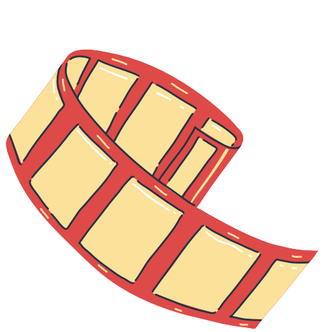
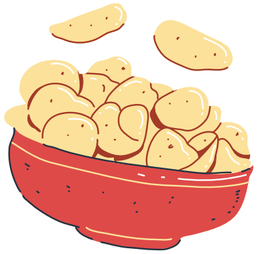
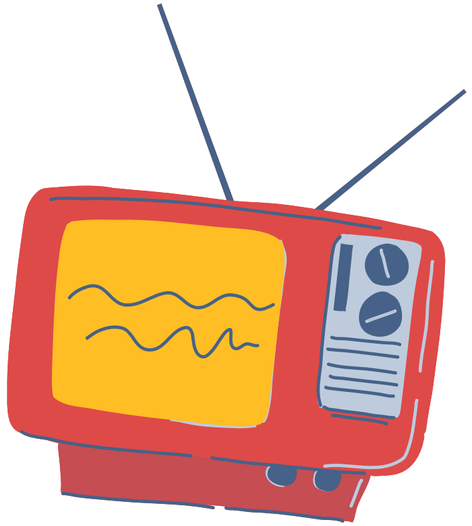
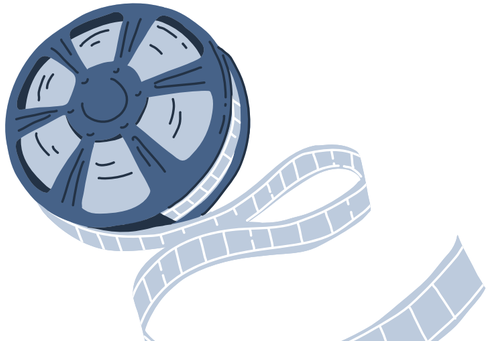
KDT
CINEMA

김동현 박희진 양현우 이화은





역할 분담

김동현

주제 선정

크롤링

이론 설계

박희진

크롤링

깃허브

메인 HTML

CSS

양현우

크롤링

데이터 취합

HTML 연결

이화은

PPT

개인 HTML

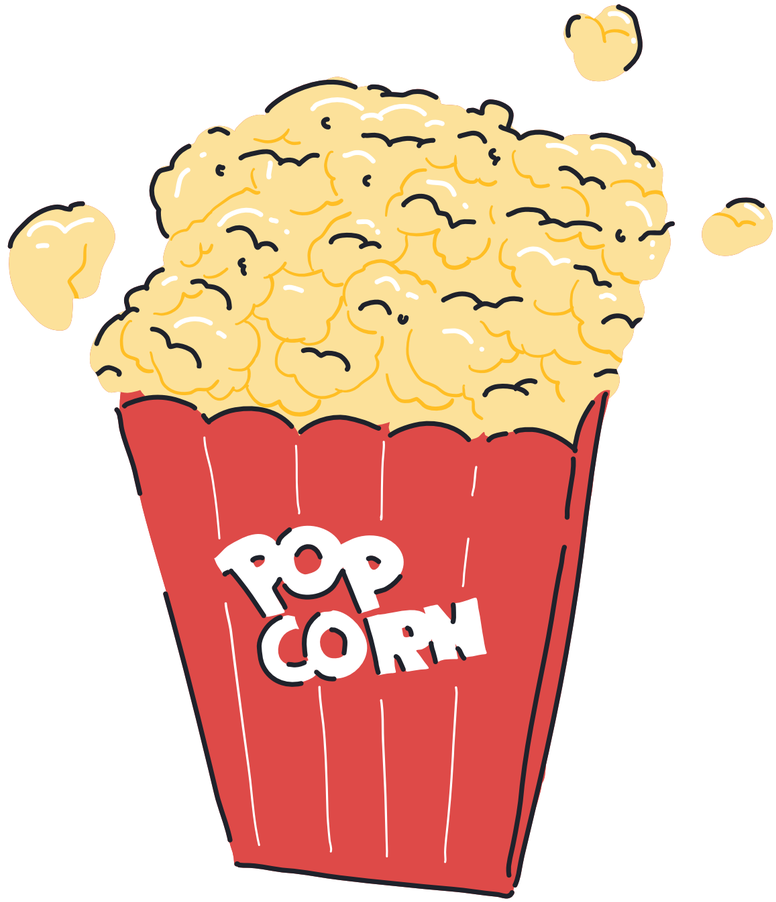
CSS

기능설계서

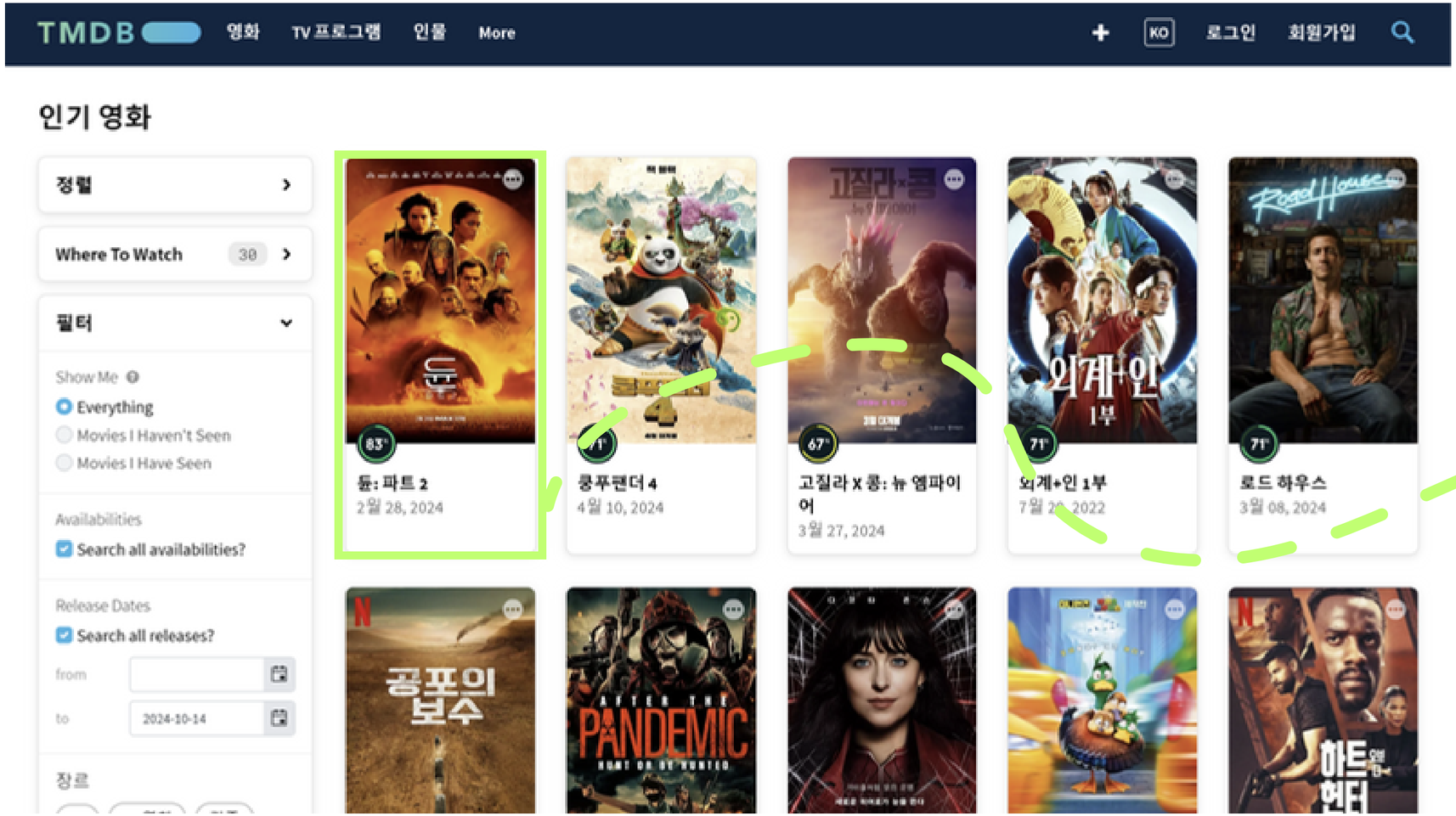
대분류	중분류	소분류	상세내역	담당자
데이터 수집	크롤링	크롤링	tbmd 데이터 크롤링	박희진
데이터 전처리	CNN 모델 데이터 전처리	데이터 전처리	결측치 제거	박희진, 김동현
		이미지 전처리	데이터 증강,resize, 텐서화, RGB정규화	박희진, 김동현
	RNN 모델 데이터 전처리	데이터 전처리	소문자 통일, 불용어 제거, 결측치 제거, 패딩 설정	이화은, 양현우
		단어사전 생성	단어사전 생성	이화은, 양현우
기능 설계	모델 기능 검토 및 비교	모델 분석	여러 모델 성능 분석 및 모델 구조 고찰	김동현
	모델 구축	분류 모델	포스터로 장르 분류 모델	박희진
			줄거리로 로맨스 영화 이진 분류 모델	이화은
		회귀 모델	포스터로 평점 예측 모델	김동현
			줄거리로 평점 예측 모델	양현우
구현	웹	모델 예측	각 모델 예측	박희진, 이화은

INSTRUCTIONS

1. 포스터로 영화 장르 분류
2. 포스터로 영화 평점 예측
3. 영화 줄거리로 로맨스 영화 분류
4. 영화 줄거리로 평점 예측



크롤링 – TVMB



각 영화 링크 추출

개요 ▾ 미디어 ▾ 팬덤 ▾ 공유 ▾

듄: 파트 2 (2024)

12 2024/02/28 (KR) • SF, 모험 2h 47m

83% 회원 점수

What's your Vibe? 0

▶ 트레일러 재생

운명의 반격이 시작된다!

개요

황제의 모략으로 멸문한 가문의 유일한 후계자 폴. 어머니 레이디 제시카와 간신히 목숨만 부지한 채 사막으로 도망친다. 그곳에서 만난 반란군들과 숨어 지내다 그들과 함께 황제의 모든 것을 파괴할 전투를 준비한다. 한편 반란군들의 기세가 높아질수록 불안해진 황제와 귀족 가문은 잔혹한 암살자 페이드 로타를 보내 반란군을 몰살하려 하는데...

Denis Villeneuve
Director, Screenplay

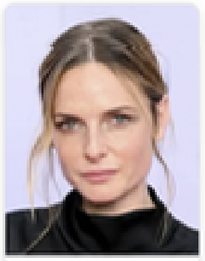
Frank Herbert
Novel

Jon Spaihts
Screenplay

주요 출연진

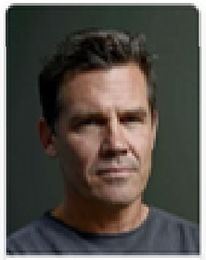
Timothée Chalamet
Paul Atreides

Zendaya
Chani

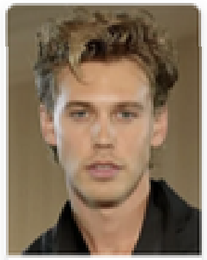


Rebecca Ferguson
Jessica

Javier Bardem
Stilgar



Josh Brolin
Gurney Halleck



Austin Butler
Feyd-Rautha

Florence P
Princess Irul

원제
Dune: Part Two

상태
개봉됨

원어
영어

제작비
$190,000,000.00

수익
$683,813,734.00

총 출연진 & 제작진

COLUMNS

포스터
제목
장르
러닝타임
회원점수
줄거리
제작비
수익
감독

크롤링 결과

```python
# 포스터
try:
    poster = driver.find_element(By.XPATH, '//*[@id="original_header"]/div[1]/div/div[1]/div/img').get_attribute('src')
    print(poster)
    poster_list.append(poster)
except:
    poster = None
    print(poster)
    poster_list.append(poster)

# 제목
try:
    title = driver.find_element(By.XPATH, '//*[@id="original_header"]/div[2]/section/div[1]/h2/a').text
    print(title)
    title_list.append(title)
except:
    title = None
    print(title)
    title_list.append(title)

# 장르
try:
    glist = []
    genres_elements = driver.find_elements(By.XPATH, './/a[starts-with(@href, "/genre/")]')
    for genre_element in genres_elements:
        glist.append(genre_element.text)
    print(glist)
    genre_list.append(', '.join(glist))
except:
    genre_list.append(None)

# 러닝타임
try:
    duration = driver.find_element(By.CLASS_NAME, 'runtime').text
    print(duration)
    duration_list.append(duration)
except:
    duration = None
    print(duration)
    duration_list.append(duration)

# 회원 점수
try:
    score_element = driver.find_element(By.XPATH, '//*[@id="consensus_pill"]/div/div[1]/div/div/div/span').get_attribute('class')
    score_value = int(re.search(r'\d+', score_element).group()) # 숫자만 추출하여 int형으로 저장
```

```
<div class="percent">flex == $0
  <span class="icon icon-r83">...</span>
</div>
```

정규표현식 사용

→ 셀레늄 이용

크롤링 결과

(9722, 9)

	포스터	제목	장르	러닝 타임	회원 점수	줄거리	제작비	수익	감독
0	https://media.themoviedb.org/t/p/w300_and_h450...	Dune: Part Two	Science Fiction, Adventure	2h 47m	83.0	Follow the mythic journey of Paul Atreides as ...	666813734.0	NaN	Denis Villeneuve
1	https://media.themoviedb.org/t/p/w300_and_h450...	Kung Fu Panda 4	Animation, Action, Adventure, Comedy, Family	1h 34m	71.0	Po is gearing up to become the spiritual leade...	410255055.0	NaN	Mike Mitchell
2	https://media.themoviedb.org/t/p/w300_and_h450...	Godzilla x Kong: The New Empire	Action, Science Fiction, Adventure, Fantasy	1h 55m	67.0	Following their explosive showdown, Godzilla a...	361305986.0	NaN	Adam Wingard
3	https://media.themoviedb.org/t/p/w300_and_h450...	Alienoid	Science Fiction, Action, Fantasy, Adventure	2h 22m	71.0	Gurus in the late Goryeo dynasty try to obtain...	24500000.0	12600000.0	Choi Dong-hoon
4	https://media.themoviedb.org/t/p/w300_and_h450...	The Wages of Fear	Action, Thrillerp	1h 47m	60.0	When an explosion at an oil well threatens hun...	NaN	NaN	Julien Leclercq
5	https://media.themoviedb.org/t/p/w300_and_h450...	Road House	Action, Thriller	2h 1m	71.0	Ex-UFC fighter Dalton takes a job as a bouncer...	NaN	NaN	Charles Mondry
6	https://media.themoviedb.org/t/p/w300_and_h450...	After the Pandemic	Science Fiction, Action	1h 24m	51.0	Set in a post-apocalyptic world where a global...	NaN	NaN	Richard Lowry
7	https://media.themoviedb.org/t/p/w300_and_h450...	Madame Web	Action, Fantasy	1h 56m	57.0	Forced to confront revelations about her past,...	99266032.0	NaN	S.J. Clarkson
8	https://media.themoviedb.org/t/p/w300_and_h450...	The Tearsmith	Romance, Drama	1h 46m	68.0	Adopted together after a tough childhood in an...	7000000.0	NaN	Alessandro Genovesi
9	https://media.themoviedb.org/t/p/w300_and_h450...	Heart of the Hunter	Action, Mystery, Thriller	1h 48m	58.0	A retired assassin is pulled back into action ...	NaN	NaN	Deon Meyer
10	https://media.themoviedb.org/t/p/w300_and_h450...	Creation of the Gods I: Kingdom of Storms	Action, Fantasy, War	2h 28m	69.0	Based on the most well-known classical fantasy...	85000000.0	372945000.0	Wuershan
11	https://media.themoviedb.org/t/p/w300_and_h450...	Migration	Animation, Action, Adventure, Comedy, Family	1h 23m	75.0	After a migrating duck family alights on their...	292496973.0	NaN	Benjamin Renner
12	https://media.themoviedb.org/t/p/w300_and_h450...	The Adventures of Maid Marian	Adventure, Action	1h 22m	49.0	Everyone knows the stories of Robin Hood and M...	NaN	NaN	Bill Thomas
13	https://media.themoviedb.org/t/p/w300_and_h450...	Imaginary	Horror, Mystery, Thriller	1h	63.0	When Jessica moves back	38000000.0	NaN	Jeff Wadlow

CATEGORY 1

포스터로 영화 장르 분류

박희진

데이터 전처리 각 라벨 당 데이터가 적고 불균형이 심함

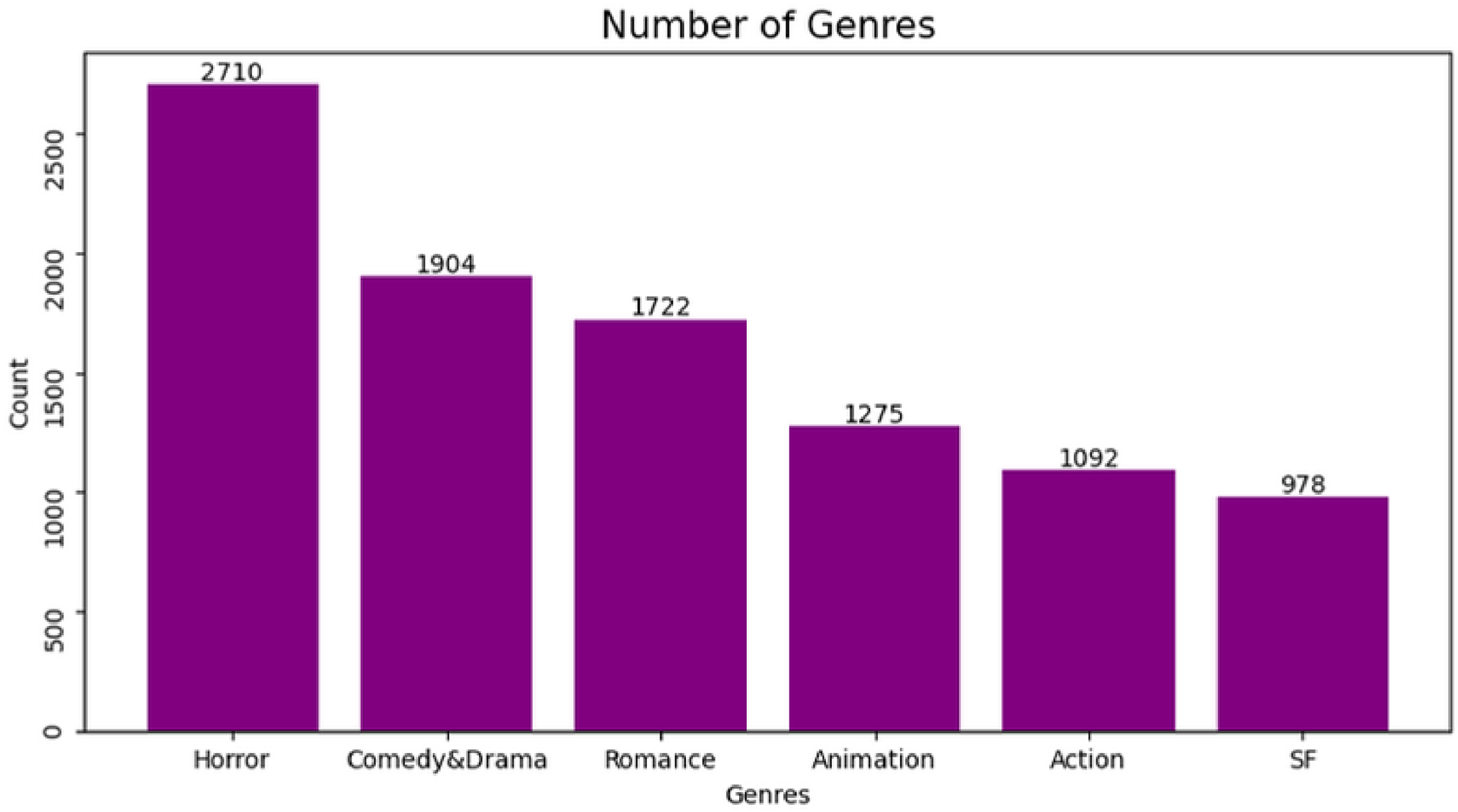
```
{'Action',
 'Adventure',
 'Animation',
 'Comedy',
 'Crime',
 'Documentary',
 'Drama',
 'Family',
 'Fantasy',
 'History',
 'Horror',
 'Music',
 'Mystery',
 'Romance',
 'Science Fiction',
 'TV Movie',
 'Thriller',
 'Thrillerp',
 'War',
 'Western'}
```

