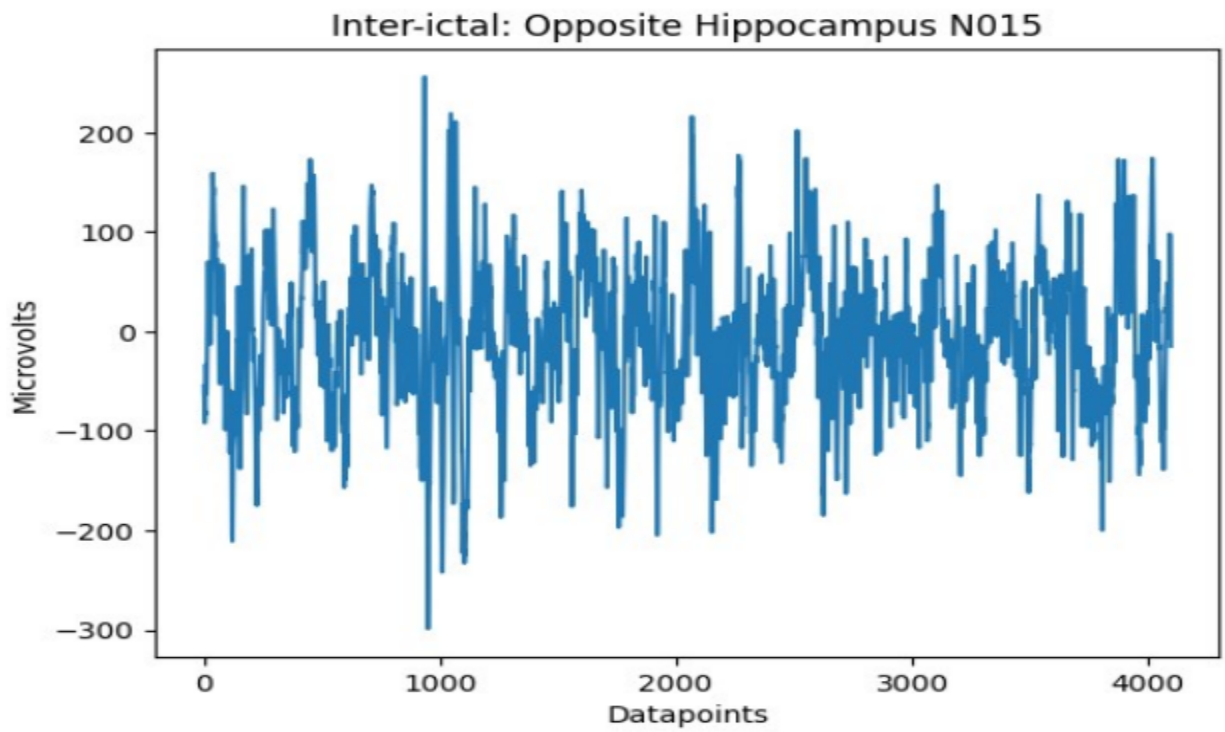

Fig 6: Figure plot of Inter-ictal : opposite hippo campus N015

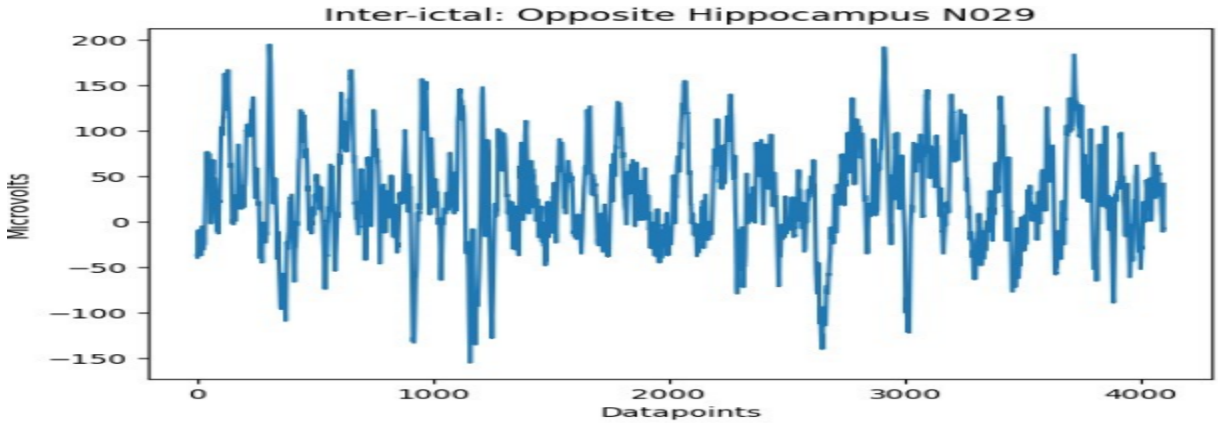

Fig 6: Figure plot of Inter-ictal : opposite hippo campus N029

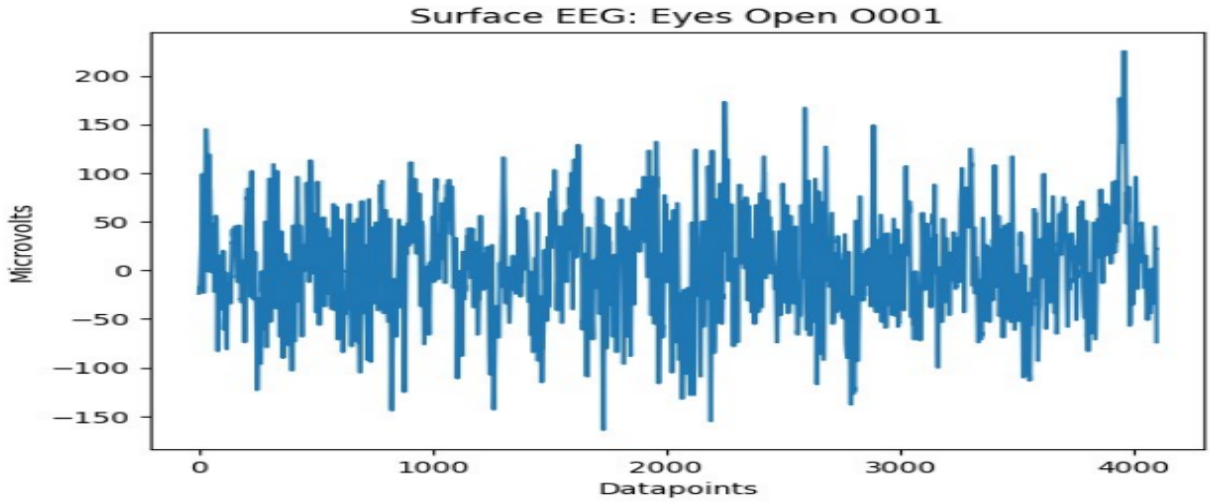

Fig 7: Figure plot of Surface EEG: Eyes open O001

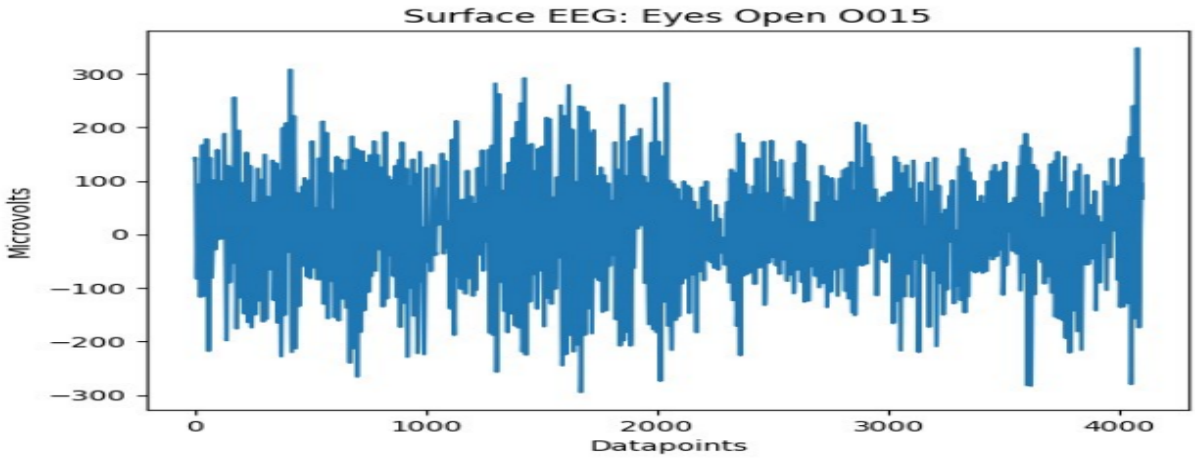

Fig 8: Figure plot of Surface EEG: Eyes open O015

