라이언로켓 데이터 연구팀 사전과제 코드리뷰

데이터사이언티스트 인턴

지원자 **이성규**

[개발 환경]

```
print("Python: ", sys.version)
print('numpy: ' + np.__version__)
print('pandas: ' + pd.__version__)
print('matplotlib: : ' + mpl.__version__)
print('librosa: ' + librosa.__version__)
print('scipy: ' + scipy.__version__)
print('dtw: ' + dtw.__version__)
```

```
Python:  3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
numpy: 1.20.1
pandas: 1.2.4
matplotlib: : 3.3.4
librosa: 0.9.1
scipy: 1.6.2
dtw: 1.1.12
```

[소과제 1]

```
import re

def clean_text(String):
    rm = re.sub('[$%^&]', '', String)
    rm = re.sub('(([.])##2{1,})', '.', rm)
    rm = re.sub('(([,])##2{1,})', ',', rm)
    rm = re.sub('(([!])##2{1,})', '!', rm)
    rm = re.sub('(([?])##2{1,})', '?', rm)
    return rm
```

Q1. 텍스트에 섞인 4가지 종류의 특수 기호($%^&) 제거

정규표현식으로 [$%^&]가 들어가면 지우고 붙임으로 대체했습니다.

Q2. 텍스트에 같은 문장 기호(.,!?)가 반복된다면 하나로 통일

정규표현식으로 '연속해서 2번이상' 반복되는 문장기호를 1개로 대체했습니다.[1]

---

[1] https://tmdrl5779.tistory.com/124

[소과제 2]

```
# pitch detection
def pitch_detection(Files):
    for i in Files:
        audio_sample, sampling_rate = librosa.load('./dataset/audio/' + i)

        # Get some useful statistics
        T = 1/sampling_rate         # Sampling period
        N = len(audio_sample)       # Signal length in samples
        t = N / sampling_rate       # Signal length in seconds

        S = np.abs(librosa.stft(audio_sample, n_fft = 512, hop_length = 108, win_length = 512, window=signal.hann))
        # FFT(고속 푸리에 변환) 알고리즘의 속도를 최적화하려면 2의 거듭제곱인 n_fft = 512로 설정
        # hop_length는 win_length의 1/4 크기인 hop_length = 108로 설정
        # win_length는 n_fft와 같게 설정

        pitches, magnitudes = librosa.piptrack(S=S, sr=sampling_rate, fmin = 100, fmax = 450)
        # fmin이 적용되지 않아 원인을 찾아봤으나, 작동하지 않아 아래에서 조건문으로 처리

        shape = np.shape(pitches)
        nb_samples = shape[0]
        nb_windows = shape[1]

        pitch_list = []
        for k in range(0, nb_windows):
            index = magnitudes[:,k].argmax()
            pitch = pitches[index,k]
            pitch_list.append(pitch)

        # 작동안된 fmin 처리
        clean_pitch = []
        for pitch in pitch_list:
            if pitch > 100:
                clean_pitch.append(pitch)
```

Q1. Pitch Detection 수행

과제 수행에 앞서, 예시자료를 불러온 결과 (최솟값, 최댓값)이 (133, 453)이고, 개수가 330개인 것을 파악했습니다. 따라서 pitch의 범위를 100~450으로 설정해야겠다고 생각했습니다.

Pitch Dectection을 위해 librosa 패키지를 사용하였습니다. FFT(고속 푸리에 변환)을 위해 librosa 패키지의 stft 함수를 적용하고 결과가 복소수이기 때문에, np.abs를 적용하였습니다. FFT 알고리즘의 속도를 최적화하기 위해서 2의 거듭제곱인 512로 n_fft를 설정하였습니다. win_length는 n_fft와 같게 설정하고, hop_length는 win_length의 1/4 크기인 108로 설정했습니다.[2]

그리고 librosa 패키지의 piptrack 함수를 사용하여 [예시파일]의 범위처럼 100~450 사이의 피치를 검출하려하였으나, fmin이 이상하게도 적용되지 않아 원인을 찾아봤으나 찾지못해 100이상 범위는 조건문으로 처리하였습니다.[3]

이후 Magnitude의 최댓값 인덱스를 이용하여 피치검출을 했습니다.