Animation > SF > Horror > Romance > Action > Comedy&Drama

```
for i in genres.index:
    if 'Animation' in genres[i]:
        genres[i] = 'Animation'
    elif 'Science Fiction' in genres[i]:
        genres[i] = 'SF'
    elif 'Horror' in genres[i] or 'Thriller' in genres[i]:
        genres[i] = 'Horror'
    elif 'Romance' in genres[i] or 'Family' in genres[i]:
        genres[i] = 'Romance'
    elif 'Action' in genres[i] or 'War' in genres[i] or 'Western' in genres[i]:
        genres[i] = 'Action'
    else:
        genres[i] = 'Comedy&Drama'
```

	포스터	장르
0	https://media.themoviedb.org/t/p/w300_and_h450...	SF
1	https://media.themoviedb.org/t/p/w300_and_h450...	Animation
2	https://media.themoviedb.org/t/p/w300_and_h450...	SF
3	https://media.themoviedb.org/t/p/w300_and_h450...	SF
4	https://media.themoviedb.org/t/p/w300_and_h450...	Horror
5	https://media.themoviedb.org/t/p/w300_and_h450...	Horror
6	https://media.themoviedb.org/t/p/w300_and_h450...	SF
7	https://media.themoviedb.org/t/p/w300_and_h450...	Action
8	https://media.themoviedb.org/t/p/w300_and_h450...	Romance
9	https://media.themoviedb.org/t/p/w300_and_h450...	Horror
10	https://media.themoviedb.org/t/p/w300_and_h450...	Action
11	https://media.themoviedb.org/t/p/w300_and_h450...	Animation
12	https://media.themoviedb.org/t/p/w300_and_h450...	Action
13	https://media.themoviedb.org/t/p/w300_and_h450...	Horror
14	https://media.themoviedb.org/t/p/w300_and_h450...	SF
15	https://media.themoviedb.org/t/p/w300_and_h450...	Horror
16	https://media.themoviedb.org/t/p/w300_and_h450...	Action
17	https://media.themoviedb.org/t/p/w300_and_h450...	Action
18	https://media.themoviedb.org/t/p/w300_and_h450...	Horror
19	https://media.themoviedb.org/t/p/w300_and_h450...	Horror
20	https://media.themoviedb.org/t/p/w300_and_h450...	SF
21	https://media.themoviedb.org/t/p/w300_and_h450...	Horror
22	https://media.themoviedb.org/t/p/w300_and_h450...	Horror
23	https://media.themoviedb.org/t/p/w300_and_h450...	Comedy&Drama
24	https://media.themoviedb.org/t/p/w300_and_h450...	Action

데이터 분포 확인 각 라벨 당 데이터가 적고 불균형이 심함



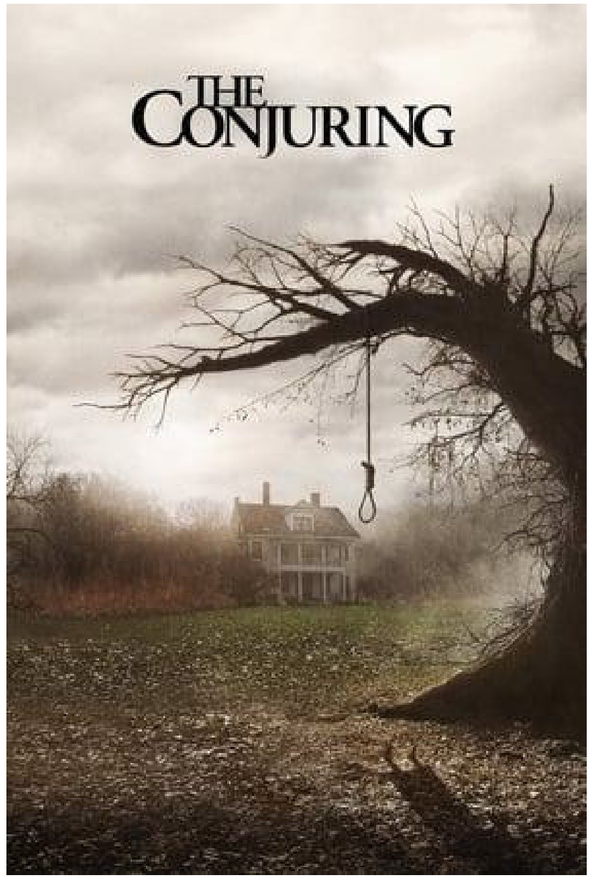
데이터 전처리

장르별 이미지 폴더 생성

```python
# 이미지 저장 폴더 생성
for genre in genres.unique():
    os.makedirs(os.path.join('data', genre), exist_ok=True)

# 이미지를 각 장르의 폴더에 저장
for idx, (image_url, genre) in enumerate(zip(image_urls, genres)):
    response = requests.get(image_url)
    if response.status_code == 200:
        image = Image.open(BytesIO(response.content))
        file_path = os.path.join('data', genre, f"img_{idx}.jpg")
        image.save(file_path)
        print(f"이미지 저장: {file_path}")
    else:
        print(f"이미지 다운로드 실패: {image_url}")
```

- Action
- Animation
- Comedy&Drama
- Horror
- Romance
- SF

이미지 확인 (3, 350, 400)



데이터 전처리 이미지 전처리

```python
preprocessing = transforms.Compose([
    transforms.Resize((300,300)),
    transforms.ToTensor()
    ]) # 각 이미지 사이즈 (300, 450) # 사이즈 모두 동일
```

```python
# 이미지와 라벨(폴더명)을 쌍으로 저장할 리스트
images = []
labels = []

# 각 폴더(장르)별로 이미지를 불러와서 리스트에 저장
for genre in os.listdir(genre_folder):
    genre_path = os.path.join(genre_folder, genre)
    if os.path.isdir(genre_path):
        for file_name in os.listdir(genre_path):
            img_path = os.path.join(genre_path, file_name)  # 이미지 파일 경로
            # 이미지 불러오기
            try:
                image = Image.open(img_path)
                image = preprocessing(image)
                if image.shape[0] == 3:  # 채널이 3인 경우에만 저장
                    images.append(image)
                    labels.append(genre)
            except Exception as e:
                print(f"이미지 불러오기 실패: {img_path}, Error: {e}")
```

```python
normalize = transforms.Normalize(mean=[meanR, meanG, meanB], std=[stdR, stdG, stdB])

# 데이터셋 내의 모든 이미지에 대해 정규화 진행
trainDS = [(normalize(img), label) for img, label in trainDS]
validDS = [(normalize(img), label) for img, label in validDS]
```

데이터 전처리 for 데이터 불균형 완화

라벨 균형 맞춰서 trainset, validset 분리

```
xtrain, xval, ytrain, yval = train_test_split(images, labels, test_size=0.2, random_state=42, stratify=labels)
```

샘플러 이용하여 데이터 로더에서 가능한 균형적이게 데이터가 나오도록 설정

```
# 클래스별 비율 계산
train_class = {label: count / len(trainDS) for label, count in Counter([label for _, label in trainDS]).items()}
valid_class = {label: count / len(validDS) for label, count in Counter([label for _, label in validDS]).items()}
```

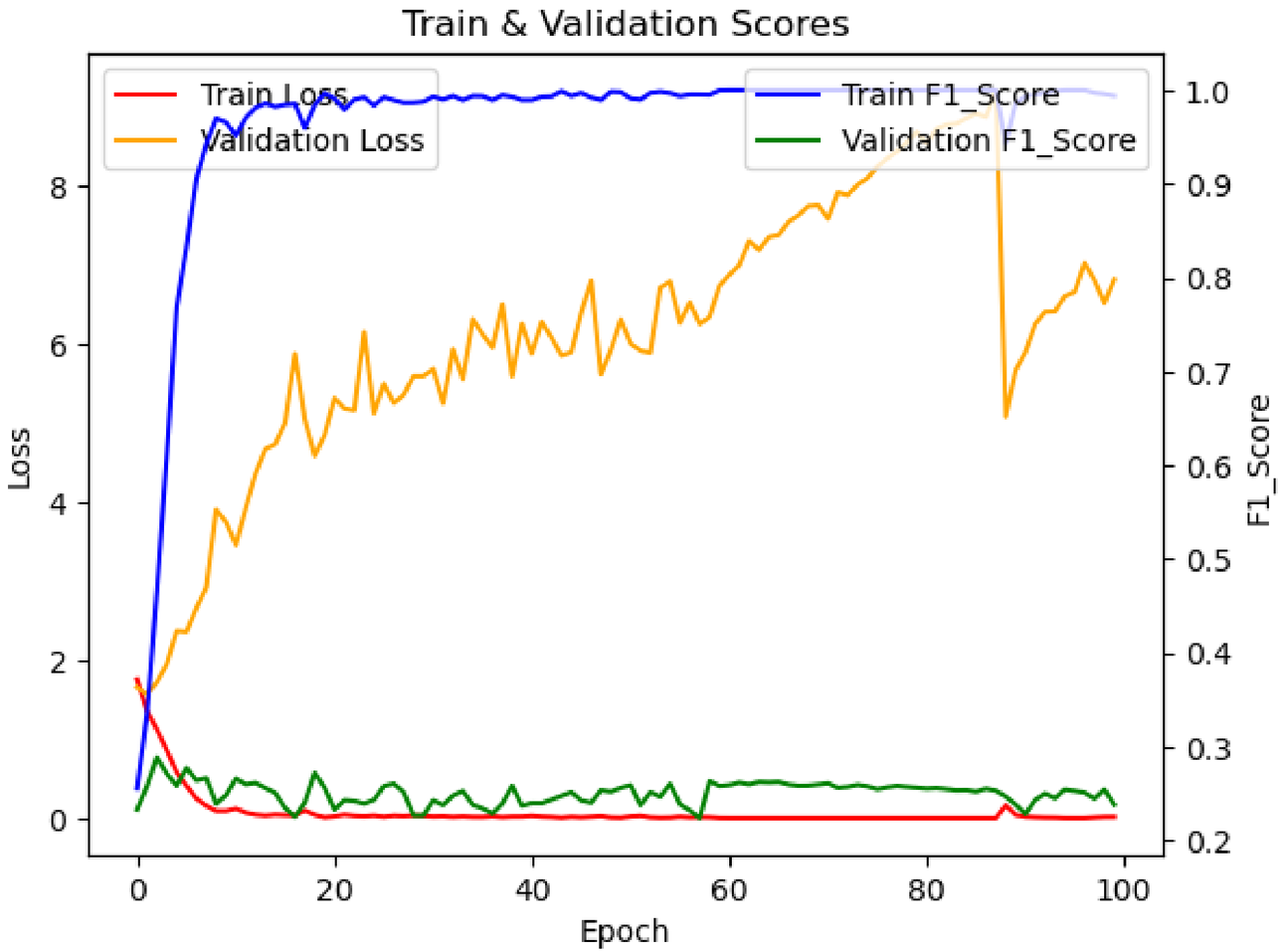
```
SAMPLER = WeightedRandomSampler(weights=[train_class[label] for _,label in trainDS], num_samples=len(trainDS), replacement=True)

BATCH = 32
# DataLoader 생성
trainDL = DataLoader(trainDS, batch_size=BATCH, sampler=SAMPLER)
validDL = DataLoader(validDS, batch_size=BATCH)
```

CNN MODEL 정의

```python
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, stride=1, padding=1)
        self.bn1 = nn.BatchNorm2d(16)  # 배치 정규화 층 추가
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv2 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=3, stride=1, padding=1)
        self.bn2 = nn.BatchNorm2d(32)  # 배치 정규화 층 추가

        self.fc1 = nn.Linear(32 * 75 * 75, 32)
        self.fc2 = nn.Linear(32, 16)
        self.fc3 = nn.Linear(16, 6)  # 클래스의 수에 맞게 수정

        nn.init.kaiming_uniform_(self.conv1.weight)
        nn.init.kaiming_uniform_(self.conv2.weight)

    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)  # 배치 정규화 적용
        x = F.relu(x)
        x = self.pool(x)  # 풀링 적용

        x = self.conv2(x)
        x = self.bn2(x)  # 배치 정규화 적용
        x = F.relu(x)
        x = self.pool(x)

        x = x.view(-1, 32 * 8 * 8)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.fc2(x)
        x = F.relu(x)
        x = self.fc3(x)

        return x
```

배치 정규화
가중치초기화

CNN MODEL 성능



F1 SCORE

0.2 대

문제점이 뭘까?

1. 적은 데이터
2. 불균형한 데이터
3. 이미지 전처리에서 설정한 이미지 크기
4. 배치사이즈
5. 학습률
6. 모델 복잡도

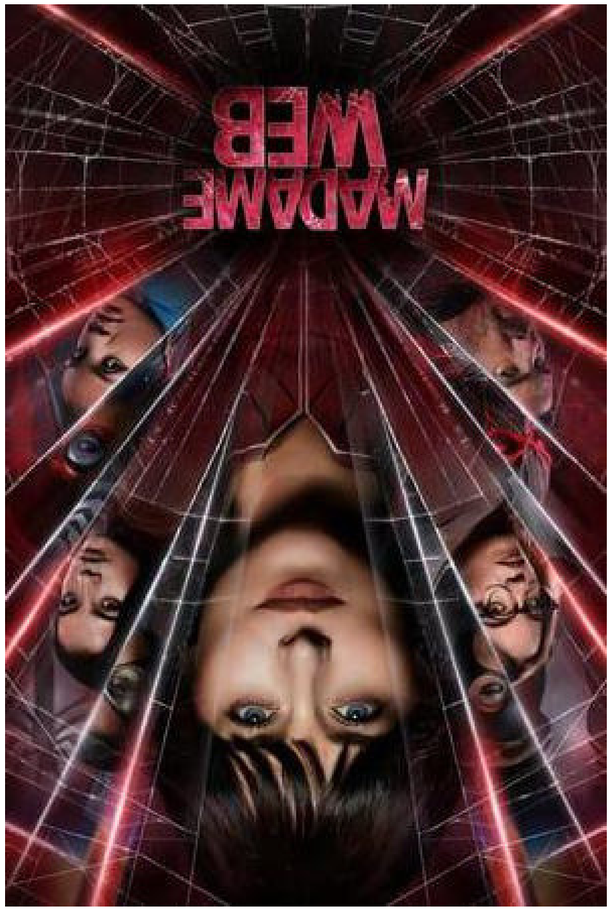
데이터 전처리 데이터 업샘플링 - rotate

```python
def img_up(genre, rotation_angle):
    # 회전 각도 설정
    folder_path = f'data/{genre}'
    rotation_angle = rotation_angle
    for filename in os.listdir(folder_path):
        # 파일 경로 확인
        file_path = os.path.join(folder_path, filename)
        # 이미지 파일 경로
        img_path = file_path
        # 이미지 불러오기
        img = Image.open(img_path)
        # 이미지 회전
        rotated_img = img.rotate(rotation_angle)
        # 회전된 이미지 저장 (파일명에 회전 각도 추가)
        rotated_img_path = os.path.join(folder_path, filename.split('.')[0] + f"_rotated_{rotation_angle}.jpg")
        rotated_img.save(rotated_img_path)
```

최대한 포스터의 이미지를 왜곡하지 않는 증강 방법 선택

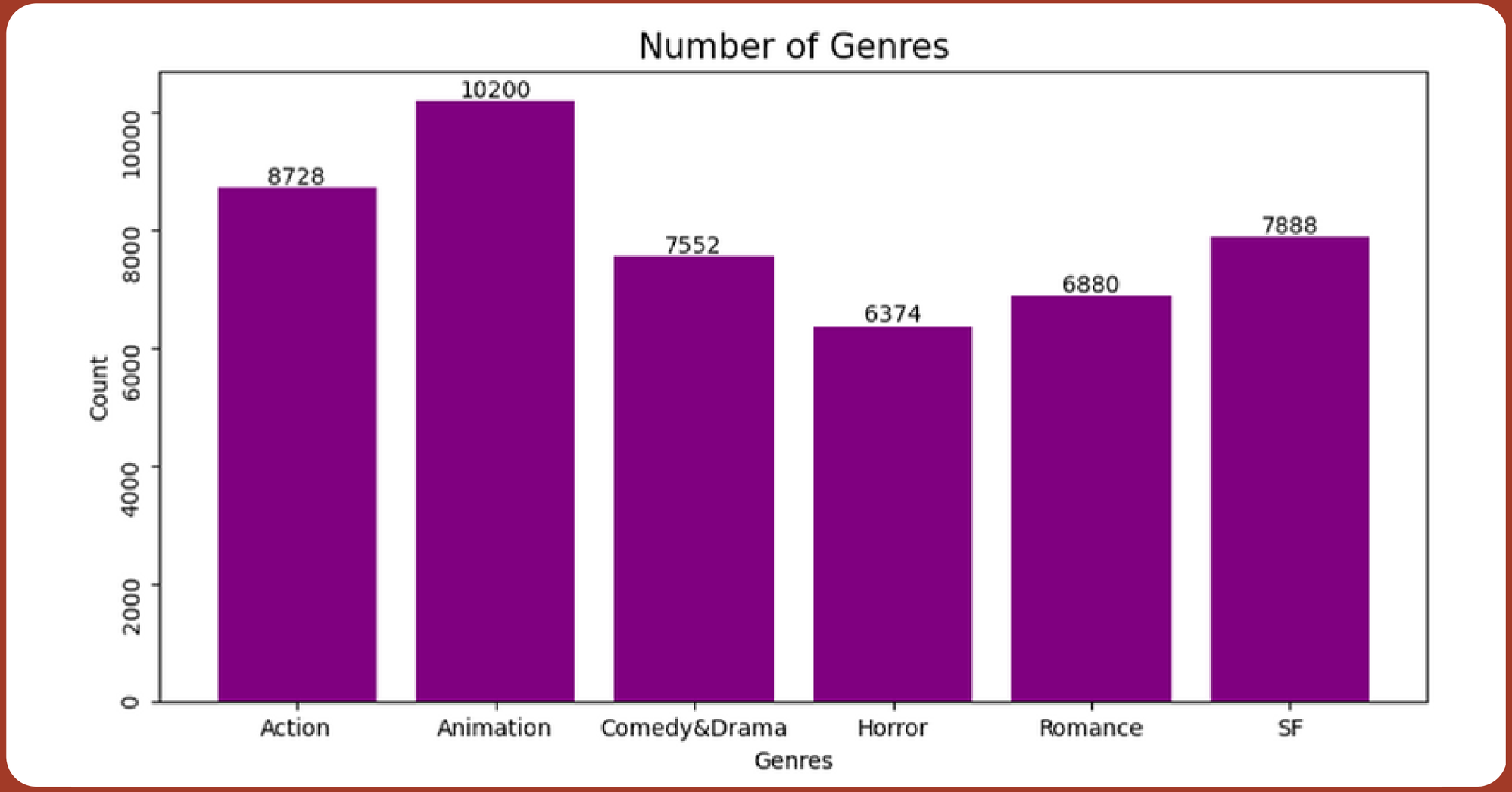
```python
img_up('SF', 45)
```

이미지 확인 (3, 350, 400)

