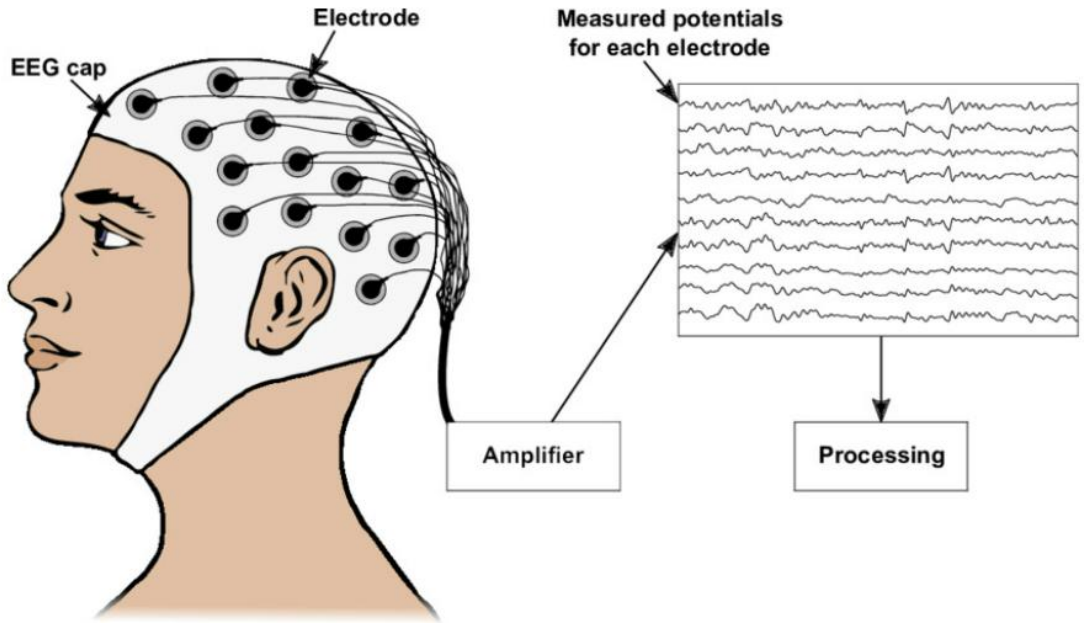
EEG data

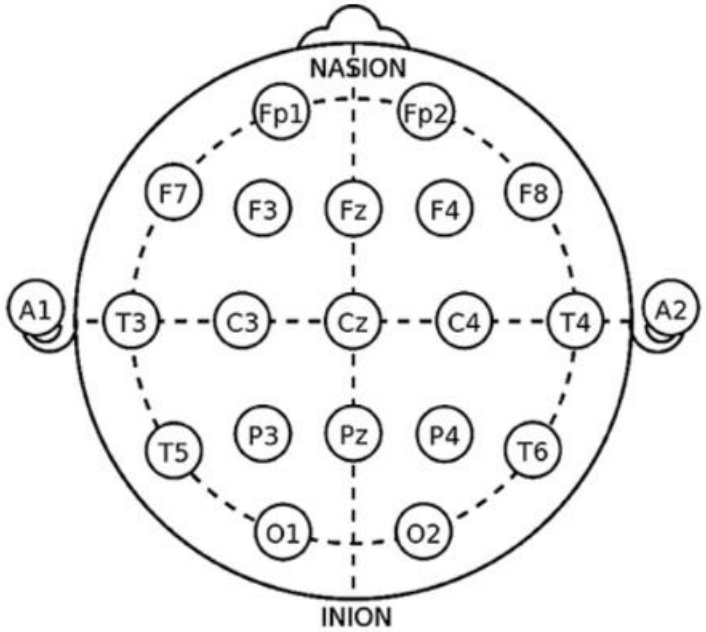
EEG data

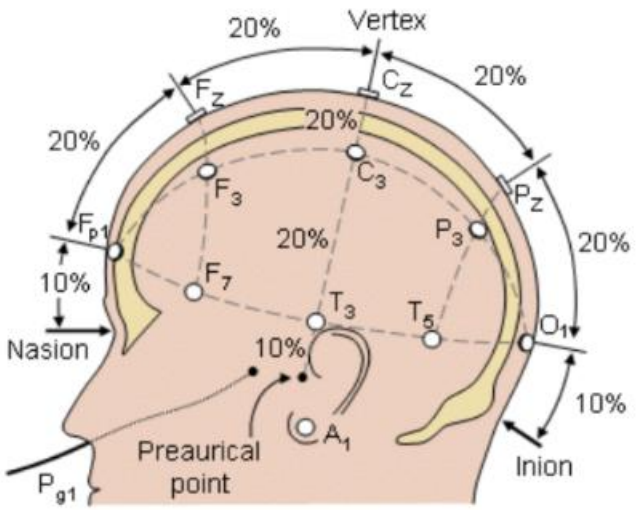


뇌의 전기적 활동을 측정하는 데 사용되는 신호

EEG 데이터를 분석하여 뇌의 활동 패턴을 이해하거나, 특정 상태나 질병의 진단 및 치료에 활용할 수 있다.

EEG data



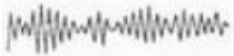


국제임상신경생리연맹이 정한 국제적 １０－２０ 시스템

전극 사이의 실제 거리가 두개골의 전체 앞뒤 또는 좌우 거리의 10%

또는 20%라는 사실을 나타냄

주파수

주파수대 (frequency Bands)	뇌파	상관되는 두뇌 상태
0.5 ~ 3Hz Delta		숙면, 뇌 손상, movement or eye blink artifact, LD(유아에게 많이 나타남)
3 ~ 5Hz Low Theta		졸음
6 ~ 7Hz High Theta		내면으로 향함, 기억 재생에 중요, 매우 창조적인 것이 특징, 읽고 경청하는 등의 외적 학습 자극에는 초점을 맞추지 못함. (어린 아동들에게 흔히 나타나는 주파수대)
7.5 ~ 8.5Hz		시각화 (Visualization)
8 ~ 10(or 11)Hz Low Alpha		내면으로 향함, 명상의 어떤 형태에서 관찰 가능, 해리 현상을 경험할 수 있다. (성인이 눈을 감았을 때의 뇌파)
12(11 ~ 13)Hz High Alpha		넓고 통찰적인 자각 상태, 고난도 기술을 구사해야 하는 운동 선수가 준비 상태에 있을 때 관찰 가능. (높은 지능을 가진 사람들에서 흔히 high peak alpha frequency가 나타남)
13 ~ 15Hz SMR		Central Cortex에서만 관찰(C3, Cz, C4), 한 곳에 집중하면서 감각과 운동의 활동성이 줄어들 때 관찰됨, 움직임이 없고 불안과 충동성이 감소되는 현상과 상관, 의식적인 활동이 감소될 때와 연관.

