Fractal Analysis of Biomedical Signals for Seizure Prediction

Draft Documentation

1. Introduction

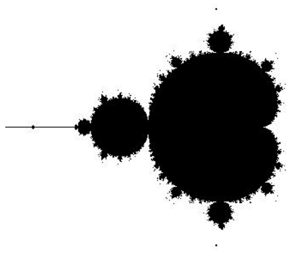
This documentation outlines a machine learning project focused on predicting seizures through the application of fractal analysis to biomedical signals such as ECG and EEG. Traditional seizure detection relies on visual EEG or ECG analysis by clinicians which is time consuming and subjective. The primary objective is to develop a reliable system for early seizure prediction. By employing fractal dimension calculations and integrating these features into a machine learning models, this initiative aims to overcome the limitations of current diagnostic approaches.

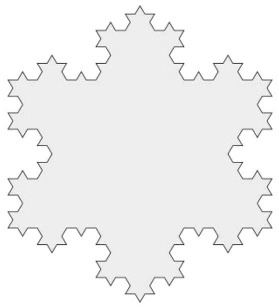
2. Overview of Biomedical signals

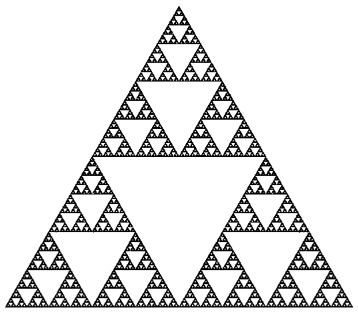
- **Electroencephalography (EEG)**: EEG stands as the most widely adopted diagnostic modality in epilepsy seizure due to its compelling advantages such as cost -effectiveness, portability. This technique quantitatively records the brain's electrical activity by capturing the summated electrical potentials generated by the vast populations of neurons.
- **Electrocardiography (ECG)**: While traditionally employed as non-invasive method for diagnosing various cardiac abnormalities, ECG is increasingly recognized for its significant potential in seizure prediction. This utility comes from the observation that dysregulation in Heart Rate Variability, a metric derived from ECG signals, frequently precedes epileptic seizures.

3. Overview of Fractal Analysis

Fractals is a concept pioneered by Benoit Manderlbrot in 1977. They are distinguished by their exhibition of scaled-invariant, self-similar behaviour. This simply implies that the shape of a fractal is similar even when it it scaled or zoomed in, with patterns often repeating across an infinite range of scales. Unlike Euclidean geometry, which describes smooth objects with integer dimensions, fractals possess an intricate "fine structure", and their inherent irregularity cannot be rendered on an integer scale. Fractals can extend beyond static geometric patterns; they can also effectively describe dynamic processes evolving over time. Some of the famous examples of fractals are the Manderlbrot Set, Koch Snowflake and Siepinski Triangle etc.


Manderlbrot Set


Koch Snowflake


Siepinski Triagnle

Fractal Dimension (FD): Fractal Dimension is a unit-less number that quantifies structural complexity and roughness of an object or phenomenon. In contrast to

topological dimensions. Fractal dimensions are typically non-integer values whereas normal Euclidean shapes have dimensions of integer values. A fractal curve's dimension, often falling between 1 and 2, indicates its efficiency in locally filling space compared to an ordinary line.
While a single, precise mathematical definition of a fractal remains elusive, key concepts for defining fractal dimension include

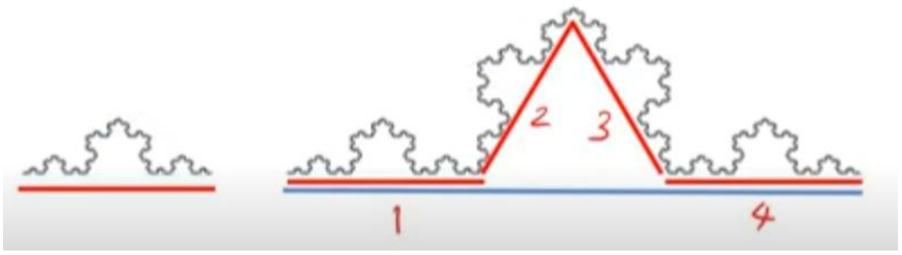
- **Box Counting Fractal Dimension**: In box counting method, the object or signal is covered with grid of boxes of a certain size, and the number of boxes that contain part of pattern is counted. As box size decreases, the number of boxes needed to cover the structure increases in a way that can be described by a power law. The slope of the relationship gives fractal dimension. Lets assume ε as the length of each square box in grid and N(ε) as number of boxes of size ε that are needed to cover the pattern. Then fractal dimension is represented by the equation

$$D = \lim_{\varepsilon \to 0} \frac{\log N(\varepsilon)}{\log\left(\frac{1}{\varepsilon}\right)}$$

