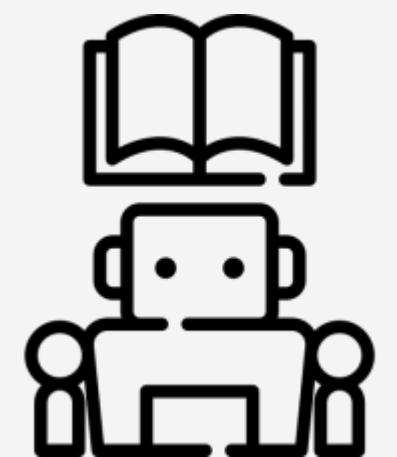


---

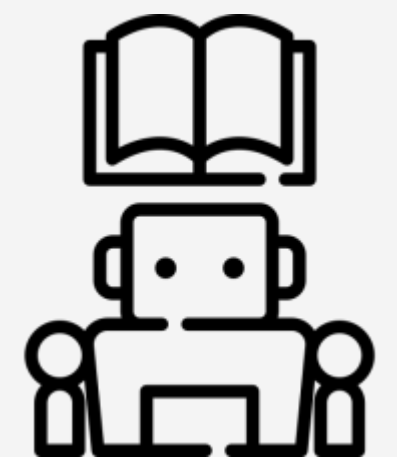
# Machine Learning Term Project #3

음성 감정 인식



---

# Overview

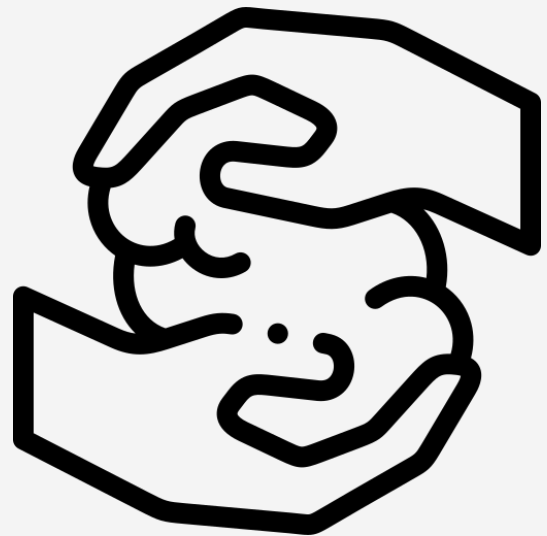


# [Overview] - 프로젝트 방향성 이해하기

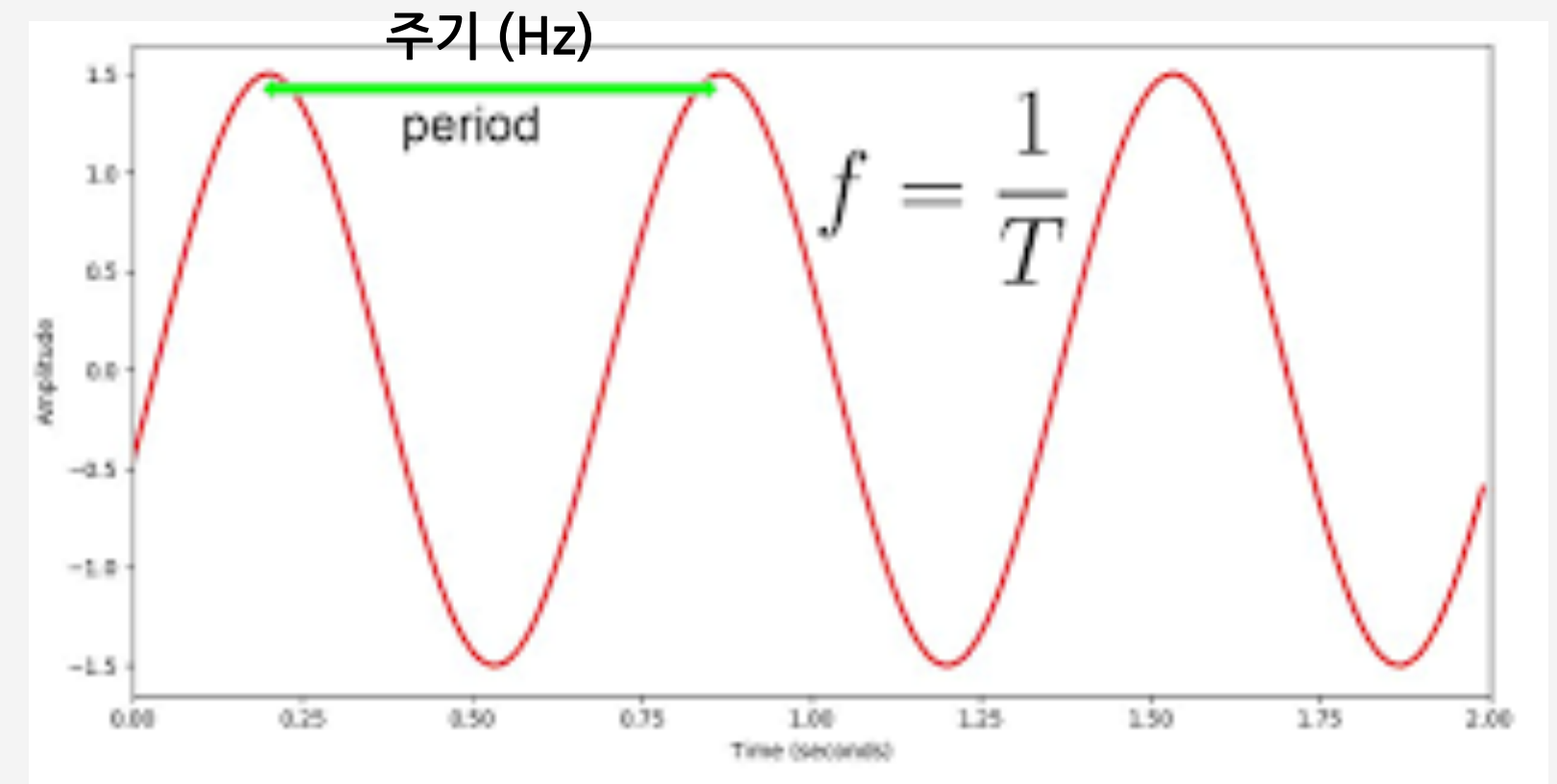
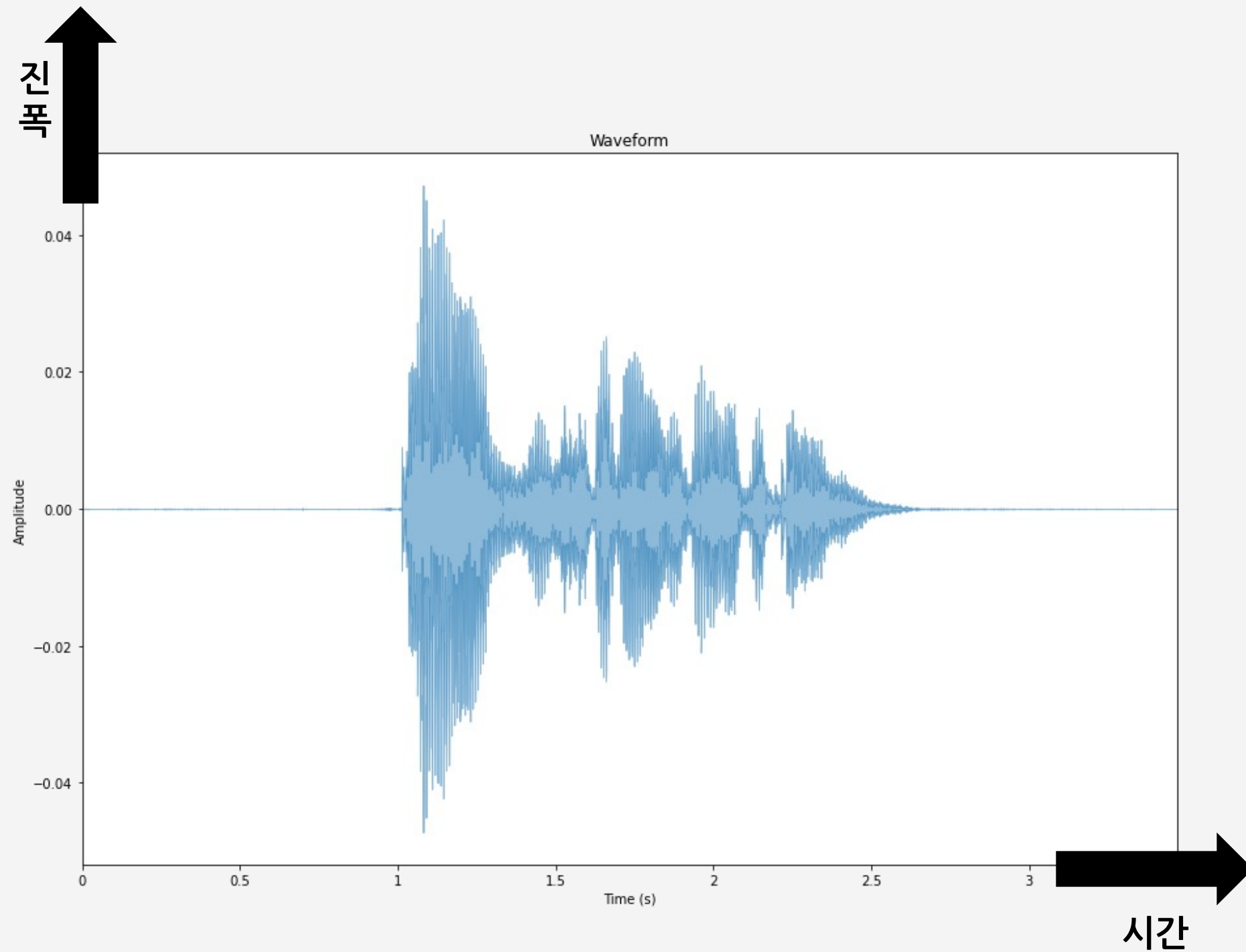
---

## 프로젝트의 목적

본 프로젝트를 통해 1D 음성 데이터를  
Handcrafted Feature 로 기술하는 법을 알 수 있다.