주파수

16 ~ 20Hz Beta		문제 해결을 위한 가장 필요한 주파수대, 학습을 할 때 필요한 베타파
19 ~ 32Hz		불안을 동반한 정서적 긴장 상태
24 ~ 36Hz		주로 부정적인 생각을 반추할 때
~ 27Hz		가족의 물질 탐닉 경향과 관계
38 ~ 42Hz Gamma		Binding Rhythm: 대상의 다른 측면을 하나로 묶어서 지각할 때, Peak Performance와 연관. (떨어지지 않기 위해 균형을 잡을 때 나타남)

EEG data

Delta (0.5 – 4 Hz):
가장 느린 뇌파로 깊은 수면 및 신체 회복과 관련됨.
꿈 없는 수면에서 두드러지며 치유 및 재생 과정에 기여.

Theta (4 – 8 Hz):
얕은 수면, 깊은 명상, REM 수면 중에 발생.
창의력, 직관, 공상과 관련됨.

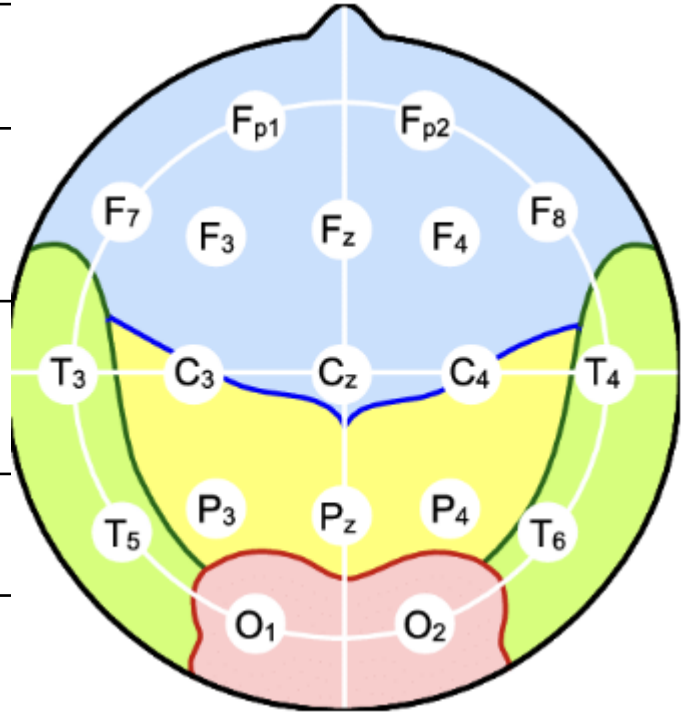
Alpha (8 – 12 Hz):
이완 상태이지만 여전히 각성된 상태에서 나타남.
정신적 조화, 차분함, 학습을 돕는 역할.

Beta (12 – 30 Hz):
깨어 있는 동안의 집중 및 사고 활동과 관련됨.
인지 작업 수행, 주의 집중, 불안한 사고 등에 관여.

Gamma (30 – 45 Hz):
고차원적 사고, 정보 처리, 기억 형성에 중요한 역할.
서로 다른 뇌 영역에서 동시에 정보를 처리하는 과정과 관련됨

EEG data

전극	위치	주요 역할
Fp1, Fp2	전두엽 앞쪽 부위, 눈썹 위	고차원적 인지 기능 (집중력, 문제 해결, 계획), 감정 및 행동 조절
F3, F4	전두엽 중간 부위, 이마 부위	운동 계획, 의사결정, 사회적 행동 및 반응 조절
C3, C4	두정엽 중앙 부위, 중앙 선 기준 좌우	감각 정보 처리, 운동 조절, 공간 인식 및 체감
P3, P4	두정엽 후방, 감각 및 공간 인식 관련	공간적 사고, 시각적 처리, 감각 정보 통합
O1, O2	후두엽, 시각 관련 영역	시각 정보 처리, 시각적 인식 및 반응
T3, T4	측두엽, 청각 관련 영역	청각 정보 처리, 언어 이해 및 기억, 감정 처리
Fz, Cz, Pz, Oz	중간선에 위치한 전극	두뇌의 기본적인 기능, 고차원적 인지, 운동 및 감각 처리, 시각적 처리



EKG: 심전도 전극. 뇌파가 아니라 심장의 전기 활동 기록.
→ EEG 분석에서 심장 잡음을 제거하거나 관련성 파악에 사용 가능

EEG 데이터 분석에 AI의 도입 필요성

전통적인 방식의 EEG 분석은 훈련된 신경과 전문의의 시각적 검사에 의존

이는 피로나 주관적 판단 등의 요인에 의해 해석이 왜곡될 가능성이 있음을 뜻한다

또한 여러 신경과 전문의들의 해석이 갈릴 가능성도 있다.

→ 자동화 기술, AI를 이용하여 빠르고 일관된 해석을 하려는 시도 증가

주요 EEG 이상 패턴

Seizure(SZ): 발작

Generalized Periodic Discharges(GPD): 전반적 주기적 방전.

뇌 전체에서 주기적으로 반복되는 이상 신호가 나타남

Lateralized Periodic Discharges(**LPD**): 국소성 주기적 방전.

뇌의 한쪽에서만 주기적인 이상 신호가 나타남

Lateralized Rhythmic Delta Activity(**LRDA**):