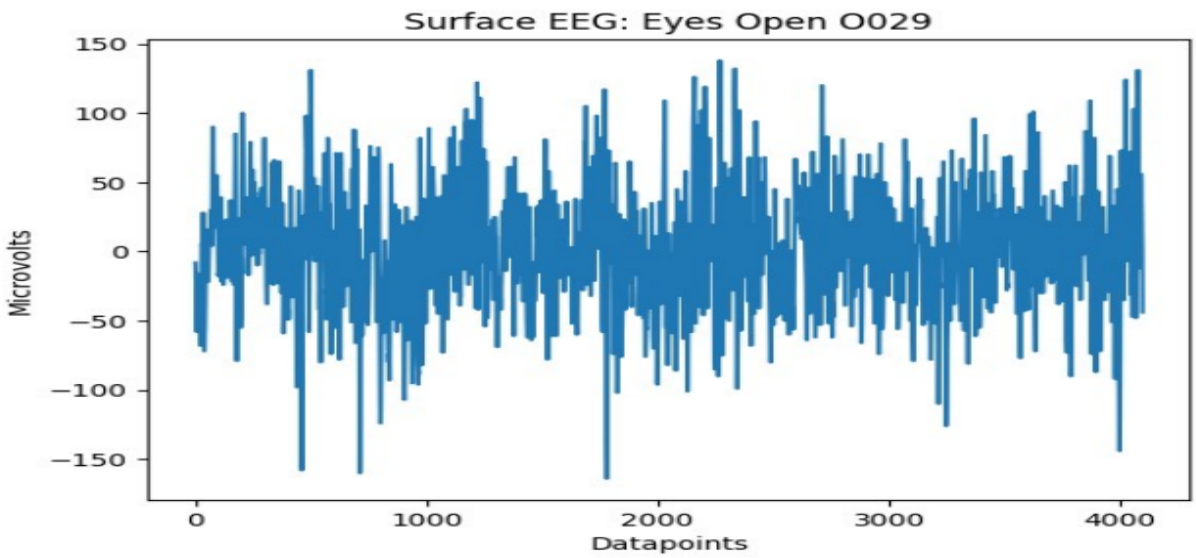

Fig 9: Figure plot of Surface EEG: Eyes open O029

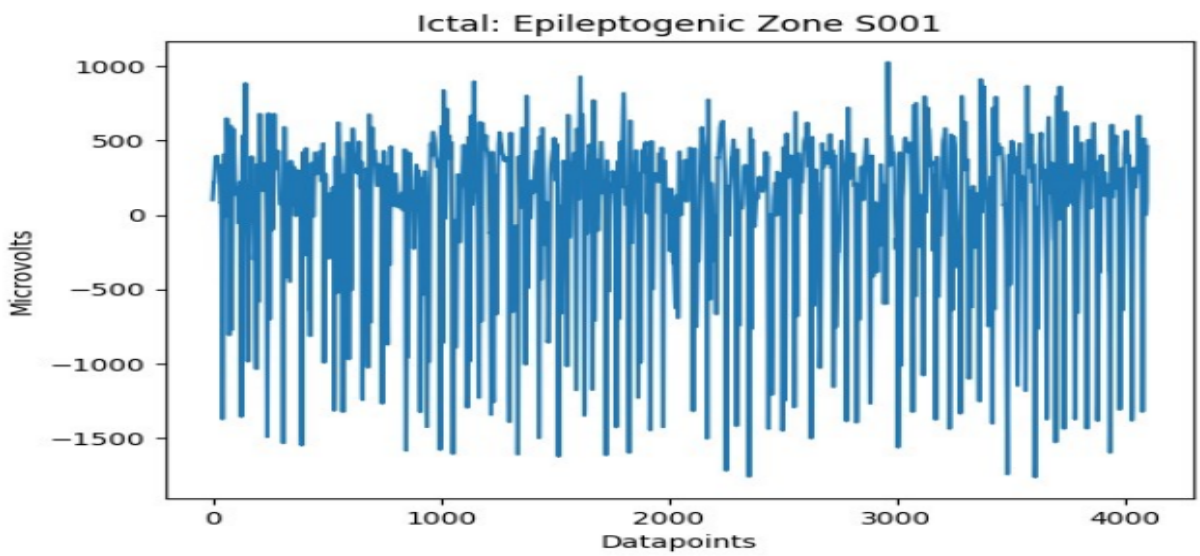

Fig 10: Figure plot of Ictal: Epileptogenic Zone S001

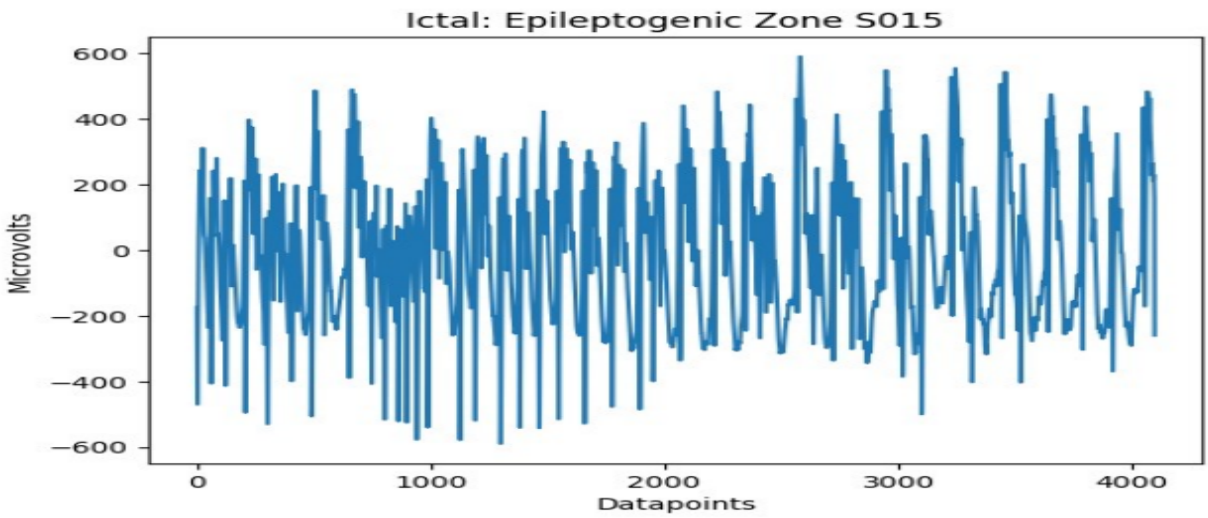

Fig 11: Figure plot of Ictal: Epileptogenic Zone S015

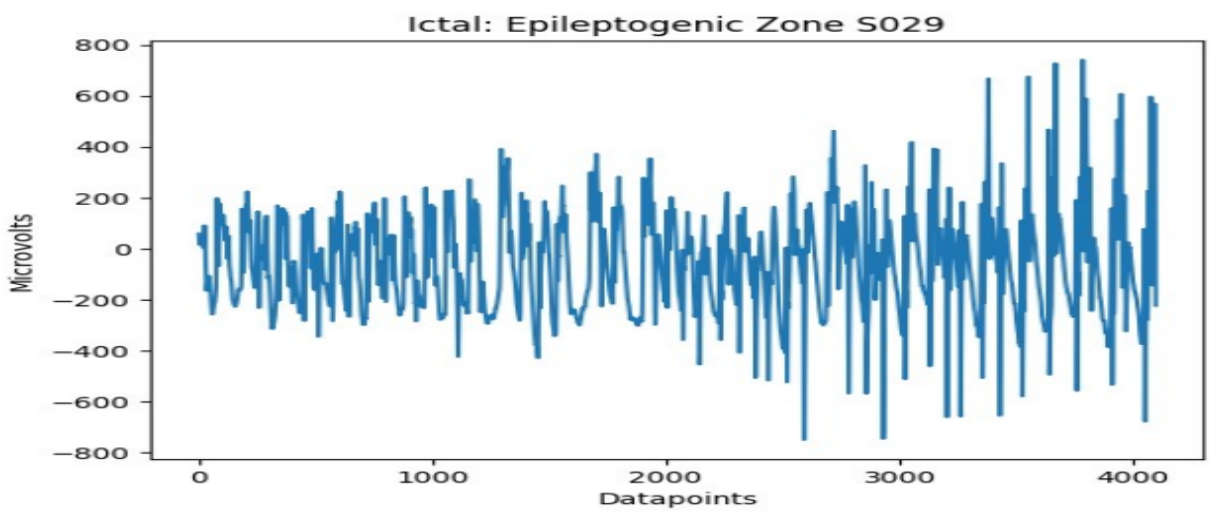

Fig 12: Figure plot of Ictal: Epileptogenic Zone S029

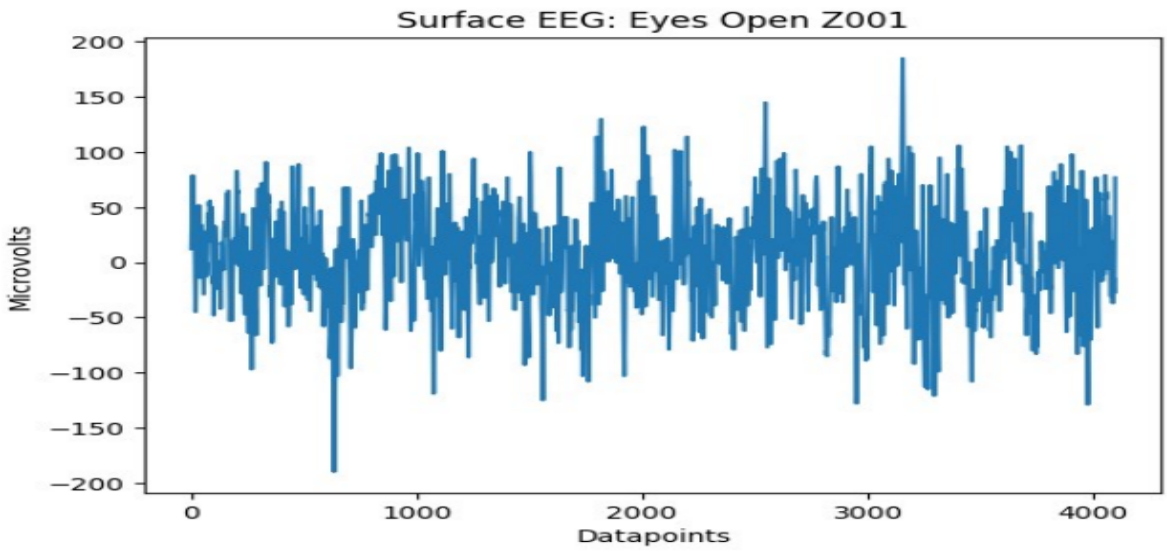