# [Overview] - 1D 음성 데이터 다루기 (감정 분류기)



Frequency → 소리의 고음, 저음

Amplitude → 소리의 크기



# [Overview] - 1D 음성 데이터 다루기 (감정 분류기)

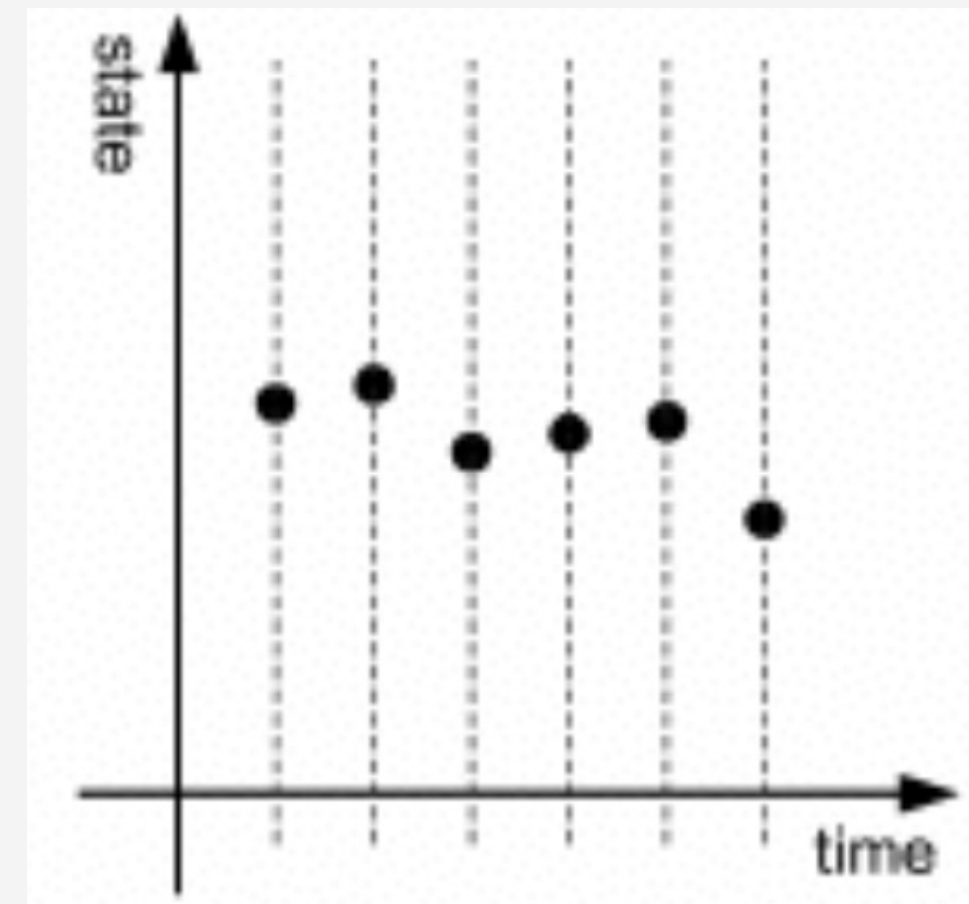
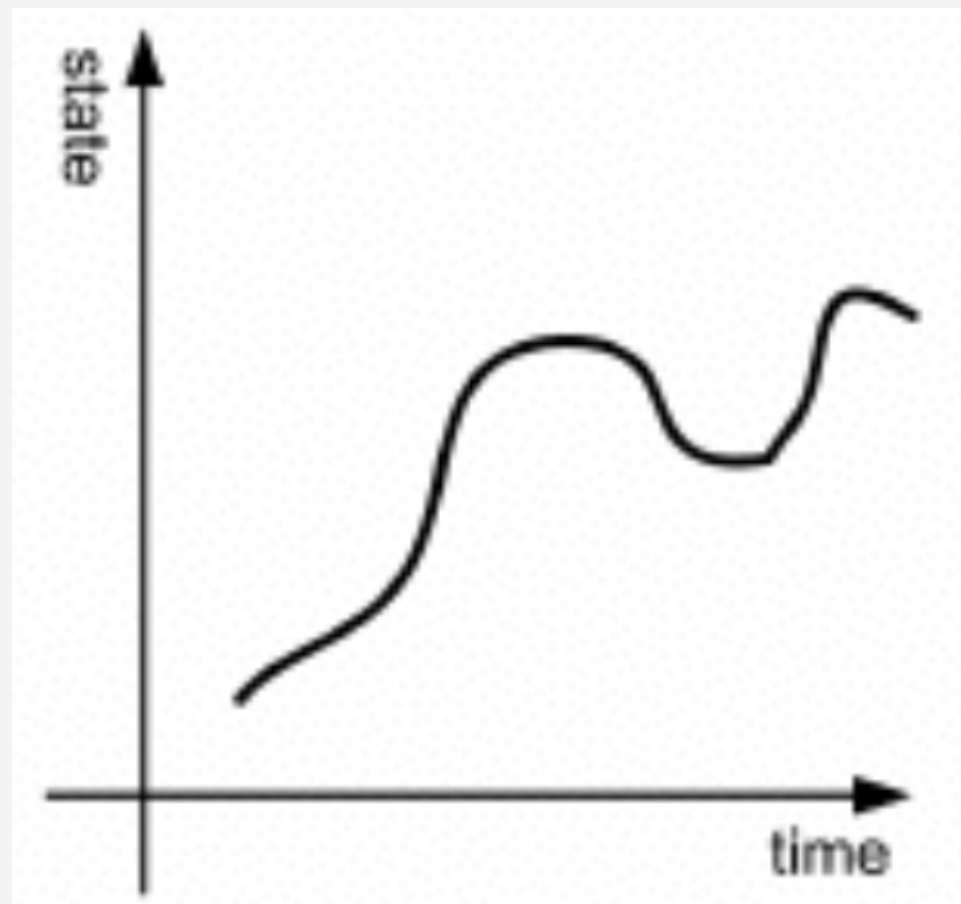


연속형  
(continuous)



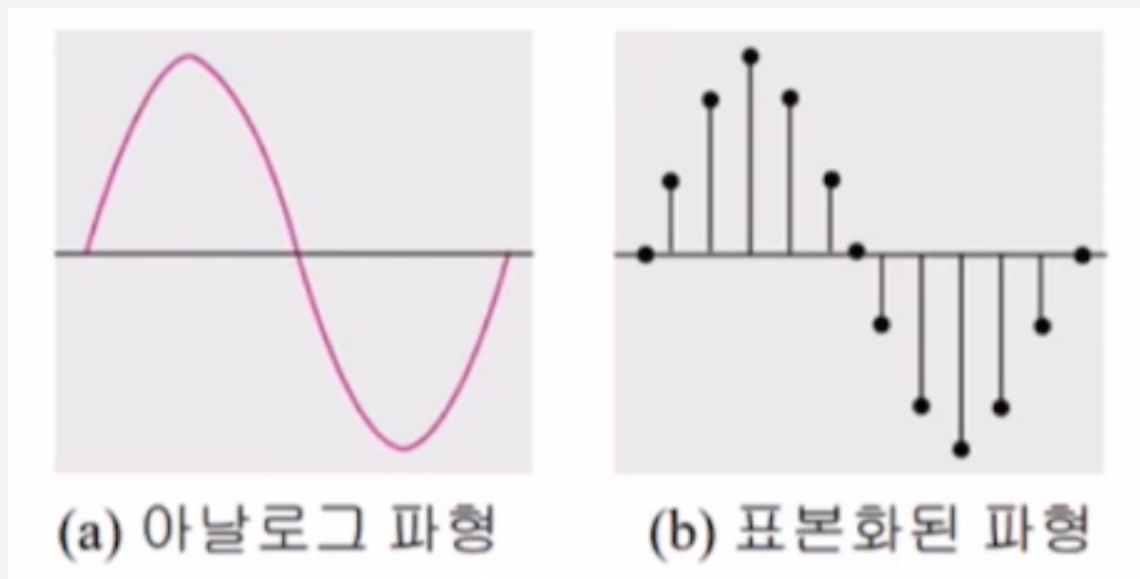
- ① sampling
- ② quantization(양자화)

이산형  
(discrete)



