기계학습 텀프로젝트 3

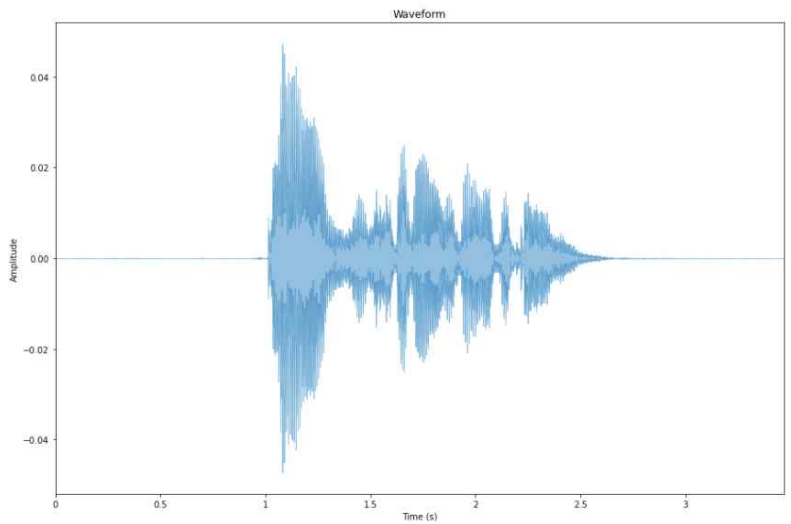
19011494 조국희

03. 음성 감정 인식

- **프로젝트 목적**

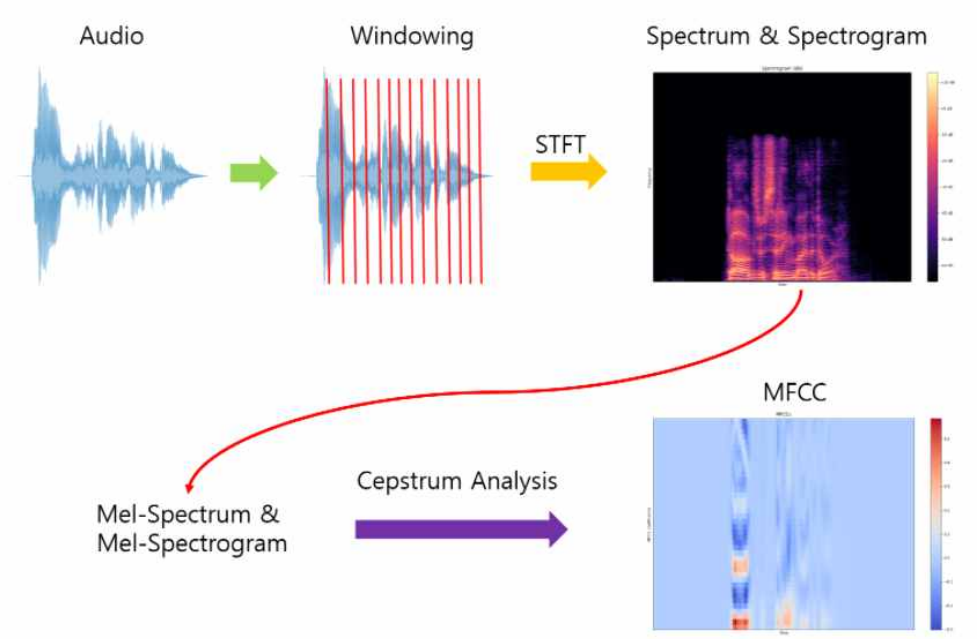
  1D 음성 데이터를 Handcrafted Feature 로 기술하는 법을 알 수 있음

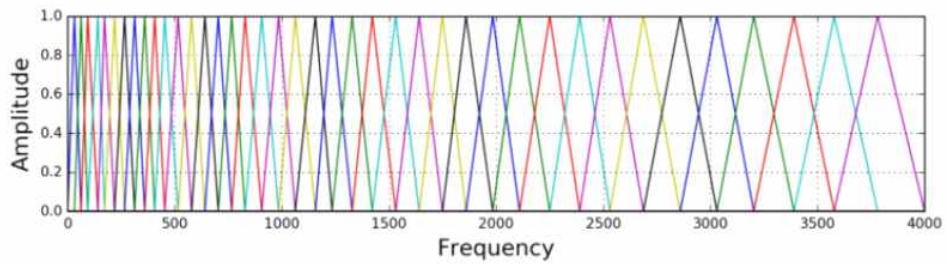
- **음성 데이터**
  - ✓ 연속형 데이터 -> sampling 및 양자화 -> 이산형
  - ✓ 음성의 대표적 성질을 보여주는 feature 추출