특정 뇌 부위에서 델타파(느린 뇌파) 규칙적으로 반복함

Generalized Rhythmic Delta Activity(**GRDA**) : 뇌 전체에서 델타파가 규칙적으로 나타남

Other patterns : 기타 비정상 패턴

GPD

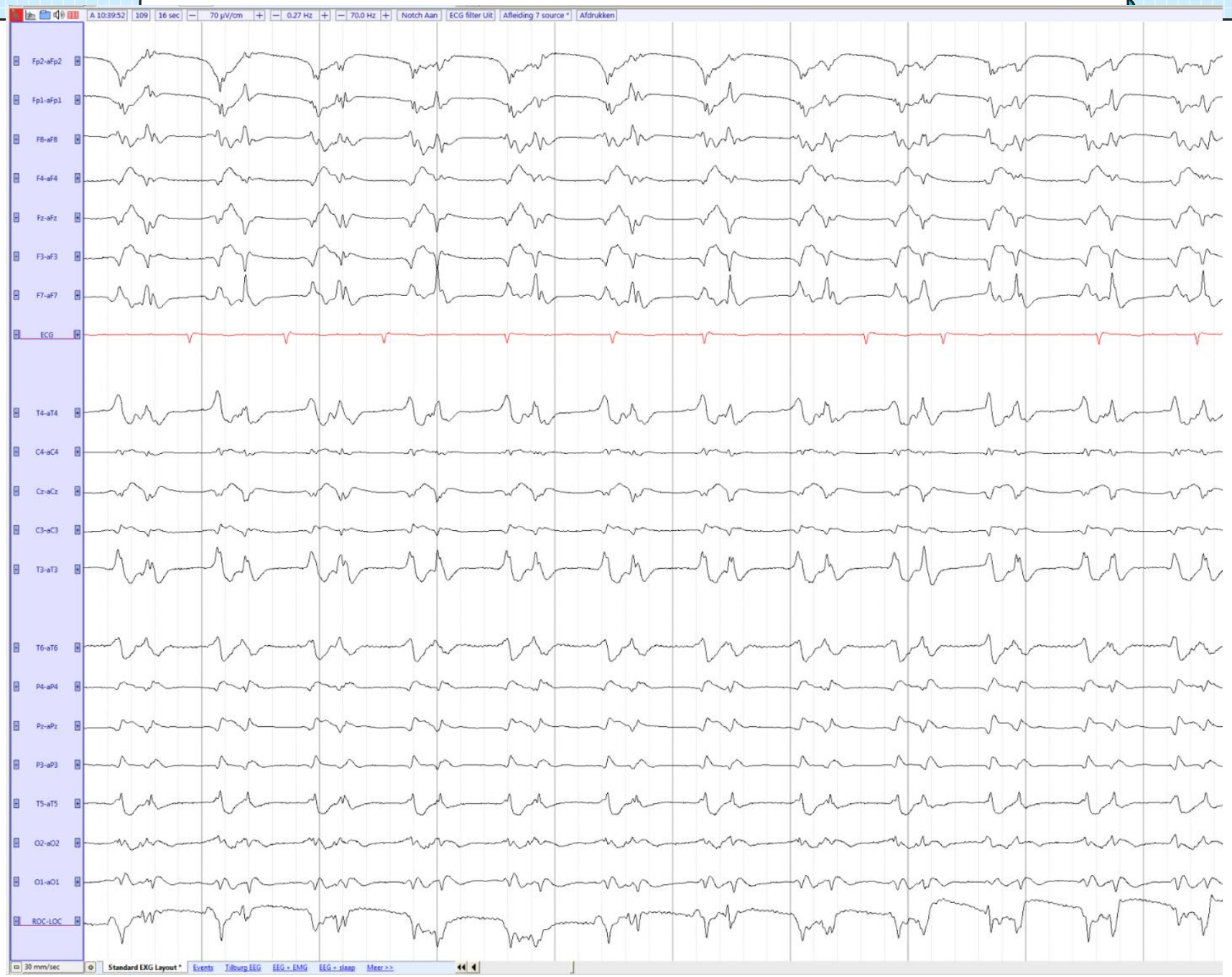
양쪽 뇌 반구에서 광범위하게 발생

반복적인 패턴을 보이며, EEG에서 다른 전반적 이상 패턴과 구별됨

반복적인 날카로운 윤곽의 파형

다른 EEG 패턴보다 동기화가 잘 되어 있음

전반적인 뇌 기능 장애의 심각성을 나타내는 지표


A 10:39:52
109
16 sec
70 µV/cm
0.27 Hz
70.0 Hz
Notch Aan
ECG filter Uit
Afleiding 7 source *
Afdrukken
Fp2-aFp2
Fp1-aFp1
F8-aF8
F4-aF4
Fz-aFz
F3-aF3
F7-aF7
ECG
T4-aT4
C4-aC4
Cz-aCz
C3-aC3
T3-aT3
T6-aT6
P4-aP4
Pz-aPz
P3-aP3
T5-aT5
O2-aO2
O1-aO1
ROC-LOC
30 mm/sec
Standard EXG Layout *
Events
Tilburg EEG
EEG + EMG
EEG + slaap
Meer >>