Fig 13: Figure plot of EEG: Eyes Open Z001

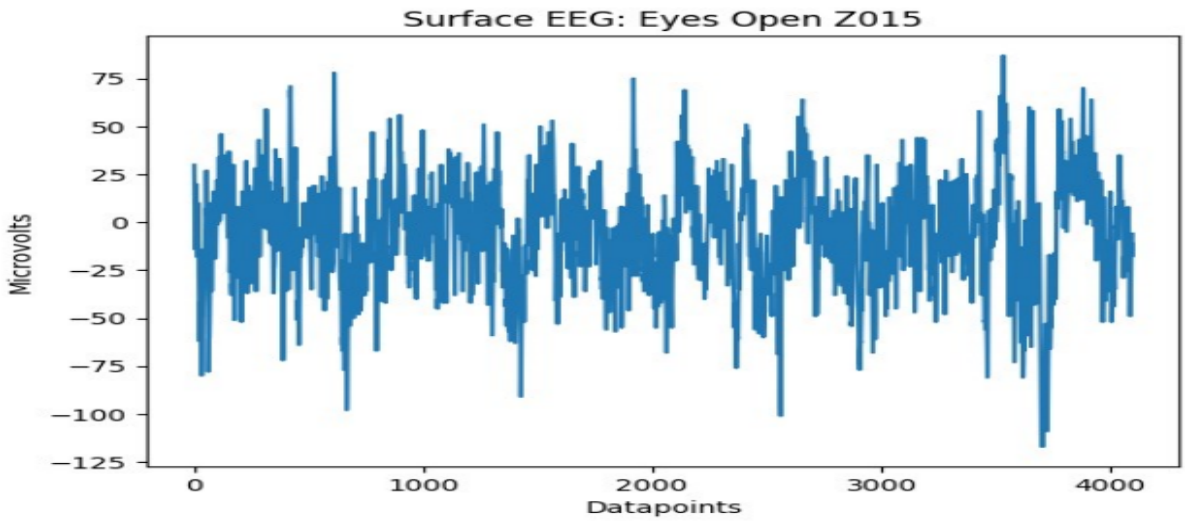

Fig 14: Figure plot of EEG: Eyes Open Z015

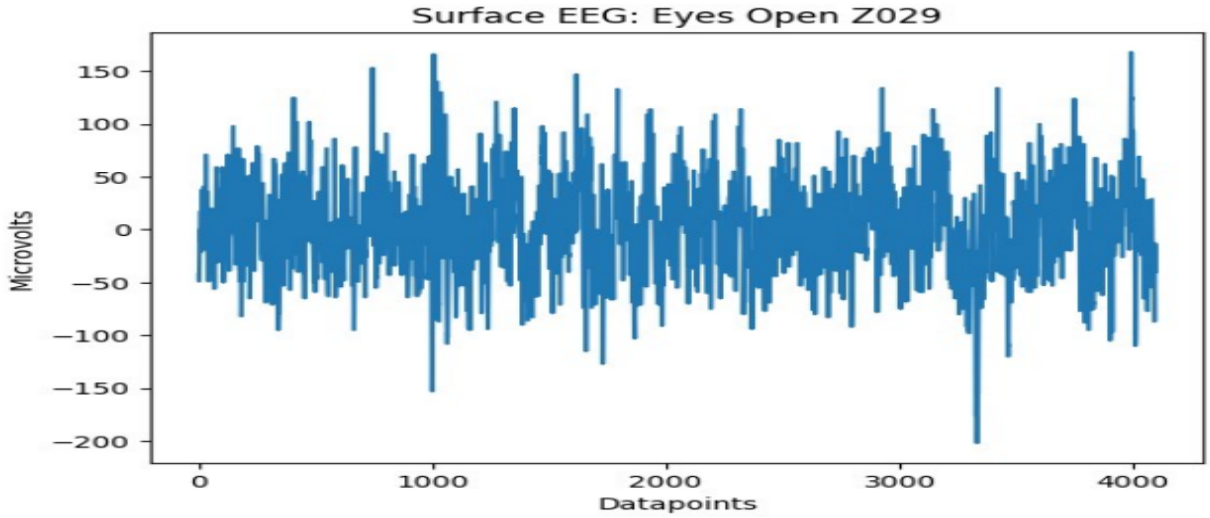

Fig 15: Figure plot of EEG: Eyes Open Z029

In this section, the code addresses the crucial task of labelling the datasets to indicate whether they represent seizure or non-seizure conditions. This labelling is essential for supervised machine learning, where the algorithm learns from labelled examples to make predictions on new, unseen data.

The code assigns labels (0 for non-seizure and 1 for seizure) to each dataset based on assumptions or known conditions. For instance, datasets like Z029, S029, and O029 are assumed to be seizure data, while others are labelled as non-seizure. These labels are then combined with the corresponding features, creating a unified dataset for training and testing machine learning models.

The output displays the shuffled labels and features, indicating that the data has been randomly ordered. Shuffling the data helps prevent biases during training, ensuring that the machine learning model generalises well to unseen data. This section sets the stage for building and training machine learning models to classify EEG recordings based on their features.