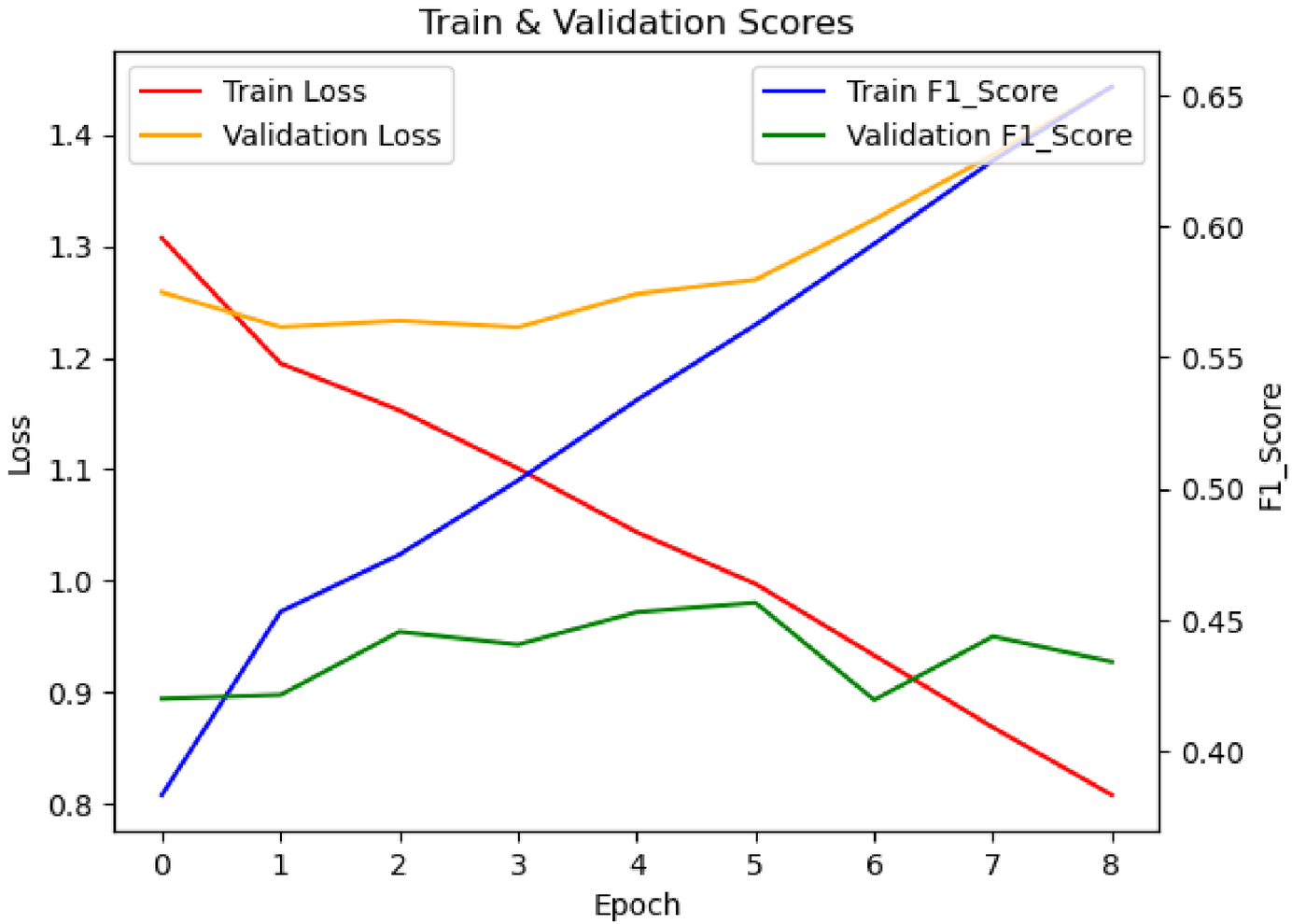


데이터 전처리

데이터 업샘플링



MODEL 성능 데이터 증강 후



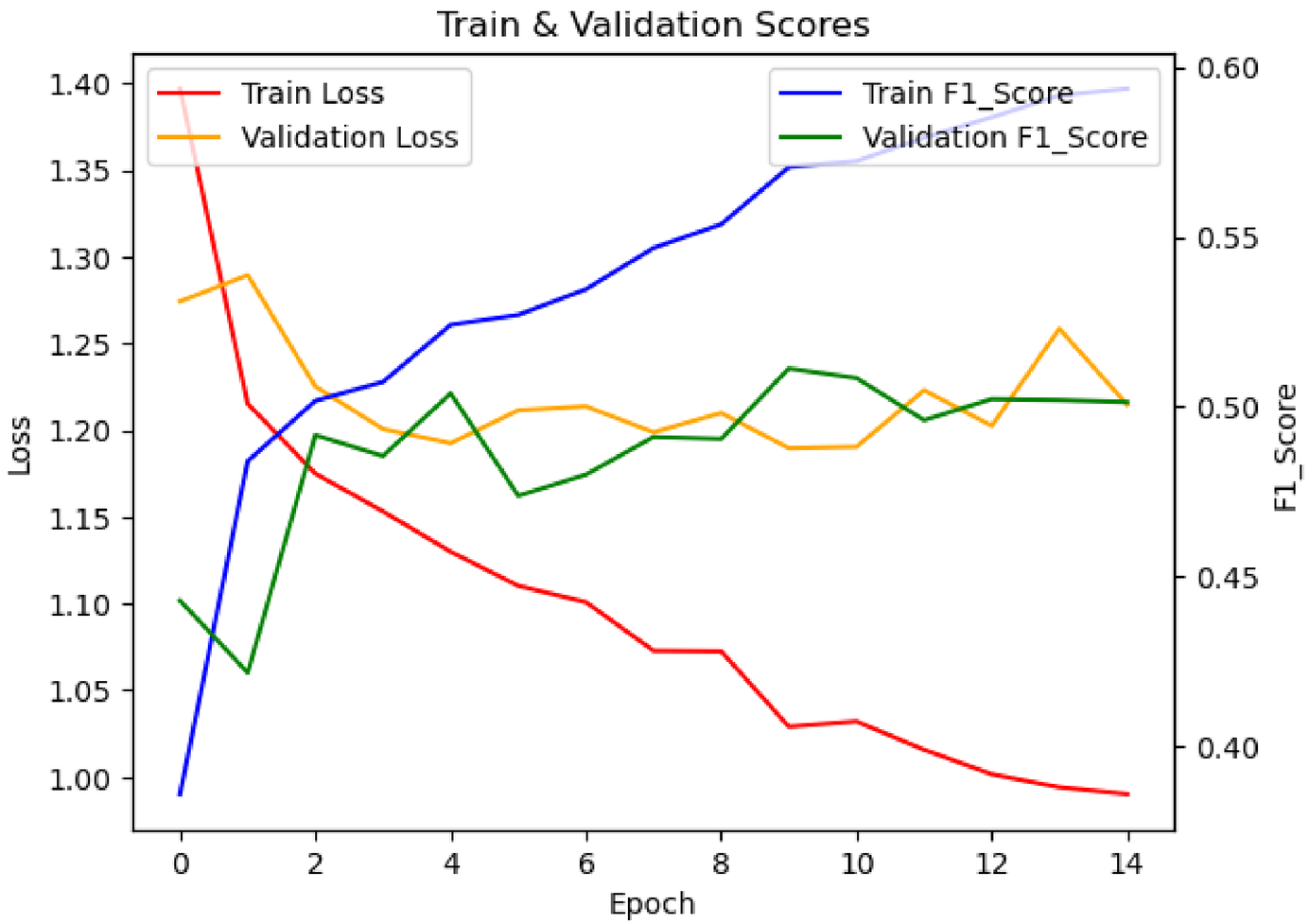
TRAIN LOSS : 0.9974

TRAIN F1 SCORE : 0.56

TEST LOSS : 1.2699

TEST F1 SCORE : 0.46

MODEL 성능 이미지 크기 조절 – (32, 32)



TRAIN LOSS : 1.0291

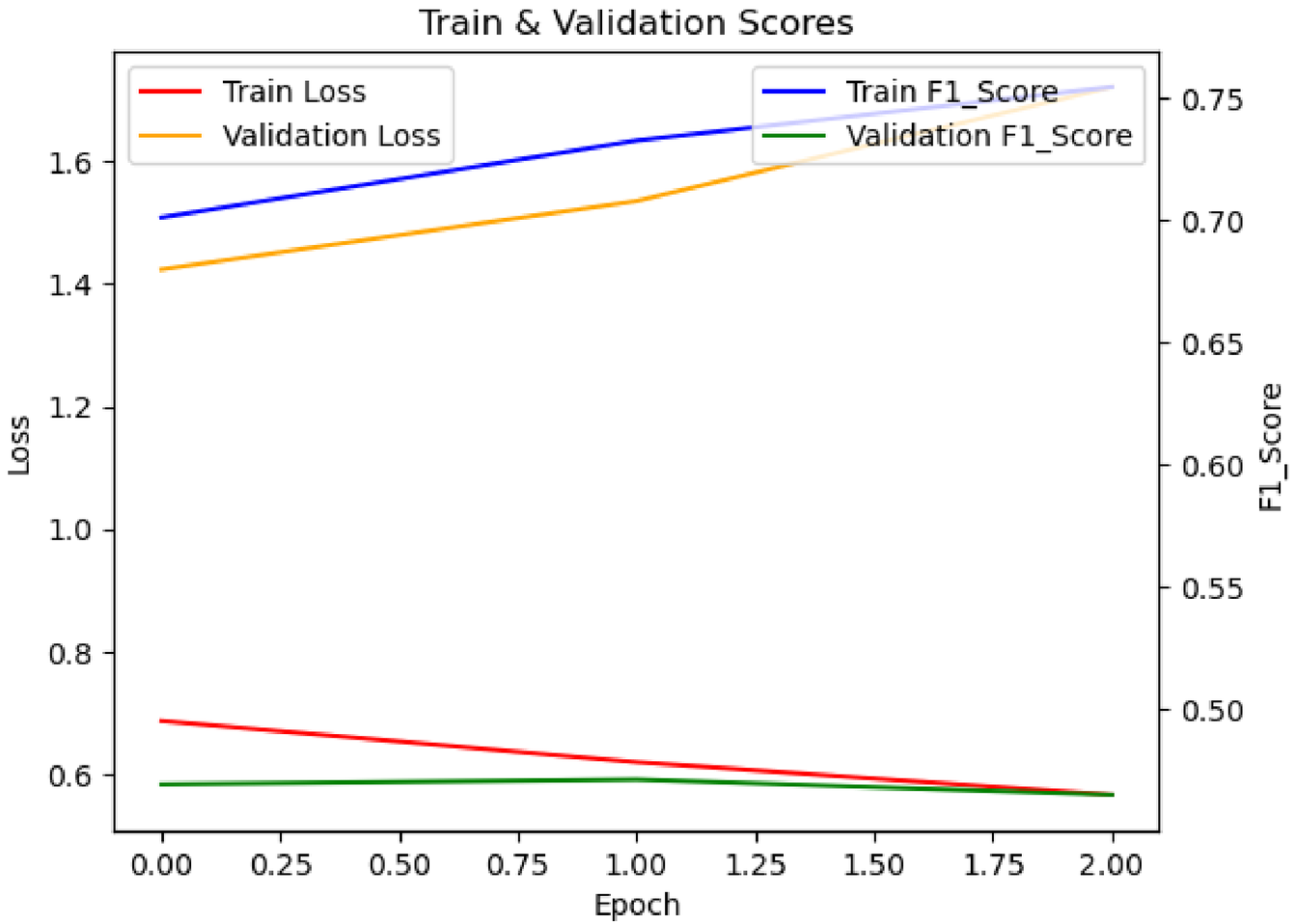
TRAIN F1 SCORE : 0.57

TEST LOSS : 1.1896

TEST F1 SCORE : 0.51

MODEL 성능

모델 복잡도 증가



매우 심한 과적합 !!

TRAIN LOSS : 0.5661

TRAIN F1 SCORE : 0.75

TEST LOSS : 1.7215

TEST F1 SCORE : 0.47

최종 모델 MODEL

이미지 크기 (32, 32)

배치 사이즈 = 256

학습률 = 0.001

모델은 처음과 동일하게

-> 합성곱층 2개 / 풀링층 2개

-> 배치 정규화 적용

-> 가중치 초기화

>>>

TRAIN LOSS : 1.0291

TRAIN F1 SCORE : 0.57

TEST LOSS : 1.1896

TEST F1 SCORE : 0.51

성능 향상

0.22 >> 0.51

아쉬운 점

SF

SF, 공포, 모험

장르는 보통 하나에 국한되지 X

다중 레이블 분류 모델을 만들었다면
더욱 성능이 좋았을 것

CATEGORY 2

포스터로 영화 회원 평점 예측

김동현

데이터 전처리

결측 데이터 처리

```
# 제목 결측된 데이터는 한개=> 'non title(missing)'으로 수정
movieDF['제목'] = movieDF['제목'].fillna('non title(missing)')
```

✓ 0.0s

```
# 제목 결측 데이터 없는지 확인
movieDF['제목'].isna().sum()
```

✓ 0.0s

```
0
```

```
# 포스터 결측된 데이터 제거
movieDF.dropna(subset = ['포스터'], inplace = True)
```

✓ 0.0s

```
# 포스터 결측 데이터 없는지 확인
movieDF['포스터'].isna().sum()
```

✓ 0.0s

```
0
```

```
movieDF.shape
```

✓ 0.0s

```
(9704, 9)
```

```
# # 회원점수가 잘못된 데이터 삭제
string = '''The Best of Enemies,Centers on the unlikely

movieDF = movieDF[movieDF['회원점수'] != string]
```

✓ 0.0s

```
# 회원점수 데이터 float으로 타입변경
movieDF['회원점수'] = movieDF['회원점수'].astype('float')
```

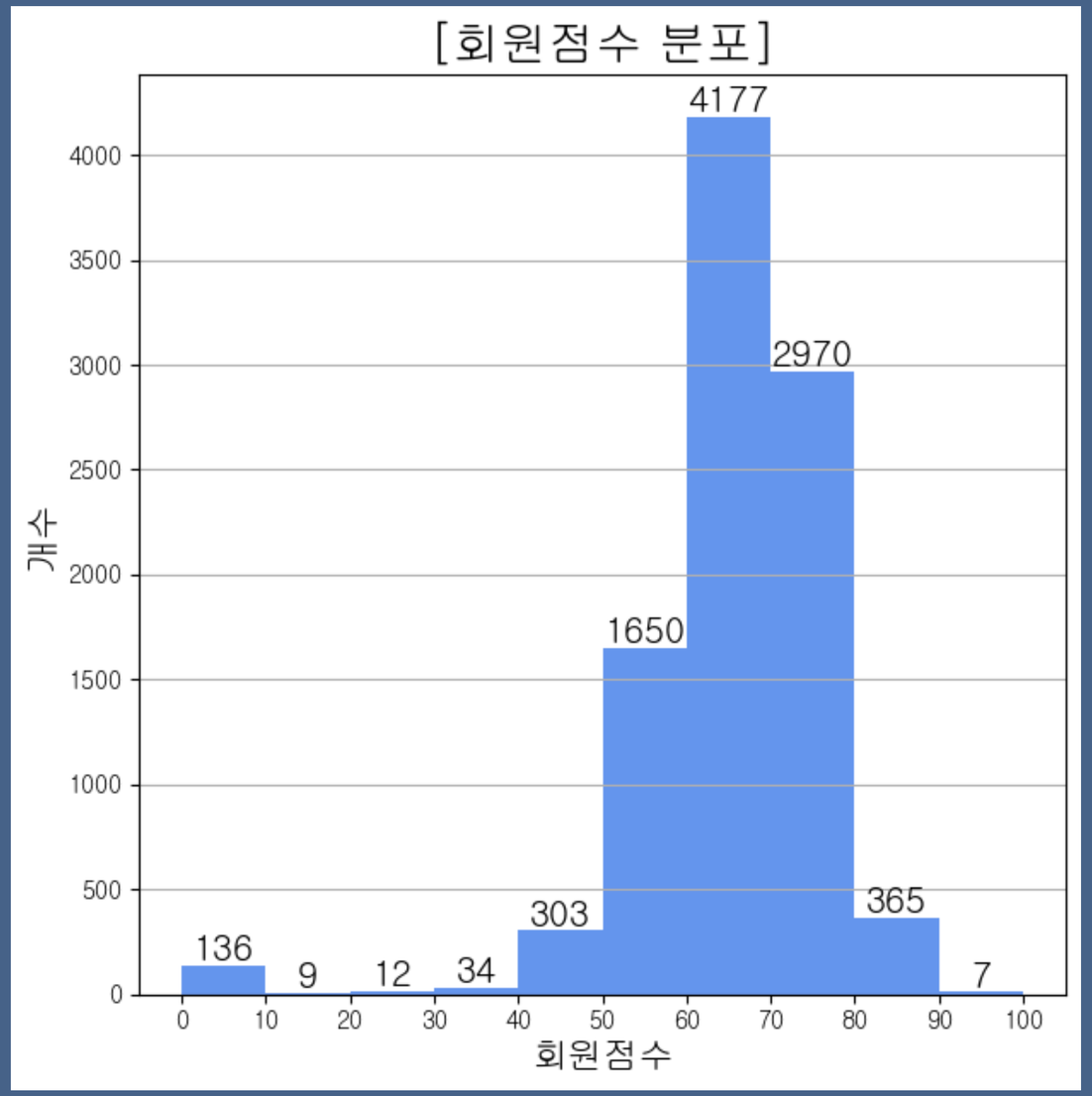
✓ 0.0s

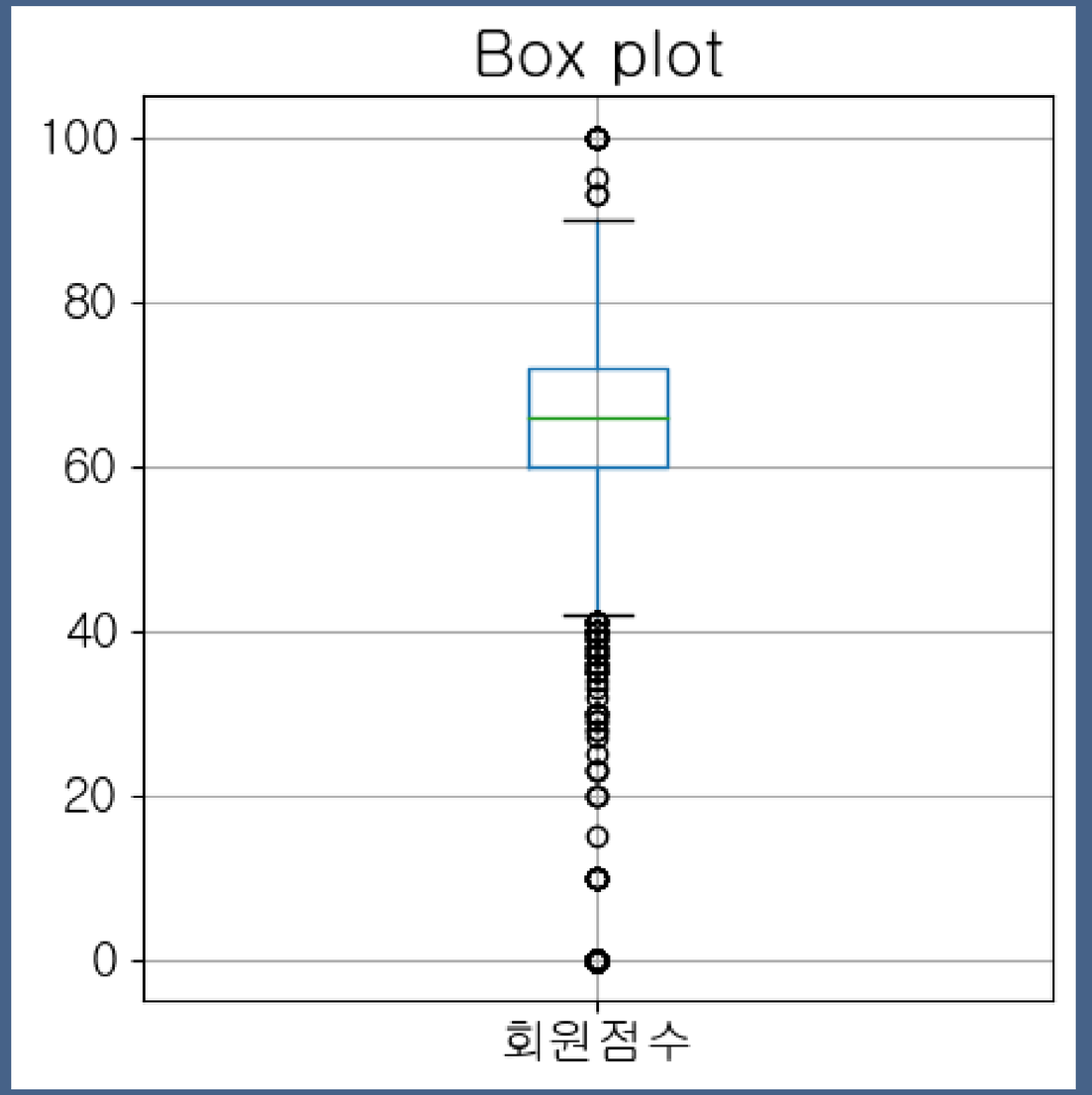
```
# 회원점수 데이터 타입 변경 후 확인
movieDF.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
Index: 9703 entries, 0 to 9721
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   포스터     9703 non-null   object
 1   이미지 경로  9703 non-null   object
 2   제목       9703 non-null   object
 3   장르       9680 non-null   object
 4   러닝타임     9600 non-null   object
 5   회원점수     9673 non-null   float64
 6   줄거리      9673 non-null   object
 7   제작비      5917 non-null   float64
 8   수익       901 non-null    object
 9   감독       9673 non-null   object
dtypes: float64(2), object(8)
memory usage: 833.9+ KB
```

