Emotion Recognition using Graph Analysis and k-Means Clustering

Under the guidance of

Dr. Jajati Keshari Sahoo

Asst. Professor and HOD

Department of Mathematics

BITS Pilani, K K Birla Goa Campus

As a part of

Study Oriented Project (MATH F266)

by

Mohd Khizir Siddiqui   Mahak Kothari

2018A8PS0439G   2018A7PS0232G

Special thanks to **Mayur Jartarkar** sir (Cognitive Neuroscience Lab, BITS Pilani, Goa) for his continued support and guidance related to the neuroscience technicalities of the project, and **Dr. Swati Agarwal** ma’am (Dept CSIS, BITS Pilani, Goa) for her guidance related to the graph and network analysis and helping us bring the project to completion.

Table of Contents

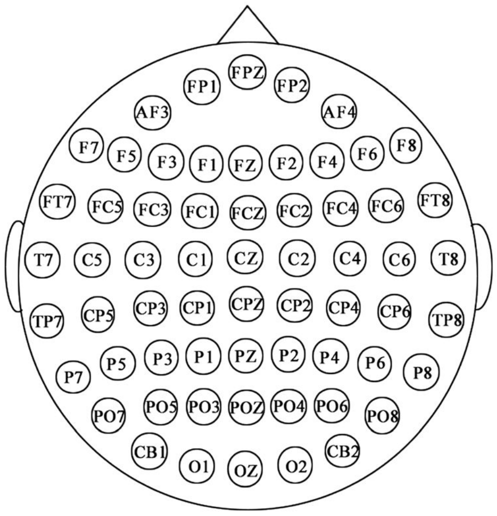
Introduction

Consumer neuroscience is the application of neuroscientific methods to the understanding of consumer behavior. By using methods such as EEG, fMRI, eye tracking, GSR, and more, consumer neuroscientists aim to better understand how consumers make decisions, what are their emotions that can lead to purchases. Consumer neuroscience widely uses EEG in order to detect brain-based signals that can relay information about purchase intentions. Real-time assessment and regulation of emotion will improve people’s lives and make them better.

To design an emotion recognition system using EEG signals, effective feature extraction, and optimal classification are the main challenges. EEG signals are non-linear, nonstationary, buried into various sources of noise, and are random in nature. Thus, handling and extracting meaningful features from EEG signals plays a crucial role in the effective designing of an emotion recognition system.

We were having 14 subjects and each of them was shown 18 advertisements videos and then the signals were recorded. We have considered 33 EEG channels for our experiment which are located at various locations of the brain equally divided into both the hemispheres. The 33rd channel chosen as the reference channel, and we have used average referencing.

Various feature extraction methods have been proposed for emotion recognition from EEG signals but we have used a Graph-based (unsupervised learning) approach in our project. We have converted the signals recorded from EEG signals into the adjacency matrices and then further converted them into the form suitable for social network analysis. To overcome the curse of dimensionality we have used dimensionality reduction techniques like Principal component analysis (PCA) and t-SNE.



Position of all the electrodes

Dataset Information and Collection

Collection

The project is in collaboration with Cognitive Neuroscience Laboratory, BITS, Pilani Goa. The data is collected from the stimulus-response of every subject on the 18 advertisements obtained from the internet. The subject uses a 33 channel headset to collect responses. A sample screenshot of a single channel for the first thousand milliseconds is shown in the figure:

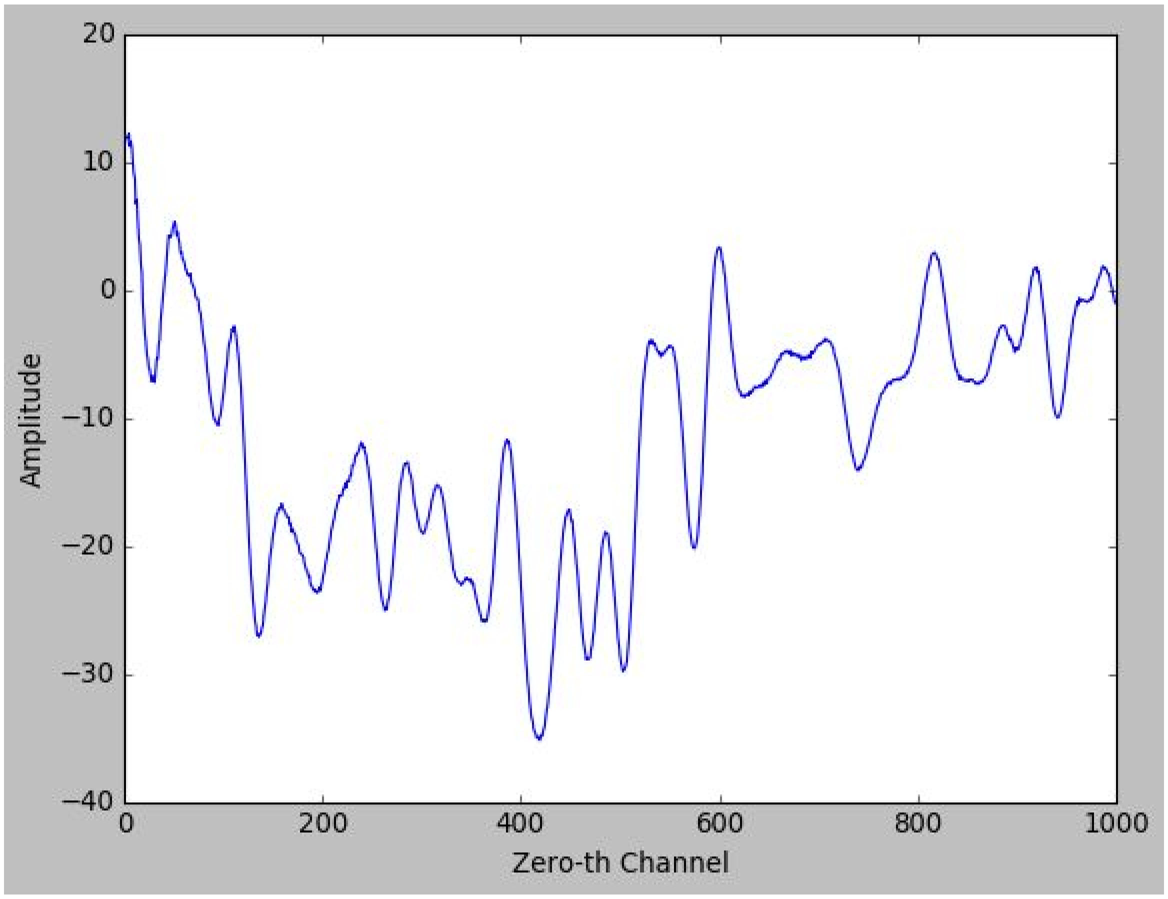


Figure 1: ECA-Anish-October31_Step1_9_1.mat 1st 1000ms

Similar data streams are available for all 18 advertisements and different subjects and 33 channels (32 channel + 1 ref channel). The data is prone to various artifacts and hence need to be denoised. Various preprocessing filters are tested and chosen accordingly.

Information

A standard EEG data contains records of electrical activity from the scalp of the brain. It measures the voltage fluctuations in the ionic activities within the neurons. The EEG data displays 5 types of broad categories of brain waves: alpha, beta, gamma, delta, and theta. At every point of your life, these waves are emitted and can be recorded using non-invasive technologies like EEG.

Band	Delta	Theta	Alpha	Beta	Gamma
Frequency Range (Hz)	< 4 Hz	4 to 7 Hz	8 to 15 Hz	16 to 31 Hz	> 32 Hz

Different parts of waves show different activities as a stimulus to the emotions of the subject. A recent study in Radboud University revealed how variations in the alpha band of EEG data can be associated with the blunders one makes

A common problem with the EEG data collected is the artifacts. EEG data is highly sensitive to the noise - eye blinks, ocular muscle activities, or could also be due to poor grounding of the electrodes, The artifacts can be removed using various signal processing or machine learning data dimension reduction techniques. We have primarily employed digital signal processing techniques to remove the artifacts, as mentioned in the afterward preprocessing section I.

Preprocessing I

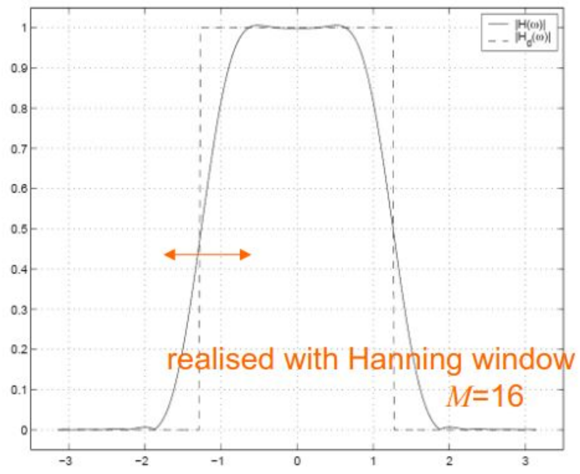
The collected data needs to be smoothed as an attempt to crop out the artifacts. Digital signal processing has various filter designs to help it out.

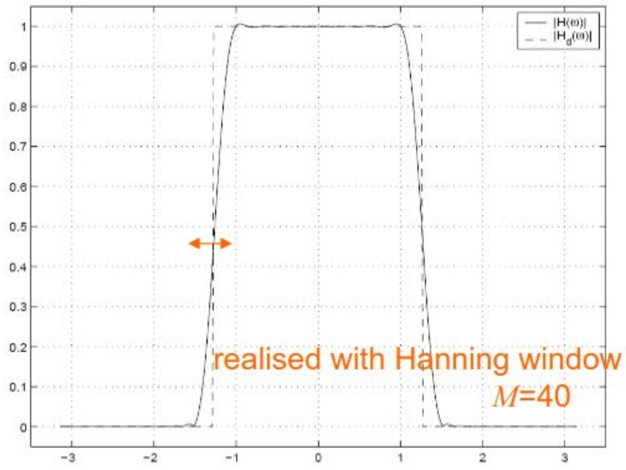
Filter Designing

We limited our choices to majorly used low pass finite impulse response (FIR) filters using window methods, namely - Rectangular, Hanning, Hamming, and Blackman.

Window's name	Mainlobe	Mainlobe/sidelobe	Peak $20\log_{10}\delta$
Rectangular	$4\pi/\ M$	-13dB	-21dB
Hanning	$8\pi/M$	-32dB	-44dB
Hamming	$8\pi/M$	-43dB	-53dB
Blackman	$12\pi/M$	-58dB	-74dB