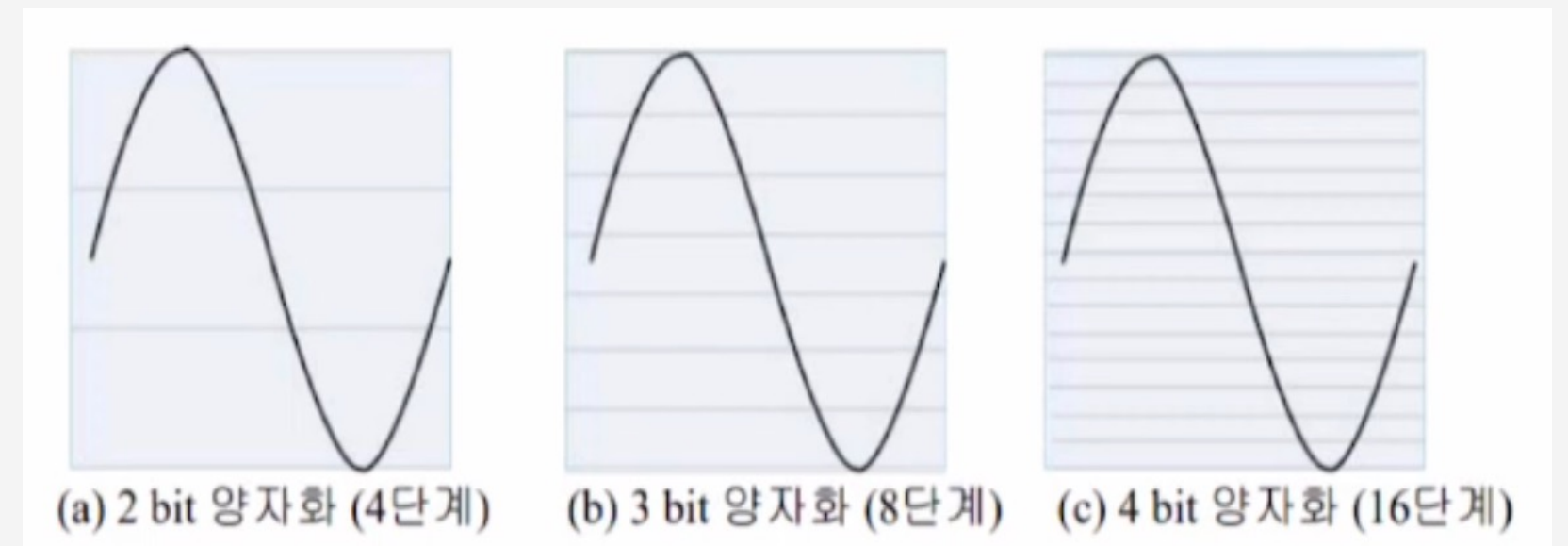
# [Overview] - 1D 음성 데이터 다루기 (감정 분류기)

## ① sampling



→ 주기

## ② quantization

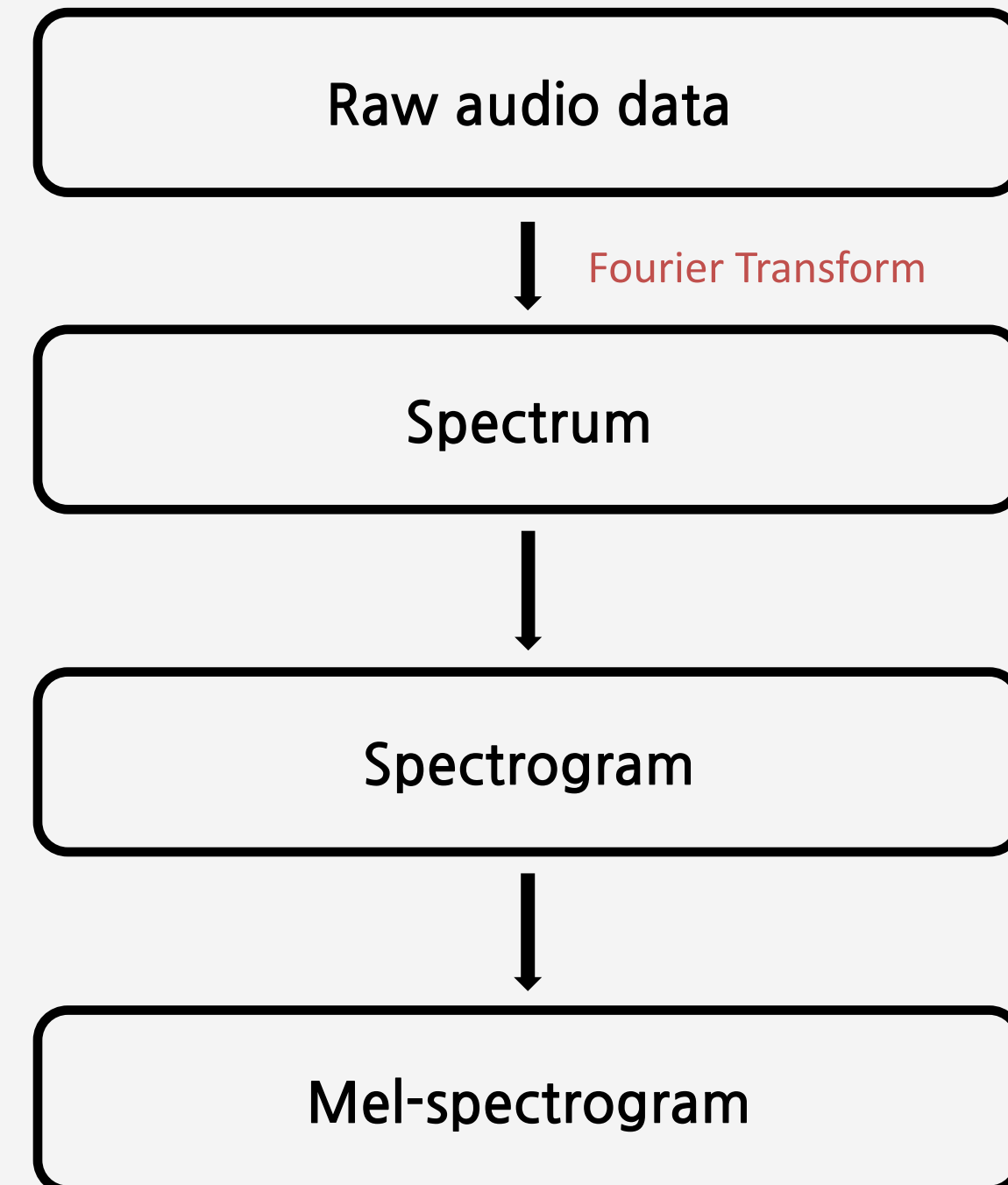
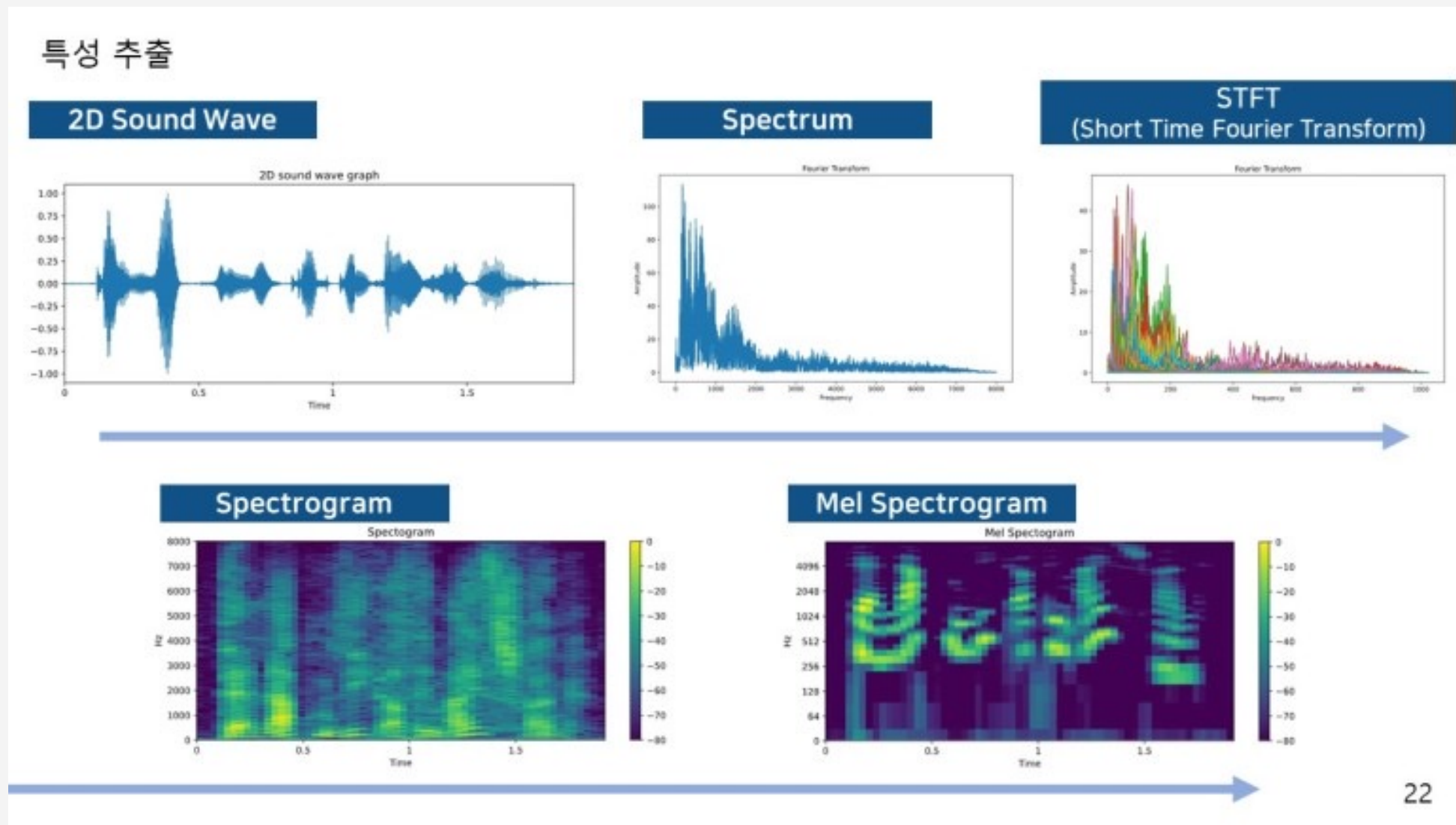