LPD

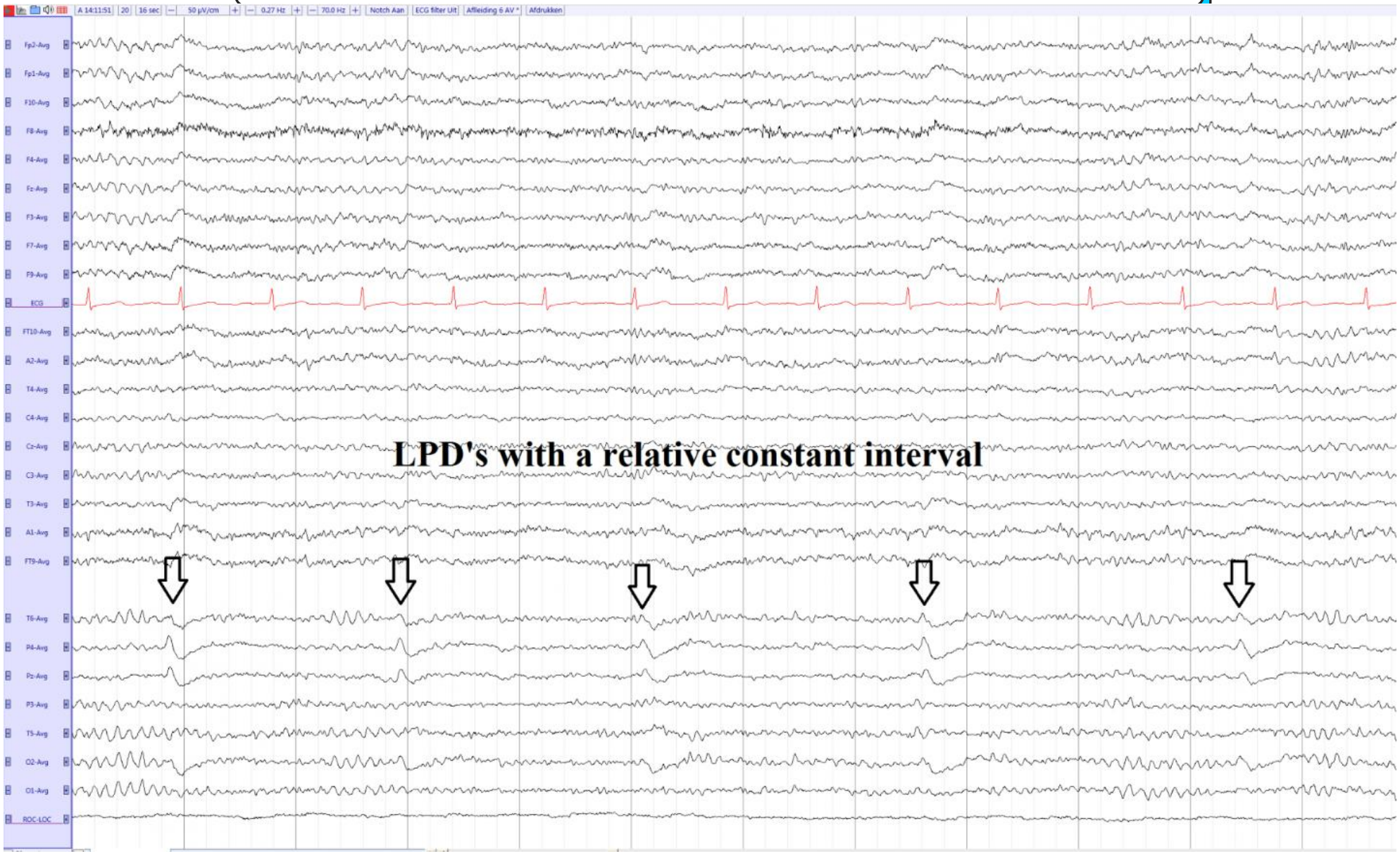
한쪽 뇌 반구에서 주로 발생

규칙적인, 뚜렷한 윤곽을 가진 파형이 반복적으로 나타남

날카로운 파형 또는 복합 파형 – EEG의 배경활동과 명확히 구별됨

일반적으로 느린 파형을 동반

주요 EEG 이상 패턴



LRDA

한쪽 뇌 반구에서만 관찰됨

국소적 뇌 병변이 있는 뇌 손상의 위치를 추정하는 데 유용

리드믹 델타파: 주로 1-4Hz의 델타 주파수 영역에서 발생

LPD와 달리 날카로운 파형이 아닌 부드러운 형태를 가짐

뇌졸중, 종양, 국소 염증, 국소 발작과 관련 있음

GRDA

LRDA와 달리, 특정 반구에 국한되지 않고 양쪽 뇌 반구에서 대칭적으로 발생

리드믹 델타파: GPD보다 더 느리고 일정한 간격으로 반복되는 파형

독성-대사성 뇌병증, 확산성 뇌 기능 장애, 저산소성 손상

일부 수면 단계에서도 나타날 수 있음

전반적인 뇌 기능 저하를 의미

주요 EEG 이상 패턴

EEG 패턴	주된 특징	주요 연관 질환
LPD (측방성 주기 방전)	한쪽 뇌 반구에서 발생, 주기적 sharp wave	급성 뇌 손상, 발작
GPD (전반성 주기 방전)	양쪽 뇌 반구에서 발생, 주기적 sharp wave	독성-대사성 뇌병증, 저산소성 뇌손상
LRDA (측방성 리드믹 델타 활동)	한쪽 뇌 반구에서 발생, 리드믹 델타파	뇌졸중, 종양, 염증
GRDA (전반성 리드믹 델타 활동)	양쪽 뇌 반구에서 발생, 리드믹 델타파	독성-대사성 뇌병증, 수면, 저산소성 손상

Data load

```
BASE_PATH = '/kaggle/input/hms-harmful-brain-activity-classification/'
df = pd.DataFrame({'path': glob(BASE_PATH + '**/*.parquet')})
#df = pd.read_parquet("파일경로.parquet") → 이런 식으로도 .parquet 파일 호출 가능
```

•glob(): glob 모듈의 glob 함수는 사용자가 제시한 조건에 맞는 파일 명을 리스트 형식으로 반환
•**/*.parquet: 모든 하위 폴더도 포함해서 .parquet 파일 찾기

path
/kaggle/input/.../train_eegs/test_1/1000913311.parquet
/kaggle/input/.../train_eegs/test_2/1000913362.parquet
...

parquet

Parquet(파케이) : **데이터를 효율적으로 저장하고 빠르게 처리할 수 있도록 설계된 파일 형식**
보통 데이터를 저장할 때 **CSV** 같은 일반적인 텍스트 파일을 사용하지만, 대량의 데이터를 다룰 때는 **저장 공간을 많이 차지하고, 처리 속도가 느려지는 문제**가 생긴다.
👉 **Parquet는 이런 문제를 해결하기 위해 만들어졌어요!**

✅ **컬럼 단위 저장 (Columnar Storage)**
CSV는 행 단위로 저장되지만, Parquet는 열(컬럼) 단위로 저장합니다. 필요한 열만 빠르게 읽을 수 있어서 성능이 좋아요!
✅ **압축 효율이 뛰어남**
같은 타입의 데이터(예: 숫자, 날짜)가 연속적으로 저장되므로 압축이 잘 돼서 파일 크기가 작아져요!
✅ **빨리 읽을 수 있음**
데이터가 필요한 부분만 읽을 수 있도록 최적화되어 있어서 속도가 빠릅니다.
예를 들어, "이름" 컬럼만 보고 싶을 때 CSV는 모든 데이터를 읽어야 하지만, Parquet는 "이름" 부분만 딱 읽으면 됩니다.

Data load

```
df['test_type'] = df['path'].str.split('/').str.get(-2).str.split('_').str.get(-1)
```

•각 파일의 경로의 상위 폴더 이름에서 test type 추출
•예: /train_eegs/test_1/1000913311.parquet라면 test_1 → _ 기준 split → 1 추출
•test type 번호를 담는 'test_type' 열 df에 추가

```
df['id'] = df['path'].str.split('/').str.get(-1).str.split('.').str.get(0)
```

•각 파일 이름에서 ID 값만 추출
•예: 1000913311.parquet → 1000913311
•ID 정보를 담은 'id' 열 df에 추가

path	test_type	id
.../test_1/1000913311.parquet	1	1000913311
.../test_2/1000913362.parquet	2	1000913362

Data load

```
df_eeg = pd.read_parquet(BASE_PATH + 'train_eegs/1000913311.parquet')
df_eeg.head()
```

	Fp1	F3	C3	P3	F7	T3	T5	O1	Fz	Cz
0	-105.849998	-89.230003	-79.459999	-49.230000	-99.730003	-87.769997	-53.330002	-50.740002	-32.250000	-42.099998
1	-85.470001	-75.070000	-60.259998	-38.919998	-73.080002	-87.510002	-39.680000	-35.630001	-76.839996	-62.740002
2	8.840000	34.849998	56.430000	67.970001	48.099998	25.350000	80.250000	48.060001	6.720000	37.880001
3	-56.320000	-37.279999	-28.100000	-2.820000	-43.430000	-35.049999	3.910000	-12.660000	8.650000	3.830000
4	-110.139999	-104.519997	-96.879997	-70.250000	-111.660004	-114.430000	-71.830002	-61.919998	-76.150002	-79.779999

Data load

```
df = pd.read_csv('/kaggle/input/hms-harmful-brain-activity-classification/train.csv')
TARGETS = df.columns[-6:]
print('Train shape:', df.shape )
print('Targets', list(TARGETS))
df.head()
```

```
Train shape: (106800, 15)
Targets ['seizure_vote', 'lpd_vote', 'gpd_vote', 'lrda_vote', 'grda_vote', 'other_vote']
```

Data load

	eeg_id	eeg_sub_id	eeg_label_offset_seconds	spectrogram_id	spectrogram_sub_id	spectrogram_label_offset_seconds
0	1628180742	0	0.0	353733	0	0.0
1	1628180742	1	6.0	353733	1	6.0
2	1628180742	2	8.0	353733	2	8.0
3	1628180742	3	18.0	353733	3	18.0
4	1628180742	4	24.0	353733	4	24.0

eeg_id	EEG 기록 전체에 대한 고유 식별자.
eeg_sub_id	이 행의 라벨이 적용되는 특정 50초 길이의 하위 샘플에 대한 ID 입니다.
eeg_label_offset_seconds	통합 EEG 시작과 이 하위 샘플 사이의 시간입니다.
spectrogram_id	EEG 기록 전체에 대한 고유 식별자.
spectrogram_sub_id	이 행의 라벨이 적용되는 특정 10분 하위 샘플에 대한 ID입니다.
spectogram_label_offset_seconds	통합 스펙트로그램의 시작과 이 하위 샘플 사이의 시간입니다.

