```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd

df = pd.read_csv("/content/drive/MyDrive/EEG/signal-8.csv.gz")
df.head()
```

	Unnamed: 0	AF3	AF4	F3	F4	F7	F8	FC5	FC6	O1	...	CQ_F
0	1	3970.769231	3412.820513	3815.384615	3223.076923	3928.717949	2892.820513	4180.512821	3528.205128	4117.435897	...	4
1	2	3974.871795	3417.435897	3815.897436	3227.179487	3927.692308	2882.051282	4182.051282	3520.000000	4122.051282	...	4
2	3	3995.384615	3430.256410	3816.923077	3231.282051	3928.205128	2883.076923	4180.000000	3511.282051	4117.948718	...	4
3	4	3997.435897	3422.051282	3819.487179	3234.871795	3927.179487	2875.384615	4172.820513	3503.589744	4105.641026	...	4
4	5	3998.974359	3416.410256	3826.666667	3248.205128	3924.102564	2844.615385	4170.256410	3503.589744	4102.564103	...	4

5 rows × 36 columns

```
!pip install mne -q
```

```
Start coding or generate with AI.
```

```
!wget https://zenodo.org/record/1252141/files/EEGs_Guinea-Bissau.zip
```

```
--2026-02-17 07:22:41--  https://zenodo.org/record/1252141/files/EEGs_Guinea-Bissau.zip
Resolving zenodo.org (zenodo.org)... 188.185.43.153, 188.184.103.118, 188.184.98.114, ...
Connecting to zenodo.org (zenodo.org)|188.185.43.153|:443... connected.
HTTP request sent, awaiting response... 301 MOVED PERMANENTLY
Location: /records/1252141/files/EEGs_Guinea-Bissau.zip [following]
```

```
--2026-02-17 07:22:41--  https://zenodo.org/records/1252141/files/EEGs_Guinea-Bissau.zip
Reusing existing connection to zenodo.org:443.
HTTP request sent, awaiting response... 200 OK
Length: 153973086 (147M) [application/octet-stream]
Saving to: ‘EEGs_Guinea-Bissau.zip’

EEGs_Guinea-Bissau. 100%[===================>] 146.84M   151MB/s    in 1.0s

2026-02-17 07:22:42 (151 MB/s) - ‘EEGs_Guinea-Bissau.zip’ saved [153973086/153973086]
```

```
#unzip the files
from zipfile import ZipFile
data = ZipFile('EEGs_Guinea-Bissau.zip')
data.extractall()
```

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

```
meta_df=pd.read_csv('https://zenodo.org/record/1252141/files/metadata_guineabissau.csv')
meta_df.head()
```

	subject.id	Group	Eyes.condition	Remarks	recordedPeriod	startTime
0	1	Epilepsy	closed-3min-then-open-2min	by 45s reposition electrodes	301	27/5/2020 14:33
1	2	Control	open-3min-then-closed-2min	NaN	309	26/5/2020 22:44
2	3	Epilepsy	closed-3min-then-open-2min	NaN	309	27/5/2020 14:26
3	4	Epilepsy	closed-3min-then-open-2min	Green lights not shown, but good EEG traces	299	27/5/2020 15:23
4	5	Control	closed-3min-then-open-2min	NaN	302	23/5/2020 19:09

Next steps: Generate code with `meta_df` New interactive sheet

```
#now i need to seprate Epilepsy vs Control subjects
EP_sub=meta_df['subject.id'][meta_df['Group']=='Epilepsy']
CT_sub=meta_df['subject.id'][meta_df['Group']=='Control']
```

```
#read csv files
Epilepsy=[pd.read_csv('EEGs_Guinea-Bissau/signal-{}.csv.gz'.format(i), compression='gzip') for i in  EP_sub]
Control=[pd.read_csv('EEGs_Guinea-Bissau/signal-{}.csv.gz'.format(i), compression='gzip') for i in  CT_sub]
```

```
Epilepsy[0].head()
```

	Unnamed: 0	AF3	AF4	F3	F4	F7	F8	FC5	FC6	O1	...	CQ_F
0	1	4426.153846	3994.871795	4408.205128	3847.692308	4690.256410	3895.897436	4702.051282	3914.871795	4049.743590	...	4
1	2	4420.512821	3986.666667	4394.358974	3836.923077	4678.461538	3886.666667	4696.410256	3910.769231	4054.358974	...	4
2	3	4413.846154	3986.153846	4386.666667	3831.794872	4654.871795	3881.025641	4690.769231	3908.205128	4066.666667	...	4
3	4	4407.692308	3984.615385	4384.102564	3832.820513	4644.615385	3883.076923	4686.153846	3910.256410	4063.076923	...	4
4	5	4407.179487	3978.974359	4382.564103	3832.307692	4647.692308	3878.974359	4685.641026	3903.076923	4057.948718	...	4