But for perfectly similar fractals such as **Koch curve** or **Sierpinski's triangle** we know that ε= 1/R(R is the magnifying factor) and the equation and be re-written as

$$D = \lim_{\varepsilon \to 0} \frac{\log N(\varepsilon)}{\log\left(\frac{1}{\varepsilon}\right)} = \frac{\log N}{\log R}$$

Let us take an example where we magnify the koch curve by 3 times and we get 4 copies



Here **N = 4 , R = 3,** So dimension of this figure is

$$D = \frac{\log 4}{\log 3} \approx 1.2619$$

- **Higuchi Fractal Dimension (HFD): HFD** quantifies how the length of signal curve changes as a function of the scale of observation. In essence, it constructs multiple subsampled versions of signal and evaluates the average length of these curves at each scale. It is a robust algorithm to estimate the fractal dimension of time-series data

For a 1-dimensional time series signal X(1), X(2),…,X(N):

1. For a given integer k, construct k new subseries:

$$X_m^{(k)} = \{X(m), X(m+k), X(m+2k), \ldots\} \quad \text{for } m = 1, 2, \ldots, k$$

2. For each subseries Xm, compute its length:

$$L_m(k) = \frac{1}{k}\left(\sum_{i=1}^{\lfloor \frac{N-m}{k} \rfloor} |X(m+ik) - X(m+(i-1)k)|\right) \cdot \frac{(N-1)}{\lfloor \frac{N-m}{k} \rfloor \cdot k}$$

3. Average over all m:

$$L(k) = \frac{1}{k}\sum_{m=1}^{k} L_m(k)$$

4. Repeat steps for k = 1,2,…,Kmax and plot
   Log(L(k)) vs log(1/k)

5. The slope of linear region gives the Higuchi Fractal Dimension D_H:
   D_H = slope of log(L(k)) vs log(1/k)

- **Lacunarity:** Lacunarity is a type of fractal analysis which quantifies the texture and heterogeneity of fractal or spatial pattern. While fractal dimension measures how much space is filled, lacunarity measures how that space is filled particularly the size and distribution of gaps (lacunae) in the structure. In fractal analysis, fractal dimension measures scaling and space-filling complexity and lacunarity measures texture variability and gap distribution. Two shapes can have same fractal dimension but different lacunarity.

Two patterns can have the same fractal dimension but very different lacunarities:

-> Low Lacunarity: homogeneous texture (uniform gaps or distribution)
-> High Lacunarity: heterogeneous texture (variable gaps, clustering)

Given a fractal image or dataset covered by boxes of isze r, define:

- Q(M): the probability that a box contains mass M
- μ: the mean mass in all boxes
- σ^2: the variance of the mass distribution
- Then, lacunarity Λ(r) is defined as:

$$\Lambda(r) = \frac{\sigma^2(r)}{\mu^2(r)} + 1$$

$$\Lambda(r) = \frac{\langle M^2 \rangle}{\langle M \rangle^2}$$

- <M>: mean mass in boxes of size r
- <M^2>: second moment of mass distribution

$\Lambda(r)$ = 1 : perfectly uniform distribution

$\Lambda(r)$ > 1 : increasing heterogeneity

- **HURST EXPONENT:** Hurst exponent is a statistical measure used to quantify the long-term dependence or memory in a time series data like stock market or ECG/EEG. It tells us whether a time series is more likely to continue in the same direction, revert to the mean or is a random walk. It is denoted by the letter **H.**
- The Hurst Exponent is directly related to the fractal dimension, which measures the smoothness of a surface, or in our case the smoothness of a time series. The relationship between the fractal dimension **D**, and the Hurst Exponent **H**, is given by
- ***D = 2 - H*** where 0 <= H <= 1. The closer the value of **H** to 0, the more jagged will be the time series be.