→ 진폭



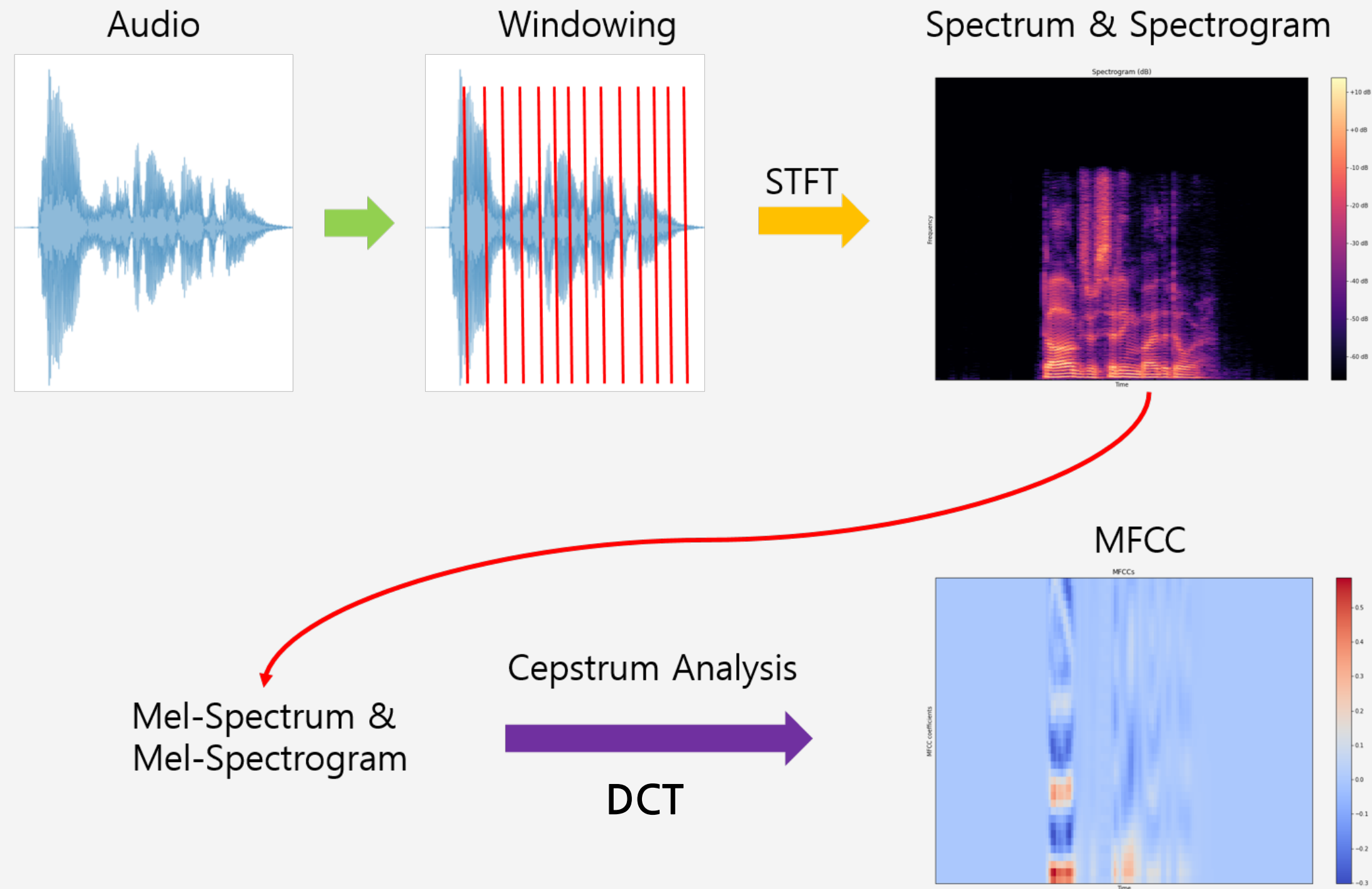
# [Overview] - 1D 음성 데이터 다루기 (감정 분류기)

## Mel-Spectrogram



# [Overview] - 1D 음성 데이터 다루기 (감정 분류기)

## MFCC (Mel Frequency Ceptral Coefficients)

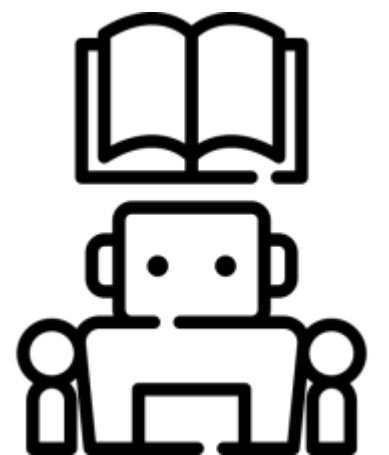




---

# Empty Module #1. 2

**학습용, 평가용 데이터 로더**



# [Empty Module #1] - 학습용 데이터 로더

```
from tqdm.notebook import tqdm
def load_data(data_info, isTrain=True):
    PATH = join('/kaggle', 'input', '2022-ml-project3')
    if isTrain:
        train_data = {'spectrogram': [], 'mel': [], 'mfcc': []} #음성 feature들을 담은 dictionary
        train_label = [] #학습에 사용할 label을 담은 list

        file_list = data_info['file_name']
        emotion_list = data_info['emotion']

        for file_name, emotion in tqdm(zip(file_list, emotion_list)): # zip을 통해 2개 이상의 변수를 한 번에 넘길 수 있다.

            # train.csv 파일에 있는 음성 파일의 이름과 emotion 정보를 통하여 학습용 데이터를 로드
            path = join(PATH, 'train_data/train_data', file_name)
            train_label.append(emotion)

            # 음성 파일의 정확한 경로 명을 extract_feature 함수의 입력으로 넣음
            spectrogram_feature, mel_spectrogram_feature, mfcc_feature = extract_feature(path)

            # extract_feature를 통해 구한 음성 feature들을 train_data 사전 속 배열에 알맞게 append
            train_data['spectrogram'].append(spectrogram_feature)
            train_data['mel'].append(mel_spectrogram_feature)
            train_data['mfcc'].append(mfcc_feature)

    return train_data, np.array(train_label)
```



## [Empty Module #2] - 평가용 데이터 로더

---

```
else:
    test_data = {'spectrogram':[], 'mel':[], 'mfcc':[]} #음성 feature들을 담은 dictionary
    file_list = data_info['file_name']

    for file_name in tqdm(file_list):

        # test.csv 파일에 있는 음성 파일의 이름정보를 통하여 평가용 데이터를 로드
        path = join(PATH, 'test_data/test_data', file_name)

        # 음성 파일의 정확한 경로 명을 extract_feature 함수의 입력으로 넣음
        spectrogram_feature, mel_spectrogram_feature, mfcc_feature = extract_feature(path)

        # extract_feature를 통해 구한 음성 feature들을 test_data 사전 속 배열에 알맞게 append
        test_data['spectrogram'].append(spectrogram_feature)
        test_data['mel'].append(mel_spectrogram_feature)
        test_data['mfcc'].append(mfcc_feature)

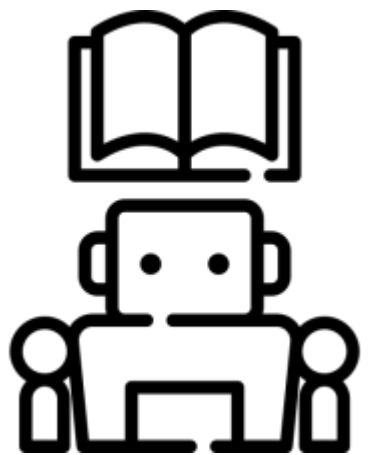
    return test_data
```



