Indian Institute of Science Education and Research Kolkata

Project Report

MA4207: Classification Of Time Series Data Using Machine Learning And Network Analysis

Abhay Kshirsagar (19MS172)
Parth Bibekar (19MS161)
G Sushanth (19MS134)

Department of Mathematics and Statistics

Supervised by
Prof. Dr. Koel Das

April 24, 2023

Table of Contents

Chapter 1

Introduction

1.1 Problem statement

The Project for MA4207 involves network analysis and pattern classification of two given Datasets. The project is divided into two components: a) Machine Learning and b) Network.

1. The data provided is divided into training and test sets and suffers from small sample size problem. You will design a pattern classification technique of your choice (not covered in class) that will classify the data with high accuracy. Achievable accuracy for given datasets are provided below. Your classifier's performance should ideally be close to the benchmark performance provided for respective datasets. You will also try to find and visualize the discriminatory regions (features) that contribute towards the classification.

2. The given data is from two different classes. Take training data and test data of the two classes together and construct the network. Now do the following: (I) Analyse the network to retrieve meaningful information. (II) Cluster the unsupervised network nodes and correlate with the original classes to see how closely the clustered groups match the original class.

1.2 About Dataset

The dataset one (Table 1.1) is from the BCI III competition and is provided by the University of Tubingen, Germany. The dataset includes imagined movements of the left small finger or tongue, and the data were recorded using an 8x8 ECoG platinum electrode grid placed on the contra-lateral motor cortex with a sampling rate of 1000Hz. The recording intervals started 0.5 seconds after the visual cue ended and lasted for 3 seconds, resulting in a series length of 3000. There are 64 dimensions in total. The training data (278 cases) and test data (100 cases) were recorded from the same subject with the same task, but on two different days with about a week in between. The dataset is balanced between the two labels viz. Finger and Tongue.

Train Size	Test Size	Length (in steps)	Number of Classes	Number of Dimensions
278 Finger: 139 Tongue: 139	100 Finger: 50 Tongue: 50	3000	2	64

Table 1.1: Summary of EEG Dataset

The dataset two (Table 1.2), in question involves predicting major earthquakes in Northern California based on the most recent readings in the surrounding area. The data is taken from the Northern California Earthquake Data Center, with each data point representing an averaged reading for one hour, spanning from Dec 1st, 1967 to 2003. A major event is defined as any reading over 5 on the Richter scale and positive cases are defined as instances where a major event is not preceded by another major event for at least 512 hours. Negative cases are constructed by considering readings below 4 and are preceded by at least 20 non-zero readings in the previous 512 hours. The dataset consists of 368 negative cases and 93 positive cases, with no overlap in time. However, the dataset is highly imbalanced, with significantly more negative cases than positive cases, which can affect the accuracy of any classifier that predicts more 0 than 1.

Train Size	Test Size	Length (in steps)	Number of Classes	Number of Dimensions
322 '0': 264 '1': 58	139 '0':104 '1': 35	512	2	1

Table 1.2: Summary of Earthquake Dataset

1.3 Feature Selection

The EEG Dataset has a really high dimensions i.e not only does it have 64 features but each has 3000 timepoints of data. This it is important to reduce this dimensionality to improve the results. For this dataset we employed a transformation commonly used for signal analysis called Power Spectral Density (PSD) Analysis.

Power Spectral Density (PSD) is a mathematical tool used to describe the distribution of power into different frequency components of a signal. It is a common analysis technique in signal processing and is used to identify the frequencies present in a signal and their respective strengths. PSD is often used to study the properties of stochastic processes and to analyze signals in fields such as electrical engineering, physics, and neuroscience. The PSD provides a way to visualize the energy distribution across different frequencies of a signal, which can be useful for understanding the characteristics of the signal and developing effective signal processing algorithms[1].

We will compare the average PSD of each channel of both the labels and select the ones which show visible difference between the two labels this will provide us the channels which are important for classification. For calculating PSD we will be using the Python library Matplotlib, the library contains an inbuilt function called `matplotlib.psd()` for calculating the Power Spectral Density and average it across all the Data Points using Numpy (Python library).

From the PSD comparison (Figure 1.1) we choose the channels `[29, 30, 31, 38, 39, 40, 46]` as these show most visible difference between the two labels.