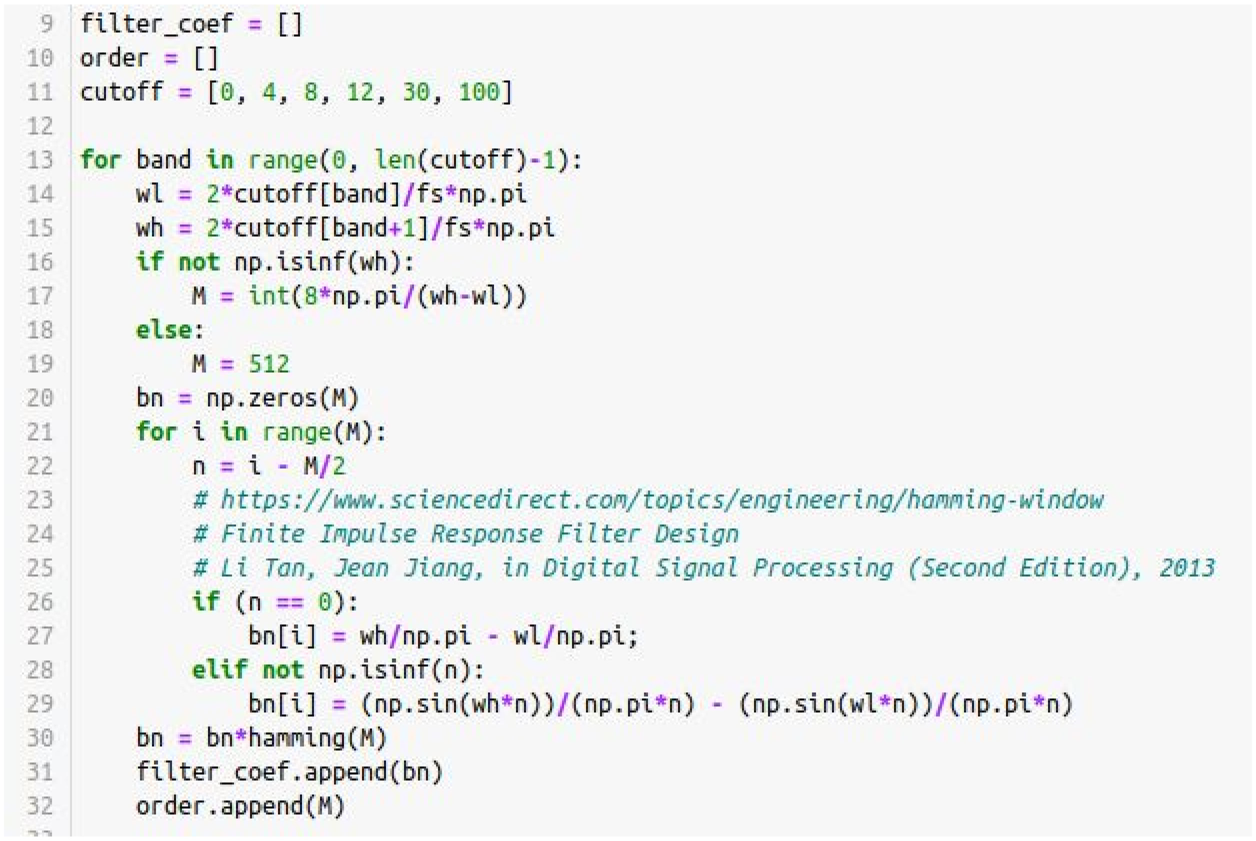
Window selection comes with various tradeoffs. Wider transition regions are compensated by the much lower sidelobes. This increases ripples in the system response. (See figure below)





There’s a fundamental trade-off between the main lobe and sidelobes; as the window smoothens more, the peak side lobes increase but the width of the main lobe decreases.

The ideal frequency response of a signal has infinite impulse responses, it has to be truncated and then shifted to the right to be made causal (necessary for FIR filter use).

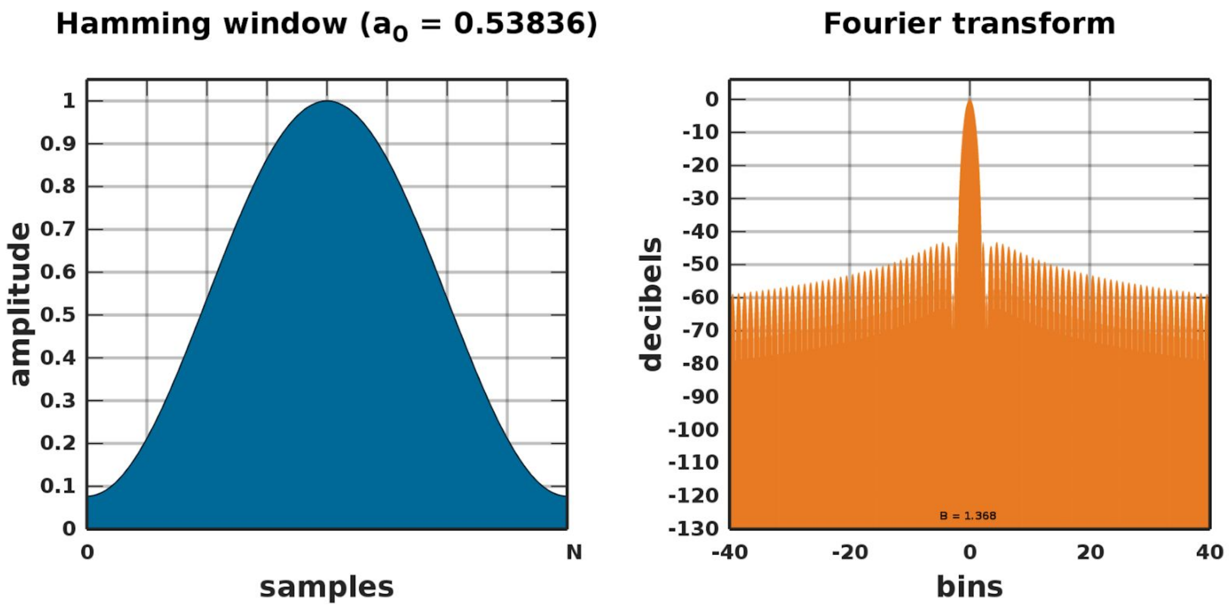


```python
filter_coef = []
order = []
cutoff = [0, 4, 8, 12, 30, 100]

for band in range(0, len(cutoff)-1):
    wl = 2*cutoff[band]/fs*np.pi
    wh = 2*cutoff[band+1]/fs*np.pi
    if not np.isinf(wh):
        M = int(8*np.pi/(wh-wl))
    else:
        M = 512
    bn = np.zeros(M)
    for i in range(M):
        n = i - M/2
        # https://www.sciencedirect.com/topics/engineering/hamming-window
        # Finite Impulse Response Filter Design
        # Li Tan, Jean Jiang, in Digital Signal Processing (Second Edition), 2013
        if (n == 0):
            bn[i] = wh/np.pi - wl/np.pi;
        elif not np.isinf(n):
            bn[i] = (np.sin(wh*n))/(np.pi*n) - (np.sin(wl*n))/(np.pi*n)
    bn = bn*hamming(M)
    filter_coef.append(bn)
    order.append(M)
```