---

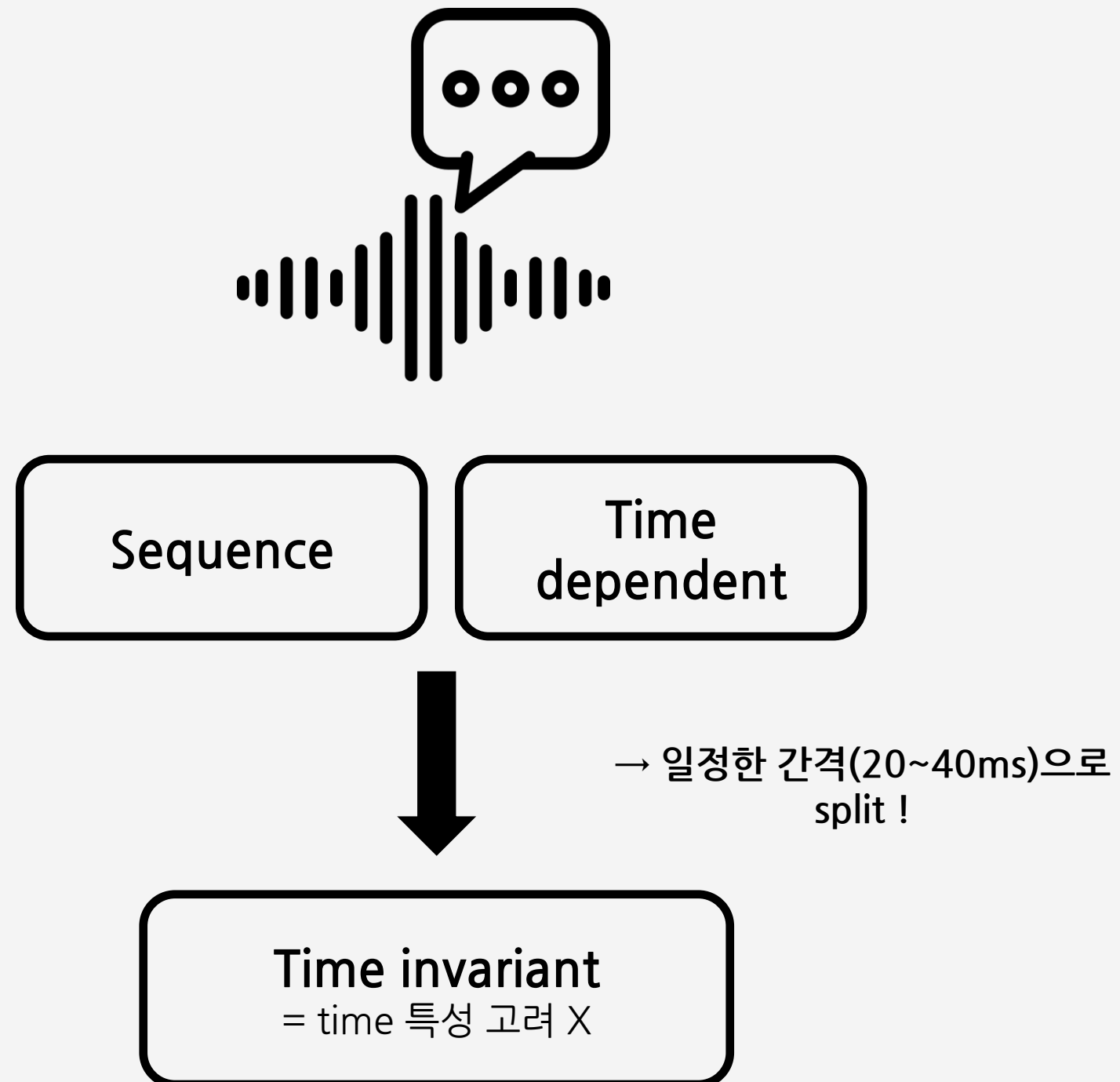
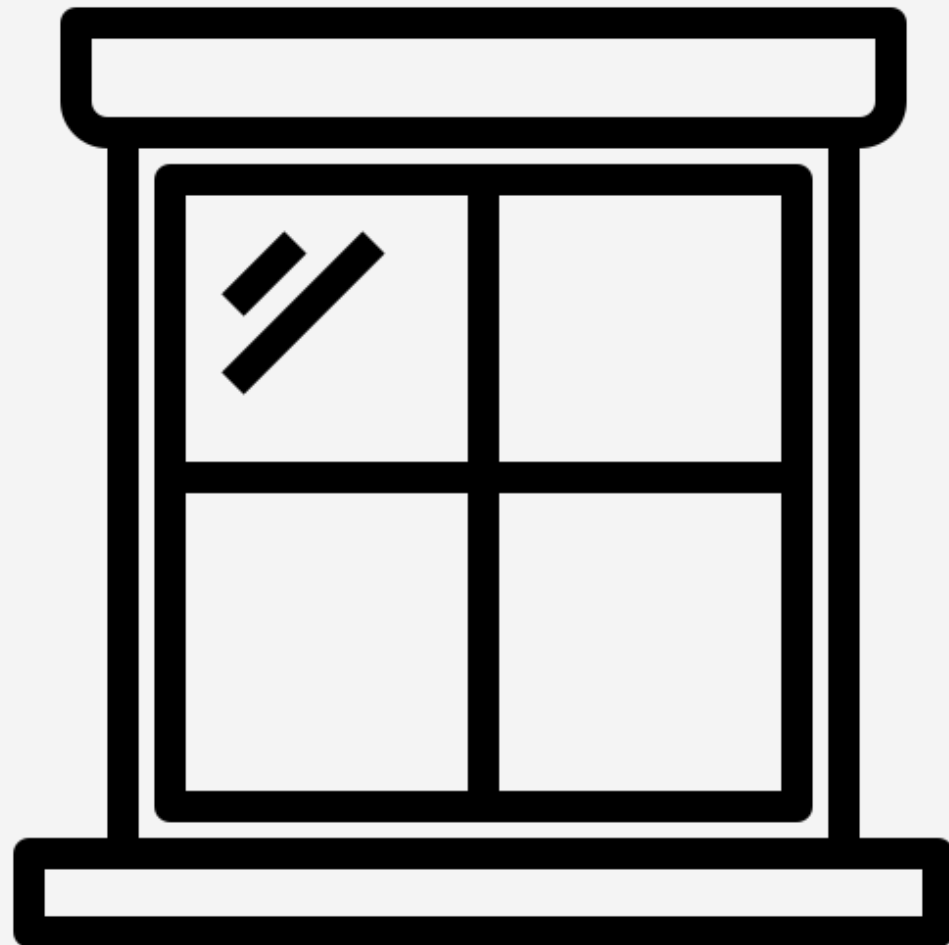
# Empty Module #3

**Spectrogram 구하기**



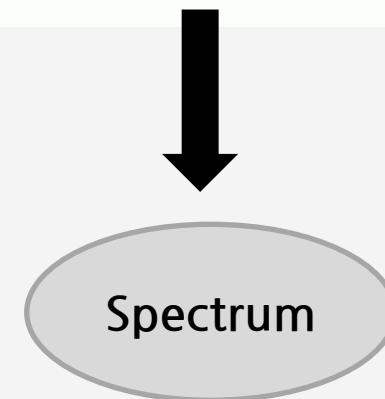
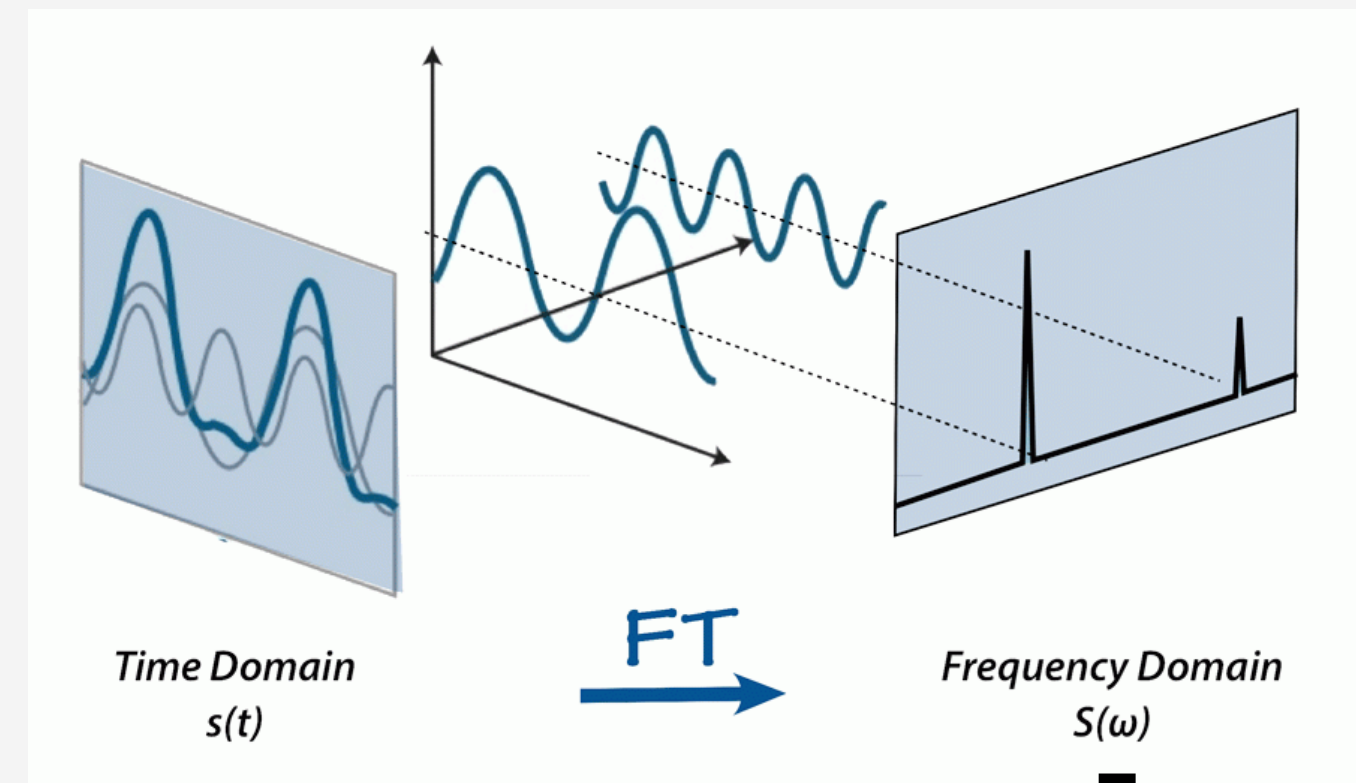
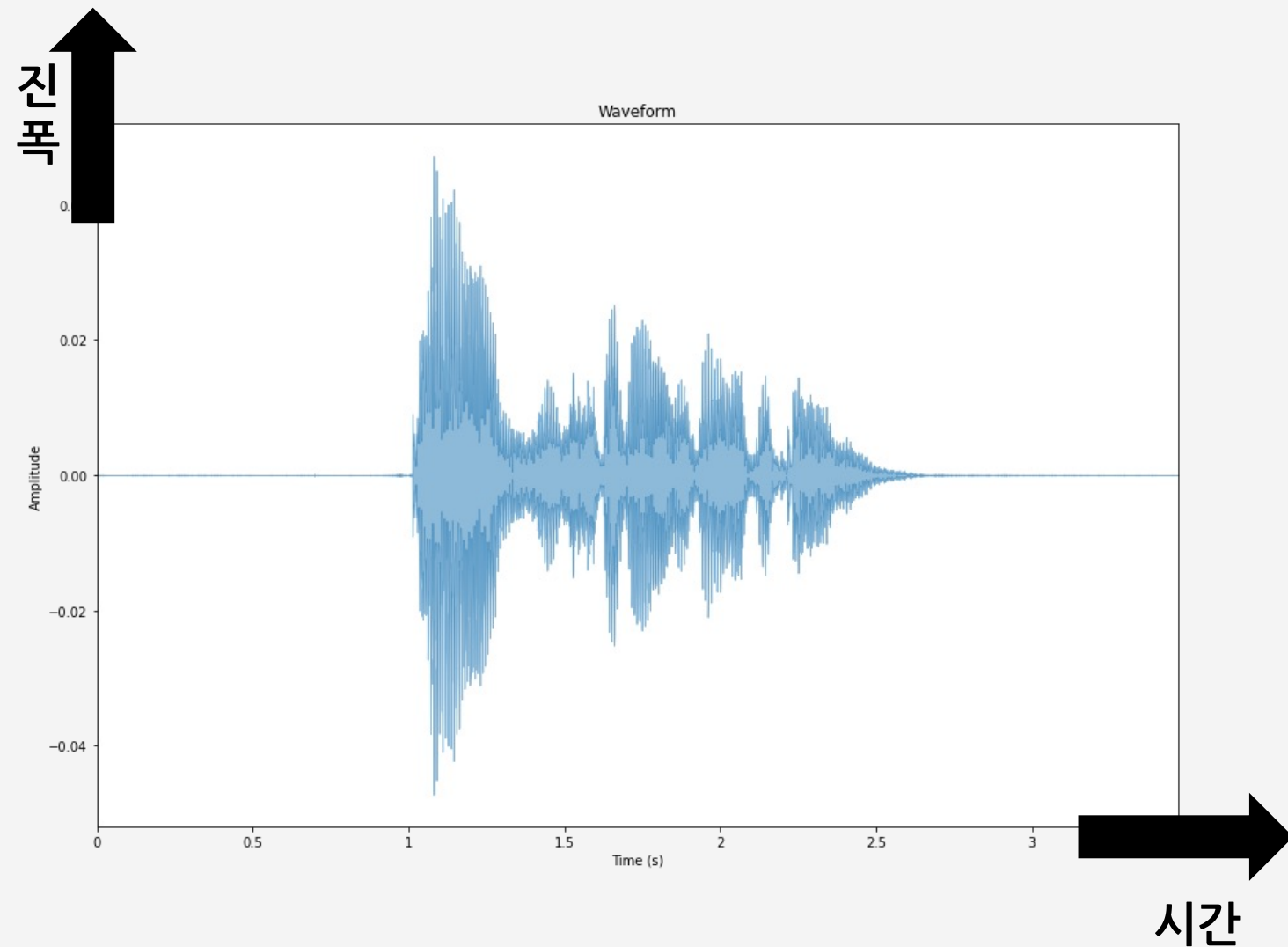
# [Empty Module #3] - Spectrogram 구하기