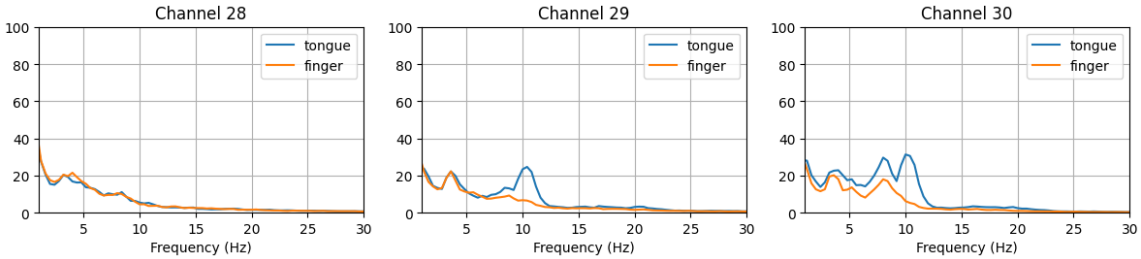


Figure 1.1: Comparison of PSD of channels 28-30. As we can see channel 28 is pretty homogeneous for both the labels but channel 29th and 30th show visible change.

For the Earthquake Dataset since we have one dimensional data we will be using it as it is. We tried multiple methods like checking the trends withing the data obtained by Autocorrelation functions however we noticed using such features reduced the accuracy. Since the dataset is very imbalanced we have used different class weights for '0' (0.2) and '1' (0.8). Further, to take care of the imbalance we also changed the classification threshold from 0.5 to 0.2.

Chapter 2

Classification Task

2.1 Learning algorithms

2.1.1 Decision Tree

Decision tree classifiers are a type of machine learning algorithm that is commonly used for classification tasks. They build a tree-like model of decisions and their possible consequences, where each node represents a feature, each branch represents a possible value for that feature, and each leaf node represents a class label[2].

To build a decision tree, the algorithm recursively partitions the data into subsets based on the values of the input features, selecting the feature that provides the best split at each level. This process continues until a stopping criterion is reached, such as a maximum depth or minimum number of samples in a leaf node.

During prediction, the algorithm follows the path down the tree based on the input feature values, and outputs the corresponding class label at the leaf node. Decision tree classifiers are easy to interpret and visualize, making them useful for gaining insights into the decision-making process of the model. However, they can be prone to overfitting if the tree is too deep, and may not perform well on datasets with many irrelevant features.

2.1.2 Random Forest

Random Forest Classifier is an ensemble learning method that combines multiple decision trees to create a more robust and accurate model. It randomly selects subsets of features and data points to build multiple decision trees, and combines their predictions to make a final classification decision[3].

During training, the algorithm generates a set of decision trees by selecting random subsets of features and data points for each tree. The trees are constructed using the same decision tree algorithm as a single decision tree classifier. During prediction, the algorithm uses each tree in the forest to make a prediction, and the final prediction is the majority vote from all the trees.

Random Forest Classifier is effective in handling high-dimensional data with a large number of features, as it can capture complex nonlinear relationships between features and labels. It is also less prone to overfitting than a single decision tree, as the randomization of feature and data point selection can reduce the correlation between trees.

2.1.3 Support Vector Machine

Support Vector Machines (SVM) is a popular machine learning algorithm for classification and regression problems. SVM works by constructing a hyperplane in a high-dimensional space that optimally separates different classes of data. During training, SVM finds the hyperplane with the maximum margin, which is the distance between the hyperplane and the closest data points from each class. The hyperplane is chosen in such a way that it maximizes the margin and minimizes the classification error. SVM is effective in handling complex datasets with high-dimensional feature spaces, as it can create non-linear decision boundaries using kernel functions. SVM is also robust to overfitting, and can handle datasets with outliers by ignoring them during training[4].

However, SVM can be sensitive to the choice of kernel function and its parameters, and can be computationally expensive for large datasets. Here we employed a Non-Linear Function like RBF (Radial Basis Funcitons) which uses them for transform the data to find a better hyperplane than just normal Linear SVM[5].