03. 음성 감정 인식

- **전체적인 과정**
  - ✓ 음성의 시간적 특성을 고려해 Windowing (사람의 발음을 바꿀 수 없는 범위인 20-40ms 사이로)
  - ✓ 시간의 흐름에 따른 주파수의 변화를 알 수 있도록 STFT 적용한 spectrogram 추출
  - ✓ 해당 spectrogram에 Mel-filter 를 적용한 mel-spectrogram 생성
  - ✓ Mel-Spectrogram에 행렬을 압축해서 표현해주는 DCT 연산을 수행해 MFCC 생성





저주파 대역을 감지하는 부분은 굵지만
고주파 대역을 감지하는 부분으로 갈수록 얇아지는
달팽이관의 특성을 고려한 mel-scale

03. 음성 감정 인식

- **Sampling & 양자화 -> 연속형 신호 이산형으로 변환**
  - ✓ 음성의 시간적 특성을 고려해 Windowing (사람의 발음을 바꿀 수 있는 범위인 20-40ms 사이로)
- **Short Time Fourier Transform -> spectrogram 추출**

```
# sample rate를 22050, n_fft를 512 설정해 23ms 간격으로 나눔 (512/ 22050 = 0.023)
# n_fft : 한  번 fft를 해 줄 만큼의 sequence 길이
X, sample_rate = librosa.load(file_name, sr=22050) # 음성데이터 load
```

```
spectrogram = np.abs(librosa.stft(X, n_fft = 512))  shape : (frequency길이, 프레임수)
# 각 프레임의 평균값 구한 후 (1, frequency의 길이) 형태로 재배열하여 spectrogram_feature에 저장
spectrogram_feature = np.mean(spectrogram, axis = 1).reshape(1, -1)
```

03. 음성 감정 인식

- **Mel-Filter 적용 -> Mel-spectrogram**

```
# spectogram 을 mel-scale 형태로 변경
power_spectrogram = np.square(spectrogram)
mel_spectrogram = librosa.feature.melspectrogram(S=power_spectrogram, n_fft=512, hop_length = hop)
# magnitude 를 dB 단위로 바꿔줌                                                    window 간의 거리
mel_spectrogram = librosa.power_to_db(mel_spectrogram)
# 각 프레임의 평균값 구한 후 (1, mel filter의 길이) 형태로 재배열하여 mel_spectrogram_feature에 저장
mel_spectrogram_feature = np.mean(mel_spectrogram, axis = 1).reshape(1, -1)
```

- **Mel-spectrogram -> MFCC 추출**

```
mfcc = librosa.feature.mfcc(sr=sample_rate,S = mel_spectrogram, n_mfcc = 20)
mfcc_feature = np.mean(mfcc, axis = 1).reshape(1, -1)
```

03. 음성 감정 인식

▪ 데이터 불러오기

```
def load_data(data_info, isTrain=True):
    PATH = join('/kaggle','input','2021-ml-tp-p6')
    if isTrain:
        train_data = {'spectrogram':[],'mel':[],'mfcc':[]}#음성 feature들을 담는 dictionary
        train_label = []#학습에 사용할 label을 담는 list
        file_list = data_info['file_name']
        emotion_list = data_info['emotion']
        for file_name, emotion in tqdm(zip(file_list, emotion_list)):

            # train data의 feature들을 뽑아내 각각 spectrogram_feature, mel_spectrogram_feature, mfcc_feature 저장
            spectrogram_feature, mel_spectrogram_feature, mfcc_feature = extract_feature(join(PATH,'train_data','train_data',file_name))
            # 각 데이터 해당하는 키에 저장
            train_data['spectrogram'].append(spectrogram_feature)
            train_data['mel'].append(mel_spectrogram_feature)
            train_data['mfcc'].append(mfcc_feature)
            train_label.append(emotion)
            #-------------------------------------------------------------------------------------

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

▪ 학습 및 예측

```
for feature_name in train_data.keys():

    X = np.array(train_data[feature_name]).reshape(1008, -1)
    y = y_train
    from sklearn.model_selection import GridSearchCV
    params = {
        'max_depth' : [14, 15, 16, 17, 18],
        'criterion' : ['entropy', 'gini'],
    }
    clf = GridSearchCV(RandomForestClassifier(random_state = 1), params, cv=5)
    clf.fit(X, y)
    predict = clf.predict(test)


    #Sample submit file 저장
    sample['emotion'] = predict.reshape(-1,1)
    sample.to_csv(join(feature_name+'.csv'),index=False,header=True)
```

03. 음성 감정 인식

▪ 성능 높이기 위한 시도

- ✓ Sampling rate = 44100으로 늘리고, n_fft를 1024로 설정
- ✓ PCA 사용

베이스라인	성능
spectrigram	0.46064
Mel-spectrogram	0.47222
mfcc	0.55555

모델	성능
RandomForest + spectrogram	0.47222
RandomForest + Mel-spectrogram	0.48379
RandomForest + mfcc	0.58101
RandomForest + mfcc + sr44100	0.57407
RandomForest + mfcc + sr44100 + PCA	0.60416