H VALUE	**Interpretation**
=0.5	Purely Random (Brownian Movement)
<0.5	Anti-persistent(reverting to mean)
>0.5	Persistent(following the trent)

- **Estimation of Hurst Exponent using Re-scaled Range(R/S) Analysis:** Rescaled Range is a statistical method used to estimate the Hurst Exponent **H,** which characterizes the long-range dependence of a time series

$$\frac{R(n)}{S(n)} \propto n^{H}$$

Where:

R(n): Range of cumulative deviations in a segment of length n
S(n): Standard deviation of the segment
H: Slope of the line in log-log space -> Hurst Exponent

Steps for R/S Estimations:

Given a time series $X = \{x_1, x_2, x_3 \ldots, x_n\}$:

Step 1: Divide the time series

- Choose various window sizes $n \in \{n_1, n_2, \ldots, n_k\}$
- For each n, divide the time series into **M** = $\lfloor N/n \rfloor$ non-overlapping chunks.

Step 2: For each chunk of size n

For a chunk $X^{(m)} = \{x_1^{(m)}, x_2^{(m)}, \ldots, x_n^{(m)}\}$:

- Compute mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i^{(m)}$$

- Create mean-adjusted series

$$Y_k = \sum_{i=1}^{k}\left(x_i^{(m)} - \bar{x}\right)$$

- Compute range

$$\mathbf{R = \max(Y_k) - \min(Y_k)}$$

- Compute standard deviation

$$S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i^{(m)} - \bar{x}\right)^2}$$

- Compute rescaled range: R/S

Step 3: Average the R/S over all the chunks of size n

Step 4: Fit a lone in log-log plot. The slope of the best fit line is Hurst Exponent

Let us take an example. Given is a sample time series data along with its Rescaled Range Analysis value

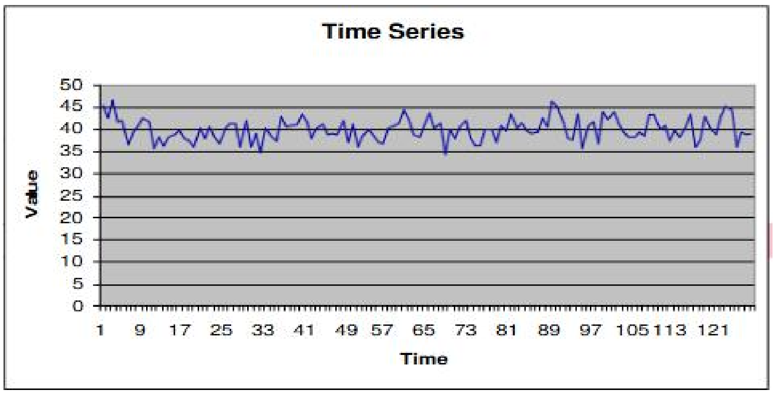
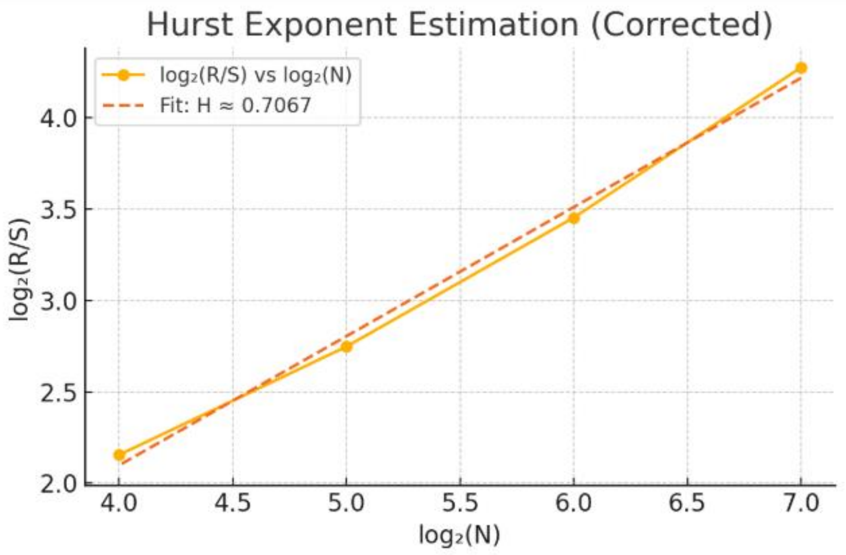


Table 1. Rescaled Range Analysis values to estimate *H*

Segment size (N)	$\log_2(N)$	$\log_2(R/S)$
128	7	4.2754
64	6	3.4539
32	5	2.7479
16	4	2.1550



In this sample the slope is close to 0.7067. Since H > 0.5 the time series shows persistent behaviour.