2.1.4 Light Gradient Boosting Model

LightGBM builds an ensemble of decision trees, where each tree is built in a gradient boosting fashion to minimize the loss function. The model is trained iteratively by adding new trees that fit the residuals of the previous trees. One of the key features of LightGBM is its ability to handle categorical features efficiently. It uses a technique called the "Gradient-based One-Side Sampling" (GOSS) algorithm, which selects only the most informative samples from the dataset to speed up the training process. Another feature of LightGBM is its ability to handle imbalanced datasets, as it allows users to assign different weights to different classes to balance the dataset during training, this was a helpful feature of this model as the Earthquake data is imbalanced [6].

2.1.5 Long Short-term Memory Networks

Long Short-Term Memory (LSTM), is a type of neural network architecture commonly used for processing sequential data. In contrast to other neural network architectures, LSTMs can capture long-term dependencies in the data by using memory cells that can store information over an extended period. This makes LSTMs particularly useful for tasks such as classification, where the input is a sequence of observations over time and the output is a label or category. LSTMs achieve this by using input and output gates to regulate the flow of information in and out of the memory cells. Overall, LSTM is a powerful tool for processing sequential data and has been shown to be effective in a wide range of applications[7].

Network Architecture

The LSTM architecture consists of memory cells, input gates, forget gates, and output gates. Each of these components plays a crucial role in regulating the flow of information in and out of the memory cells, allowing the network to learn and remember long-term dependencies in the input sequence.

At each time step, the LSTM receives an input vector, x_t, which is processed by the input gate, i_t. The input gate decides which information to keep and which to discard, by applying a sigmoid activation function to the input vector and a weight matrix. The output of the sigmoid is then

multiplied by a candidate activation vector, $\tilde{C}_t$, which contains the proposed new values for the memory cell state.

The forget gate, f_t, is responsible for deciding which information to forget from the previous memory cell state, C_{t-1}. The forget gate receives the same input as the input gate and produces a value between 0 and 1 for each element of the previous cell state. This value is then used to selectively erase information from the previous cell state that is no longer relevant.

The memory cell state, C_t, is updated by a combination of the input and forget gates. The updated cell state is then used to compute the output of the network, h_t. The output gate, o_t, decides which information from the updated memory cell state to output by using a sigmoid activation function and a weight matrix to control the output[8, 9].

In our model we use a LSTM based classifier on the ECoG data. We use a sequential model with 64 LSTM blocks followed by a neural network with the sigmoid activation function in the output layer. Fig 2.1 shows the model architecture we used, $\hat{y}$ denotes the predicted label of the model.

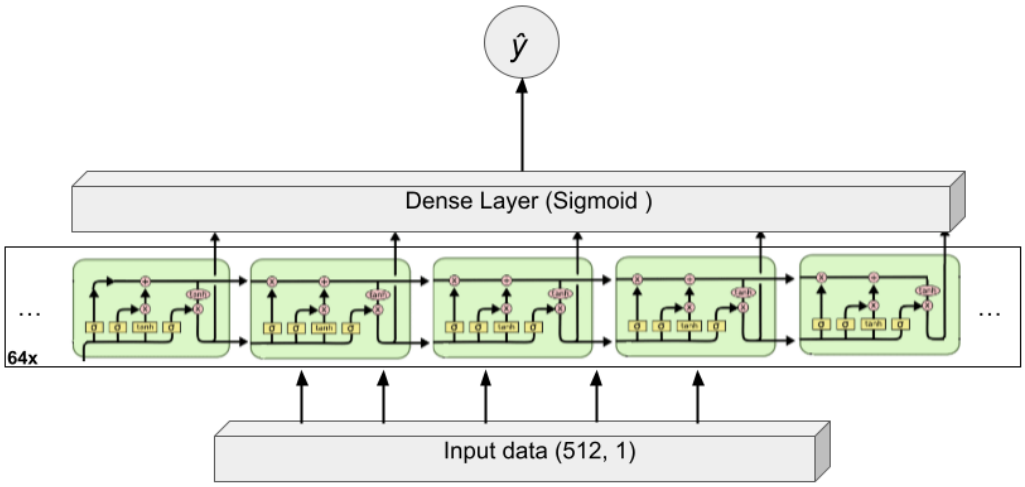


Figure 2.1: Model architecture used in this study. The data is input in the LSTM layer which is connected to a dense layer with the sigmoid activation function which outputs the label of the input data.