After the calculations required, accounting for the limits, we decided to use the Hamming Window :

$$H(\theta) = 0.53836 + 0.46 \ cos[(2\pi/N)n]$$

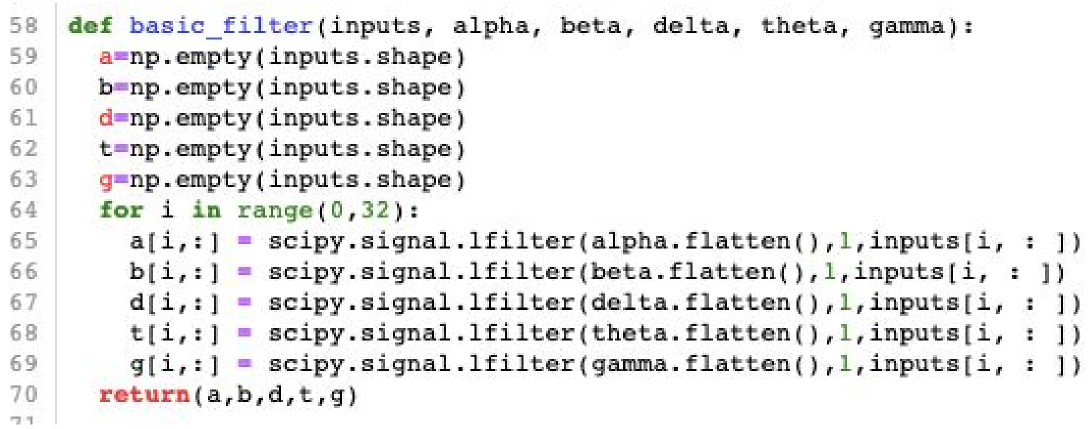


Spectral Response of the Hamming window

Preprocessing II

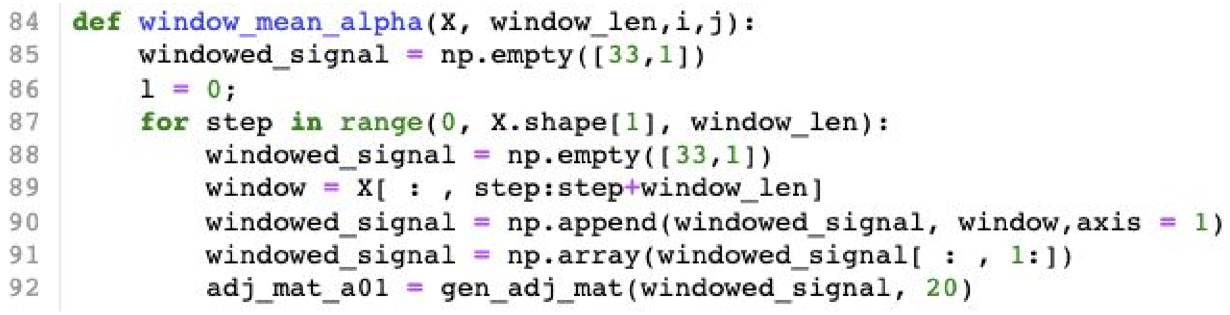
To generate adjacency matrices and the Social Network from the EEG signals the following steps were followed:-

1. The signals were passed through a function basic_filter which filtered the data frequency wise.



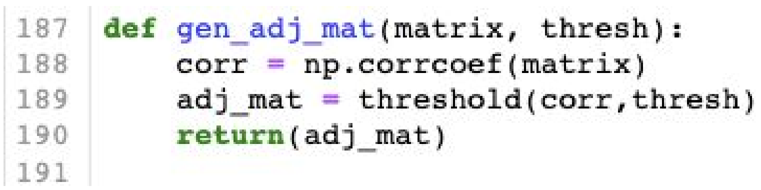
```python
def basic_filter(inputs, alpha, beta, delta, theta, gamma):
  a=np.empty(inputs.shape)
  b=np.empty(inputs.shape)
  d=np.empty(inputs.shape)
  t=np.empty(inputs.shape)
  g=np.empty(inputs.shape)
  for i in range(0,32):
    a[i,:] = scipy.signal.lfilter(alpha.flatten(),1,inputs[i, : ])
    b[i,:] = scipy.signal.lfilter(beta.flatten(),1,inputs[i, : ])
    d[i,:] = scipy.signal.lfilter(delta.flatten(),1,inputs[i, : ])
    t[i,:] = scipy.signal.lfilter(theta.flatten(),1,inputs[i, : ])
    g[i,:] = scipy.signal.lfilter(gamma.flatten(),1,inputs[i, : ])
  return(a,b,d,t,g)
```

2. The windowed mean of these filtered signals was taken. The length of the window was 200 i.e., we have divided the data in the interval of 2ms and replaced them by their mean.



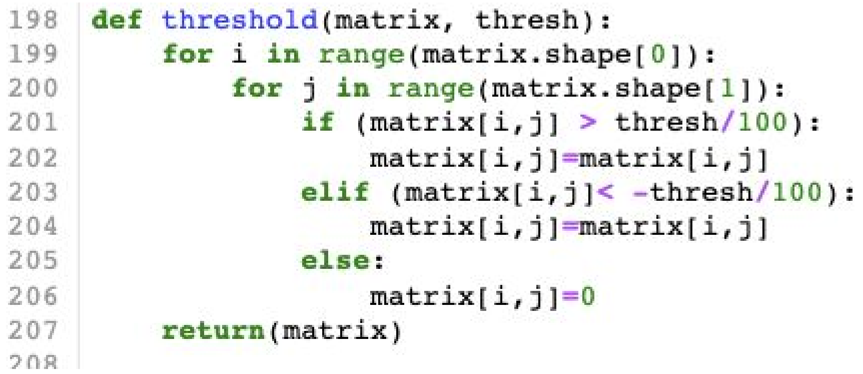
```python
def window_mean_alpha(X, window_len,i,j):
    windowed_signal = np.empty([33,1])
    l = 0;
    for step in range(0, X.shape[1], window_len):
        windowed_signal = np.empty([33,1])
        window = X[ : , step:step+window_len]
        windowed_signal = np.append(windowed_signal, window,axis = 1)
        windowed_signal = np.array(windowed_signal[ : , 1:])
        adj_mat_a01 = gen_adj_mat(windowed_signal, 20)
```