5 rows × 36 columns

```
#remove non eeg channels
Epilepsy=[i.iloc[:,1:15] for i in  Epilepsy]
Control=[i.iloc[:,1:15] for i in  Control]
```

```
import mne
def convertDF2MNE(sub):
    info = mne.create_info(list(sub.columns), ch_types=['eeg'] * len(sub.columns), sfreq=128)
    info.set_montage('standard_1020')
    data=mne.io.RawArray(sub.T, info)
    data.set_eeg_reference()
    data.filter(l_freq=0.1,h_freq=45)
    epochs=mne.make_fixed_length_epochs(data,duration=5,overlap=1)
    epochs=epochs.drop_bad()

    return epochs
```

```
%%capture
#Convert each dataframe to mne object
```

```
Epilepsy=[convertDF2MNE(i) for i in  Epilepsy]
Control=[convertDF2MNE(i) for i in  Control]
```

```
%%capture
#concatenate the epochs
Epilepsy_epochs=mne.concatenate_epochs(Epilepsy)
Control_epochs=mne.concatenate_epochs(Control)
```

```
Epilepsy_group=np.concatenate([[i]*len(Epilepsy[i]) for i in range(len(Epilepsy))])#create a list of list where each sub list correspo
Control_group=np.concatenate([[i]*len(Control[i]) for i in range(len(Control))])#create a list of list where each sub list corresponds

Epilepsy_label=np.concatenate([[0]*len(Epilepsy[i]) for i in range(len(Epilepsy))])
Control_label=np.concatenate([[1]*len(Control[i]) for i in range(len(Control))])
```

```
Epilepsy_group.shape,Control_group.shape,Epilepsy_label.shape,Control_label.shape
```

```
((3995,), (3461,), (3995,), (3461,))
```

```
#combine data
data=mne.concatenate_epochs([Epilepsy_epochs,Control_epochs])
group=np.concatenate((Epilepsy_group,Control_group))
label=np.concatenate((Epilepsy_label,Control_label))
print(len(data),len(group),len(label))
```

```
Not setting metadata
7456 matching events found
No baseline correction applied
7456 7456 7456
```

```
Start coding or generate with AI.
```

```
# source: https://mne.tools/stable/auto_tutorials/clinical/60_sleep.html#sphx-glr-auto-tutorials-clinical-60-sleep-py

def eeg_power_band(epochs):
    """EEG relative power band feature extraction.

    This function takes an ``mne.Epochs`` object and creates EEG features based
    on relative power in specific frequency bands that are compatible with
    scikit-learn.

    Parameters
    ----------
    epochs : Epochs
        The data.

    Returns
    -------
    X : numpy array of shape [n_samples, 5]
        Transformed data.
    """
    # specific frequency bands
    FREQ_BANDS = {"delta": [0.5, 4.5],
                  "theta": [4.5, 8.5],
                  "alpha": [8.5, 11.5],
                  "sigma": [11.5, 15.5],
                  "beta": [15.5, 30],
                  "gamma": [30, 45],
                  }

    # Compute the PSD using the Welch method
    spectrum = epochs.compute_psd(method='welch', picks='eeg', fmin=0.5, fmax=45)
    psds = spectrum.get_data()
    freqs = spectrum.freqs
    psds /= np.sum(psds, axis=-1, keepdims=True)    # Normalize the PSDs

    X = []#For each frequency band, compute the mean PSD in that band
    for fmin, fmax in FREQ_BANDS.values():
        psds_band = psds[:, :, (freqs >= fmin) & (freqs < fmax)].mean(axis=-1)# Compute the mean PSD in each frequency band.
        X.append(psds_band)

    return np.concatenate(X, axis=1)#Concatenate the mean PSDs for each band into a single feature vector
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
```

```
subset = 2000 # Increased subset to include both 'Epilepsy' and 'Control' samples
```

```
    features = []
    for d in range(subset):
        features.append(eeg_power_band(data[d]))

    features = np.concatenate(features)
```

```
    Effective window size : 5.000 (s)
```

```
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
```

```
# rebuild labels from Epilepsy and Control lists
labels = np.array([1]*len(Epilepsy) + [0]*len(Control))

np.unique(labels, return_counts=True)
```

```
(array([0, 1]), array([46, 51]))
```

```
np.unique(y, return_counts=True)
```

```
(array([0]), array([2000]))
```

```
subset = 2000 if len(data) > 2000 else len(data)

X_list = []
y = []

for i in range(subset):
    feat = eeg_power_band(data[i])  # (n_epochs_i, 70)
    X_list.append(feat)

    n_epochs_i = feat.shape[0]
```

```
        if i < len(Epilepsy):
            y.extend([1]*n_epochs_i)
        else:
            y.extend([0]*n_epochs_i)

    X = np.vstack(X_list)
    y = np.array(y)

    X.shape, y.shape, np.unique(y, return_counts=True)
```

```
    Effective window size : 5.000 (s)
```

```
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
Effective window size : 5.000 (s)
((2000, 84), (2000,), (array([0, 1]), array([1949,   51])))
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

model = RandomForestClassifier(
    n_estimators=300,
    class_weight="balanced",
    n_jobs=-1,
    random_state=42
)
```

```
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9875

Confusion Matrix:
 [[390   0]
 [  5   5]]

Classification Report:
               precision    recall  f1-score   support

           0       0.99      1.00      0.99       390
           1       1.00      0.50      0.67        10

    accuracy                           0.99       400
   macro avg       0.99      0.75      0.83       400
weighted avg       0.99      0.99      0.99       400
```

✦ Gemini

```
#do 5 fold cross validation
clf=RandomForestClassifier()
accuracies=cross_val_score(clf, X, y, groups=group[:len(X)], cv=5)
print('Five fold accuracies',accuracies)
print('Average accuracy',np.mean(accuracies))
```