Binary Cross Entropy Loss Function

Binary cross-entropy is a commonly used loss function in machine learning, particularly in binary classification problems. It measures the difference between two probability distributions: the true distribution and the predicted distribution[10].

The formula for binary cross-entropy takes the form of:

$$Loss = \frac{1}{N}\sum_{i=1}^{N} -(y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)) \tag{2.1}$$

where y_i is the true binary label (0 or 1), and p_i is the predicted probability of the positive class (i.e., the class with label 1).

The function penalizes the model more heavily for making incorrect predictions, and is used as an objective function to minimize during training. It is effective for optimizing models that output a probability for a binary outcome.

Adam Optimizer

We have used the Adam optimizer as the optimization algoeithm to train our model. The Adam optimizer maintains an exponential moving average of the gradients and the squared gradients. It uses the moving averages to update the learning rate adaptively for each weight parameter. This helps to achieve faster convergence during training by adjusting the learning rate based on the gradients[11].

Adam optimizer minizes the loss function as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\Big[\frac{\delta L}{\delta \omega_t}\Big] v_t \tag{2.2}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\Big[\frac{\delta L}{\delta \omega_t}\Big]^2 \tag{2.3}$$

Where, m_t is the first moment estimate at time t v_t is the second moment estimate at time t β_1 and β_2 are the exponential decay rates for the first and second moments, respectively. From eq. 2.2 and eq. 2.3 we get:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.4}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.5}$$

$\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected estimates of the first and second moments, respectively.

2.2 Results

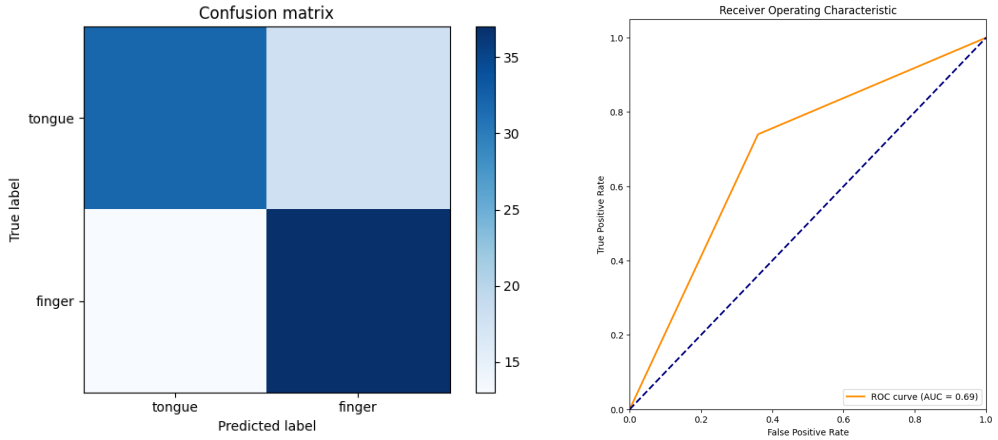
2.2.1 ECoG Dataset

For the ECoG dataset, we use an ensemble model with a decision tree, a random forest, a SVM with the Radial Basis Function and a LightGBM. The decision tree and the random forest used 100 estimators and the depth of the tree was set at 10. Data from the channels 29, 30, 31, 38, 39, 40 and 46 was used as the input for our ensemble model. Finally, the outputs of all the methods were used in a hard voting criterion and the output was predicted as either 0 for 'tongue' and 1 for 'finger'. To evaluate the performance of our method we used the accuracy, precision, recall, and the area under the receiver operating characteristic curve (ROC) as the metrics.

Fig 2.2 shows the confusion matrix and the ROC curve of the model predictions and table 2.1 shows the scores of all the evaluation metrics on a independent balanced test set with 100 examples.