Data load

label_id	patient_id	expert_consensus	seizure_vote	lpd_vote	gpd_vote	lrda_vote	grda_vote	other_vote
127492639	42516	Seizure	3	0	0	0	0	0
3887563113	42516	Seizure	3	0	0	0	0	0
1142670488	42516	Seizure	3	0	0	0	0	0
2718991173	42516	Seizure	3	0	0	0	0	0
3080632009	42516	Seizure	3	0	0	0	0	0

label_id	이 라벨 세트에 대한 ID입니다.
patient_id	데이터를 기증한 환자의 ID입니다.
expert_consensus	합의 주석자 라벨

```
train = df.groupby('eeg_id')[['spectrogram_id','spectrogram_label_offset_seconds']].agg(
    {'spectrogram_id':'first','spectrogram_label_offset_seconds':'min'})
train.columns = ['spec_id','min']
```

'eeg_id'로 데이터를 묶어서,

각 그룹의 첫 번째 'spectrogram_id ' 와 가장 작은 'spectrogram_label_offset_seconds'를 뽑아서

새로운 데이터프레임 train을 만들고, 열 이름을 'spec_id'와 'min'으로 바꿈

```
tmp = df.groupby('eeg_id')[['spectrogram_id', 'spectrogram_label_offset_seconds']].agg(
    {'spectrogram_label_offset_seconds': 'max'})
train['max'] = tmp
```

train 데이터프레임에 max 열로 추가

```
tmp = df.groupby('eeg_id')[['patient_id']].agg('first')
train['patient_id'] = tmp
```

eeg_id에 대해 첫 번째 patient_id 값을 가져와 train 데이터프레임에 추가

```
tmp = df.groupby('eeg_id')[TARGETS].agg('sum')
for t in TARGETS:
    train[t] = tmp[t].values

y_data = train[TARGETS].values
y_data = y_data / y_data.sum(axis=1, keepdims=True)
train[TARGETS] = y_data
```

'eeg_id'별로 TARGETS 열들의 합계를 구해서 tmp에 저장.

그 합계를 train에 옮김.

train에서 TARGETS 값을 배열로 뽑아서(y_data) 정규화

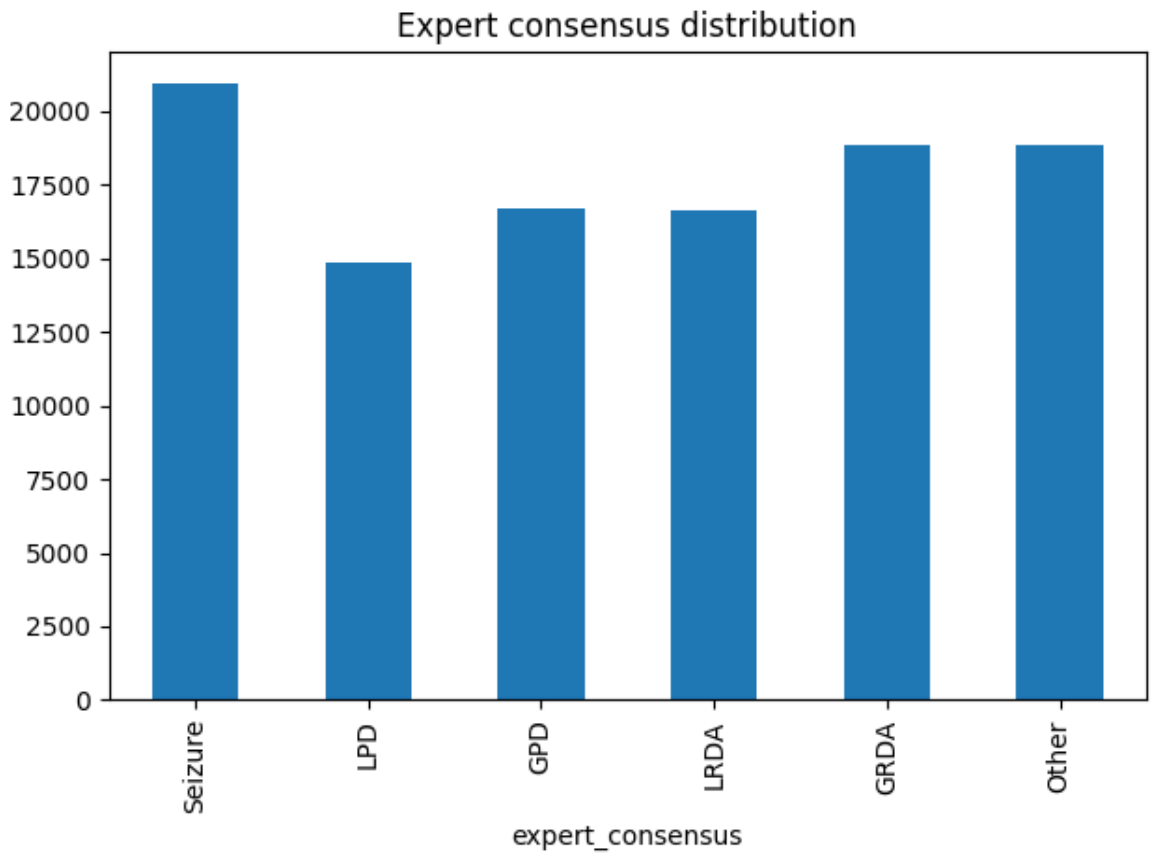
그 비율을 다시 train에 넣음.

```
tmp = df.groupby('eeg_id')[['expert_consensus']].agg('first')
train['target'] = tmp
```

각 eeg_id에 대해 첫 번째 expert_consensus 값을 가져와 train 데이터프레임에 target 열로 추가

	eeg_id	spec_id	min	max	patient_id	seizure_vote	lpd_vote	gpd_vote	lrda_vote	grda_vote	other_vote	target
0	568657	789577333	0.0	16.0	20654	0.0	0.000000	0.25	0.000000	0.166667	0.583333	Other
1	582999	1552638400	0.0	38.0	20230	0.0	0.857143	0.00	0.071429	0.000000	0.071429	LPD
2	642382	14960202	1008.0	1032.0	5955	0.0	0.000000	0.00	0.000000	0.000000	1.000000	Other
3	751790	618728447	908.0	908.0	38549	0.0	0.000000	1.00	0.000000	0.000000	0.000000	GPD
4	778705	52296320	0.0	0.0	40955	0.0	0.000000	0.00	0.000000	0.000000	1.000000	Other