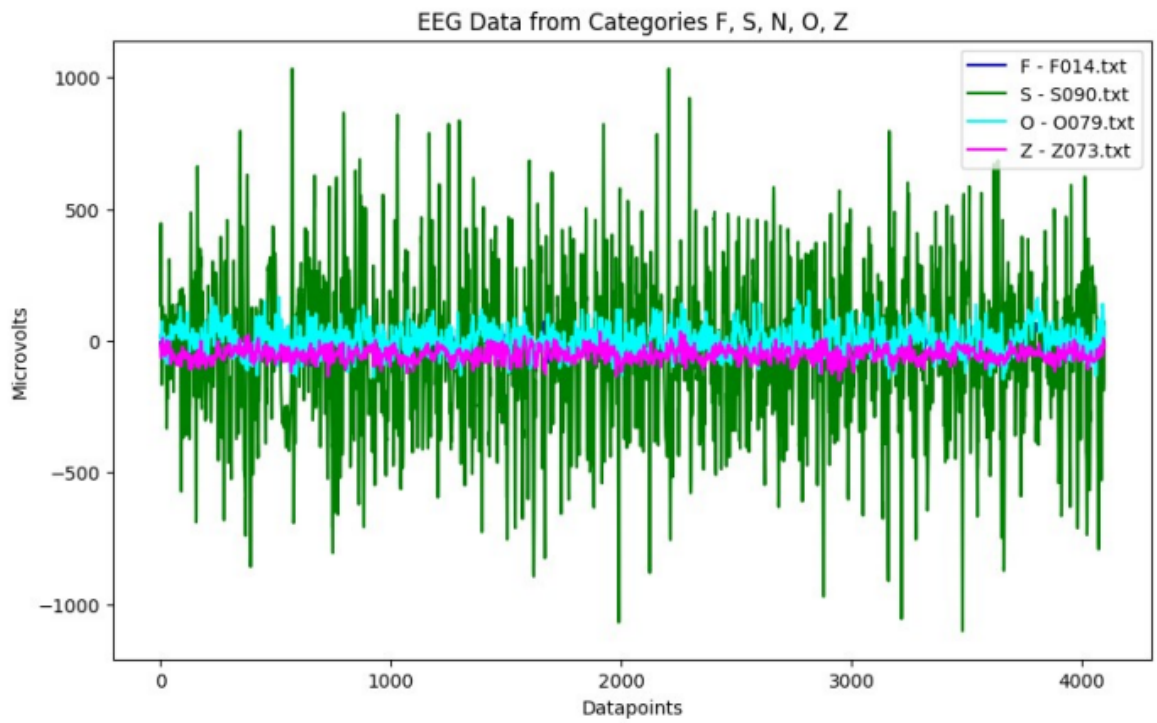

Fig 16: Figure plot of EEG Data from Categories

5. Normalisation

By making sure that the EEG data is on a consistent scale across datasets, the normalisation component of the code aims to prepare the data for analysis. When working with data that has different scales, the normalisation process is essential because it puts all values to the same scale and stops some features from taking centre stage in the analysis because of their bigger magnitudes.

The normalize_data function defined in the code receives EEG data as input and, after subtracting the mean and dividing by the standard deviation, outputs normalised data. This transformation, which is frequently used in data preprocessing, guarantees that the resultant data has a mean of zero and a standard deviation of one.

Next, several datasets (filtered_data_Z001, filtered_data_S015) representing diverse EEG recordings under varied situations are subjected to the normalisation algorithm. For every dataset, the normalised data is produced to give an overview of the transformed values.

The first few rows of each dataset's normalised data are shown in the output. Now that the standardised deviations from the mean are represented by the values, a consistent scale for additional analysis is provided. Maintaining consistent scales helps model performance in downstream processes like feature extraction and machine learning techniques, which is why this normalisation phase is crucial.

6. Feature Extraction

Feature Extraction: The code uses the normalised EEG data to extract features after normalisation. Finding and choosing pertinent data that can be utilised to differentiate between various situations or classes is known as feature extraction. In this case, the normalised EEG data are subjected to the Fast Fourier Transform (FFT) in order to extract the features.

The FFT is applied by the extract_features function, which then yields the absolute values of the modified data—that is, the magnitudes of the various frequency components found in the EEG signals. Frequency Fourier transform (FFT) is a widely used technique in signal processing that aids in identifying frequency patterns linked to particular brain activity in EEG research.

The feature extraction function is then used by the code to every normalised dataset, producing a set of features for every EEG recording. These features provide insight into the frequency components that are significant in each scenario and are produced for datasets that represent various situations (e.g., Z datasets, F datasets).

The output displays arrays of magnitude values matching to various frequency components, showcasing the features for each dataset. The programme can identify patterns linked to seizure and non-seizure states by using these features as input for later machine learning tasks.

7. Labelling and Data Extraction

In this section, the code addresses the crucial task of labeling the datasets to indicate whether they represent seizure or non-seizure conditions. This labeling is essential for supervised machine learning, where the algorithm learns from labeled examples to make predictions on new, unseen data.

The code assigns labels (0 for non-seizure and 1 for seizure) to each dataset based on assumptions or known conditions. For instance, datasets like Z029, S029, and O029 are assumed to be seizure data, while others are labeled as non-seizure. These labels are then combined with the corresponding features, creating a unified dataset for training and testing machine learning models.