Python Implementation

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.polynomial.polynomial import polyfit

def rs_analysis(time_series, min_chunk=8, max_chunk=None):
   N = len(time_series)
   if max_chunk is None:
      max_chunk = N // 2

   chunk_sizes = []
   rs_values = []

   for n in range(min_chunk, max_chunk, 2):
      M = N // n
      if M < 2:
         continue
      rs = []
      for i in range(M):
         chunk = time_series[i * n : (i + 1) * n]
         mean = np.mean(chunk)
         Y = np.cumsum(chunk - mean)
         R = np.max(Y) - np.min(Y)
         S = np.std(chunk, ddof=1)
         if S != 0:
            rs.append(R / S)

      if len(rs) > 0:
         chunk_sizes.append(n)
         rs_values.append(np.mean(rs))

   log_n = np.log(chunk_sizes)
   log_rs = np.log(rs_values)

   slope, intercept = polyfit(log_n, log_rs, 1)
   return slope, log_n, log_rs
```

- **Lyapunov Exponent:** The Lyapunov Exponent is a fundamental concept in dynamical systems and chaos theory. It quantifies how sensitive is to initial conditions, which is the hallmark of chaotic behavior. Consider two very close initial points in a system. Over time, these points may Converge (stable behaviour) or stay close (neutral) or diverge exponentially (chaotic behaviour). The Lyapunov exponent λ measures the average exponential rate of divergence between nearby trajectories in phase space.

For two trajectories separated by an initial distance d_0 their separation after time t is approximately:

$$\delta(t) \approx \delta_0 e^{\lambda t}$$

Taking logs and solving for λ:

$$\lambda = \lim_{t\to\infty} \frac{1}{t} \ln \frac{\delta(t)}{\delta_0}$$

Interpretation of λ

Lyapunov Exponent λ	System behaviour
> 0	Chaotic (sensitive to initial conditions)
= 0	Neutral/Stable

< 0	Stable (convergent trajectories)

- **Katz Fractal Dimension:** Katz Fractal Dimension estimates how complicated or irregular waveform is by comparing the total length of diameter (the greatest distance from the first point). It assumes the signal is a geometric curve and evaluates how tightly the path winds over a time interval. The Katz Fractal Dimension (KFD) is a widely used method for estimating the complexity of 1D signals, particularly biomedical signals like EEG and ECG. It is simple, fast, and robust to noise, making it popular in seizure prediction, emotion recognition, and other time-series tasks.

  Given a time series X = {x_1, x_2, x_3, …. , x_n}, compute:

  - **L:** Total length of the signal

$$L = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + 1}$$

  - **d:** The maximum distance from the first point

$$d = \max_i \sqrt{(x_i - x_1)^2 + (i - 1)^2}$$

  - Then, Katz Fractal Dimensio is:

$$D_{\text{Katz}} = \frac{\log_{10}(L)}{\log_{10}(d) + \log_{10}(n)}$$

  Where:
  - n: number of signals in the point
  - L: total signal length
  - d: Maximum distance from origin

Python Implementation:

```
import numpy as np

def katz_fd(signal):
    n = len(signal)
    L = 0
    for i in range(1, n):
        L += np.sqrt((signal[i] - signal[i - 1])**2 + 1)
    d = max([np.sqrt((signal[i]-signal[0]**2+(i)**2)for i in
range(1,n)])
    return np.log10(L) / (np.log10(d) + np.log10(n))
```

Interpretation

$D \approx 1$	Smooth, regular signal like sinusoid
$D > 1.2$	Irregular or complex waveform
$D > 1.5$	Possibly chaotic behaviour

- Advantages of Katz Exponent:
    - Computationally efficient and easy to implement
    - Works directly on raw time-series
    - Robust to short signal strength
    - Less sensitive to signal amplitude

Higher Katz FD indicates greater complexity, which often correlates with seizure onset, arrhythmia, or cognitive stress in biomedical signals.

- **Petrosian Fractal Dimension:** The Petrosian Fractal Dimension quantifies the "instantaneous" complexity of a signal by analyzing how often the signal changes direction (sign) — i.e., how many times it crosses itself or reverses slope. It's a compact way of measuring how irregular or jagged a waveform is. EEG signals are highly dynamic, showing fluctuations in amplitude and frequency based on brain state. Seizures can cause sudden structural changes in EEG. These result in more zero crossing and greater signal irregularity and thus increasing Petrosian Fractal Dimension. ECG reflects the electrical activity of heart. **Heart Rate Variability(HRV)** becomes chaotic before some seizure types or during stress, arrhythmia or automatic dysfunction. ECG signals may show irregular intervals and welcome deformations before critical events thus showing higher PFD value

Formula

Given a signal $X = \{x_1, x_2, \ldots, x_n\}$:

1. Compute the number of sign changes in the signal's derivative (i.e, zero-crossings):

N_δ = **number of sign changes in diff(x)**

2. Compute Petrosian FD:

$$D_{PFD} = \frac{\log_{10}(n)}{\log_{10}(n) + \log_{10}\left(\frac{n}{n+0.4N_\delta}\right)}$$

n = number of points in the signal

- Python Implementation:

```
import numpy as np

def petrosian_fd(signal):
    n = len(signal)
    diff = np.diff(signal)
    # Zero-crossings in derivative
    N_delta = np.sum(diff[1:] * diff[:-1] < 0)
    return np.log10(n)/(np.log10(n)+np.log10(n/(n+0.4* N_delta)))
```

- **Detrended Fluctuation Analysis**

Detrended Fluctuation Analysis (DFA) is a powerful method for analyzing long-range correlations and fractal scaling in nonstationary time-series data, such as EEG and ECG.DFA quantifies the self-similarity and scaling behavior of a signal, even in the presence of local trends or non-stationarities.

It computes a scaling exponent α that indicates whether a signal exhibits long-range dependence, randomness, or anti-persistent behavior.

DFA is important for EEG/ECG because biomedical signals often contain:
- Trends (eg. Drift in ECG)
- Noise
- Complex temporal dynamics

DFA is a robust to these trends and can identify subtle fractal patterns and correlations that precede events like seizures or arrhythmia.
Formulas: Given a 1D signal $X = \{x_1, x_2, x_3, \ldots, x_n\}$

- Integrate the signal

$$Y(k) = \sum_{i=1}^{k} [x_i - \bar{x}]$$

- Divide into equal sized windows
- Detrend each segment: In each segment, fit a polynomial trend and subtract this local trend from the data.
- Compute the root mean square fluctuation

$$F(s) = \sqrt{\frac{1}{N} \sum_{k=1}^{N} [Y(k) - Y_{\text{fit}}(k)]^2}$$

- Repeat for multiple segment sizes *s:* Vary *s* and compute ***F(s)*** for each.
- Plot in log-log scale: the slope α of the linear portion is the DFA exponent.

Interpretation of DFA exponent α

α value	Signal type
≈ 0.5	Uncorrelated white noise (random walk)
< 0.5	Anti-persistent (alternates direction)
> 0.5	Persistent (trend-following behavior)
> 1.0	Nonstationary with strong correlations

4. Fractal Analysis in Biomedical Signals

Biomedical signals are nonlinear, non-stationary, and scale-invariant, properties that fractal analysis captures better than traditional methods. Fractals provides powerful tools to quantify the complexity, self similarity and irregularity of biomedical signals like EEG and ECG.

Steps for finding dimensionality using box counting fractal dimension

- Preprocess Signal:
    - Filter EEG or ECG to a standard frequency
    - Normalize amplitude to [0,1] range

- Box-Counting Algorithm:
    - Discretize the signal into a grid of box sizes ε
    - For each ε, count boxes $N(\varepsilon)$ that intersect the signal
    - Plot $\log N(\varepsilon)$ vs $\log(1/\varepsilon)$; the slope is D
    - If dimension D is similar to 1 then it is a simple curve,
    - $D > 1.3$ shows some level of irregularity
    - $D > 1.5$ shows that the signals are highly chaotic

  We can interpret the possibility of seizure by analyzing the Dimensions of ECG and EEG signals

5. Machine Learning Models for Seizure Prediction

Fractal features alone are not sufficient unless combined with a robust decision-making system. The goal is to classify EEG or ECG segments into seizure (ictal/preictal) or non-seizure (interictal) states. This section outlines commonly used machine learning and deep learning models for seizure prediction using fractal and nonlinear features.

5.1 Feature Extraction

The following features are typically extracted from EEG/ECG segments:

- Box Counting Fractal Dimension
- Higuchi Fractal Dimension
- Petrosian Fractal Dimension
- Katz Fractal Dimension
- Hurst Exponent
- Lancunarity
- Lyapunov Exponent
- Detrended Fluctuation Analysis
- Multifractal DFA

These features can be computed from raw time series (wavelets) or frequency-transformed representations

5.2 Classifiers and Models for Seizure Prediction

Traditional ML classifiers and modern deep networks can both be used. **Support Vector Machines (SVM)** are very common for EEG seizure detection. They handle hight-dimensional feature vectors well. SVM uses RBF kernels on fractal and entropy features. **Random Forest(RF)** is also another popular choice-its ensemble of decision trees handles non-linear feature interactions and often yield high accuracy. **kNN** is a simple instance based classifier also used in many studies. Other standard classifiers include Logistic Regression, Decision Trees, and feed forward Neural Networks.