데이터 전처리


[회원점수 분포]
개수
회원점수
4177
2970
1650
303
365
136
9
12
34
7
0
500
1000
1500
2000
2500
3000
3500
4000
0
10
20
30
40
50
60
70
80
90
100


Box plot
회원점수
0
20
40
60
80
100

회원 점수 이상치 제거

```
# 사분위수 기반 이상치 제거 후 확인
movieDF = delete_q_outlier(['회원점수'])
movieDF.shape
```

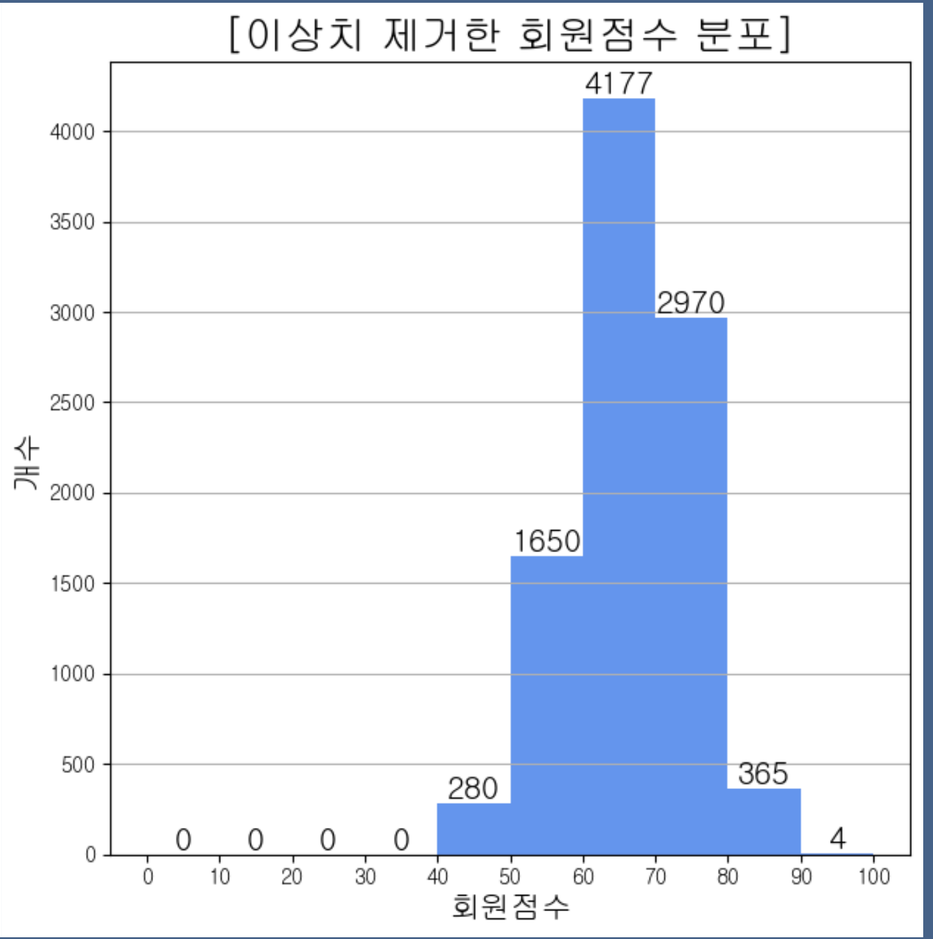
✓ 0.0s

```
(9446, 10)
```

```
find_q_outlier(['회원점수'])
```

✓ 0.0s

회원점수 특성의 사분위수 기반 이상치 개수 : 227개



흑백 포스터 제거

```
# jpg 포스터가 아닌 데이터는 삭제
ind = 0
for poster in movieDF['이미지 경로'].values:
    if os.path.splitext(poster)[1] != '.jpg':
        movieDF.drop(ind, inplace = True)
    ind += 1
```

✓ 0.0s

```
# jpg 형식이 아닌 포스터 없음
movieDF.shape
```

✓ 0.0s

```
(9446, 10)
```

```
movieDF.reset_index(drop = True, inplace = True)
```

✓ 0.0s

```
# 컬러 포스터가 아닌 데이터는 삭제
ind = 0
for poster in movieDF['이미지 경로'].values:
    if np.array(Image.open(poster)).ndim != 3:
        movieDF.drop(ind, inplace = True)
    ind += 1
```

✓ 2m 31.3s

```
# 컬러 포스터가 아닌 데이터 30개 삭제
movieDF.shape
```

✓ 0.0s

```
(9416, 10)
```

```
movieDF.reset_index(drop = True, inplace = True)
```

✓ 0.0s

```
# 흑백이미지 존재 여부 확인 => 없음
for poster in movieDF['이미지 경로'].values:
    if np.array(Image.open(poster)).ndim != 3:
        print('2차원 이미지')
```

✓ 53.2s

IMAGE 데이터 처리

```
transform = transforms.Compose([
    transforms.Resize((64, 64)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],  # ImageNet 데이터셋의 평균값
                         std=[0.229, 0.224, 0.225])]) # ImageNet 데이터셋의 표준편차 std=[0.229, 0.224, 0.225]
```

✓ 0.0s

원본 사이즈 (450, 300)에서 (64, 64)로 변경

ImageNet이 학습한 수백만장의 이미지의 RGB 각각의 채널에 대한 평균은 0.485, 0.456, 0.406 그리고 표준편차는 0.229, 0.224, 0.225.
만약, 일반적인 조도, 각도, 배경을 포함하는 평범한 이미지의 경우는 (0.485, 0.456, 0.406), (0.229, 0.224, 0.225)으로 정규화하는 것을 추천한다는 커뮤니티 의견이 지배적

모델 구성

모델 => ResNet18, 사전학습 X

Linear층 3개 추가

-> 배치 정규화, 드롭아웃(0.5)

학습률 = 0.01

활성화 함수 ReLU

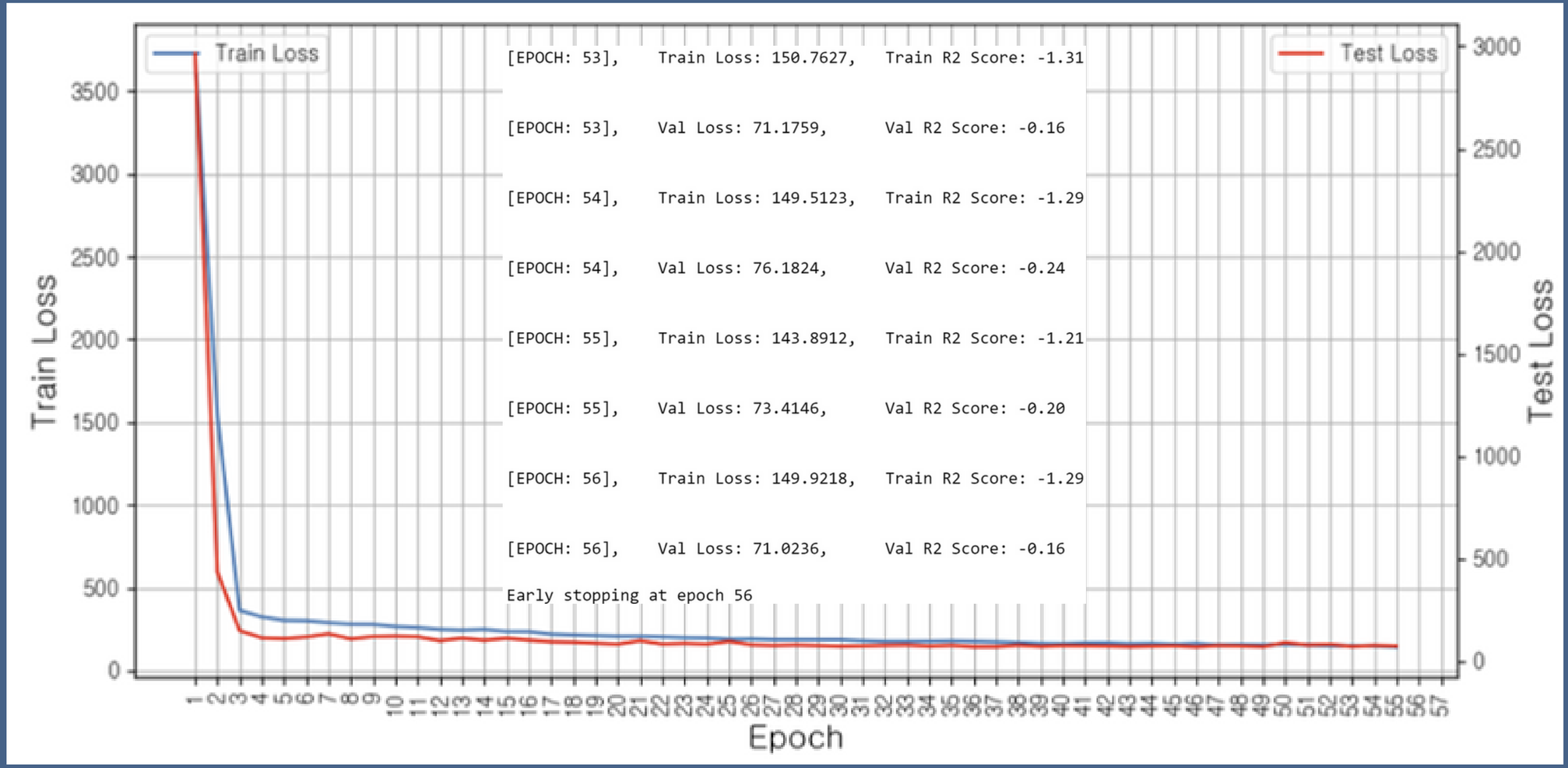
전체 층 He 가중치 초기화

배치 사이즈 = 128

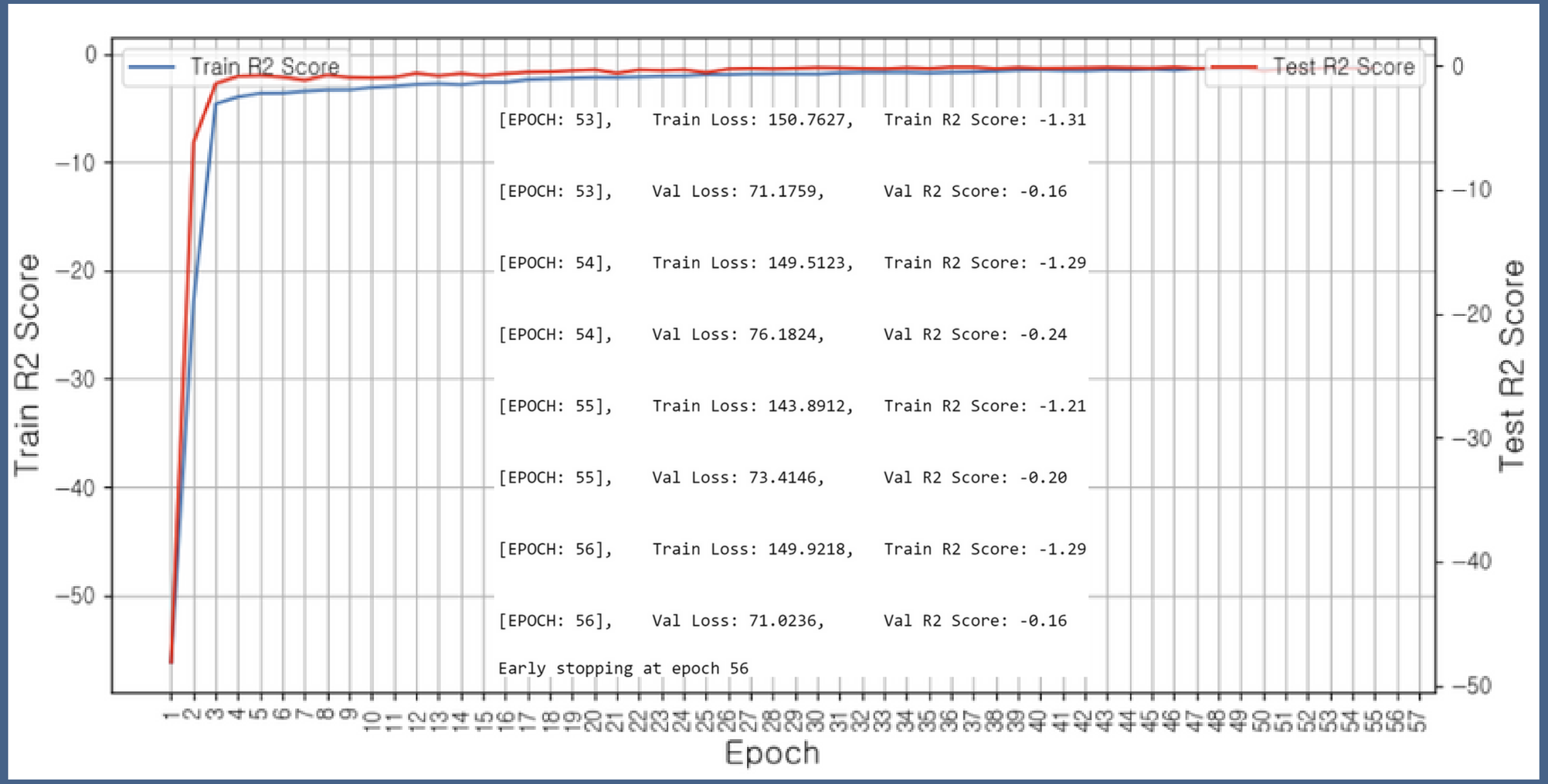
EPOCH = 100

학습 스케쥴러 PATIENCE = 10

MSE LOSS

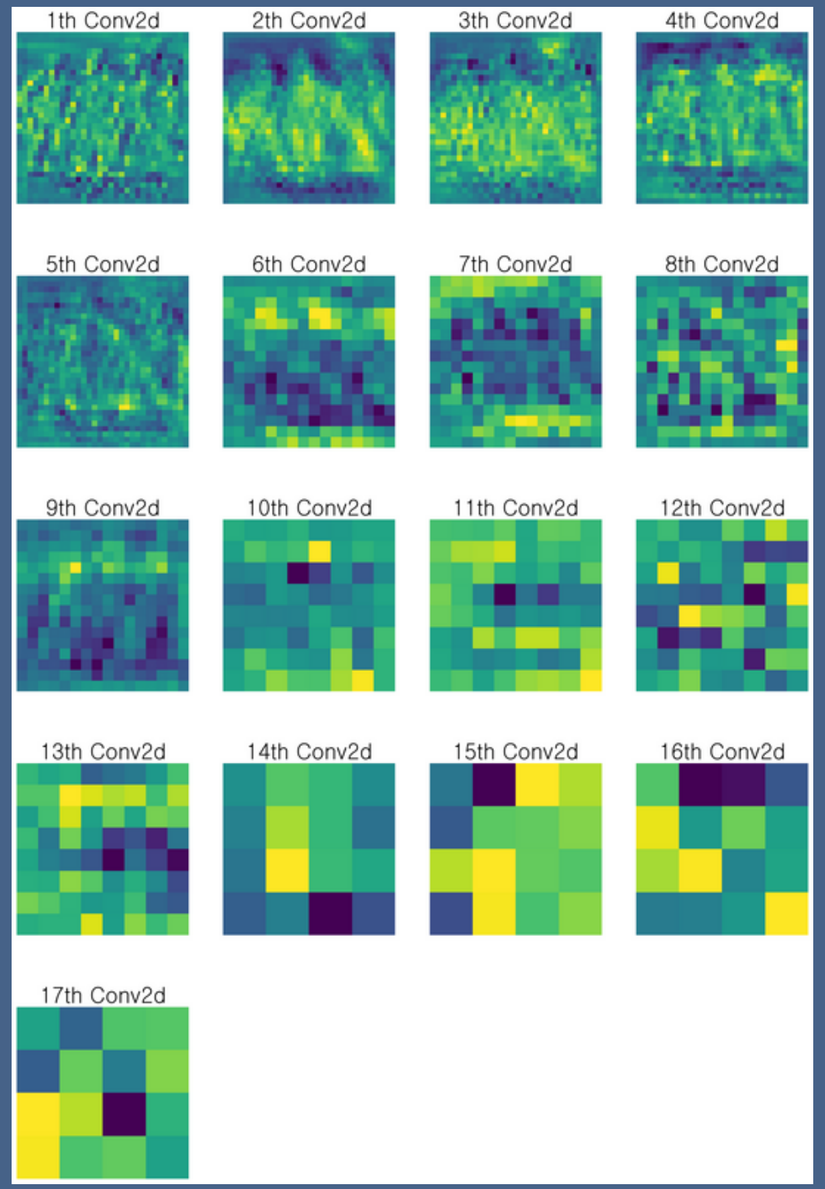

Train Loss
Test Loss
[EPOCH: 53], Train Loss: 150.7627, Train R2 Score: -1.31
[EPOCH: 53], Val Loss: 71.1759, Val R2 Score: -0.16
[EPOCH: 54], Train Loss: 149.5123, Train R2 Score: -1.29
[EPOCH: 54], Val Loss: 76.1824, Val R2 Score: -0.24
[EPOCH: 55], Train Loss: 143.8912, Train R2 Score: -1.21
[EPOCH: 55], Val Loss: 73.4146, Val R2 Score: -0.20
[EPOCH: 56], Train Loss: 149.9218, Train R2 Score: -1.29
[EPOCH: 56], Val Loss: 71.0236, Val R2 Score: -0.16
Early stopping at epoch 56
Train Loss
Test Loss
Epoch

R2 SCORE


Train R2 Score
Test R2 Score
[EPOCH: 53], Train Loss: 150.7627, Train R2 Score: -1.31
[EPOCH: 53], Val Loss: 71.1759, Val R2 Score: -0.16
[EPOCH: 54], Train Loss: 149.5123, Train R2 Score: -1.29
[EPOCH: 54], Val Loss: 76.1824, Val R2 Score: -0.24
[EPOCH: 55], Train Loss: 143.8912, Train R2 Score: -1.21
[EPOCH: 55], Val Loss: 73.4146, Val R2 Score: -0.20
[EPOCH: 56], Train Loss: 149.9218, Train R2 Score: -1.29
[EPOCH: 56], Val Loss: 71.0236, Val R2 Score: -0.16
Early stopping at epoch 56
Train R2 Score
Test R2 Score
Epoch

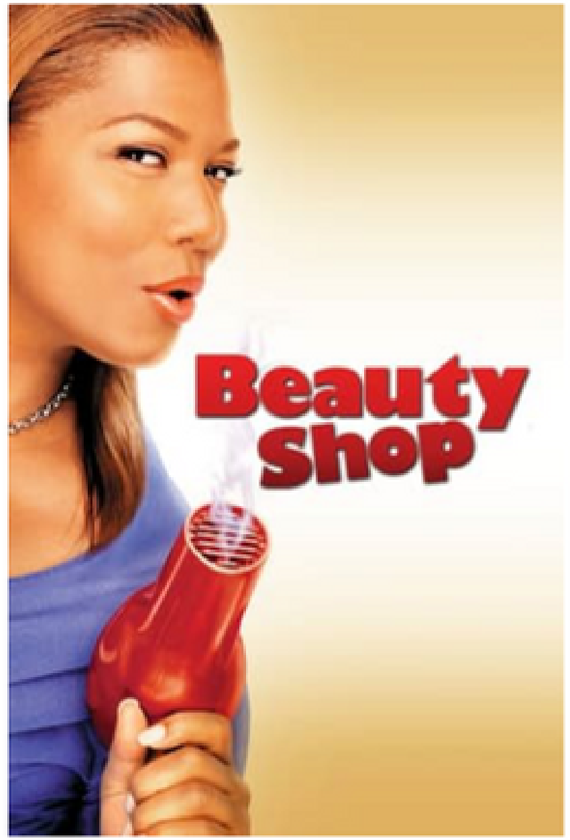
추출된 피처맵 확인


THE WAGES OF FEAR


1th Conv2d
2th Conv2d
3th Conv2d
4th Conv2d
5th Conv2d
6th Conv2d
7th Conv2d
8th Conv2d
9th Conv2d
10th Conv2d
11th Conv2d
12th Conv2d
13th Conv2d
14th Conv2d
15th Conv2d
16th Conv2d
17th Conv2d

예측 결과 확인


```
영화 ['Halloween'] 포스터에 대한 예측 회원 점수는 65.1점
실제 회원 점수는 62.0점


영화 ['Fearless'] 포스터에 대한 예측 회원 점수는 70.3점
실제 회원 점수는 75.0점


영화 ['Oblivion'] 포스터에 대한 예측 회원 점수는 69.2점
실제 회원 점수는 66.0점
```