3. Next, the gen_adj_mat function was called which created the adjacency matrices.



```python
def gen_adj_mat(matrix, thresh):
    corr = np.corrcoef(matrix)
    adj_mat = threshold(corr,thresh)
    return(adj_mat)
```

4. We have done thresholding i.e., if Correlation is between -threshold and threshold then Correlation = 0.



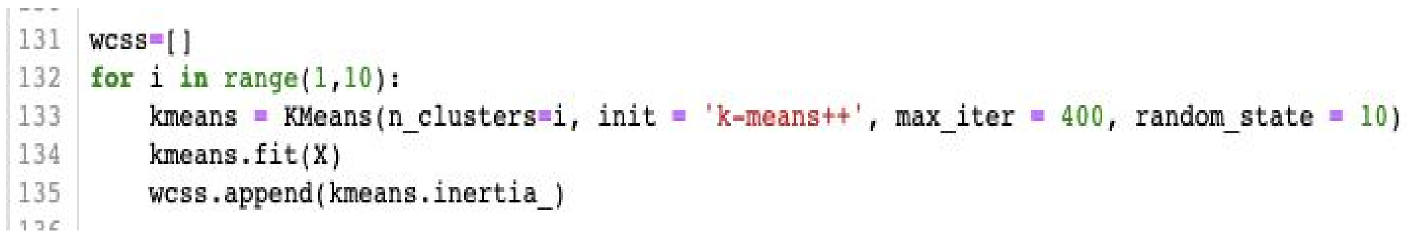
```
def threshold(matrix, thresh):
    for i in range(matrix.shape[0]):
        for j in range(matrix.shape[1]):
            if (matrix[i,j] > thresh/100):
                matrix[i,j]=matrix[i,j]
            elif (matrix[i,j]< -thresh/100):
                matrix[i,j]=matrix[i,j]
            else:
                matrix[i,j]=0
    return(matrix)
```

5. To convert the adjacency matrices into the form suitable for social network analysis graph_transform.py was used.
6. Further algorithms were applied to that social network.

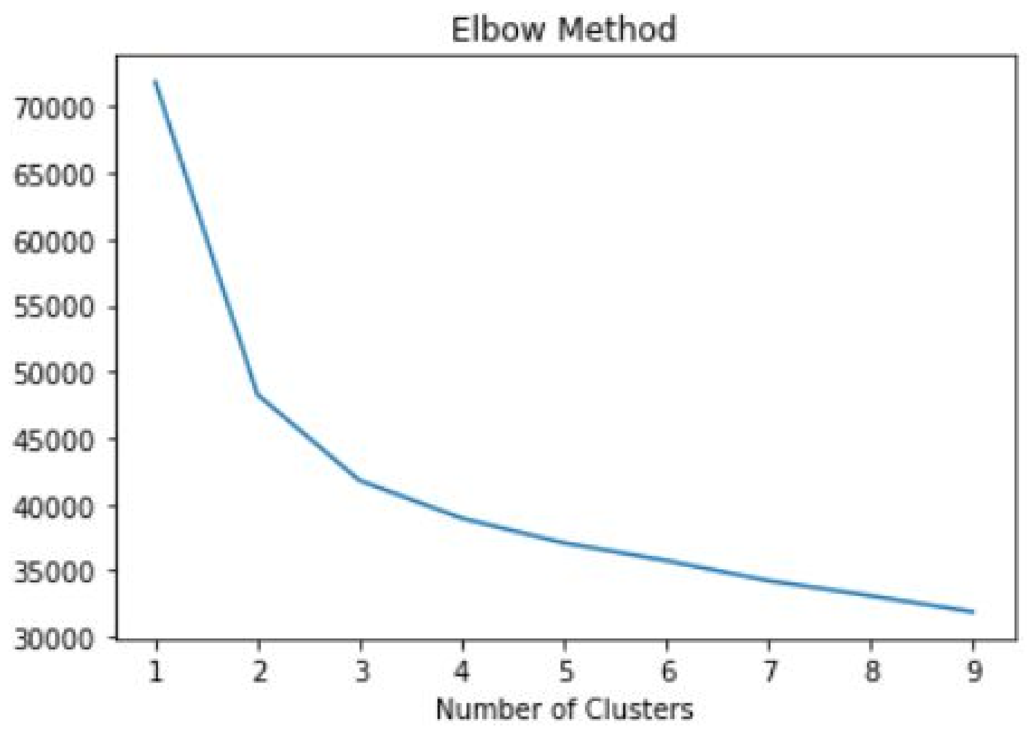
K-Means Clustering

K-Means clustering is one of the most popular clustering algorithms used to group data points into distinct non-overlapping subgroups.