Deep learning methods bypass manual feature extraction by learning hierarchical patterns. Convolutional Neural Network are widely used for seizure classification. CNNs automatically learn spatial or special patterns and have achieved very high accuracies in recent studies. Recurrent Neural Networks like **LSTM (Long Short-Term Memory)** capture temporal dynamics of EEG, making them suited to sequential data.

In summary, classification of preictal vs interictal EEG typically uses feature-based ML pipelines (feature selection + SVM/RF) or end-to-end deep nets (CNNs, LSTMs, hybrids). Simpler models (LR, kNN, Naive Bayes) are often outperformed but may still serve as benchmarks. State-of-the-art systems tend to use complex architectures (deep CNNs or CNN-LSTM) or ensembles, achieving sensitivities in the 90–99% range on datasets like CHB-MIT.

Comparative Classifier Summary

Model	**Key Characteristics**
SVM	Margin-based classifier; effective in high-dim feature spaces. Often used with RBF kernel on fractal/entropy features. Reported accuracies up to ~99%
Random Forest	Ensemble of decision trees; handles nonlinear feature interactions, robust to noise. Widely applied to EEG with fractal features. Accuracies ~93–99%
kNN	Classifies based on nearest labeled examples; simple but sensitive to feature scaling. Used in some studies for EEG. Performance is dataset-dependent
Logistic Regression	Linear classifier (sigmoid output); quick to train, interpretable but may underperform on complex EEG features. Used as baseline in some works.
Decision Tree / AdaBoost	Tree-based classifiers; AdaBoost combines weak learners. Occasionally used (e.g. tree selects fractal features). Less common for seizure but available in scikit-learn.
CNN	1D/2D convolutions extract local patterns (spatial/frequency). Learns features from raw EEG or spectrograms.
LSTM (RNN)	Captures temporal sequences in EEG.

	Often combined with CNN. Less common standalone for raw EEG, but effective for time-series patterns.
CNN–LSTM (Hybrid)	Combines CNN spatial filtering + LSTM memory. “Remembers” previous EEG states; often outperforms single models Achieved near-perfect accuracy in some studies.
Autoencoder + Classifier	Uses unsupervised feature learning (e.g. CNN-Autoencoder) combined with ML classifier. E.g., CNN-AE features fused with fractal features fed to SVM/kNN.

5.3 Model Pipeline

The General workflow is as follows:

1. **Preprocessing:** Filter, normalize and window EEG/ECG data.
2. **Feature Extraction:** Apply fractal and entropy analysis on each segment.
3. **Model Training:**
    - Traditional ML: Train classifier on feature vectors
    - Deep Learning: Feed raw or spectrogram-transformed signals into CNN or CNN-LSTM
4. **Evaluation:** Use cross-validation or test set to report metrics using Accuracy, Precision, Recall, F1-Score ,ROC-AUC
5. **Interpretation:** Persistent or chaotic signals during preictal periods may be indicated by increasing **FD** or Lyapunov Exponent and decreasing Entropy

6. Python Libraries and Tools

The following open-source Python libraries support fractal analysis, features extraction and classifier training

Library	**Purpose**
nolds	Computes Hurst Exponent, Lyapunov, DFA, Higuchu/Katz/Petrosian FD
pyEEG	EEG-specific library for ApEn, SampEn, FD and other features
Neurokit2	Broad library for biosignal analysis including entropy, HRV, complexity.
antropy	Lightweight library for entropy measures: ApEn, SampEn, PermEn, etc.
Scikit-learn	Classic ML algorithms (SVM, RF, kNN, LR), feature scaling, metrics.
TensorFlow/Kerals/PyTorch	Build deep learning models: CNN, LSTM, autoencoders.
MNE-Python	EEG signal preprocessing: filtering, epoching, sensor data handling.