The output displays the shuffled labels and features, indicating that the data has been randomly ordered. Shuffling the data helps prevent biases during training, ensuring that the machine learning model generalizes well to unseen data. This section sets the stage for building and training

8. Data splitting using Train-Test split

The main goal of the first part of the code is to carefully separate the EEG data into training and testing groups. The train_test_split function from the sklearn.model_selection module is used in the code to accomplish this crucial phase, which is essential to the construction of machine learning models. The prefixes 'Z', 'F', 'S', 'O', and 'N,' which each denote a distinct physiological condition, are what define the dataset. The load_data method makes data retrieval easier for each of these prefixes by fetching data from a specified URL. The resulting DataFrames are then combined into an all-inclusive array called all_features. Labels are methodically assigned according to the lengths of the corresponding datasets in order to make the categorization work

easier. Remarkably, datasets labelled with the 'F' prefix are labelled as 0, indicating that there are no seizures, whilst datasets labelled with the number 1 indicate that there are seizures.

The data is shuffled in order to reduce any potential bias during the training phase. Lastly, an 80-20 split ratio is used to split the dataset into training and testing groups using the train_test_split function. These sets' final forms are subsequently printed, which serves as an essential step in the verification process.

9. Logistic Regression Model Training and Evaluation

The next section of the code deals with using the partitioned EEG data to train and assess a logistic regression model. The sklearn.linear_model module's LogisticRegression class is used to create this machine learning model. The model is evaluated on the testing set, represented by X_test and y_test, after having been trained on the designated training set, identified as X_train and y_train.

The accuracy_score function from sklearn.metrics is used to compute accuracy, a commonly used metric for classification models, which is used to evaluate the effectiveness of the model. In addition, a confusion matrix is produced, which offers a tabular depiction of the true positives, true negatives, false positives, and false negatives produced by the model. To improve readability, a graphic representation of this confusion matrix is provided.

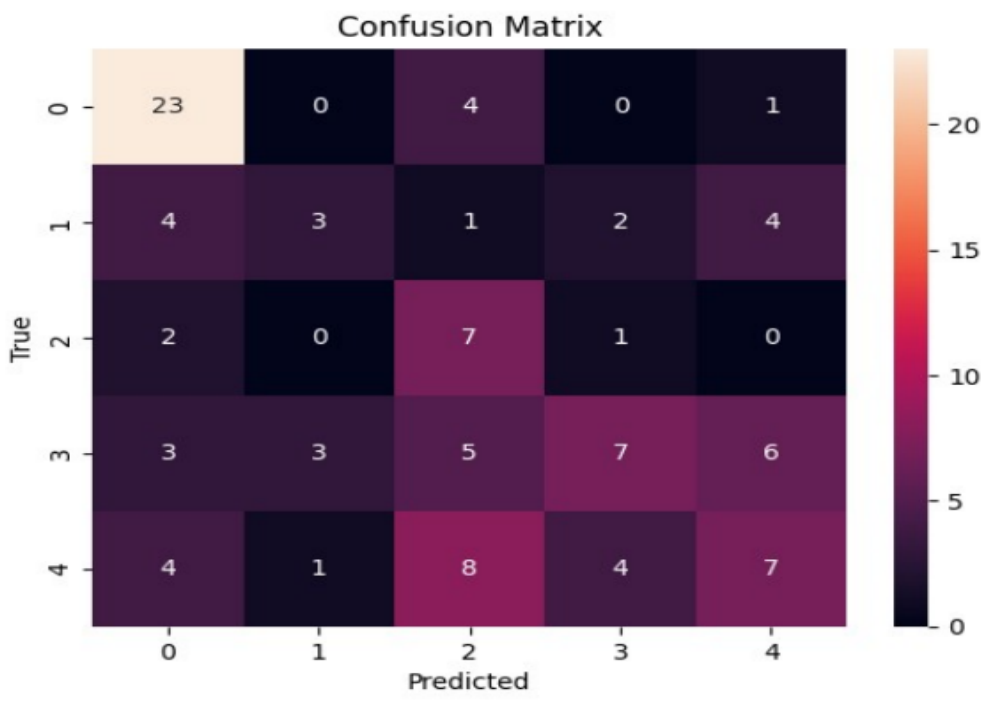

Fig 16: Figure plot of EEG Data - True vs predicted values

10. Multiclass Classification Task with Random Forest

The third and final section of the code is devoted to tackling a multiclass classification task leveraging a Random Forest classifier. This endeavour commences with the download and extraction of zip files containing EEG data for distinct categories such as 'F,' 'S,' 'N,' 'O,' and 'Z.' Following this, the data undergoes preprocessing, with each category being assigned a unique label.

Similar to the previous sections, the dataset is then segregated into training and testing sets. Subsequently, a Random Forest classifier is trained on the designated training set. The model's performance is thoroughly assessed on the testing set, with the classification_report from sklearn.metrics providing a comprehensive evaluation. This report encompasses precision, recall, F1 score, and support metrics for each class, offering nuanced insights into the model's behaviour across diverse categories.