영화 ['The Dressmaker'] 포스터에 대한 예측 회원 점수는 58.5점
실제 회원 점수는 70.0점

영화 ['Wendell & Wild'] 포스터에 대한 예측 회원 점수는 65.9점
실제 회원 점수는 67.0점

영화 ['Beauty Shop'] 포스터에 대한 예측 회원 점수는 60.6점
실제 회원 점수는 63.0점


```
영화 ['Taylor Swift: Reputation Stadium Tour'] 포스터에 대한 예측 회원 점수는 66.6점
실제 회원 점수는 84.0점


영화 ['The Bad Guys: A Very Bad Holiday'] 포스터에 대한 예측 회원 점수는 66.9점
실제 회원 점수는 67.0점


영화 ['Annie'] 포스터에 대한 예측 회원 점수는 65.4점
실제 회원 점수는 65.0점
```

결론 및 앞으로 할 것

1. R2 점수가 왜 음수가 나오는지 원인을 찾고 이를 보완하여 다시 학습

2. 이미지 사이즈 조정, 이미지 컷 등 다양한 시도로 성능을 측정

   => 이미지 사이즈 224, 256 등으로 조정하여 학습해 볼 것, 64인 경우 정보 손실 확인

3. 배치사이즈, 학습률, 옵티마이저 등 하이퍼 파라미터 조정하면서 학습하여 성능 비교

4. 모델이 반환하는 피처맵을 통해 모델이 어떻게 데이터를 학습하는지 좀 더 자세한 분석이 필요

5. 줄거리로 예측한 모델의 예측 점수와 합산하여 평균 점수를 도출한 후 성능 비교

6. 회원 점수를 그룹화하여 분류 문제로 바꾸면 괜찮은 성능을 기대할 수 있음

CATEGORY 3

영화 줄거리로 로맨스 영화 분류

이화은

목차

1) 데이터 준비

2) 전처리

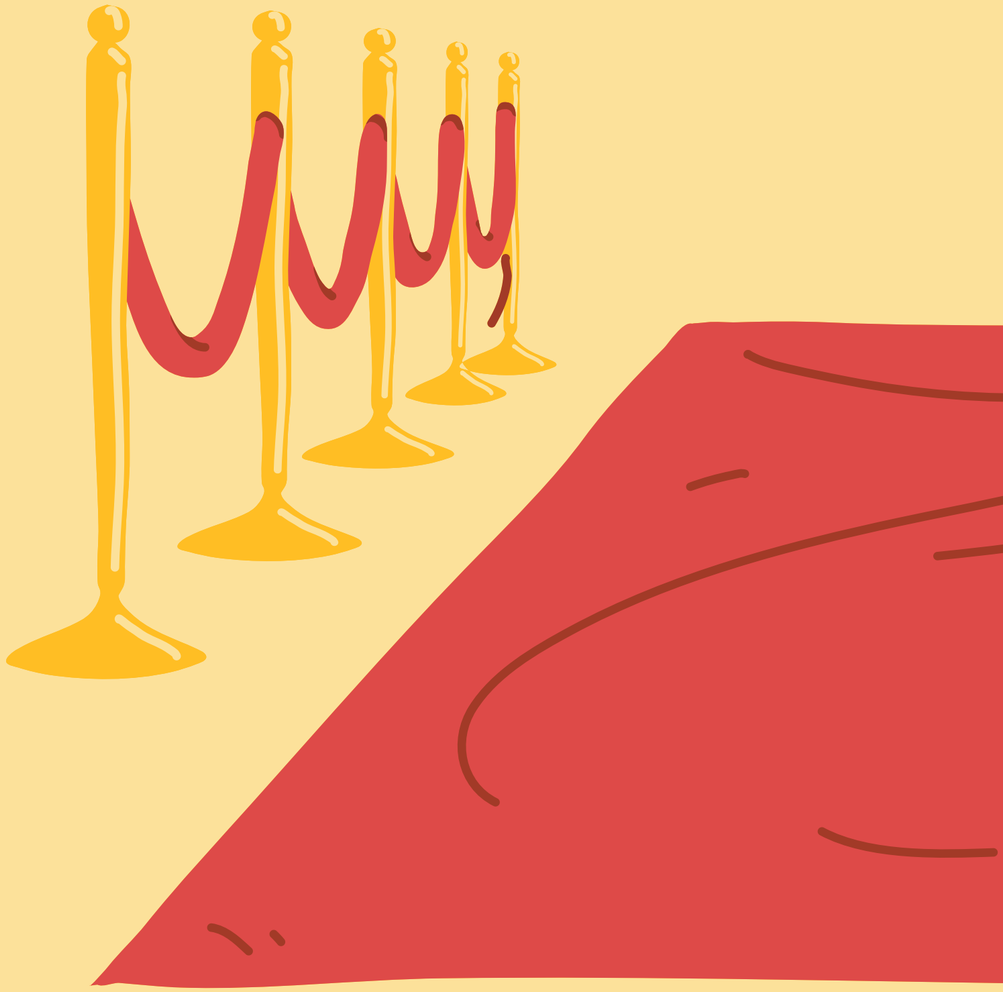
3) 모델 생성

LSTM

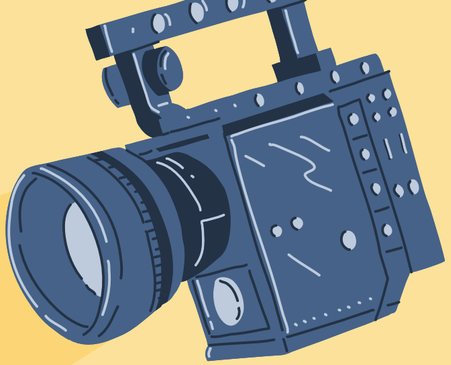
4) 학습

5) 예측

6) HTML



1) 데이터 준비



[1] 데이터 준비

```
# 모듈 불러오기
import pandas as pd
```

[107] ✓ 0.0s Python

```
# 파일 불러오기
movieDF = pd.read_csv('movie_data_all.csv', names=['source', 'title', 'genre', 'ru
#movieDF.head()
```

[108] ✓ 0.0s Python

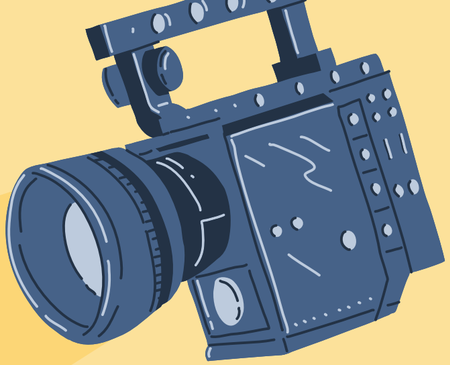
```
# 불필요한 컬럼 삭제
movieDF = movieDF[['genre', 'overview']]
```

[109] ✓ 0.0s Python

```
print(f'movieDF.shape = {movieDF.shape}]\n')
print(f'movieDF.info()\n{movieDF.info()}\n')
print(f'movieDF.head()\n{movieDF.head()}')
```

[110] ✓ 0.0s Python

2) 전처리



2-1) 기본 전처리

```
1  print('결측값 :\n', movieDF.isnull().sum(), '\n')
2  print('중복값 :', movieDF.duplicated().sum())
```

✓ 0.0s Python

```
결측값 :
 genre       30
overview     33
dtype: int64

중복값 : 31
```

```
1  # 결측값 & 중복값 삭제
2  movieDF.dropna(inplace=True)
3  movieDF.drop_duplicates(keep='first', inplace=True)
4
5  print('결측값 :\n', movieDF.isnull().sum(), '\n')
6  print('중복값 :', movieDF.duplicated().sum())
```

✓ 0.0s

```
결측값 :
 genre       0
overview     0
dtype: int64

중복값 : 0
```

결측값, 중복값 삭제

2) 전처리

```
genre
Drama                                                      465
Comedy                                                     306
Drama, Romance                                             276
Comedy, Romance                                            177
Horror                                                     167
                                                          ...
Comedy, Adventure, Family                                    1
Horror, Romance                                              1
Science Fiction, Animation, Action, Crime, Thriller          1
War, History, Action, Adventure, Drama, Romance              1
Science Fiction, Horror, Mystery, Thriller                   1
Name: count, Length: 2234, dtype: int64
```

```python
# 장르 안에 로맨스가 포함되어 있으면 로맨스로 1, 그 밖의 영화는 0으로 분류
def classify_romance_genre(genre) :
    if 'Romance' in genre :
        return 1
    else :
        return 0
```

```python
# 새 컬럼 추가
movieDF['genre_romance'] = movieDF['genre'].apply(classify_romance_genre)
```

```python
# 클래스 확인.
movieDF['genre_romance'].value_counts()
```

불균형 데이터

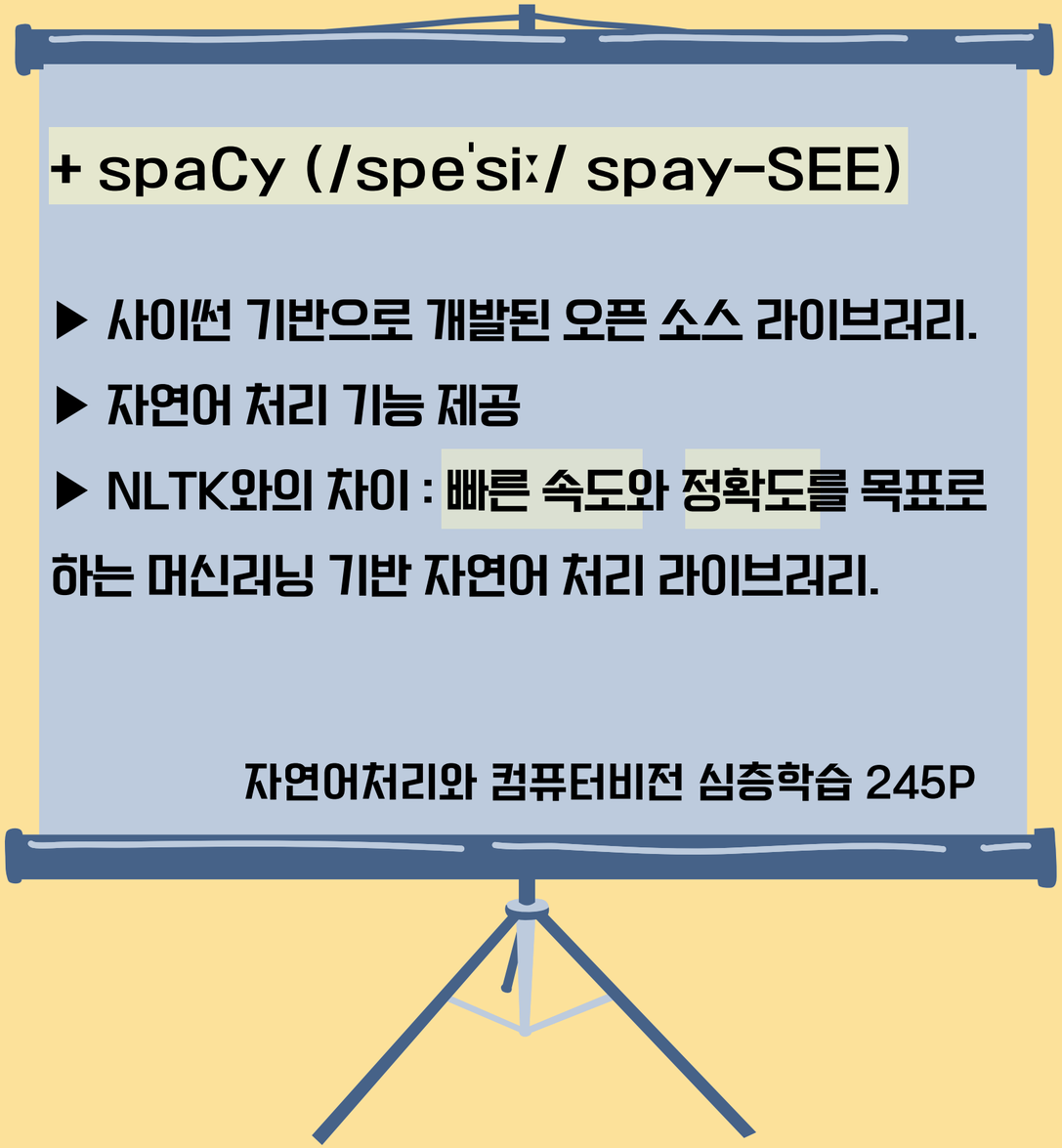
```
genre_romance
0    7737
1    1489
Name: count, dtype: int64
```

2) 전처리

```
# 피처, 레이블 분리
labelSR = movieDF['genre_romance']
featureDF = movieDF['overview'].to_frame()
print(type(labelSR), type(featureDF))
```

2-2) 텍스트 전처리

```
# spacy 설치
#!pip install spacy
#!pip show spacy

# 토큰화에 필요한 모듈 설치
#!python -m spacy download en_core_web_sm
```



+ spaCy (/speˈsiː/ spay-SEE)

▶ 사이썬 기반으로 개발된 오픈 소스 라이브러리.

▶ 자연어 처리 기능 제공

▶ NLTK와의 차이 : 빠른 속도와 정확도를 목표로 하는 머신러닝 기반 자연어 처리 라이브러리.

자연어처리와 컴퓨터비전 심층학습 245P

2) 전처리

```
1  import spacy
2  nlp = spacy.load('en_core_web_sm')
```

```
import re
from spacy.lang.en.stop_words import STOP_WORDS

tokenList = []
count = 0
for idx in range(featureDF.shape[0]):
    sentence = featureDF.iloc[idx][0]
    #print(f'[{idx}] :', sentence)
    doc = nlp(sentence)
    tokens = []
    for token in doc :
        #print(token.text)

        #영어와 숫자를 제외 삭제. 소문자로 통일.
        token = re.sub(r"[^a-z0-9]", "", token.text.lower())

        # 불용어 불포함, 공백 없음, 2글자 이상일 경우 tokenList에 추가
        if token not in STOP_WORDS and token not in "" and len(token)>=2 :
            tokens.append(token)
    tokenList.append(tokens)
```

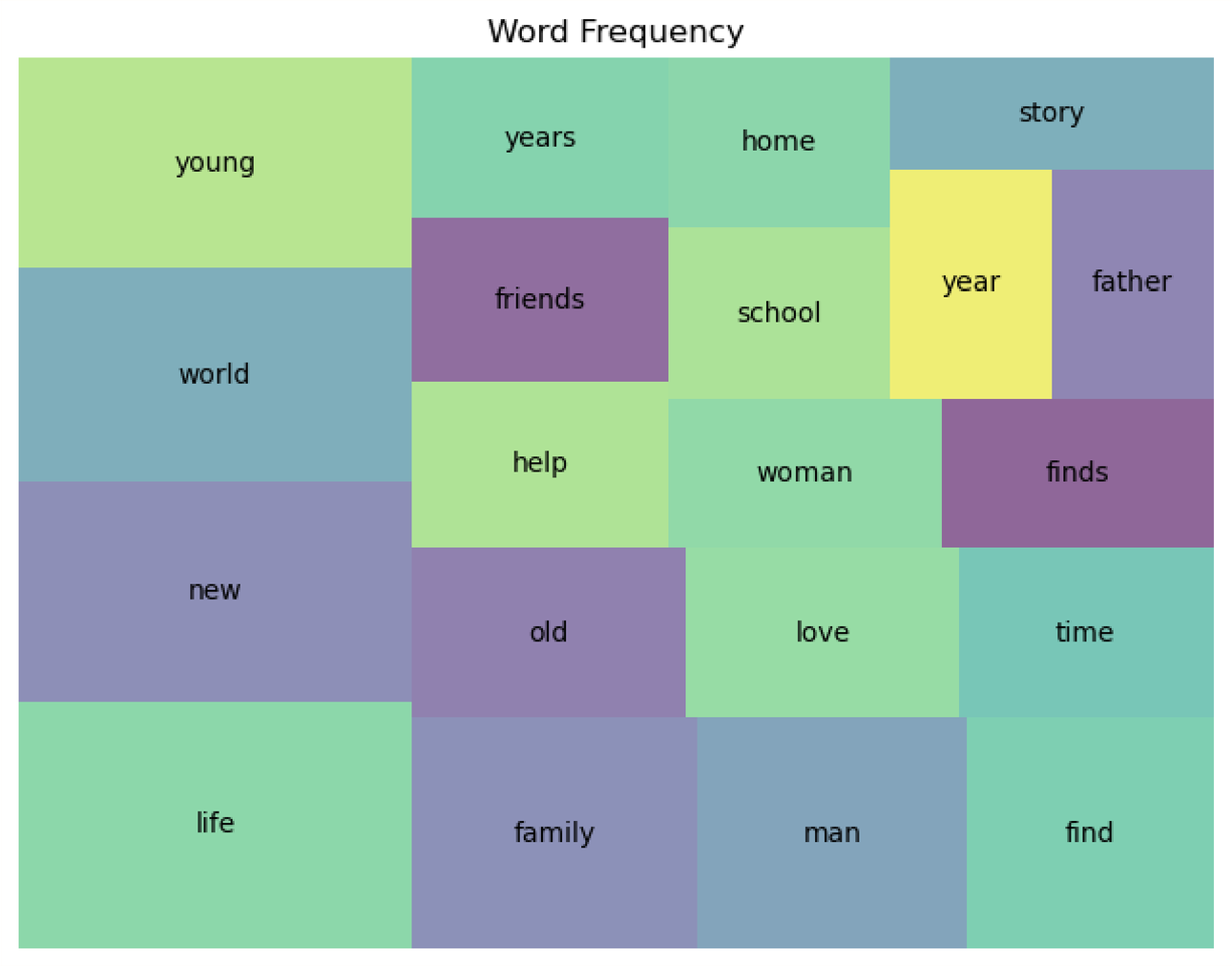
영어, 숫자 제외 삭제
소문자 통일
불용어 불포함, 공백 삭제
2글자 이상일 경우 tokenList에 추가

2) 전처리

```
from collections import Counter
word_counts = Counter()

'''
- Counter 객체는 리스트 요소의 값과 요소의 갯수를 카운트하여 저장.
- 카운터 객체는 .update 메소드로 계속 업데이트 가능.'''

# 각 행의 토큰 리스트를 가져와서 Counter 객체 업데이트
for tokens in featureDF['tokens']:
    word_counts.update(tokens)

# 업데이트된 Counter 객체를 출력하거나 필요한 작업 수행
print(word_counts)

sorted_word_counts = sorted(word_counts.items(), key=lambda x: x[1], reverse=True)
```

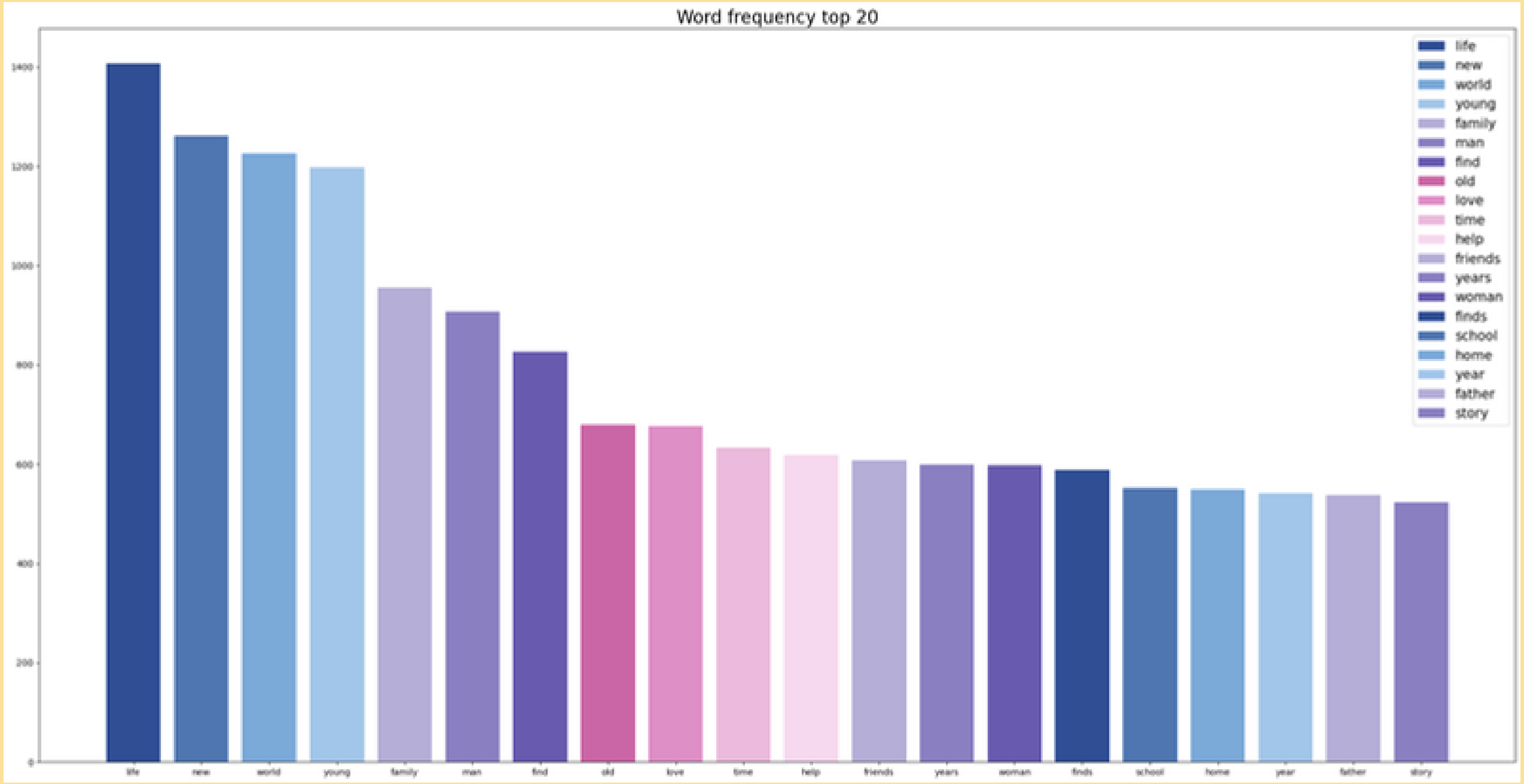
```
Counter({'life': 1407, 'new': 1261, 'world': 1226, 'young': 1197, 'family': 956, 'man': 907, 'find': 827, '
```