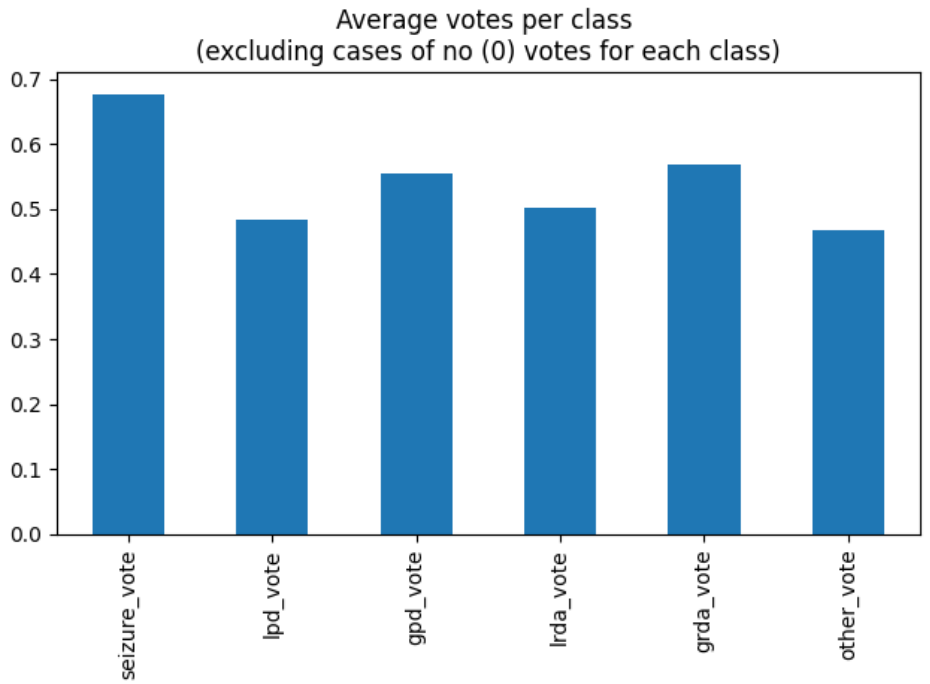
```
# distribution of expert consensus
df['expert_consensus'].value_counts()[['Seizure', 'LPD', 'GPD', 'LRDA', 'GRDA', 'Other']].plot(ki
nd='bar', title="Expert consensus distribution")
plt.tight_layout()
plt.savefig('/kaggle/working/expertConsensusDistribution.png')
```



환자들에게 가장 많이 내려진 최종 진단은 무엇인가?

특정 진단이 얼마나 자주 등장하는지 확인

```
# distrubution of average expert votes per class
votes = df[['seizure_vote', 'lpd_vote', 'gpd_vote', 'lrda_vote', 'grda_vote', 'other_vote']].appl
y(lambda x: x/x.sum(), axis=1)
# print(votes[:10])
votesMean = votes.replace(0, np.NaN).mean(axis=0)
votesMean.plot(kind='bar', title='Average votes per class \n(excluding cases of no (0) votes for
each class)')
plt.tight_layout()
plt.savefig('/kaggle/working/averageVoteDistribution.png')
```



0을 NaN으로 변경하여 제외 후 평균값 계산

각 진단(Seizure, LPD 등)이 전문가 투표에서 평균적으로 얼마나 높은 비율로 선택되었는가?

특정 진단이 전문가들 사이에서 얼마나 강하게 진단받는지 확인

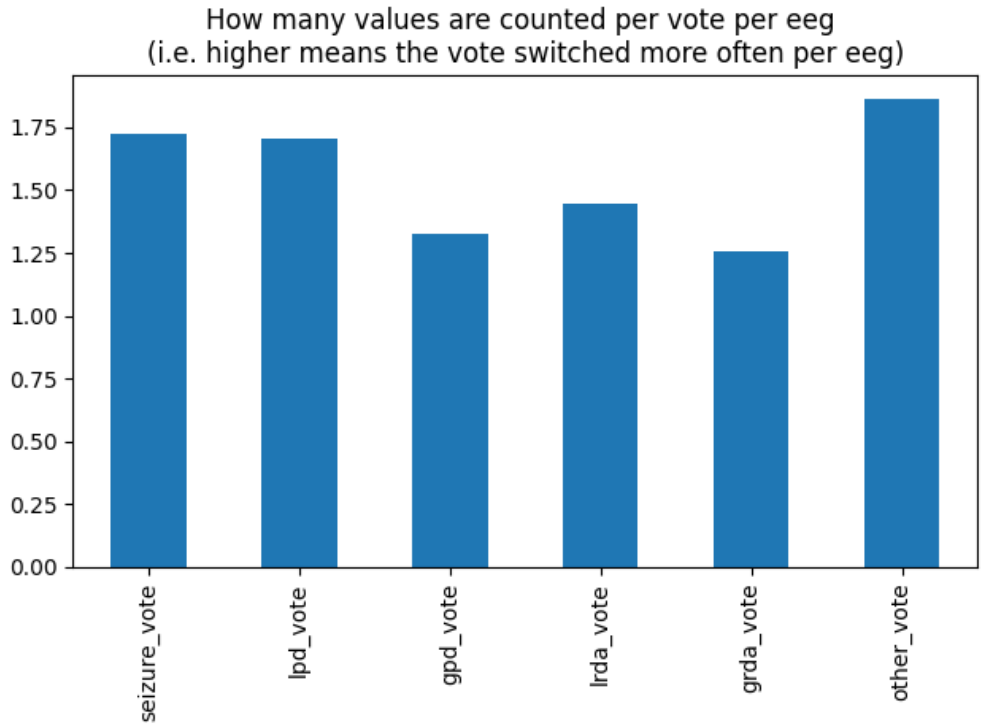
```
V = ['seizure_vote', 'lpd_vote', 'gpd_vote', 'lrda_vote', 'grda_vote', 'other_vote']
df = pd.read_csv('/kaggle/input/hms-harmful-brain-activity-classification/train.csv')
votes = df[['eeg_id', 'eeg_label_offset_seconds', 'seizure_vote', 'lpd_vote', 'gpd_vote', 'lrda_v
ote', 'grda_vote', 'other_vote']]

# counts the number of votes per class per eeg_id
votes = votes.groupby('eeg_id')[V].nunique().reset_index()

#if anywhere there were more than 1 different value in the votes, that means it changed
votes = votes.assign(changes = lambda x: x['seizure_vote'] + x['lpd_vote'] + x['gpd_vote'] + x['l
rda_vote'] + x['grda_vote'] + x['other_vote'] > 6)
votesChange = votes[votes['changes']==True]
votesSame = votes[votes['changes']==False]
```