Evaluation metric	Score
Accuracy	0.690
Recall	0.740
Precision	0.673
ROC-AUC	0.690

Table 2.1: Evaluation metrics on the ECoG test dataset



(a) (b)

Figure 2.2: (a) The confusion matrix (b) ROC curve with an AUC of 0.69

2.2.2 Earthquake Dataset

For the Earthquake dataset, we use a LSTM model with the architecture as shown in fig 2.1. The outputs of the model predicted 0 for 'no earthquake' and 1 for 'earthquake'. We evaluated the performance of out model a test set with 139 examples, with 104 '0s' and 35 '1s'. Since the test set is very imbalanced towards the negative cases, to evaluate the performance of our method we not only used the accuracy, precision, recall, and the area under the receiver operating characteristic curve (ROC) but also used the F-1 score which the is harmonic mean of the precision and recall and the area under the precision-recall curve (PR) as the evalution metrics[12].

Fig 2.3 shows the confusion matrix and fig 2.4 shows the ROC curve and the PR curve of the model predictions. Table 2.2 shows the scores of all the evaluation metrics on a independent balanced test set with 100 examples.

Evaluation metric	Score
Accuracy	0.770
Recall	0.629
Precision	0.407
F-1	0.494
ROC-AUC	0.730
PR-AUC	0.500

Table 2.2: Evaluation metrics on the earthquake test dataset

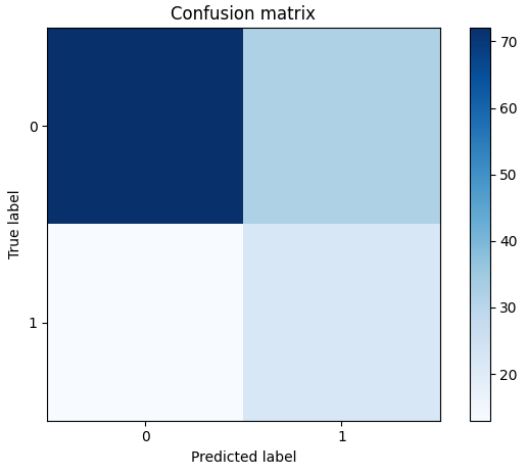


Figure 2.3: Confusion matrix

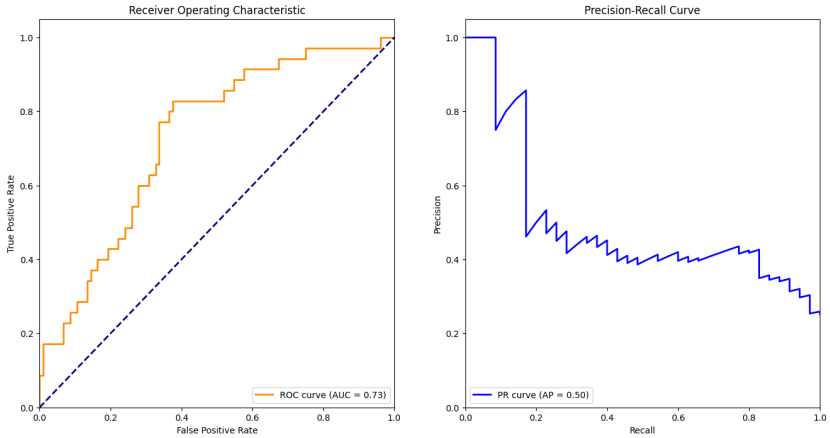


Figure 2.4: ROC curve with an AUC of of 0.73 and the PR curve with an AUC of 0.50

Chapter 3

Network Analysis

Firstly to form a network we need to define what our nodes are going to be. And then to form edges we need a similarity metric (which would be a function of the feature vector of the node) between nodes. Finally, we need to set a threshold for the similarity value above which a node is formed and below which it is not.

3.1 Similarity metric

3.1.1 EEG Dataset

For EEG Dataset each of data points (described by a 64x3000 array) are considered to be nodes. We used Pearson Correlation Coefficients [13] between each dataset for each of the 64 electrodes separately and averaged all the 64, 378x378 matrices to get a matrix shown in Figure 3.1. From this we choose the median of the values as our threshold and formed an edge if the correlation is higher than this threshold.