2) 전처리




```
import squarify
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
squarify.plot(sizes=counts, label=labels, alpha=0.6)
plt.axis('off')  # 축 제거
plt.title('Word Frequency top 20')
plt.show()
```

2) 전처리


Word frequency top 20
life
new
world
young
family
man
find
old
love
time
help
friends
years
woman
finds
school
home
year
father
story
0
200
400
600
800
1000
1200
1400

2) 전처리

어휘 사전 생성

```
wordList = ['<pad>', '<unk>']
for word in vocab :
    wordList.append(word)
```

인코딩

```
# 인코딩 : 문자 -> 숫자
encode = {token : idx for idx, token in enumerate(wordList)}
print(encode)
```

```
# overview 토큰을 정수로 변환
UNK_ID = encode.get('<unk>')
id = [[encode.get(token, UNK_ID) for token in text] for text in featureDF['tokens']]
print(id)
print(len(id))
```

패딩

```
cnt = 0
for encoded in id :
    #print(encoded)
    max_length = 0
    for i in id :
        if max_length<len(i) :
            max_length = len(i)
    padded_id = []
    for i in id :
        if len(i) < max_length :
            padded_id.append(i + [0]*(max_length-len(i)))
        else:
            padded_id.append(i)
```

3) 모델 생성

LSTM 모델 생성 – 교재 참고

```
from torch import nn as nn

class LSTM_with_dropout(nn.Module) :
    def __init__(self, n_vocab, hidden_dim, embedding_dim, n_layers, dropout, bidirectional=False) :
        super().__init__()

        self.embedding = nn.Embedding(num_embeddings = n_vocab,
                                      embedding_dim = embedding_dim,
                                      padding_idx = 0)

        self.model = nn.LSTM(input_size = embedding_dim,
                             hidden_size = hidden_dim,
                             num_layers = n_layers,
                             bidirectional=bidirectional,
                             dropout = dropout,
                             batch_first = True)
        self.classifier = nn.Linear(hidden_dim, 1)
        self.dropout = nn.Dropout(dropout)

    def forward(self, inputs) :
        embeddings = self.embedding(inputs)
        output, _ = self.model(embeddings)
        last_output = output[:, -1, :]
        logits = self.classifier(last_output)
        return logits
```

3) 모델 생성

train – test 데이터 분리

```
from sklearn.model_selection import train_test_split
Xtrn, Xtst, ytrn, ytst = train_test_split(dataDF, labelSR, test_size=0.2, random_state = 10, shuffle = True, stratify = labelSR)
```

numpy로 변환 후 데이터셋, 데이터 로더 생성

```
class myDataset(Dataset) :
    def __init__(self, feature, target) :
        super().__init__()
        self.feature = torch.IntTensor(feature)
        self.target = torch.FloatTensor(target)
        self.length = self.feature.shape[0]

    def __len__(self) :
        return self.length

    def __getitem__(self, index) :
        feature = self.feature[index]
        target = self.target[index]
        return feature, target
```

```
1  trainDS = myDataset(XtrnNP, ytrnNP)
2  testDS = myDataset(XtstNP, ytstNP)
```

✓ 0.0s

```
1  BATCH_SIZE = 32
2  trainDL = DataLoader(trainDS, BATCH_SIZE, shuffle = True)
3  testDL = DataLoader(testDS, BATCH_SIZE, shuffle = True)
```

✓ 0.0s

4) 학습

학습 + 테스트 함수

```python
def train(model, dataLoader, criterion, optimizer, device) :
    model.eval()
    lossList = []
    accuracyList = []

    for step, (input_ids, labels) in enumerate(dataLoader) :
        input_ids = input_ids.to(device)
        labels = labels.to(device).unsqueeze(1)

        logits = model(input_ids)
        loss = criterion(logits, labels)
        lossList.append(loss.item())
        ypre = torch.sigmoid(logits)>.5
        accuracyList.extend(torch.eq(ypre, labels).cpu().tolist())

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()


        print(f"Train Loss : {np.mean(lossList)}")
        print(f'Train accuracy : {np.mean(accuracyList)}\n')
        break
```

```python
def test(model, dataLoader, criterion, device) :
    model.eval()
    lossList = []
    accuracyList = []

    for step, (input_ids, labels) in enumerate(dataLoader) :

        input_ids = input_ids.to(device)
        labels = labels.to(device).unsqueeze(1)

        logits = model(input_ids)
        loss = criterion(logits, labels)
        lossList.append(loss.item())
        ypre = torch.sigmoid(logits)>.5
        accuracyList.extend(torch.eq(ypre, labels).cpu().tolist())

        print(f"Val Loss : {np.mean(lossList)}")
        print(f'Val accuracy: {np.mean(accuracyList)}\n')
        break
```

```python
CRITERION = nn.BCEWithLogitsLoss()
OPTIMIZER = optim.AdamW(mdl.parameters(), lr=LR)
```

```python
HIDDEN_SIZE = 32 # 64 -> 32
EMBEDD_DIM  = 64 # 128 -> 64
VOCAB_SIZE  = len(wordList)
NUM_LAYERS = 1
BATCH_SIZE  = 32
LR = 0.1
DROPOUT = 0.55
```

4) 학습

학습 결과

```
[998]
Train Loss : 0.4622824490070343
Train accuracy : 0.8125

Val Loss : 0.4548819065093994
Val accuracy: 0.84375

[999]
Train Loss : 0.418160080909729
Train accuracy : 0.84375

Val Loss : 0.480965256690979
Val accuracy: 0.8125

[1000]
Train Loss : 0.2758491337299347
Train accuracy : 0.9375

Val Loss : 0.2612428367137909
Val accuracy: 0.9375
```

EPOCH = 1000 일 때,

Train Loss : 0.2758491337299347

Train accuracy : 0.9375

Val Loss : 0.2612428367137909

Val accuracy : 0.9375

5) 예측

```python
def predict(model, text):
    with torch.no_grad():
        text = torch.tensor((text), dtype=torch.int64).to(device)
        text = text.unsqueeze(0)
        predicted_label = model(text)
        return predicted_label.argmax(1).item()
```

```python
def tokentext(text, vocaDF) :
    import spacy
    import re
    from spacy.lang.en.stop_words import STOP_WORDS
    nlp = spacy.load('en_core_web_sm')
    doc = nlp(text)
    textToken =[token.text for token in doc]

    voca_dict = {word: idx for idx, word in enumerate(vocaDF['0'].to_list())}
    encoded = [voca_dict[token] for token in textToken if token in voca_dict]

    # 인코딩 결과 출력
    print("토큰화된 단어들:", textToken)
    print("인코딩 결과:", encoded)

    padded_id=[]
    max_length = df.shape[1]

    if len(encoded) < max_length :
        padded_id.append(encoded + [0]*(max_length-len(encoded)))
    else :
        padded_id.append(encoded)
    return padded_id[0]
```

```python
#text = "Young Harry is in love and wants to marry an actress
#text ="When a crown prince visits, the princess's mother tri
text = "As an avid RPG player, half the fun of watching 'Solo
```

```python
1  result = predict(mdl, token)
2  if result == 0: print('other genre')
3  elif result == 1: print('romance movie')
```

✓ 0.0s

```
other genre
```

 # 원인 분석

=> 불균형 데이터로 인한 결과

결론 : 데이터 불균형을 무시하지 말자!

CATEGORY 4

영화 줄거리로 영화 평점 예측

양현우

목차

1) 전처리

2) 모델 생성

RNN

LSTM

3) 학습

4) 예측

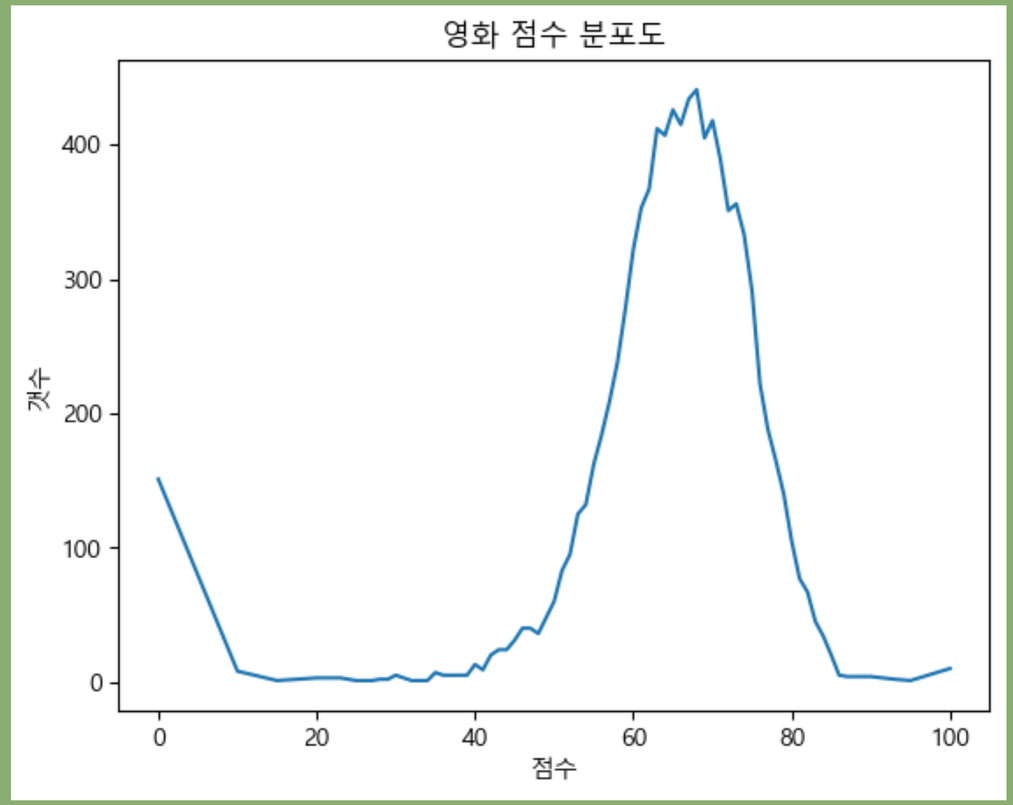
1) 데이터 수집 및 전처리

```
clear_movieDF=movieDF.dropna(subset=['
clear_movieDF.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
Index: 9267 entries, 0 to 9299
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   포스터     9250 non-null   object
 1   제목      9267 non-null   object
 2   장르      9241 non-null   object
 3   러닝타임    9174 non-null   object
 4   회원점수    9267 non-null   float64
 5   줄거리     9267 non-null   object
 6   제작비     5713 non-null   float64
 7   수익      860 non-null    float64
 8   감독      9267 non-null   object
dtypes: float64(3), object(6)
memory usage: 724.0+ KB
```

점수분포 시각화



1) 데이터 수집 및 전처리

```
# 구두점 및 특수문자 처리
clear_movieDF['줄거리']=clear_movieDF['줄거리'].str.replace('[^a-zA-Z0-9\s]','',regex=True)
clear_movieDF['줄거리'][0]
```

✓ 0.0s

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# NLTK의 불용어 다운로드
nltk.download('stopwords')
nltk.download('punkt')
```

```
tokens= []
token_count = []
for senten in list(featureSR.values):
    token = word_tokenize(senten)
    filtered_words = [word for word in token if word.lower() not in stop_words]
    tokens.append(filtered_words)
    token_count.append(len(filtered_words))
    # print(filtered_words,len(filtered_words))
```

✓ 1.5s

```
print(len(token_count))
```

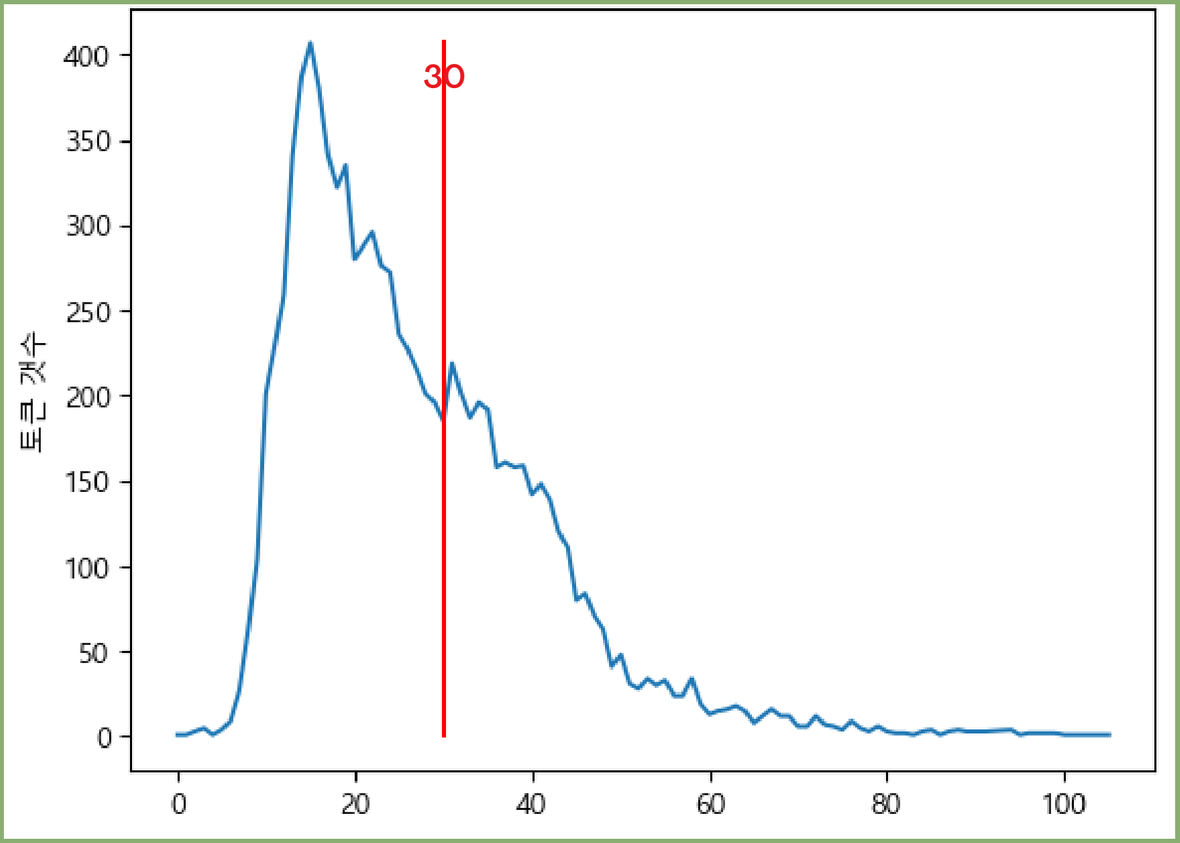
✓ 0.0s

```
9267
```

라이브러리 설정

특수문자 처리

1) 데이터 수집 및 전처리



토큰 갯수 시각화

```
def pad_sequences(sentences, maxlen, pad, start='R'):
    result = []
    for sen in sentences:
        sen = sen[:maxlen] if start == 'R' else sen[:maxlen*(-1)]
        padd_sen = sen + [pad] * (maxlen-len(sen)) if start == 'R'
        result.append(padd_sen)

    return result
```

✓ 0.0s

```
### 데이터 패딩 처리
PAD_ID = vocab1.get_stoi().get('<PAD>')
MAX_LENGTH = 30

data_ids = pad_sequences(data_ids, MAX_LENGTH, PAD_ID)
```

2) 모델 생성

```
class ScoreRegression(nn.Module):
    def __init__(self,
                 n_vocab,
                 hidden_dim,
                 embedding_dim,
                 n_layers,
                 dropout=0.5,
                 bidirectional=True,
                 model_type='lstm'):
        super(ScoreRegression, self).__init__()

        self.embedding = nn.Embedding(
            num_embeddings=n_vocab,
            embedding_dim=embedding_dim,
            padding_idx=0)

        if model_type == 'rnn':
            self.model = nn.RNN(
                input_size=embedding_dim,
                hidden_size=hidden_dim,
                num_layers=n_layers,
                bidirectional=bidirectional,
                dropout=dropout,
                batch_first=True,
            )
```

```
    elif model_type == 'lstm':
        self.model = nn.LSTM(
            input_size=embedding_dim,
            hidden_size=hidden_dim,
            num_layers=n_layers,
            bidirectional=bidirectional,
            dropout=dropout,
            batch_first=True,
        )
    if bidirectional:
        self.regression = nn.Linear(hidden_dim*2, 1)
    else:
        self.regression = nn.Linear(hidden_dim,1)
    self.dropout = nn.Dropout(dropout)

def forward(self, inputs):
    embeddings = self.embedding(inputs)
    output, _ = self.model(embeddings)
    last_output = output[:, -1, :]
    last_output = self.dropout(last_output)
    logits = self.regression(last_output)

    return logits
```

2) 모델 생성

RNN, LSTM 모델 생성

```
from torch import optim

n_vocab = len(token_to_id)
hidden_dim = 64
embedding_dim = 128
n_layers = 2
interval = 100
```

```
# rnn 모델
regression_rnn = ScoreRegression(
    n_vocab=n_vocab, hidden_dim=hidden_dim, embedding_dim=embedding_dim,mod
).to(device)

criterion = nn.MSELoss().to(device)
optimizer_rnn = optim.Adam(regression_rnn.parameters(), lr=0.001)
scheduler_rnn = optim.lr_scheduler.StepLR(optimizer_rnn, 1.0, gamma=0.1)
```

0.0s

```
# lstm 모델
regression_lstm = ScoreRegression(
    n_vocab=n_vocab, hidden_dim=hidden_dim, embedding_dim=embedding_dim ,n_
).to(device)

optimizer_lstm = optim.Adam(regression_lstm.parameters(), lr=0.001)
scheduler_lstm = optim.lr_scheduler.StepLR(optimizer_lstm, 1.0, gamma=0.1)
```

3) 학습

```
0/232-> Train Loss : 4371.741211, Train MSE: 4371.741211
                        Train r2_score : -62.209169
100/232-> Train Loss : 3118.259200, Train MSE: 3118.259521
                        Train r2_score : -34.875387
200/232-> Train Loss : 2496.829590, Train MSE: 2496.829590
                        Train r2_score : -27.099530
[epoch:1] Val Loss : 1196.718194, Val MSE: 1196.718262
                        Val r2_score : -11.223616

0/232-> Train Loss : 1341.306396, Train MSE: 1341.306396
                        Train r2_score : -20.949274
100/232-> Train Loss : 980.555355, Train MSE: 980.555298
                        Train r2_score : -10.149719
200/232-> Train Loss : 810.636886, Train MSE: 810.636841
                        Train r2_score : -7.716984
[epoch:2] Val Loss : 434.902993, Val MSE: 434.903015
                        Val r2_score : -3.085443