We have used the elbow method to find the optimal number of clusters.



```
wcss=[]
for i in range(1,10):
    kmeans = KMeans(n_clusters=i, init = 'k-means++', max_iter = 400, random_state = 10)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```



The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters which in this case is 3.

Silhouette Score

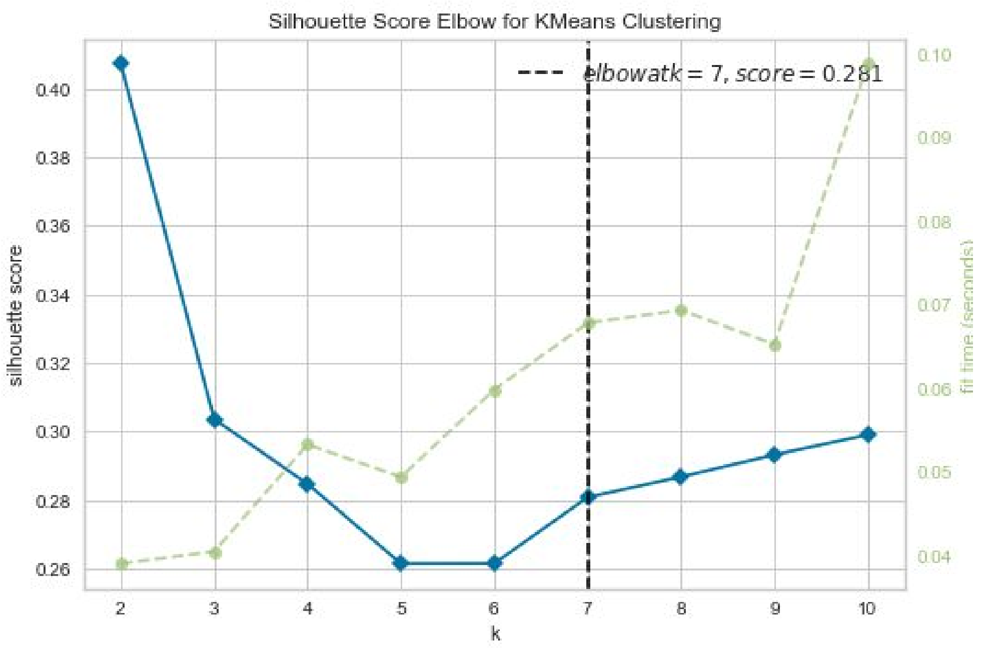
To determine the quality of clusters, we used the silhouette score analysis. It is a heuristic tool to determine whether the parameters used, for example, k in k-means clustering are reasonable or not.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

s(i) denotes the silhouette score of a single data point in the dataset. *a(i)* and *b(i)* account for the inter-cluster and intra-cluster distances. The score is bounded between -1 and 1. Higher the value, the better the clusters formed.

We used the yellowbricks library in python to help out with the silhouette score analysis.

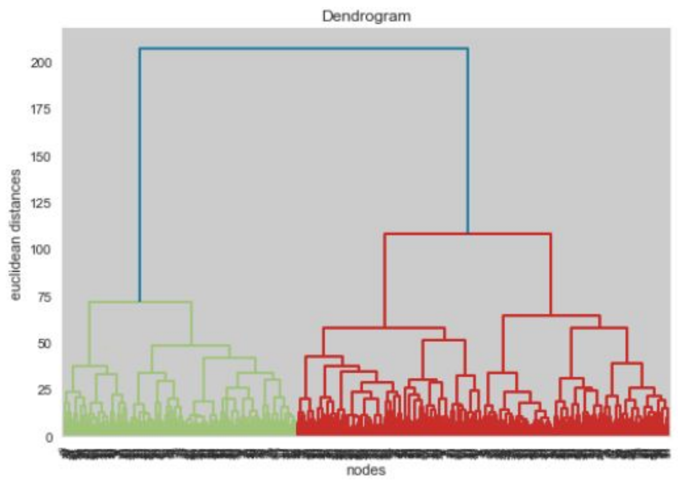
The silhouette analysis with all 365 steps in this data file gives a very low score of 0.281 even for 7 (=k) clusters. We moved forward to deterministic dimensionality reduction techniques like PCA to try out the analysis again.



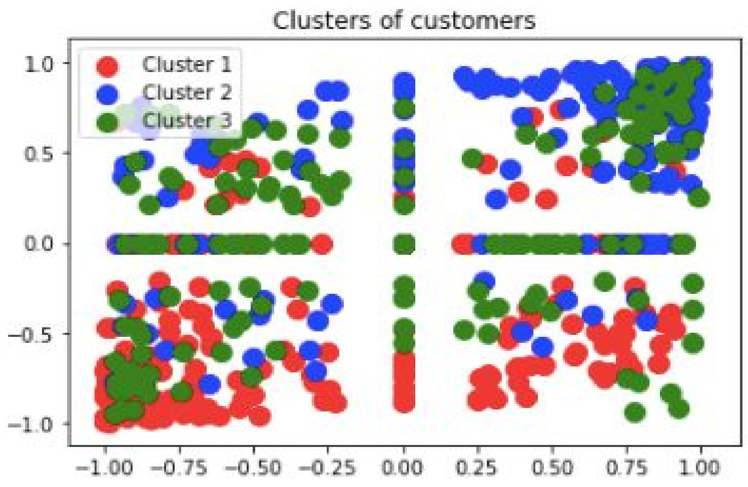
Hierarchical Clustering

There are two types of hierarchical clustering, Divisive and Agglomerative. We have used Agglomerative clustering. The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects. The result is a tree-based representation of the objects, named dendrogram.

```python
# Using the dendrogram to find the optimal number of clusters
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
```

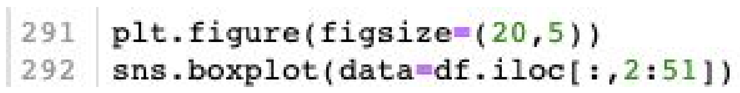


```python
hc = AgglomerativeClustering(n_clusters = 3,
                             affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(X)
```

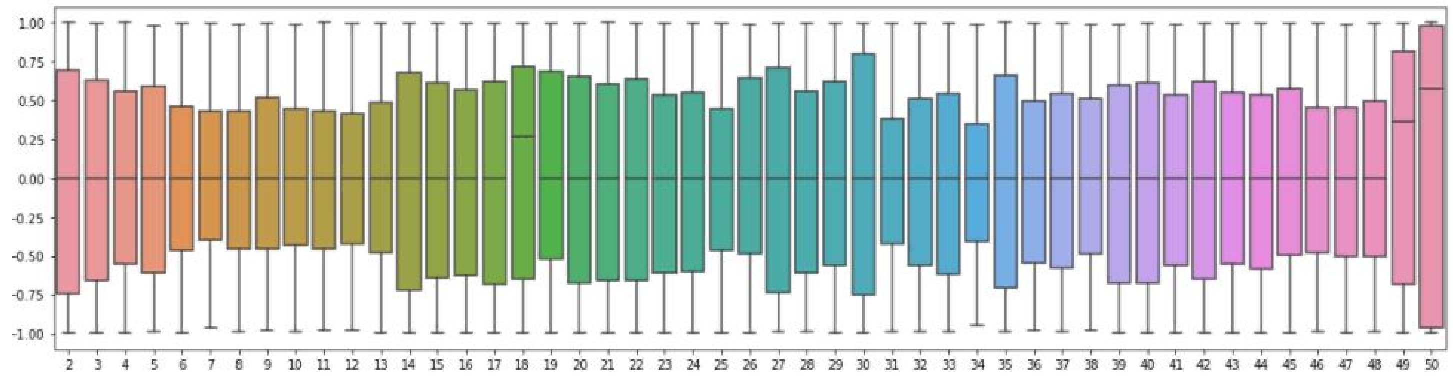


BoxPlot

A boxplot is a standardized way of displaying the distribution of data based on a five-number summary (minimum, first quartile, median, third quartile, and maximum). It can tell about outliers and what their values are. It can also tell if your data is symmetrical, how tightly the data is grouped, and if and how the data is skewed. We have used the boxplot to analyze our data in the batch of 50.