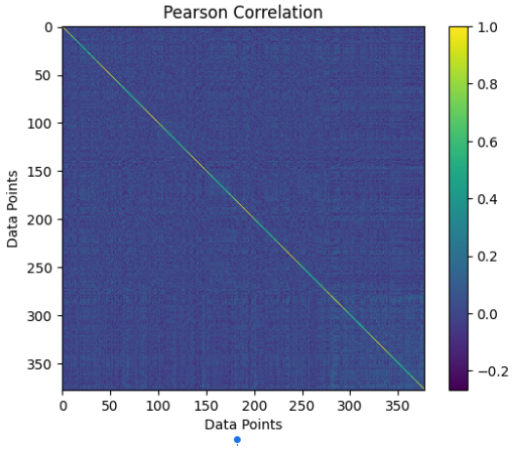


Figure 3.1: Pearson Correlation Coefficients for EEG Dataset

$$r_{ij}^{n} = \frac{\sum_{k=1}^{T}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{T}(x_k - \bar{x})^2}\sqrt{\sum_{k=1}^{T}(y_k - \bar{y})^2}} \tag{3.1}$$

where r is the Pearson Correlation Coefficient between the EEG Dataset for the i^{th} and j^{th} data point and the n^{th} feature (the superscript is not the power), these matrices were calculated separately for each feature and averaged for all 64 features.

$$R_{ij} = \frac{1}{64}\sum_{n=1}^{64} r_{ij}^{n} \tag{3.2}$$

We then used a similarity value around the sum of mean and std. dev of our correlation values as our threshold. Which comes to around 0.09 after finetuning it around that range. We make an edge if the correlation value is above this value.

3.1.2 Earthquake Dataset

For the Earthquake dataset again each data point formed our nodes and each data point had only one feature vector (1 x 512) describing it. We used Dynamic time wrapping (DTW) to obtain our similarities since the time series data is very sparsely distributed. DTW is a technique used to find similarity between time series signals where there might be difference in speed or timing between the signals. The goal of DTW is to find an optimal alignment between the two series by warping them in time, stretching or compressing one of them as necessary. The basic idea is to compare each point in one series to all points in the other series and find the optimal path that minimizes the sum of the distances between corresponding points[14]. In DTW we used the Euclidean norm as our distance function.

We then used a similarity value around the mean value to be our threshold. We had to fine-tune the value since the sparseness of the matrix undergoes a second-order phase transition around the mean and we do not want the sparseness to be too high.

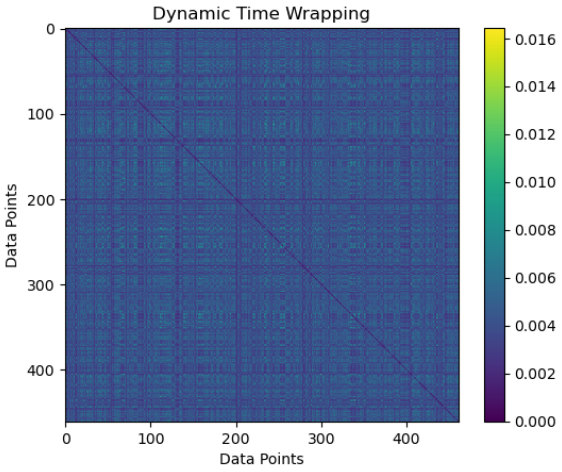


Figure 3.2: DTW Correlation Coefficients for Earthequake Dataset

3.2 Network

3.2.1 EEG Dataset

We obtain the network as given in Fig:3.3. The two classes are indicated by the colors red and blue.

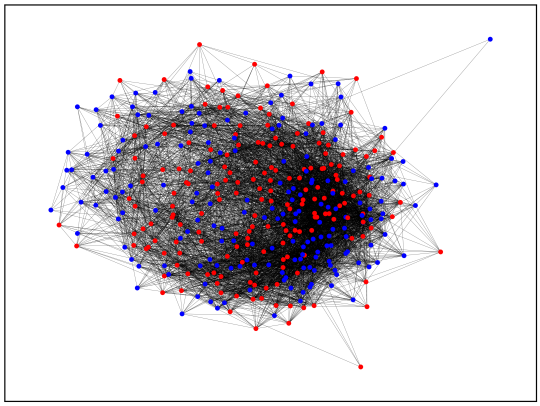


Figure 3.3: The obtained network for the EEG Datas

We note that the graph obtained above is fully connected. The average clustering coefficient of the network is 0.303. The network has a transitivity of 0.325. The network density (Sparseness) of the

network is 0.1035. The obtained diameter of the network is 4.