## 3-1. Windowing



# [Empty Module #3] - Spectrogram 구하기

## 3-2. Fast Fourier Transform(FFT)

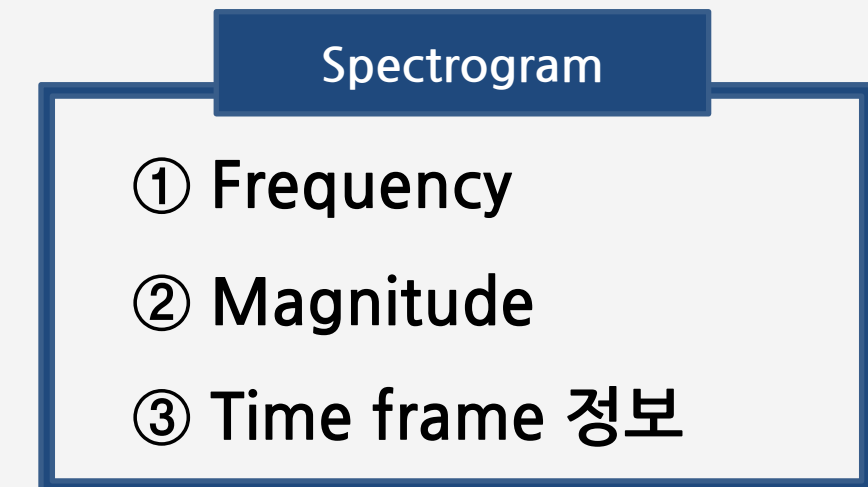
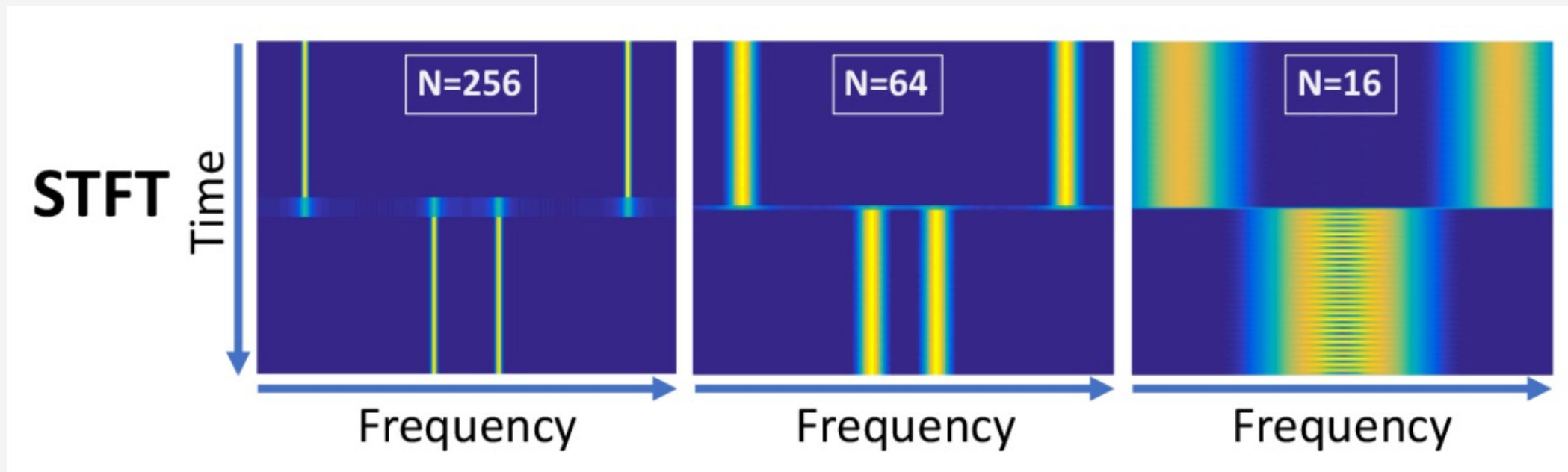


고유 배음 구조  
→ 소리의 고유한 특징 설계



# [Empty Module #3] - Spectrogram 구하기

## 3-3. Short Time Fourier Transform(STFT)



- Windowing으로 나눈 후 FFT를 적용한 각각의 frame들을 다시 시간 순서로 이어 붙임
- 시간적 특성까지 고려한 Spectrum





# [Empty Module #3] - Spectrogram 구하기

```
# spectrogram
```

```
result=np.array([])
```

```
X, sample_rate = librosa.load(file_name, sr=22050) # 입력 신호(X)를 librosa.stft 함수의 입력으로 하여 spectrogram 구하기
```

```
# - 참고) 사람의 음성은 20~40(ms) 사이 시간 내에서 현재 말하는 발음을 변경할 수 없다고 합니다.
```

```
# 시간축에 대한 구간을 나눌 때 20~40(ms) 이내 간격으로 나누기 위하여 n_fft 파라미터 값을 조정해주세요. (베이스라인 성능은 23ms 간격으로 나누었습니다.)
```

```
spectrogram = np.abs(librosa.stft(y = X, n_fft = 512)) # spectrogram에 절대값을 취하여 복소수 형태의 값 바꾸기
```

```
spectrogram_feature = np.mean(spectrogram, axis = 1) # 프레임 축의 평균값을 취한 뒤 spectrogram_feature에 저장
```

`n_fft : int > 0 [scalar]`

length of the windowed signal after padding with zeros. The number of rows in the STFT matrix `D` is `(1 + n_fft/2)`. The default value, `n_fft=2048` samples, corresponds to a physical duration of 93 milliseconds at a sample rate of 22050 Hz, i.e. the default sample rate in librosa. This value is well adapted for music signals. However, in speech processing, the recommended value is 512, corresponding to 23 milliseconds at a sample rate of 22050 Hz. In any case, we recommend setting `n_fft` to a power of two for optimizing the speed of the fast Fourier transform (FFT) algorithm.

→ `n_fft` 파라미터 512로 설정 (23ms)

