14주차(06.03~06.09) 주간 보고서

1. 효과성 입증 테스트
    1-1. 2차 테스트
        1, 2차테스트 결과 분석
    1-2. 3차 테스트
2. 보완사항
    2-1. 웹페이지 3차 테스트에 맞게 변경
    2-2. 감정분류 알고리즘 변경
        STT 구두점
        문장 나누기
        감정점수 산출

1. 효과성 입증 테스트

1-1. 2차 테스트

05/30~06/05 2차 테스트 진행 완료.

1, 2차테스트 결과 분석

결과 분석을 위해 사용할 비교 방법은 그룹간 중심 비교(Wilcoxn Signed-Rank Test)이다.

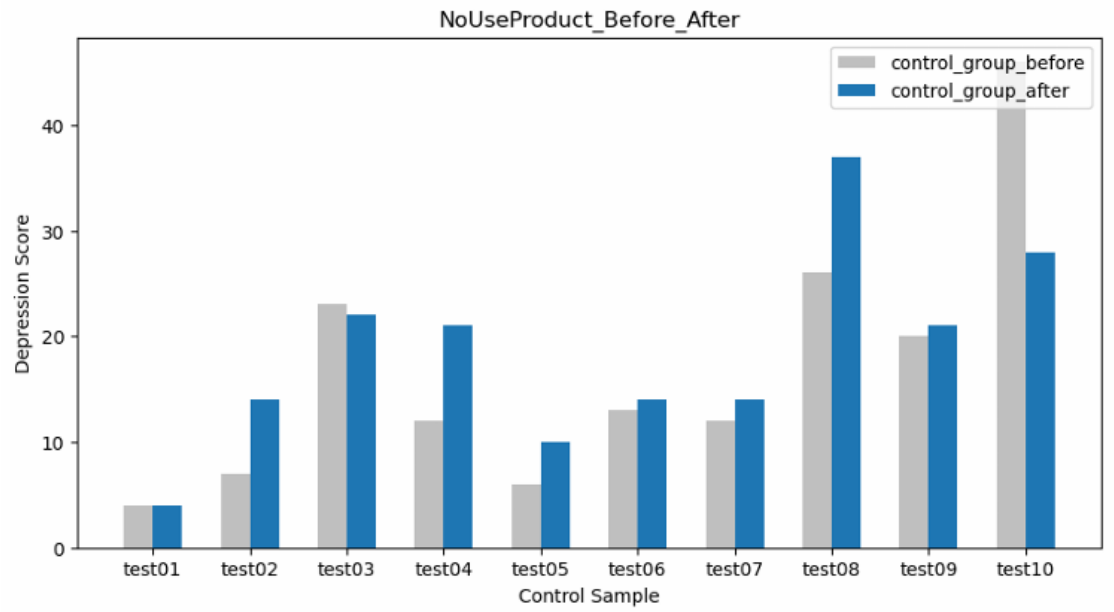
그룹간 중심 비교: 정규성을 띄지 않는 비모수 데이터의 실험군과 대조군의 차이를 검증하기 위해 사용하는 방법.

- 테스트 방법

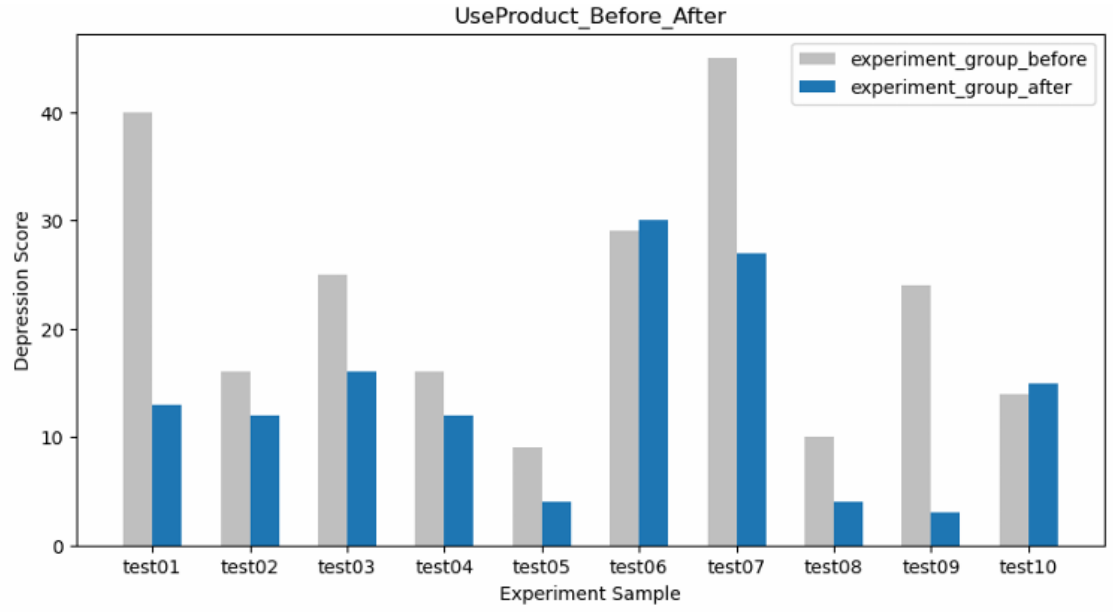
1. 실험군과 대조군에게 1주일 간격으로 CES-D 테스트 진행.
2. 귀무가설: 일기 인형을 사용한 실험군과 일기 인형을 사용하지 않은 대조군 과의 우울 점수에는 변화가 없다.
3. 대립가설: 일기 인형은 우울함을 감소하는 데 효과가 있다.