We also find the degree centrality measure of the graph. We find the mean centrality to be 39 and the maximum to be 103.

Parameter	Value
Average Clustering Coefficient	0.303
Transitivity	0.325
Density	0.1035
Diameter (Large component)	4

Table 3.1: Summary of the Earthquake Dataset Network

3.2.2 Earthquake Dataset

We obtain the network as given in Fig:3.4. The two classes are indicated by the colors red and blue.

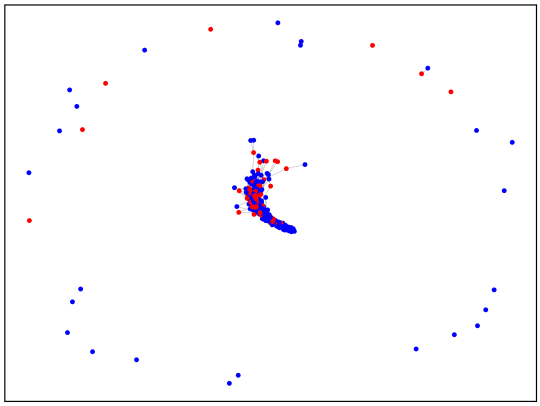


Figure 3.4: The obtained network for the Earthquake Dataset

We note that the graph obtained above is disconnected with one large component and many distributed nodes. Now we will first get the large component of the above network formed (Fig:3.5. We note that the large component contains 430 of the 461 nodes.

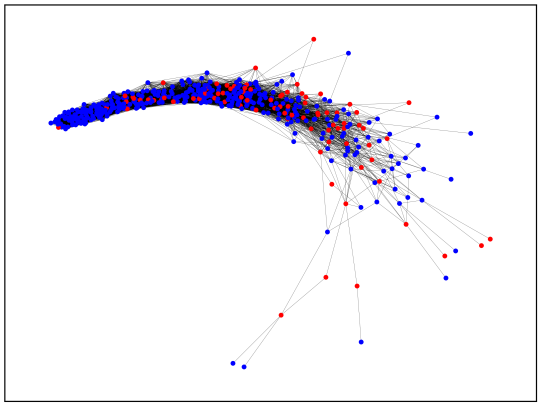


Figure 3.5: Large component of the Earthquake Dataset network

Firstly The average clustering coefficient of the network is 0.43. The network has a transitivity of 0.563. The network density (Sparseness) of the network is 0.1089. Since the network is disconnected we will find the diameter of the large component. The obtained diameter of the large component of the network is 8.

We also find the degree centrality measure of the graph. We find the mean centrality to be 50 and the maximum to be 114.

Parameter	Value
Average Clustering Coefficient	0.43
Transitivity	0.563
Density	0.1089
Diameter (Large component)	8

Table 3.2: Summary of the Earthquake Dataset Network

3.3 Clustering

Now to cluster the nodes in the network we will use one of the community clustering methods, specifically the Greedy Modularity maximization algorithm. This algorithm partitions the network into densely connected groups or modules such that the modularity score is maximized. Modularity score indicated the quality of the partitions. The Modularity score (Q) is given as follows where m is the total number of edges, A is the adjacency matrix, γ is the resolution parameter, k_i is the degree of i th node, δ is the kroneker-delta function, and c_i is the i th community[15].

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \gamma \frac{k_i k_j}{2m} \right) \delta_{c_i c_j} \tag{3.3}$$

3.3.1 EEG Dataset

We obtain the following network after clustering given in Fig:3.6. The two obtained communities are indicated by the colors yellow and green. Comparing the obtained labels to the true labels we obtain a classification accuracy of 52.64%.