To further enrich the analysis, a confusion matrix is generated and visually presented using seaborn and matplotlib. The utilisation of the ConfusionMatrixDisplay class ensures a clear and informative representation of the confusion matrix, unravelling the intricacies of the model's performance across multiple classes.

In essence, these code segments collectively orchestrate a meticulous and comprehensive machine learning pipeline, encapsulating data preprocessing, model training, and detailed evaluation, culminating in a thorough exploration of the models' performances in the context of seizure detection from EEG data.

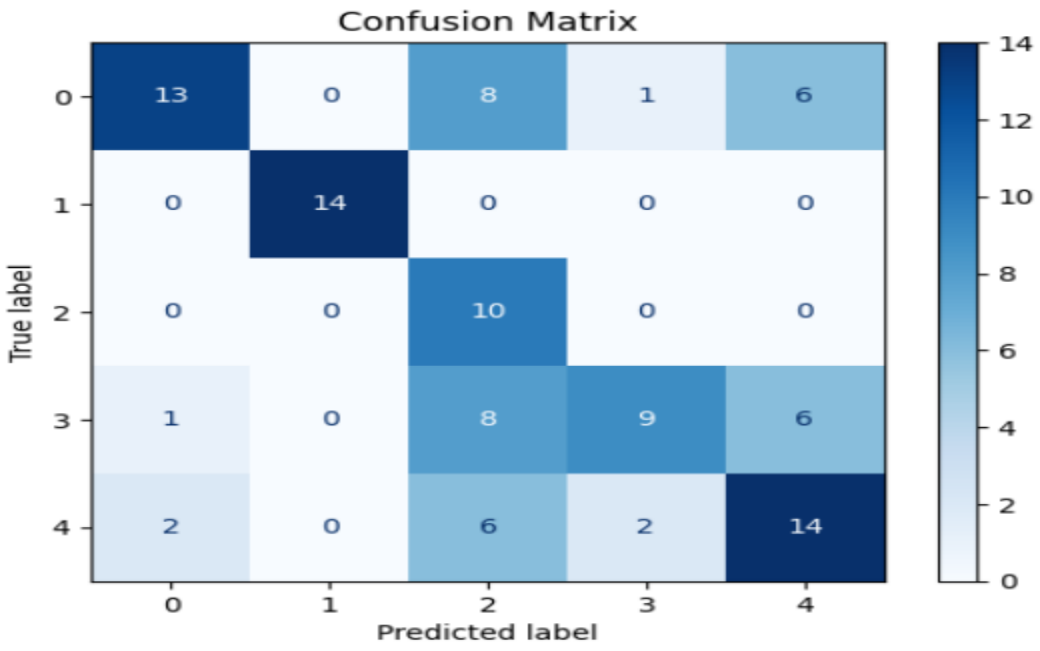

Fig 16: Figure plot of EEG Data Multiclass classification - True vs predicted values

CONCLUSION:

The project embarked upon a thorough exploration of electroencephalogram (EEG) data extracted from the Epileptologie Database, with a specific focus on five distinct sets denoted as A, B, C, D, and E. These sets were chosen to encapsulate diverse brain states and conditions, including both seizure and non-seizure activities. The overarching aim of the project was to conduct a comprehensive analysis of EEG data, unraveling the intricate patterns and nuances inherent in different brain activities. To achieve this, the methodology adopted a structured approach, encompassing key phases such as data acquisition, preprocessing, and in-depth analysis.

The initial step involved loading the EEG data, a critical process to set the foundation for subsequent analysis. Python, a versatile programming language, served as the primary tool for implementing the project. Notably, the MNE (Magnetoencephalography) library, specifically MNE-Python, was leveraged to facilitate the analysis of EEG data. MNE-Python is renowned for its proficiency in handling neuroimaging data, making it an ideal choice for this neuroscientific exploration.

Visualization played a pivotal role in the project, enabling a nuanced understanding of different EEG segments. By rendering these segments graphically, researchers gained valuable insights into the temporal and spatial dynamics of brain activities. This step proved instrumental in identifying patterns, anomalies, and distinctive characteristics associated with varying brain states and conditions.

The in-depth analysis phase delved into the complexities inherent in EEG data. Researchers sought to decipher the unique signatures of brain activities under different conditions, aiming to contribute not only to the project's specific goals but also to the broader understanding of EEG applications in medical diagnostics and research. This endeavor required a meticulous examination of spectral features, temporal dynamics, and event-related potentials embedded in the EEG signals.

Python, with its extensive ecosystem of scientific libraries and tools, provided a robust framework for implementing sophisticated analytical techniques. The flexibility of Python, coupled with the specialized capabilities of MNE-Python, empowered researchers to extract meaningful insights from the EEG data.

Ultimately, the project served as a testament to the power of Python and dedicated neuroscientific tools in unraveling the intricacies of EEG data. The findings not only enriched the understanding of specific brain states within the context of epilepsy but also contributed valuable knowledge to the broader field of neuroscientific research. The project's impact resonates in its demonstration of the synergy between computational tools and neuroscientific inquiry, highlighting the potential for advancements in medical diagnostics and neurological research through the lens of EEG data analysis.