```
plt.figure(figsize=(20,5))
sns.boxplot(data=df.iloc[:,2:51])
```



Inferences from the boxplot:

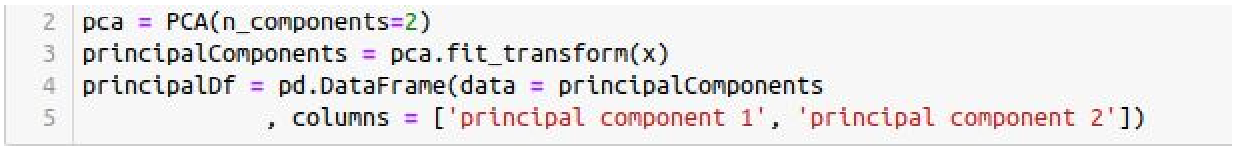
1. The median for most of the data elements is 0.
2. First quartile:- the middle number between the smallest number (not the "minimum") and the median of the dataset. It varied from -0.25 to -0.75.
3. Third quartile:- the middle value between the median and the highest value (not the "maximum") of the dataset. It varied from 0.25 to 0.75
4. interquartile range (IQR):- between first quartile and third quartile. It varied from 0.5 to 1.5.
5. There were no outliers.
6. Except for a few data elements (17, 49, and 50) the data is uniformly distributed.

Dimensionality Reduction

Principal Component Analysis

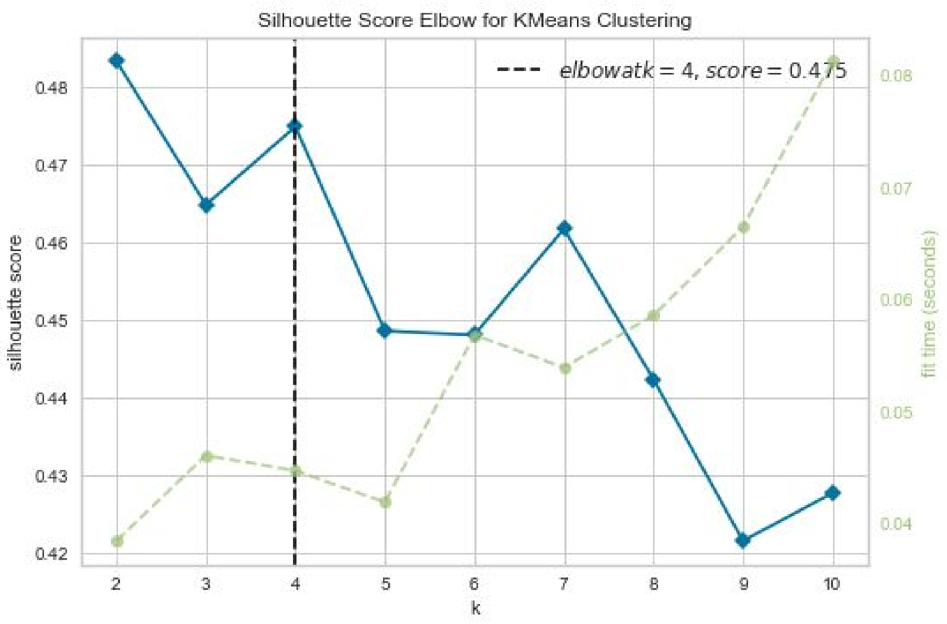
PCA is a deterministic dimensionality reduction technique. It is a nonlinear algorithm. Intuitively, it fits a p-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. The principal components being the axis, are orthogonal to each other. If an axis is found to be small, it can be omitted with small and not-so-affecting loss of information.

We used the scikit-learn library's implementation to do the PCA for us:



```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
```

PCA reduces the dimensions of our data to 2. We have come from 365 dimensions to mere 2 dimensions now with the loss of very small information. Silhouette analysis is done again to see the impact.



Silhouette analysis gives out a reasonable score of 0.475 at the elbow - k=4 clusters. 0.475 means satisfactory clustering quality and can be used for further analysis.

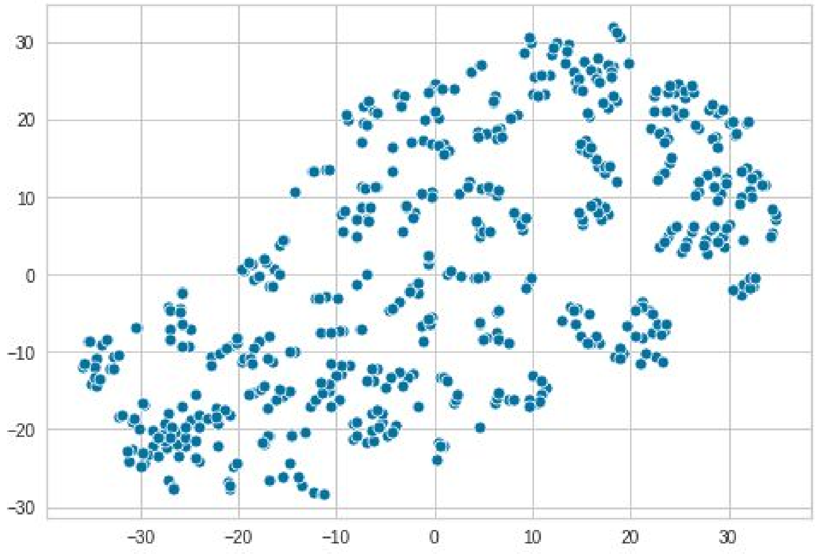
t-Distributed Stochastic Neighbor Embedding (t-SNE)

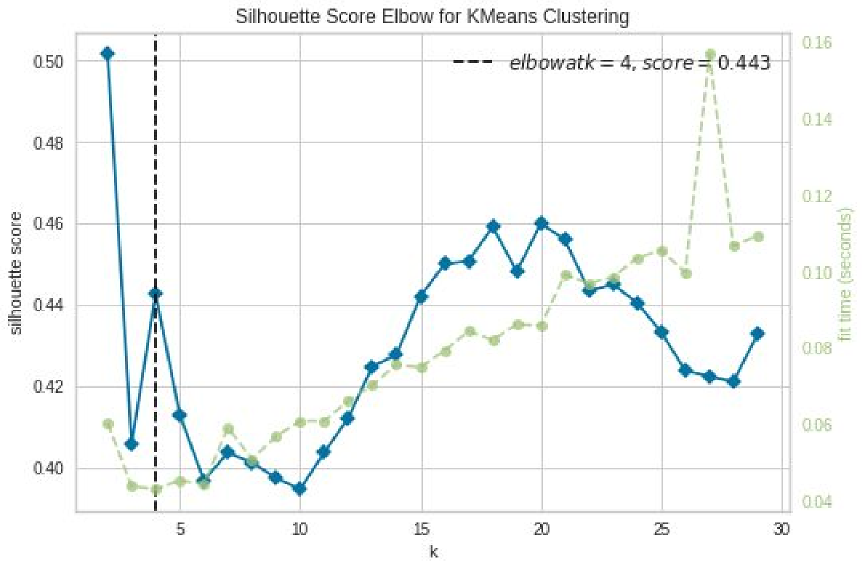
This is a recent prize-winning dimensionality reduction technique developed in 2008. It is a non-linear dimensionality reduction technique that performs quite well, in general, to reduce high dimensional data to two or three dimensions and hence can also be visualized easily.

It projects the data points such that the distance between similar points is lower and the dissimilar points are farther with high probability. It doesn't trade-off with a loss in information, directly, when reducing dimensions, unlike PCA.

```
tsne = TSNE(learning_rate=500, n_components=2)
X_tsne = tsne.fit_transform(dropped_cols_dataset_vals)
```

Here also, like PCA, we reduce the dimensions of the dataset to 2 dimensions. t-SNE allows dimension reduction to at max 3 dimensions. Given in the right is the visual representation of the t-SNE reduced data. We can distinguish the clusters forming. SIlhouette analysis can give better insights.





Silhouette analysis gives a similar result to PCA. Elbow at 4 (=k) clusters with a silhouette score of 0.443. This can also be used for further procedures, but since t-SNE is not as explainable as PCA, we will stick to PCA for our study.

Technology Stack

1. Matlab: It was used for creating the frequency wise coefficients which were used in filtering.
2. Python: All further work was done in python. The following libraries were used:
    a. Scikit learn
    b. Numpy
    c. Seaborn
    d. Scipy
    e. Matplotlib
    f. Pandas
    g. Yellowbrick

Future Work

Finally, we have reduced the dimensions by PCA, now it's time to work on the reduced dataset and extract valuable information from that.

Since we left our work on finding the optimal number of clusters, some work is still left. We are analyzing some more techniques for increasing the Silhouette score and then do clustering on that data.

Graph neural networks (GNNs) are connectionist models that capture the dependence of graphs via message passing between the nodes of graphs. Unlike standard neural networks, graph neural networks retain a state that can represent information from its neighborhood with arbitrary depth. Nowadays, GNNs are gaining popularity so we are planning to use them and therefore we are thinking to relabel our dataset.

Since many papers are using Deep Learning algorithms like CNNs or ANNs, they can also be used in our work. Further RNNs and LSTMs deal with time-series data they can also be used.

Source Code

All the source code is in python and in form of Jupiter notebooks available publicly in this repo hosted on Github: GitHub-link-here

References:

1. Fundamentals of EEG Measurement, M Teplan
2. B. H. Fadlallah, A. Keil and J. C. Príncipe, "Functional dependence in the human brain: A graph theoretical analysis," 2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Osaka, 2013, pp. 2948-2951, doi: 10.1109/EMBC.2013.6610158.
3. S. Koelstra et al., "DEAP: A Database for Emotion Analysis ;Using Physiological Signals," in IEEE Transactions on Affective Computing, vol. 3, no. 1, pp. 18-31, Jan.-March 2012, doi: 10.1109/T-AFFC.2011.15.
4. Y. Liu, O. Sourina and M. K. Nguyen, "Real-Time EEG-Based Human Emotion Recognition and Visualization," *2010 International Conference on Cyberworlds*, Singapore, 2010, pp. 262-269, doi: 10.1109/CW.2010.37.
5. Moon, K.R., van Dijk, D., Wang, Z. et al. Visualizing structure and transitions in high-dimensional biological data. Nat Biotechnol 37, 1482–1492 (2019), doi:10.1038/s41587-019-0336-3
6. Ahmed Mahfouz, Martijn van de Giessen, Laurens van der Maaten, Sjoerd Huisman, Marcel Reinders, Michael J. Hawrylycz, Boudewijn P.F. Lelieveldt, Visualizing the spatial gene expression organization in the brain through non-linear similarity embeddings, doi:10.1016/j.ymeth.2014.10.004.
7. Bullmore, E., Sporns, O. Complex brain networks: graph theoretical analysis of structural and functional systems. Nat Rev Neurosci 10, 186–198 (2009). https://doi.org/10.1038/nrn2575
8. Maja Stikic, Robin R. Johnson, Veasna Tan & Chris Berka (2014) EEG-based classification of positive and negative affective states, Brain-Computer Interfaces, 1:2, 99-112, DOI: 10.1080/2326263X.2014.912883
9. Bos,D.O.:EEG-based emotion recognition.Emotion 57(7),1798–1806 (2006)