---

[2] https://librosa.org/doc/main/generated/librosa.stft.html#librosa.stft

[3] https://librosa.org/doc/main/generated/librosa.piptrack.html

Q2. np.array 타입의 pickle 파일로 Pitch Detection 결과 저장

```
# pitch_array: np.array 타입의 pickle 파일로 /result/pitch_array/ 경로에 저장
data = np.array(clean_pitch)
with open(f'./result/pitch_array/{i}.pkl', 'wb') as f:
    pickle.dump(data, f)
```

리스트 형태로 저장되어 있는 피치들을 np.array를 활용하여 타입을 변경하고 With open 구문과 pickle 라이브러리를 이용하여 .pkl 형태로 저장했습니다.

Q3. time-frequency graph로 시각화하여 결과 저장

```
# pitch_graph: time-frequency graph로 시각화하여 /result/pitch_graph/ 경로에 저장
time = np.linspace(0, t, len(data)) # x축 시간 표현을 위해 설정

fig = plt.figure(figsize = (14,5))
plt.plot(time, data)
plt.xlabel('time(sec)')
plt.ylabel('frequency(Hz)')
plt.ioff()
plt.savefig(f'./result/pitch_graph/{i}.png')
```

x축에 시간표현을 위해 앞서 구해두었던 wav파일의 시간(t)을 np.linspace를 이용하여 피치데이터의 수만큼 나눠서 배열하여 x축에는 time, y축에는 fequency를 배치한 time-frequency graph로 시각화하여 .png 형태로 저장했습니다.

[소과제 3]

```
def similarity(pitch_array):
    similarity_list = []

    for i in pitch_array:
        similarity_list.append(dtw.dtw(example, i, keep_internals = True).distance)

    with open('./result/similarities.pkl', 'wb') as f:
        pickle.dump(similarity_list, f)
```

Q1. [소과제2]의 결과물에 대해서, 예시자료와의 유사도를 분석하여 저장

소과제 3의 Hint에서 결과물과 예시자료가 대부분 length가 다르다는 것에 주목하여 서로 다른 길이의 시계열의 유사도분석이 가능한 DTW(Dynamic Time Wrapping 동적 시간 워핑) 패키지를 유사도 분석에 사용해야겠다는 인사이트를 얻었습니다.[4]

[4] https://leo-bb.tistory.com/58

[소과제 4]

```
def ci(a, inp):
    rv = stats.norm()
    lower_ci = np.mean(inp) - (rv.isf(a/2) * (np.std(inp) / np.sqrt(len(inp))))
    upper_ci = np.mean(inp) - (rv.isf(1-a/2) * (np.std(inp) / np.sqrt(len(inp))))

    with open('./result/confidence_interval.pkl', 'wb') as f:
        pickle.dump((lower_ci, upper_ci), f)

    return lower_ci, upper_ci
```

Q1. [소과제 3] 결과물로 95% 신뢰 수준 신뢰 구간 범위의 최솟값과 최댓값 구하기

모집단의 평균과 분산을 구할 수 있다고 생각해서 정규분포를 가정하고, 신뢰수준 95%는 100(1-a)%이므로, a(유의수준)은 0.05인 것을 구하여 신뢰구간을 구현한 함수 ci를 만들어 신뢰하한과 신뢰상한을 구했습니다.

[사전과제 단계별 느낀점]

[소과제 1]

텍스트 데이터는 수집부터 전처리, 분석까지 다뤄본 경험이 있어서 수월했습니다.

[소과제 2]

오디오 데이터는 이번 사전과제로 처음 다뤄봤습니다. 패키지와 주제가 생소했고, 소과제 3,4의 과제 결과와 연결되기 때문에 가장 많은 시간을 투자하여 풀었습니다.

처음 피치검출에서 피치의 범위가 예시자료에 비해 너무 크다고 생각해서 임의로 범위를 줄여서 추출했는데, 시각화 결과도 예시자료와 많이 상이하다고 생각해서 정답을 확신하지 못하고 다른 여러 방법 (Autocorrelation, Spectrogram)도 시도해보았으나, FFT를 이용하여 magnitude의 peak를 찾는 것이 가장 예시자료와 비슷하다고 생각해서 제출했습니다.

[소과제 3]

유사도 분석은 수행했으나, 차이가 너무 큰 값이 나와 답은 확신하지 못했습니다.

[소과제 4]

신뢰구간 범위는 찾았으나, 답은 확신하지 못했습니다.