같은 EEG 기록에서 시간이 지나면서 전문가들의 투표가 변하는지를 분석하는 것

특정 진단(예: Other)이 시간에 따라 전문가들 사이에서 더 자주 의견이 바뀌는 것을 확인 가능

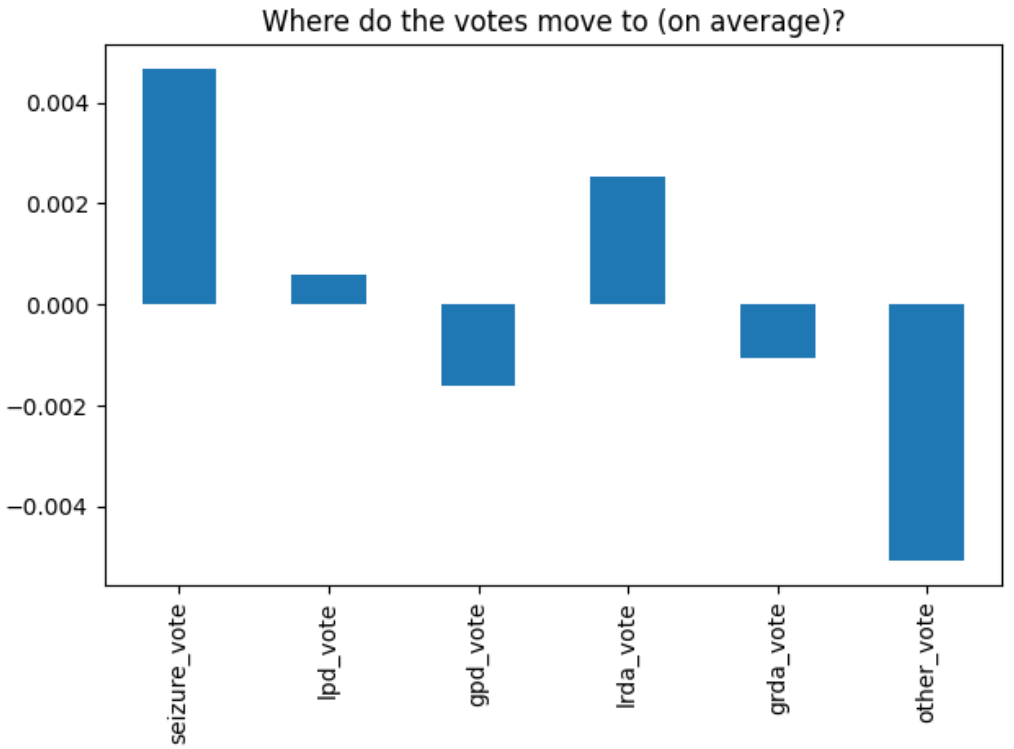
EEG 패턴이 안정적인 경우(예: GRDA, GPD) 투표 변동이 거의 없음

이 분석은 EEG 데이터의 신뢰도 평가나 전문가 간의 합의 정도를 파악하는 데 유용할 수 있음



```
# calculate percentage of votes per class
votesDevelopment[V] = votesDevelopment[V].apply(lambda x: x/x.sum(), axis=1)

# sort by offset per eeg_id so that we really only go forward in time
votesDevelopment = votesDevelopment.sort_values(by=['eeg_id', 'eeg_label_offset_seconds'])

# group by eeg_id and then transform to see the differences
votesDevelopment[V] = votesDevelopment.groupby(['eeg_id'])[V].transform(lambda x: x.diff())

# drop the NaN values which are always given to the first row of diff
votesDevelopment = votesDevelopment.dropna()

# to only take the times into account when things change we remove cases of 0 (no change)
votesDevelopment = votesDevelopment[(votesDevelopment[V] != 0).any(axis=1)]
votesDevelopmentMean = votesDevelopment[V].mean(axis=0)
```

"투표가 변할 때, 어떤 진단이 증가하고 어떤 진단이 감소하는지"를 분석



EEG를 처음 분석할 때, 전문가들이 "Other"로 판단하는 경우가 많음

시간이 지나면서 특정 패턴이 명확해지고, 전문가들의 의견이 "Seizure"로 변경되는 경향이 있음.

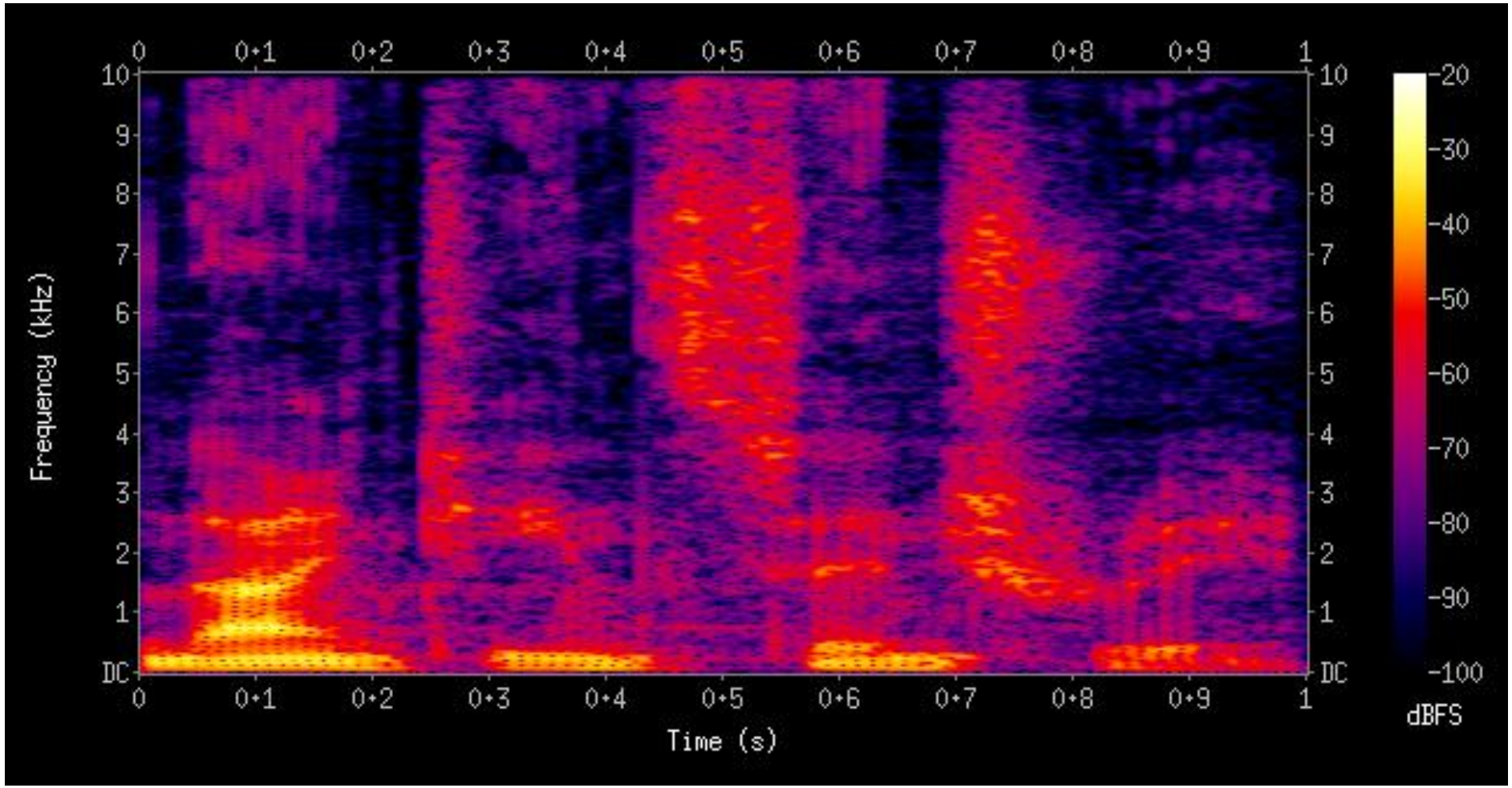
이는 EEG 데이터를 해석할 때
시간에 따른 변화도 고려해야 한다는 점을 시사.