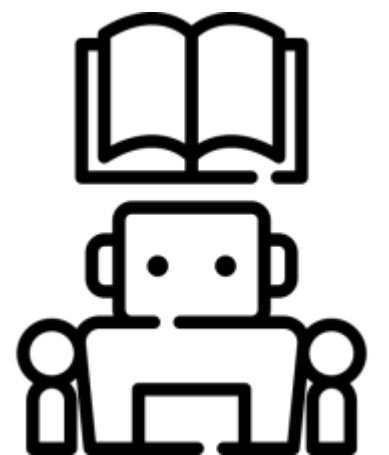




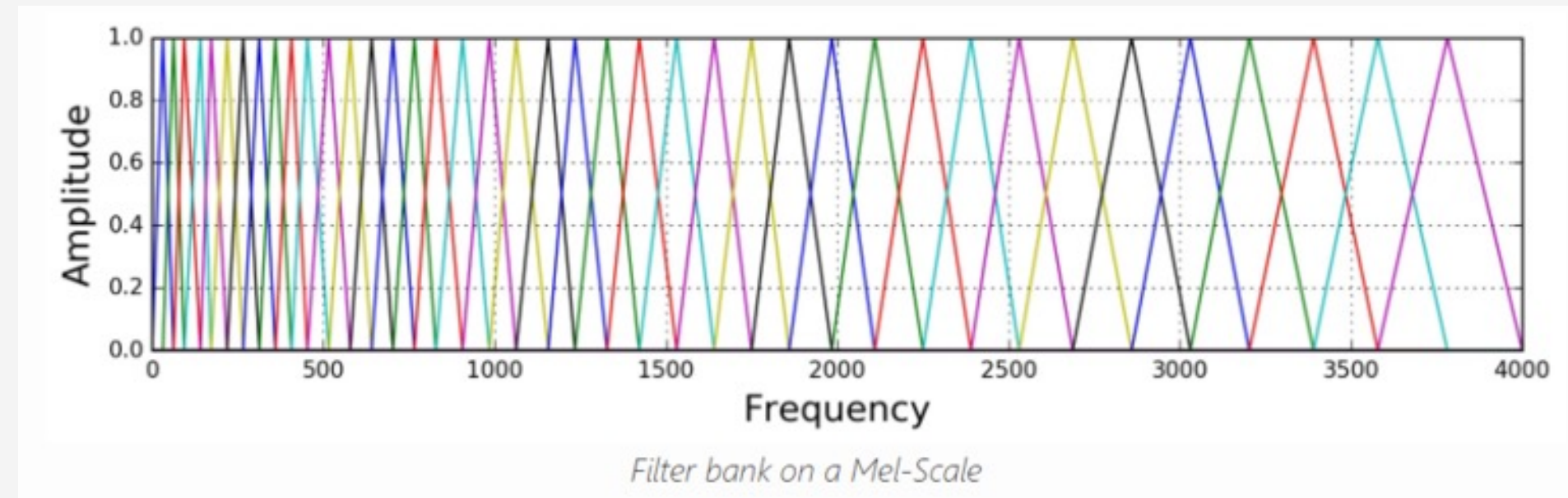
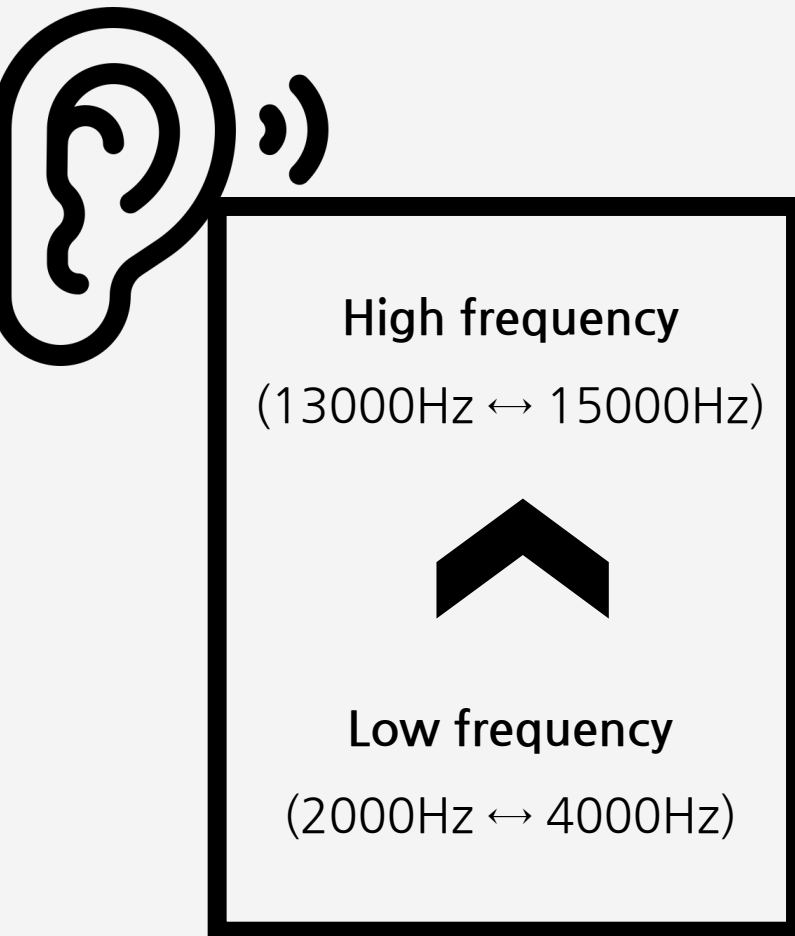
---

# Empty Module #4

**Mel-Filter-Bank를 이용한  
Mel-spectrogram 구하기**



# [Empty Module #4] - Mel-Filter-Bank를 이용한 Mel-spectrogram 구하기



# Mel-spectrogram

`power_spectrogram = spectrogram**2` # *spectrogram*을 제공하여 *power spectrogram* 구하기

`mel_spectrogram = librosa.feature.melspectrogram(S = power_spectrogram)` # *power spectrogram*을 *melspectrogram* 함수의 입력으로 해 *mel-spectrogram* 구하기

`power_magnitude_to_db = librosa.power_to_db(S = mel_spectrogram)` # *librosa.power\_to\_db*함수를 통하여 *power magnitude*를 데시벨(*db*)로 변환

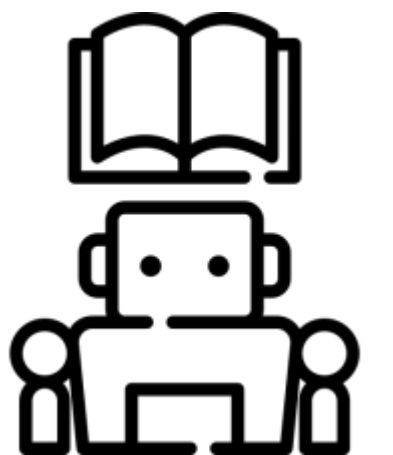
`mel_spectrogram_feature = np.mean(power_magnitude_to_db, axis = 1)` # *프레임* 축의 평균값을 취한 뒤 *mel\_spectrogram\_feature*에 저장



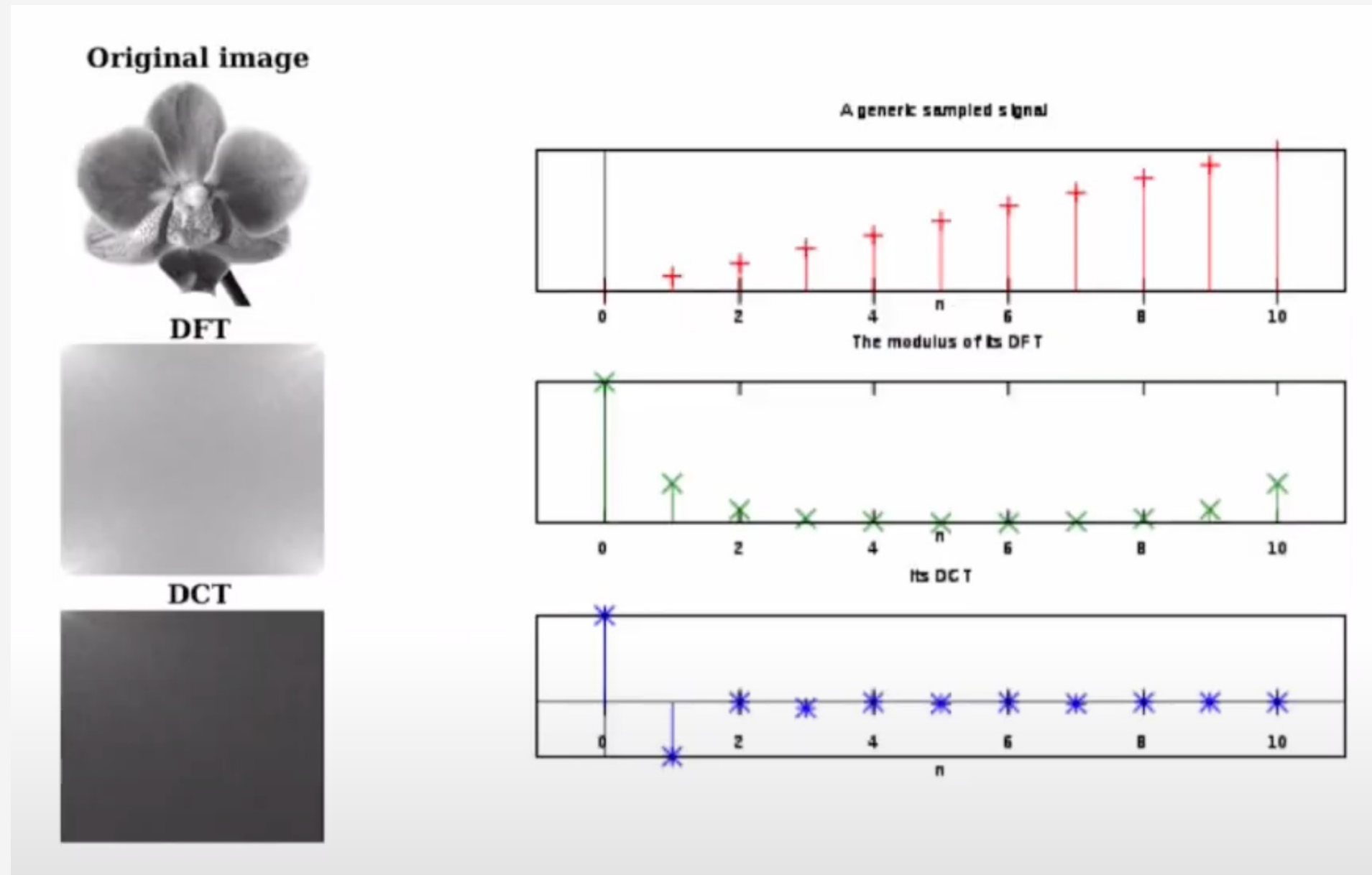
---

# Empty Module #5

**DCT를 이용한 MFCC 구하기**



# [Empty Module #5] - DCT를 이용한 MFCC 구하기



→ DCT(Discrete cosine transform)

# MFCC

```
MFCC = librosa.feature.mfcc(S = power_magnitude_to_db) # 데시벨로 변환한 mel-spectrogram을 librosa.feature.mfcc 함수의 입력으로 하여 MFCC를 구하기
mfcc_feature = np.mean(MFCC, axis = 1) # 프레임 축의 평균값을 취한 뒤 mfcc_feature에 저장
```