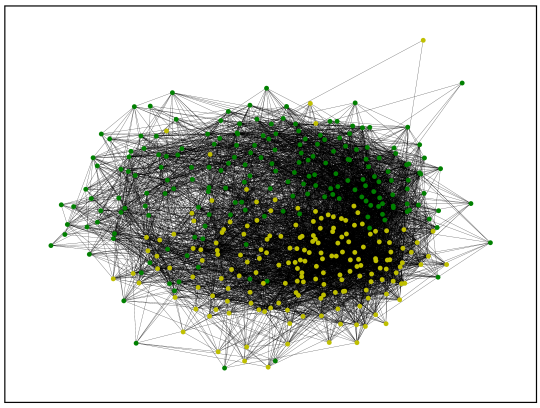


Figure 3.6: EEG Dataset Network after clustering

3.3.2 Earthquake Dataset

We obtain the following network after clustering given in Fig:3.7. The two obtained communities are indicated by the colors yellow and green. Comparing the obtained labels to the true labels we obtain a classification accuracy of 55.81%.

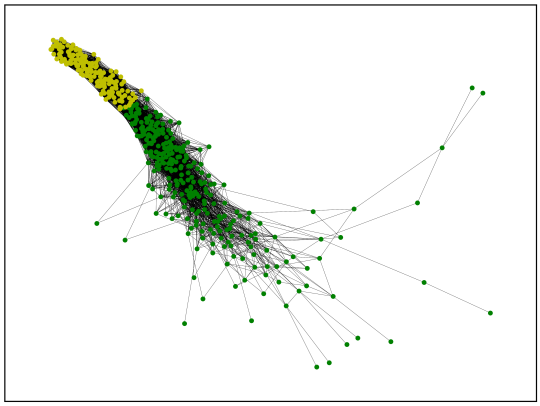


Figure 3.7: Earthquake Dataset Network after clustering

Chapter 4

Conclusion

Overall, this project presented a comprehensive analysis of two distinct datasets, focusing on classification and networking tasks. The results obtained from the ECoG dataset showcased the effectiveness of selecting top seven features based on power spectral density and using ensemble methods for classification. On the other hand, the earthquake dataset demonstrated that training a LSTM-based model on the entire dataset while addressing class imbalance achieved strong performance, evaluated using the F1 score. In addition, for the networking task, we applied different similarity measures, including the Pearson correlation coefficient and dynamic time wrapping, to calculate similarity scores between data points and utilized the greedy modularity maximization algorithm for community clustering.

While the results presented in this project are promising, there is still room for improvement in both the classification and networking tasks performed on the ECoG and the earthquake datasets. One area where further improvement can be made is through better feature engineering. While selecting top seven features based on power spectral density and addressing class imbalance were effective strategies, exploring additional features or engineering existing ones could lead to even better results. Additionally, hyperparameter tuning can also play a critical role in improving model performance. By systematically optimizing model hyperparameters, we can further enhance the accuracy and generalizability of the models. Therefore, further work should focus on developing better feature engineering techniques and applying more sophisticated hyperparameter tuning methods to improve the results presented in this project.

References

[1] S. Di Matteo, N. M. Viall, and L. Kepko. Power spectral density background estimate and signal detection via the multitaper method. *Journal of Geophysical Research: Space Physics*, 126(2), February 2021.

[2] S. B. Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261–283, June 2011.

[3] Aakash Parmar, Rakesh Katariya, and Vatsal Patel. A review on random forest: An ensemble classifier. In *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018*, pages 758–763. Springer International Publishing, December 2018.

[4] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, September 2020.

[5] Karl Thurnhofer-Hemsi, Ezequiel López-Rubio, Miguel A. Molina-Cabello, and Kayvan Najarian. Radial basis function kernel optimization for support vector machine classifiers, 2020.

[6] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[8] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8):5929–5955, May 2020.

[9] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:132306, March 2020.

[10] .Usha Ruby Dr.A. Binary cross entropy with deep learning technique for image classification. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(4):5393–5397, August 2020.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[12] Peter Flach and Meelis Kull. Precision-recall-gain curves: Pr analysis done right. *Advances in neural information processing systems*, 28, 2015.

[13] Per Ahlgren, Bo Jarneving, and Ronald Rousseau. Requirements for a cocitation similarity measure, with special reference to pearson's correlation coefficient. *Journal of the American Society for Information Science and Technology*, 54(6):550–560, 2003.

[14] Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin Heidelberg, 2007.

[15] Nicolas Dugué and Anthony Perez. Direction matters in complex networks: A theoretical and applied study for greedy modularity optimization. *Physica A: Statistical Mechanics and its Applications*, 603:127798, October 2022.