0/232-> Train Loss : 463.466003, Train MSE: 463.466034
                        Train r2_score : -1.807024
100/232-> Train Loss : 445.224614, Train MSE: 445.224640
                        Train r2_score : -3.241494
200/232-> Train Loss : 374.025242, Train MSE: 374.025238
                        Train r2_score : -2.634956
[epoch:3] Val Loss : 217.728527, Val MSE: 217.728531
...
                        Train r2_score : 0.661533
[epoch:1000] Val Loss : 228.770871, Val MSE: 228.770859
                        Val r2_score : -0.929122
```

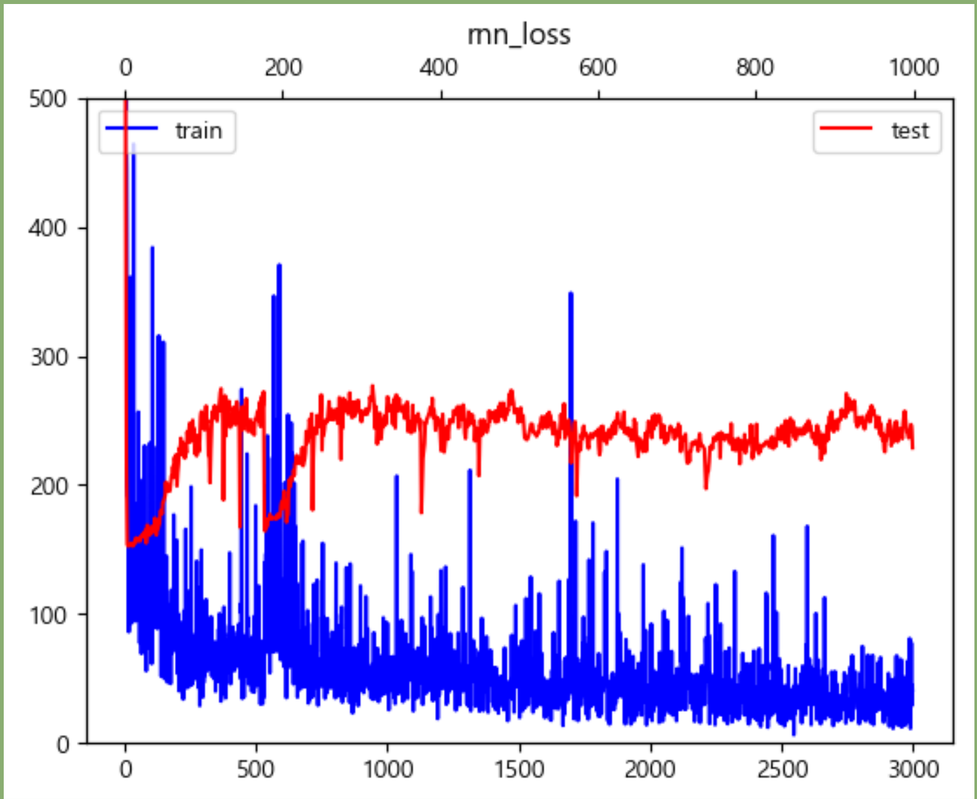
RNN – epoch1000

```
0/232-> Train Loss : 4737.291992, Train MSE: 4737.291992
                        Train r2_score : -94.710425
100/232-> Train Loss : 3341.301765, Train MSE: 3341.302246
                        Train r2_score : -40.610268
200/232-> Train Loss : 2614.885725, Train MSE: 2614.885986
                        Train r2_score : -28.966577
[epoch:1] Val Loss : 1238.095598, Val MSE: 1238.095581
                        Val r2_score : -11.838214

0/232-> Train Loss : 1155.207764, Train MSE: 1155.207764
                        Train r2_score : -12.830295
100/232-> Train Loss : 983.988441, Train MSE: 983.988464
                        Train r2_score : -10.077665
200/232-> Train Loss : 798.370419, Train MSE: 798.370422
                        Train r2_score : -7.542659
[epoch:2] Val Loss : 402.949345, Val MSE: 402.949341
                        Val r2_score : -2.793379