# [Empty Module #3, 4, 5]

```
import librosa
import glob, pickle
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import librosa, librosa.display

def extract_feature(file_name):
    # spectrogram
    result=np.array([])
    X, sample_rate = librosa.load(file_name, sr=22050) # 입력 신호(X)를 librosa.stft 함수의 입력으로 하여 spectrogram 구하기

    # - 참고) 사람의 음성은 20~40(ms) 사이 시간 내에서 현재 말하는 발음을 변경할 수 없다고 합니다.
    # 시간축에 대한 구간을 나눌 때 20~40(ms) 이내 간격으로 나누기 위하여 n_fft 파라미터 값을 조정해주세요. (베이스라인 성능은 23ms 간격으로 나누었습니다.)
    spectrogram = np.abs(librosa.stft(y = X, n_fft = 512)) # spectrogram에 절대값을 취하여 복소수 형태의 값 바꾸기
    spectrogram_feature = np.mean(spectrogram, axis = 1) # 프레임 축의 평균값을 취한 뒤 spectrogram_feature에 저장

    # Mel-spectrogram
    power_spectrogram = spectrogram**2 # spectrogram을 제곱하여 power spectrogram 구하기
    mel_spectrogram = librosa.feature.melspectrogram(S = power_spectrogram) # power spectrogram을 melspectrogram 함수의 입력으로 해 mel-spectrogram 구하기
    power_magnitude_to_db = librosa.power_to_db(S = mel_spectrogram) # librosa.power_to_db함수를 통하여 power magnitude를 데시벨(db)로 변환
    mel_spectrogram_feature= np.mean(power_magnitude_to_db, axis = 1) # 프레임 축의 평균값을 취한 뒤 mel_spectrogram_feature에 저장

    # MFCC
    MFCC = librosa.feature.mfcc(S = power_magnitude_to_db) # 데시벨로 변환한 mel-spectrogram을 librosa.feature.mfcc 함수의 입력으로 하여 MFCC를 구하기
    mfcc_feature = np.mean(MFCC, axis = 1) # 프레임 축의 평균값을 취한 뒤 mfcc_feature에 저장

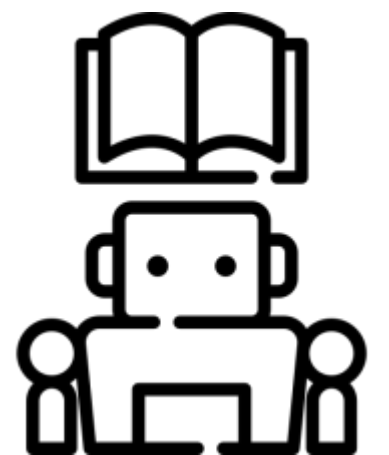
    return spectrogram_feature, mel_spectrogram_feature, mfcc_feature
```



---

# Empty Module #6

**MFCC 피처를 이용한 모델 학습하기**



# [Empty Module #6] - MFCC 피처를 이용한 모델 학습하기

## Random Forest Classifier

```
#RandomForestClassifier로 음성 감정 분류 학습 및 평가
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(random_state = 1) # baseline : 0.57870

for feature_name in train_data.keys():
    # train_data에 있는 각 feature load

    # 해당 feature의 train_data, test_data load
    x_train = np.array(train_data[feature_name])
    x_test = np.array(test_data[feature_name])

    # StandardScaler로 표준화
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    x_train = scaler.fit_transform(x_train)
    x_test = scaler.transform(x_test)

    # classifier
    classifier.fit(x_train, y_train)
    predict = classifier.predict(x_test)

#Sample submit file 저장
sample = pd.read_csv(join(DATA_PATH, 'sample_submit.csv'))
sample['emotion'] = predict.reshape(-1,1)
sample.to_csv(feature_name+'.csv', index=False, header = True)
if feature_name == 'mfcc': sample.to_csv('submission.csv', index=False, header=True)
```





# [Empty Module #6] - MFCC 피처를 이용한 모델 학습하기

## Support Vector Classifier

```
#SVC로 음성 감정 분류 학습 및 평가
from sklearn.svm import SVC
classifier = SVC(random_state = 1, C = 100, gamma = 'auto', tol=0.4, probability = True)

for feature_name in train_data.keys():
    # train_data에 있는 각 feature load

    # 해당 feature의 train_data, test_data load
    x_train = np.array(train_data[feature_name])
    x_test = np.array(test_data[feature_name])

    # StandardScaler로 표준화
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    x_train = scaler.fit_transform(x_train)
    x_test = scaler.transform(x_test)

    # classifier
    classifier.fit(x_train, y_train)
    predict = classifier.predict(x_test)

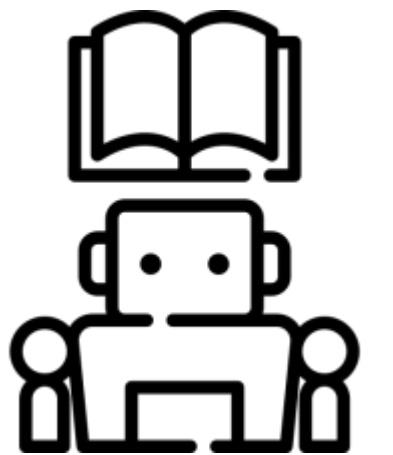
#Sample submit file 저장
sample = pd.read_csv(join(DATA_PATH, 'sample_submit.csv'))
sample['emotion'] = predict.reshape(-1,1)
sample.to_csv(feature_name+'.csv', index=False, header = True)
if feature_name == 'mfcc': sample.to_csv('submission.csv', index=False, header=True)
```





---

# Hyperparameter Tuning



# Hyperparameter Tuning

---

## Baseline


: RandomForestClassifier(random\_state = 1)

Feature name	Score(baseline)
spectrogram	0.46296
mel	0.45601
✓ mfcc	0.57870



# Hyperparameter Tuning

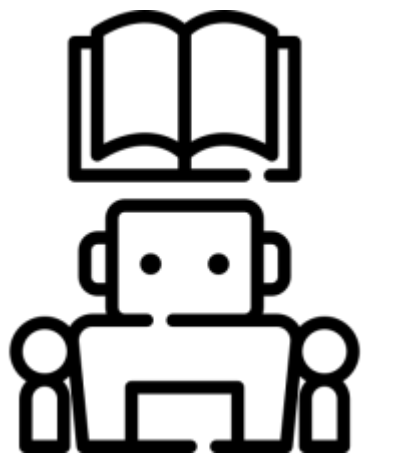
---

Hyperparameter	score
CountVectorizer(max_features = 100) SVC(gamma = 'auto')	0.94439(Baseline)
 SVC(random_state = 1, C = 100, gamma = 'auto', tol=0.45, probability = True)	0.67129
SVC(random_state = 1, C = 100, gamma = 'auto', tol=0.5, probability = True)	0.67129
SVC(random_state = 1, C = 100, gamma = 'auto', tol=0.55, probability = True)	0.65972



---

# Final Score



# Final Score

## [2022-ML][P3]20011844\_안수경

Python · 음성 감정 인식

Notebook Data Logs Comments (0) Settings



Competition Notebook  
음성 감정 인식

Run  
247.9s

Public Score  
0.45601

Best Score  
0.5787 V9



Mel baseline score (0.45601)

## [2022-ML][P3]20011844\_안수경

Python · 음성 감정 인식

Notebook Data Logs Comments (0) Settings



Competition Notebook  
음성 감정 인식

Run  
255.5s

Public Score  
0.46296

Best Score  
0.5787 V9



Spectrogram baseline score (0.46296)

## [2022-ML][P3]20011844\_안수경

Python · 음성 감정 인식

Notebook Data Logs Comments (0) Settings



Competition Notebook  
음성 감정 인식

Run  
249.5s

Public Score  
0.57870

Best Score  
0.5787 V14



MFCC baseline score (0.57870)



# Final Score

## [2022-ML][P3]20011844\_안수경

Python · [음성 감정 인식](#)

Notebook

Data

Logs

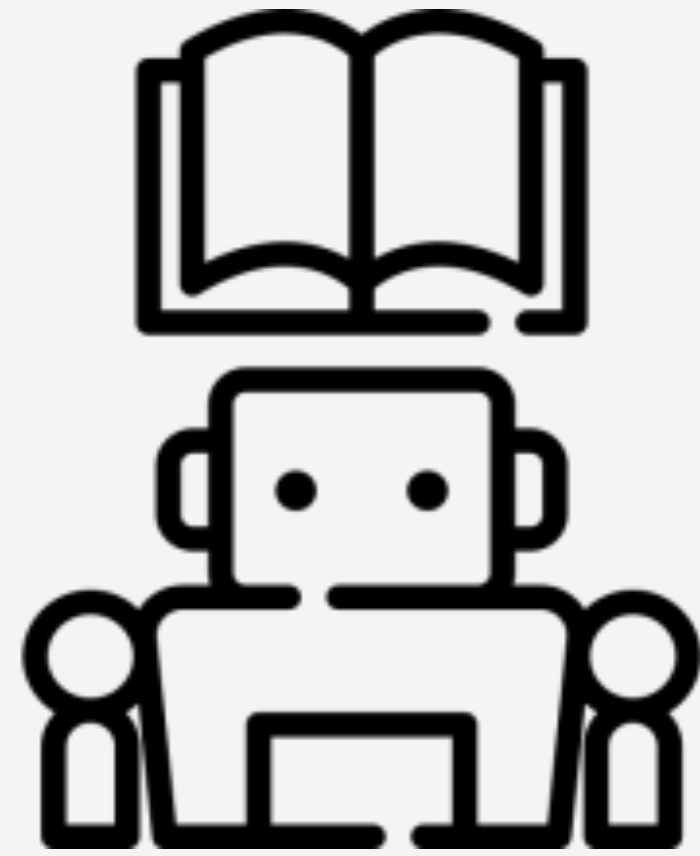
Comments (0)

Settings

	Competition Notebook	Run	Private Score	Public Score	Best Score
	<a href="#">음성 감정 인식</a>	252.5s	0.67129	0.67129	<u>0.67129 V31</u>

← MFCC best score (0.67129)





# THANK YOU

*Machine learning term project #3*

---