- Kaggle 제공 train data 기반 Spectrogram 바탕으로 진행
- https://www.kaggle.com/datasets/cdeotte/brain-spectrograms
- 1600개 feature 생성 (400 x 2 x 2)
  - 주파수를 400개 구간으로 분할
  - 각 주파수 구간마다 10분, 20초 window(시간 구간)
  - 평균과 최솟값 계산
- 긴 구간(큰 흐름)과 짧은 구간(세부적 특징) 모두를 모델이 활용할 수 있는 특성이 만들어짐

Spectrogram

- 소리나 파동을 시각화하여 파악하기 위한 도구
- 파형과 스펙트럼의 특징이 결합됨
- 파형: 시간 축의 변화에 따른 진폭 축의 변화
- 스펙트럼: 주파수 축의 변화에 따른 진폭 축의 변화
- 스펙트로그램:
  시간 축과 주파수 축의 변화에 따라 진폭의 차이를
  인쇄 농도/ 표시 색상의 차이로 나타냄
  x축 – 시간, y축 – 주파수, z축 - 진폭

Spectrogram



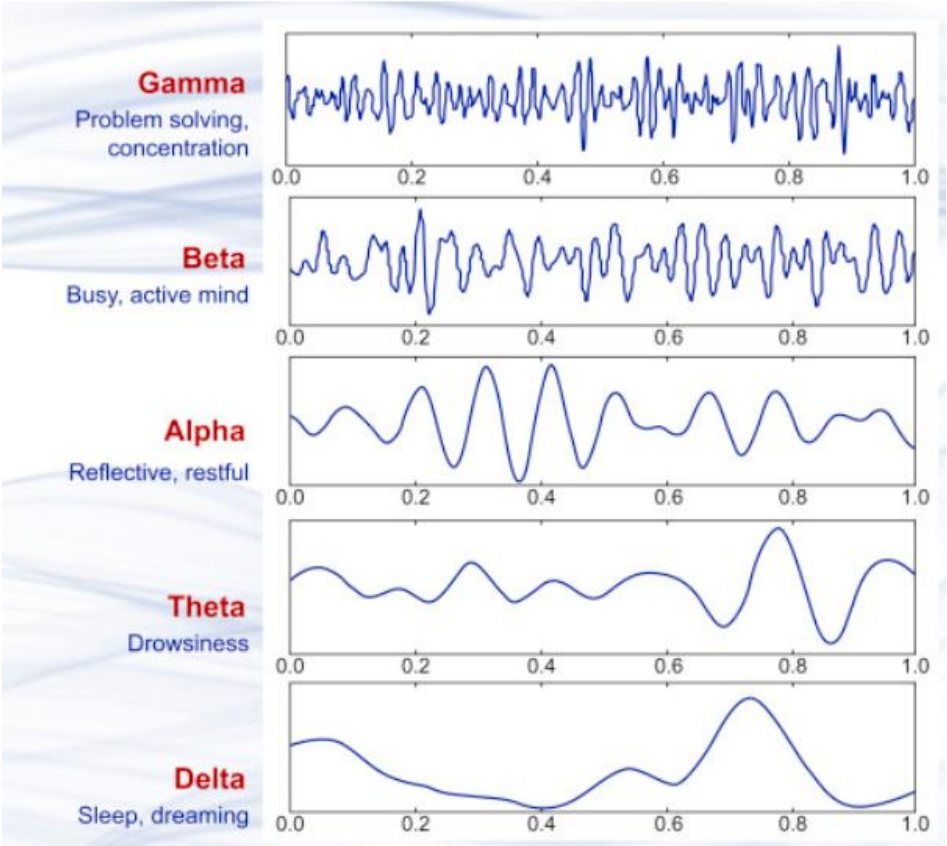
Feature Engineering

```
if FEATURE_ENGINEER:
    data = np.zeros((len(train),len(FEATURES)))
    for k in range(len(train)):
        if k%100==0: print(k,', ',end='')
        row = train.iloc[k]
        r = int( (row['min'] + row['max'])//4 )

        # 10 MINUTE WINDOW FEATURES (MEANS and MINS)
        x = np.nanmean(spectrograms[row.spec_id][r:r+300,:],axis=0)
        data[k,:400] = x
        x = np.nanmin(spectrograms[row.spec_id][r:r+300,:],axis=0)
        data[k,400:800] = x

        # 20 SECOND WINDOW FEATURES (MEANS and MINS)
        x = np.nanmean(spectrograms[row.spec_id][r+145:r+155,:],axis=0)
        data[k,800:1200] = x
        x = np.nanmin(spectrograms[row.spec_id][r+145:r+155,:],axis=0)
        data[k,1200:1600] = x
```

Frequency band feature extraction

- EEG 데이터 segment에서 주파수 대역별로 신호를 필터링
- 각 대역의 통계적 특성 (평균, 표준편차, 최댓값, 최솟값) 추출
- 각 채널당 총 20개 특성 생성

```
def extract_frequency_band_features(segment):
    # Define EEG frequency bands
    eeg_bands = {'Delta': (0.5, 4), 'Theta': (4, 8), 'Alpha': (8, 12), 'Beta': (12, 30), 'Gamma':
(30, 45)}

    band_features = []
    for band in eeg_bands:
        low, high = eeg_bands[band]
        # Filter signal for the specific band
        band_pass_filter = signal.butter(3, [low, high], btype='bandpass', fs=200, output='sos')
        filtered = signal.sosfilt(band_pass_filter, segment)
        # Extract features like mean, standard deviation, etc.
        band_features.extend([np.nanmean(filtered), np.nanstd(filtered), np.nanmax(filtered), np.
nanmin(filtered)])

    return band_features
```



Feature Aggregation and PCA

- 추출된 특성 중 결측치는 평균값으로 대체
- 정리된 데이터를 PCA(주성분 분석)을 통해 더 낮은 차원으로 축소
- PCA를 진행한 특성을 원본 데이터셋 (train)에 추가
- What is PCA?

출처

https://brunch.co.kr/@looxidlabs/16
https://www.kaggle.com/code/mariesophiesimon/dataexploration
https://www.kaggle.com/code/mvvppp/hms-eda-and-domain-journey
https://www.kaggle.com/code/yorkyong/exploring-eeg-a-beginner-s-guide