4. 실험군과 대조군의 CES-D 테스트 결과에 대한 그룹간 중심 비교 진행.
5. p-value > 0.05라면 귀무가설 채택
6. p-value ≤ 0.05라면 대립가설 채택

- 대조군 결과



- 실험군 결과



그래프를 보면 아무것도 하지 않은 대조군은 대부분 실험 후에 큰 변화가 없거나, 점수가 나빠진 반면에, 인형을 사용한 실험군은 대부분 사용 후에 점수가 낮아진 것을 확인할 수 있다.

- 그룹간 중심 비교 진행

```
from scipy.stats import wilcoxon
control_diff = []
experiment_diff = []

for before,after in zip(control_group_before,control_group_a
    control_diff.append(after - before)

for before,after in zip(experiment_group_before,experiment_g
    experiment_diff.append(after - before)

stat, p_value = wilcoxon(control_diff,experiment_diff)

print(f"Wilcoxon Test Statistic : {stat}")
print(f"p-value : {p_value}")
```

```
Wilcoxon Test Statistic : 6.0
p-value : 0.050612432239184685
```

p-value = 0.0506로 0.05보다 높은 결과가 나왔다.

→ 테스트 결과 '일기 인형을 사용한 실험군과 일기 인형을 사용하지 않은 대조군 과의 우울 점수에는 변화가 없다.' 라는 결과가 나왔지만, 지난주 1차 테스트 만으로 진행했을때의 p-value인 0.0625보다는 확실히 낮아진 모습이다. 현재 진행중인 3차테스트 까지 진행 했을 때 표본이 더 많아져 좋은 결과가 나올것이라 예상한다.

1-2. 3차 테스트

06/06~ 06/12 일정으로 3차 테스트 진행중.

2. 보완사항

2-1. 웹페이지 3차 테스트에 맞게 변경

1. 6월 6일 ~ 6월 12일까지 데이터 표시
2. 6월 달력으로 변경
3. 그래프 변경
4. 로그아웃 기능 추가

2-2. 감정분류 알고리즘 변경

기존에는 일기 전체 텍스트를 한번에 모델에 넣은 방식으로 감정을 분류했다. 그러나 여기서 감정분류가 문장이 길어지면 정확도가 떨어지는 문제점을 발견하여 문장별로 모델을 돌리는 방식으로 변경하였다.

STT 구두점

전체 텍스트를 문장 별로 나눠야 하기 때문에 STT에 구두점 가져오기를 추가하였다.

```
config = speech.RecognitionConfig(
    encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
```

```
    sample_rate_hertz=16000,
    language_code="ko-KR",
    enable_automatic_punctuation=True,   # 구두점 가져오기
)
```

https://cloud.google.com/speech-to-text/docs/automatic-punctuation?hl=ko

문장 나누기

다음 함수에서 문장을 분할 한 뒤, 각각의 문장의 감정에 대한 빈도수로 전체 일기에 대한 감정을 분류한다.

```
# 여러 문장에 대해 예측을 수행하는 함수
def predict(predict_sentence):
    # 문장을 '.', '?', '!'로 분할
    sentences = re.split(r'[.?!]', predict_sentence)

    # 빈 문장 제거
    sentences = [sentence.strip() for sentence in sentences i

    # 감정 빈도수 저장
    emotion_counts = {
        "행복": 0,
        "보통": 0,
        "화남": 0,
        "슬픔": 0
    }

    for sentence in sentences:
        result = predict_single_sentence(sentence)
        emotion_counts[result] += 1

    # 감정 빈도수 출력
    print("감정 예측 빈도수")
    for emotion, count in emotion_counts.items():
        print(f"{emotion}: {count}")
```

```
    return emotion_counts
```

감정점수 산출

감정분류 알고리즘 변경에 따라 감정점수 산출 알고리즘 역시 전체 문장에 대한 각 감정의 비율을 통해 산출하도록 변경하였다.

```
def calculate_emotion_score(emotion_counts):
    # 전체 문장의 수
    total_sentences = sum(emotion_counts.values())

    # 감정별 가중치
    weights = {
        "행복": 100,
        "보통": 50,
        "화남": 25,
        "슬픔": 0
    }

    # 점수 계산
    score = 0
    for emotion, count in emotion_counts.items():
        score += (count / total_sentences) * weights[emotion]

    return score
```