0/232-> Train Loss : 559.481079, Train MSE: 559.481079
                        Train r2_score : -1.577697
100/232-> Train Loss : 344.329632, Train MSE: 344.329620
                        Train r2_score : -2.634281
200/232-> Train Loss : 298.249903, Train MSE: 298.249878
                        Train r2_score : -1.940613
[epoch:3] Val Loss : 191.312294, Val MSE: 191.312271
...
                        Train r2_score : 0.963549
[epoch:1000] Val Loss : 164.191024, Val MSE: 164.191025
                        Val r2_score : -0.370059
```

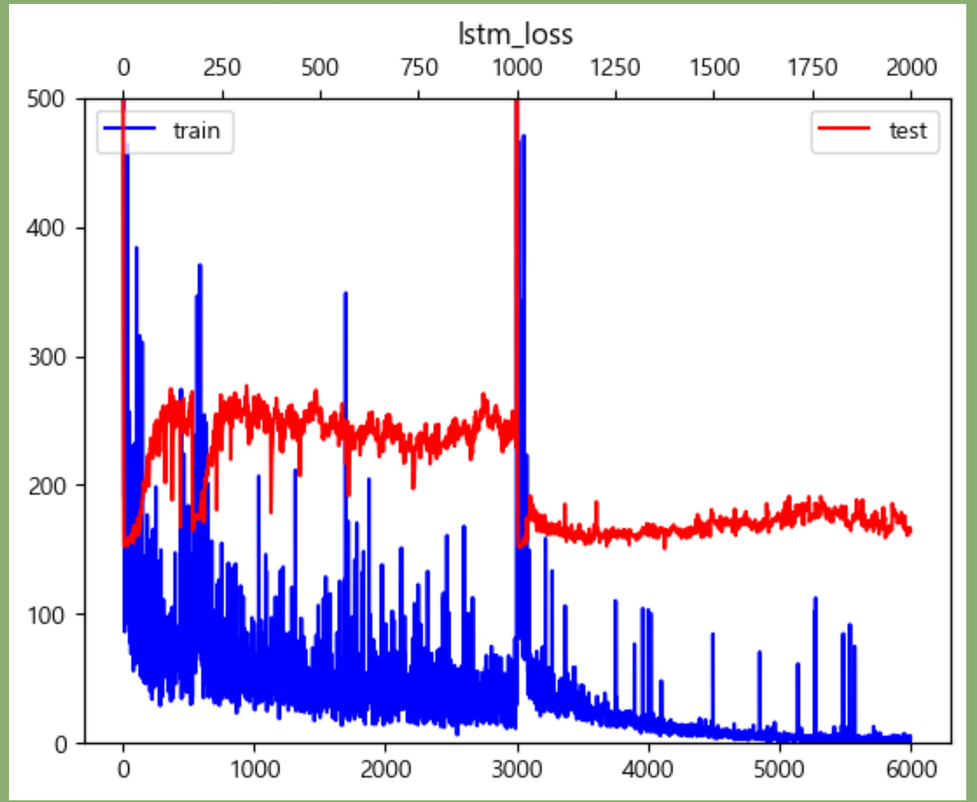
LSTM – epoch1000

3) 학습결과 – LOSS

 # RNN – epoch1000

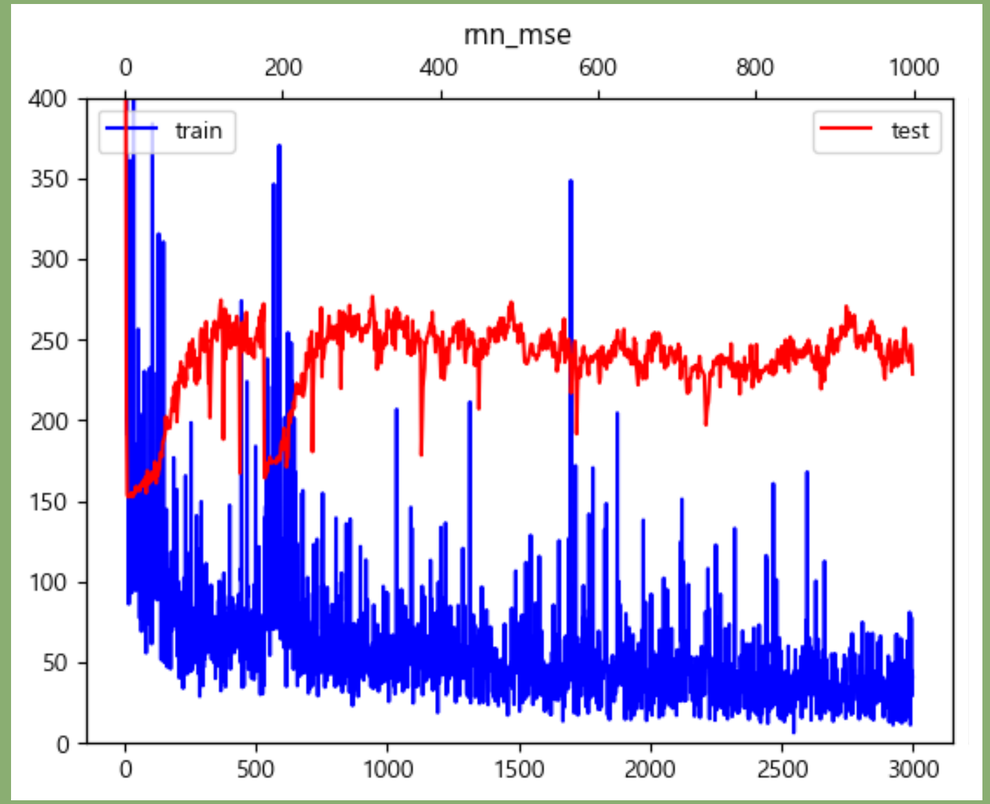


 # LSTM – epoch1000

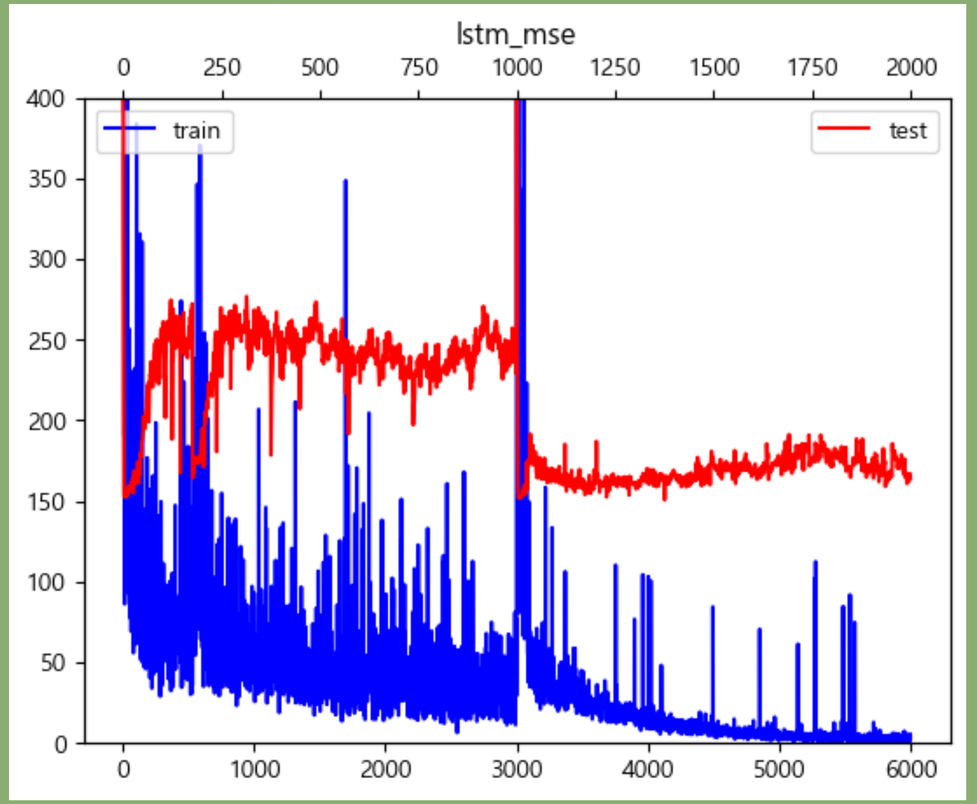


3) 학습결과-MSE

RNN – epoch1000

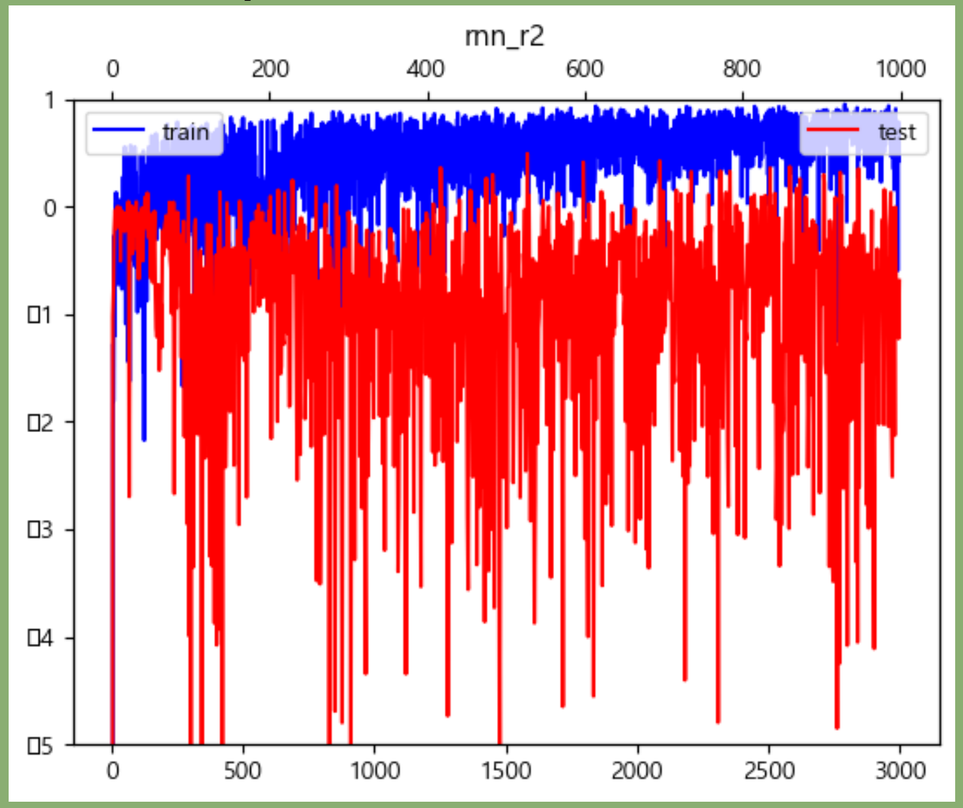


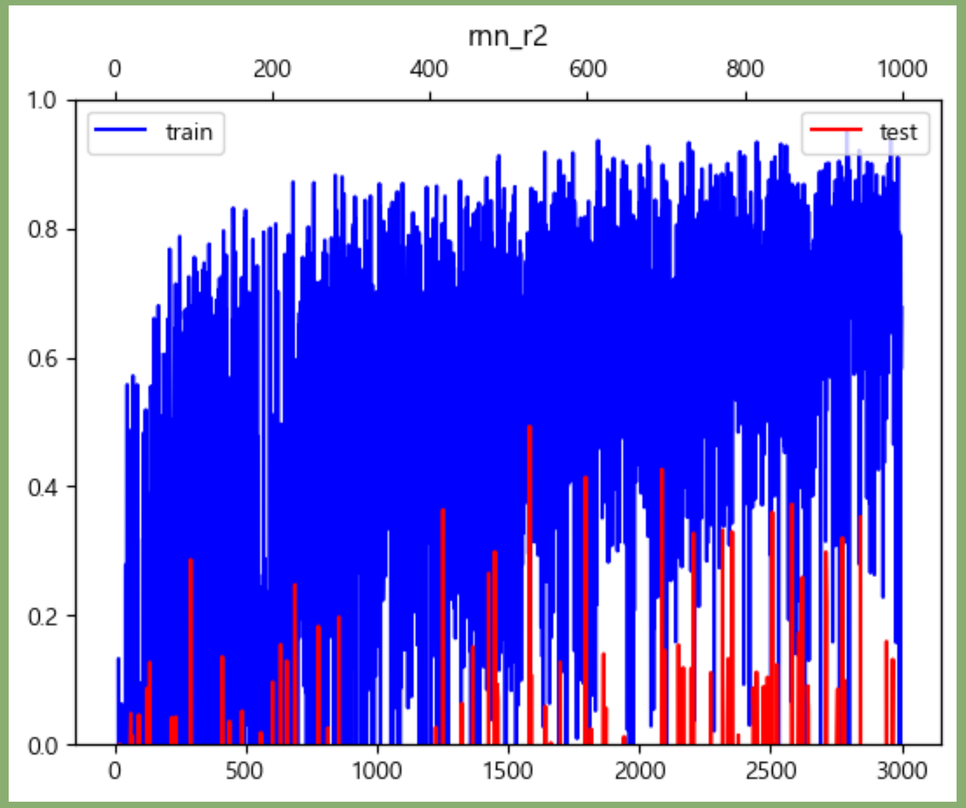
LSTM – epoch1000



3) 학습결과-R2_SCORE

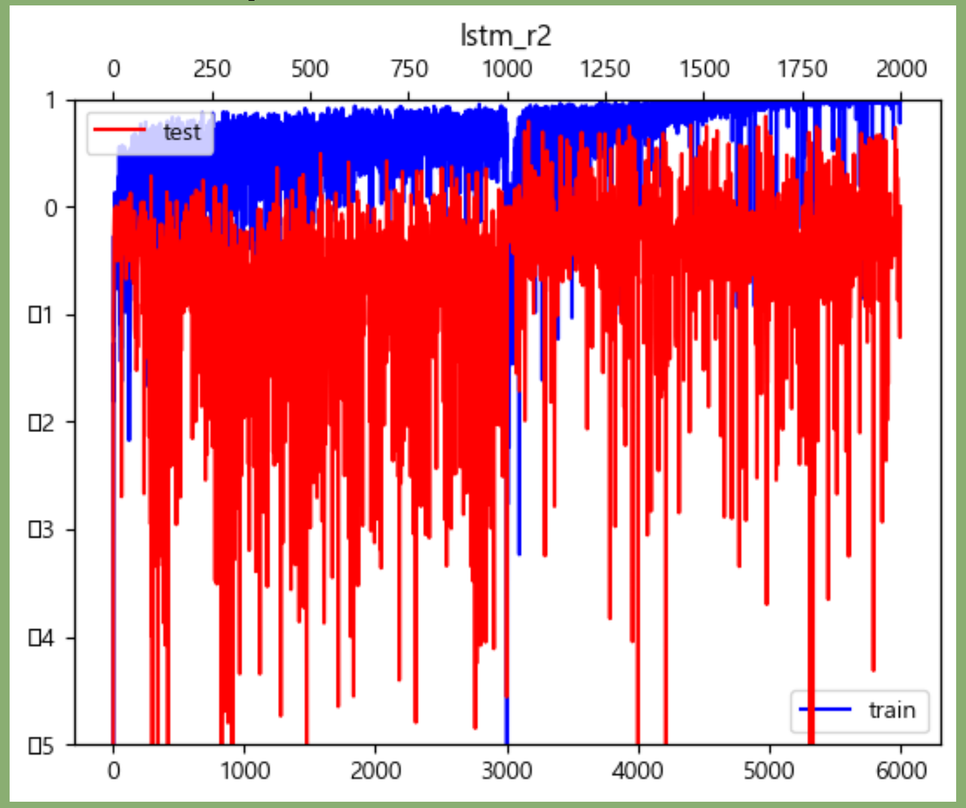
 # RNN – epoch1000

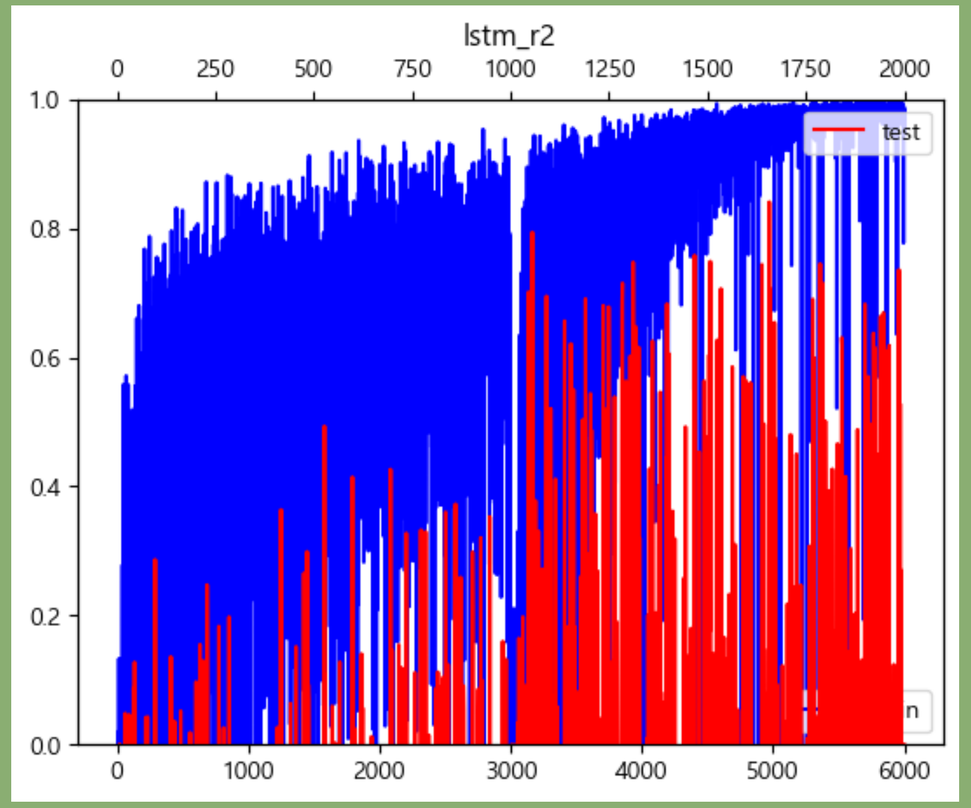




3) 학습결과-R2_SCORE

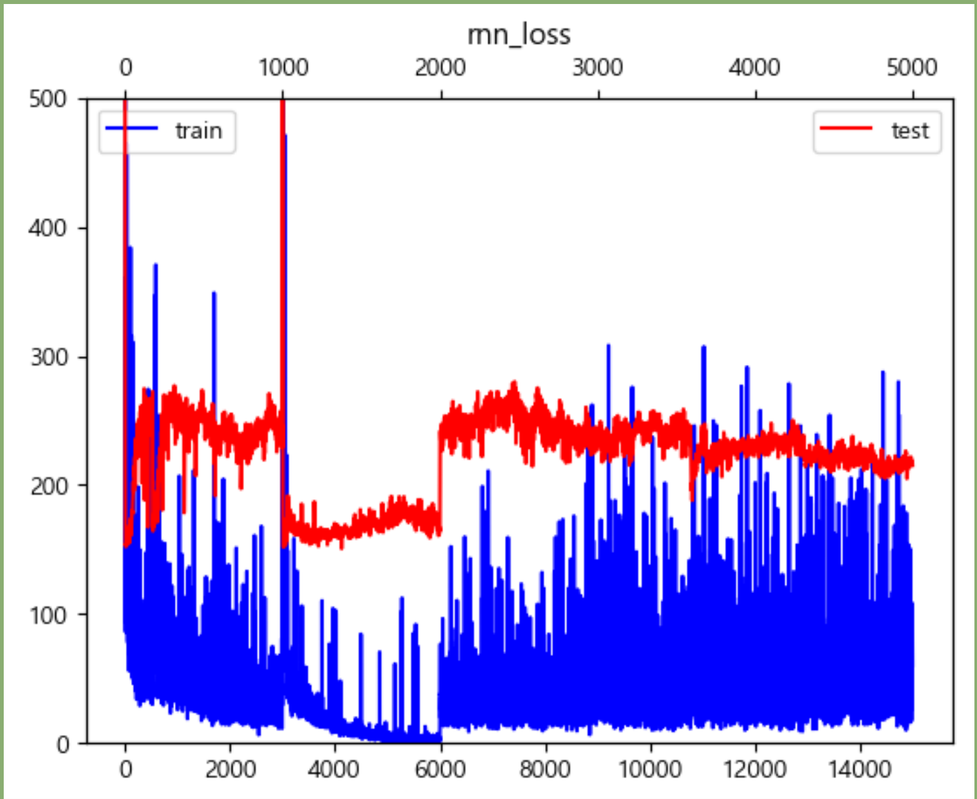
 # lstm - epoch1000



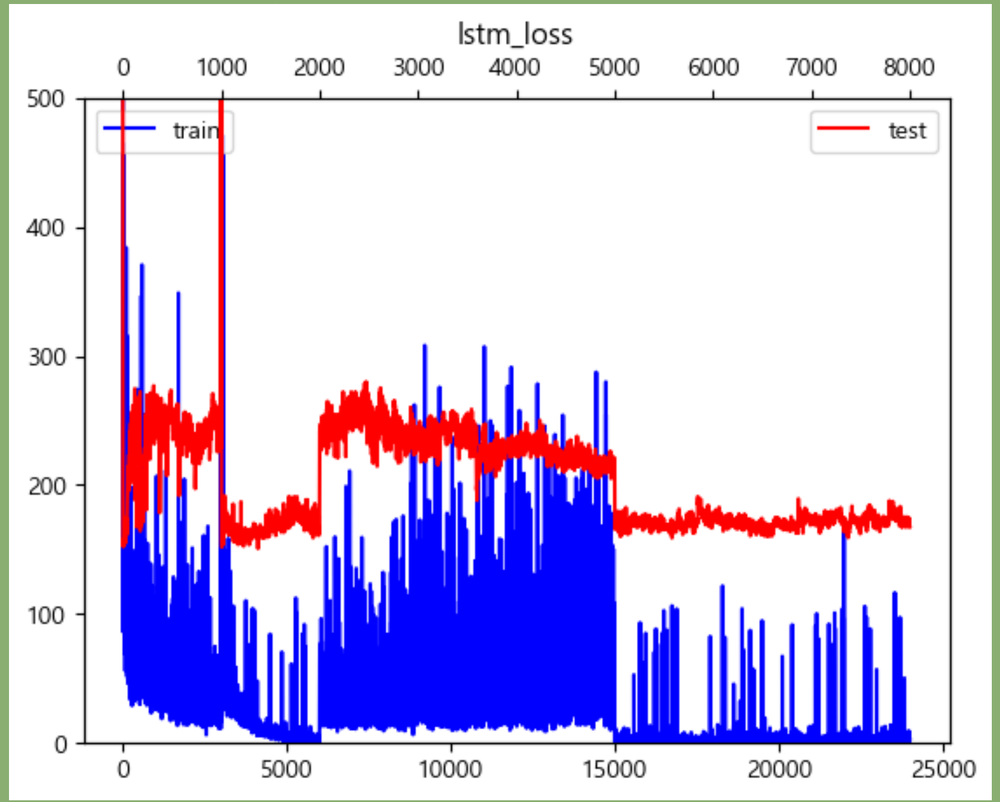


3) 학습결과 – LOSS

 # RNN – epoch4000

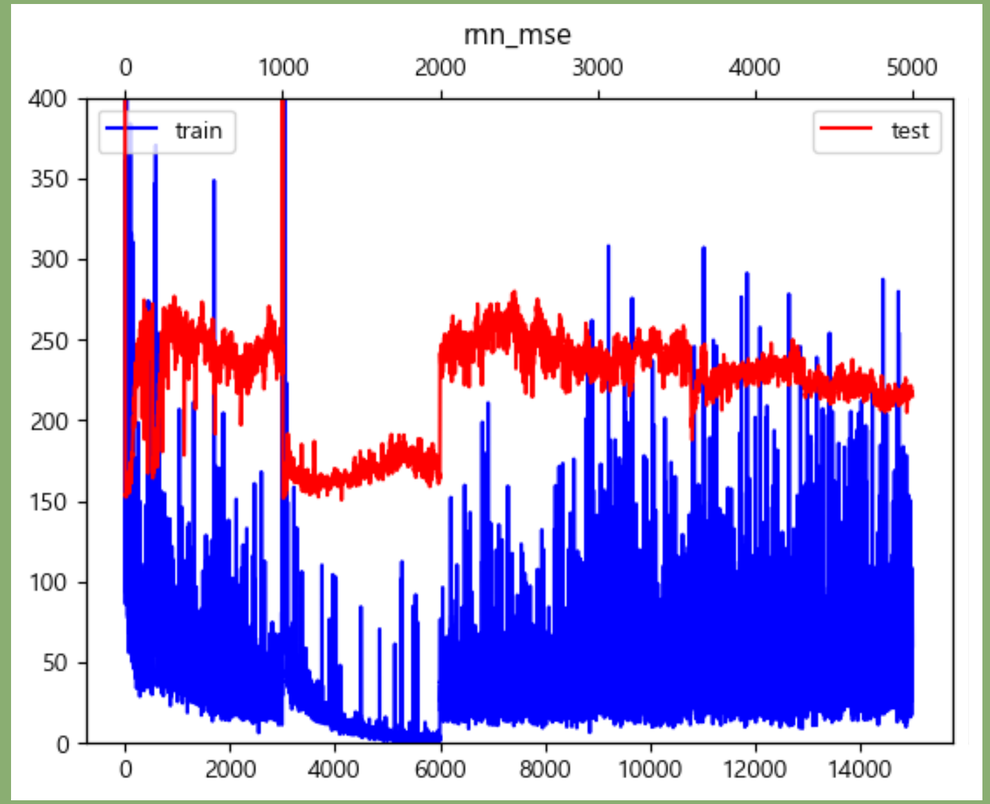


 # LSTM – epoch4000

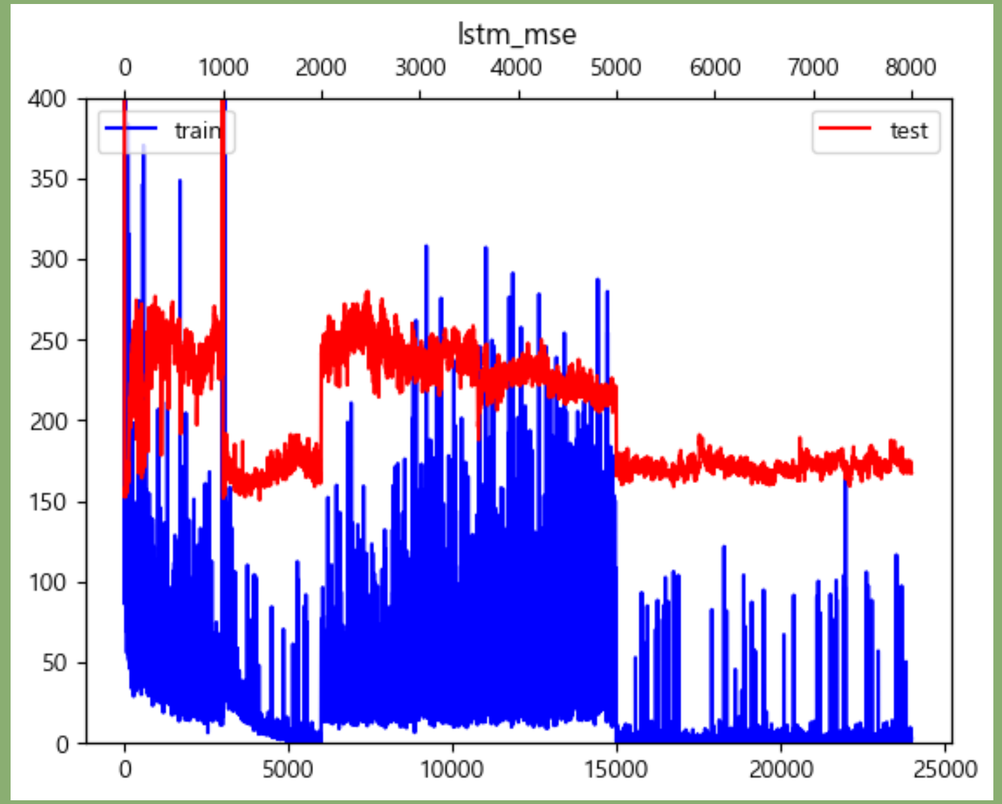


3) 학습결과-MSE

RNN – epoch4000

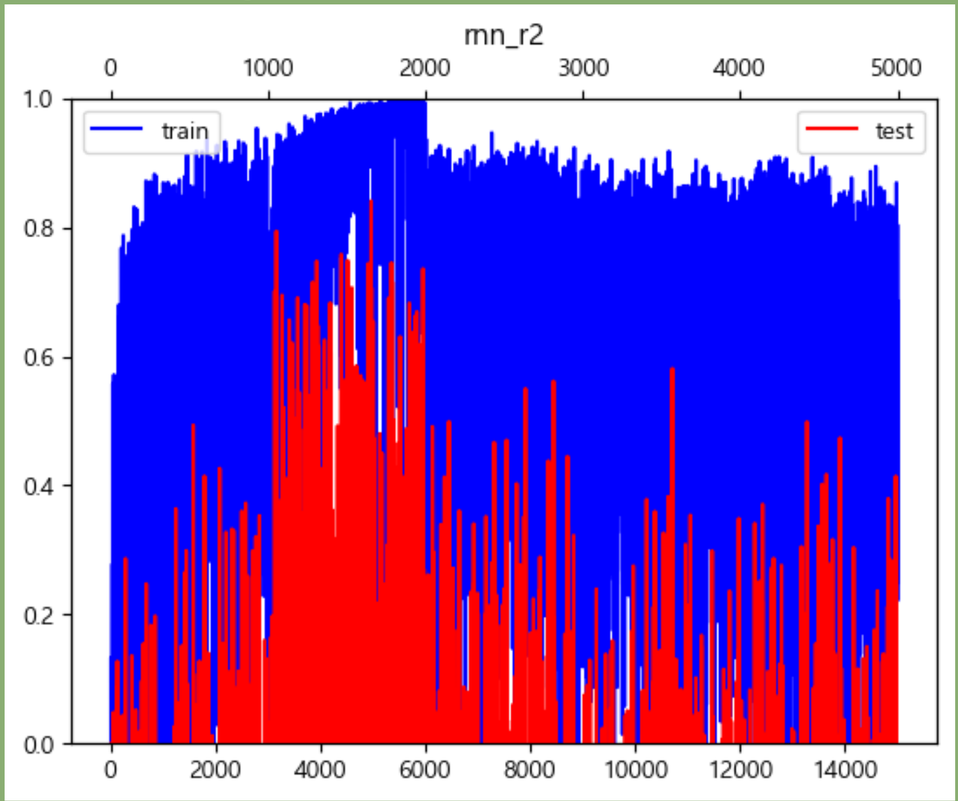


LSTM – epoch4000

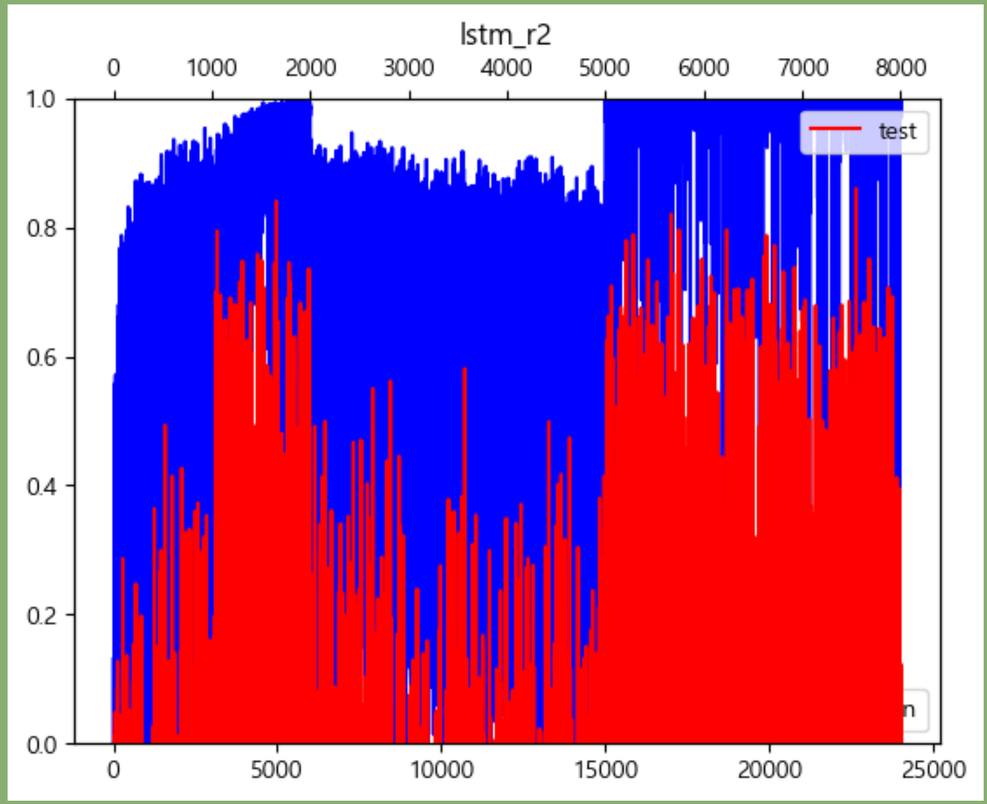


3) 학습결과-R2_SCORE

RNN – epoch4000



LSTM – epoch4000



4) 예측

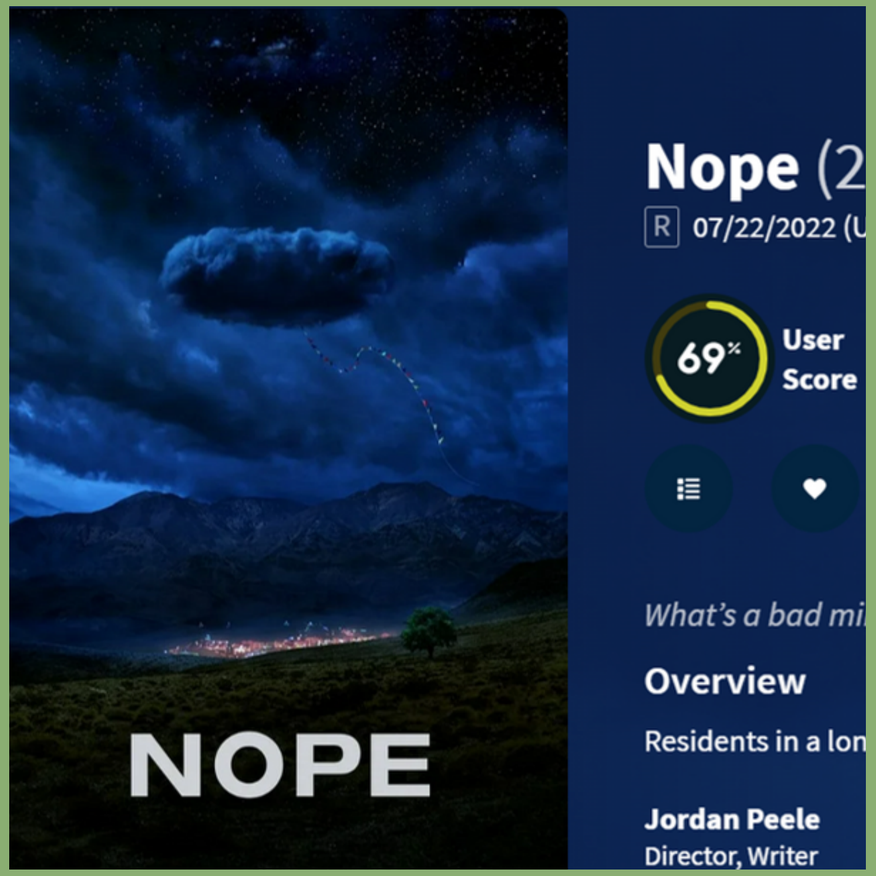

```
text2="Forced to confront revelations about her pas
predict(ScoreRegression_rnn1,text2,text_pipeline)
predict(ScoreRegression_lstm1,text2,text_pipeline)
predict(ScoreRegression_rnn2,text2,text_pipeline)
predict(ScoreRegression_lstm2,text2,text_pipeline)
```

✓ 0.0s

```
58.87356948852539
60.241355895996094
60.9906005859375
61.69707489013672
```

rnn에 조금 더 가까움

4) 예측




```
text = " Following their father's shocking death,
predict(ScoreRegression_rnn1,text,text_pipeline)
predict(ScoreRegression_lstm1,text,text_pipeline)
predict(ScoreRegression_rnn2,text,text_pipeline)
predict(ScoreRegression_lstm2,text,text_pipeline)
```

✓ 0.0s

```
74.47527313232422
62.812408447265625
73.87064361572266
63.54216003417969
```

두 모델의 평균(68)에 가깝다

결론 : 영화의 평점은 줄거리로는 알 수 없다.

하지만, 어느정도 먹힘은 알 수 있다.

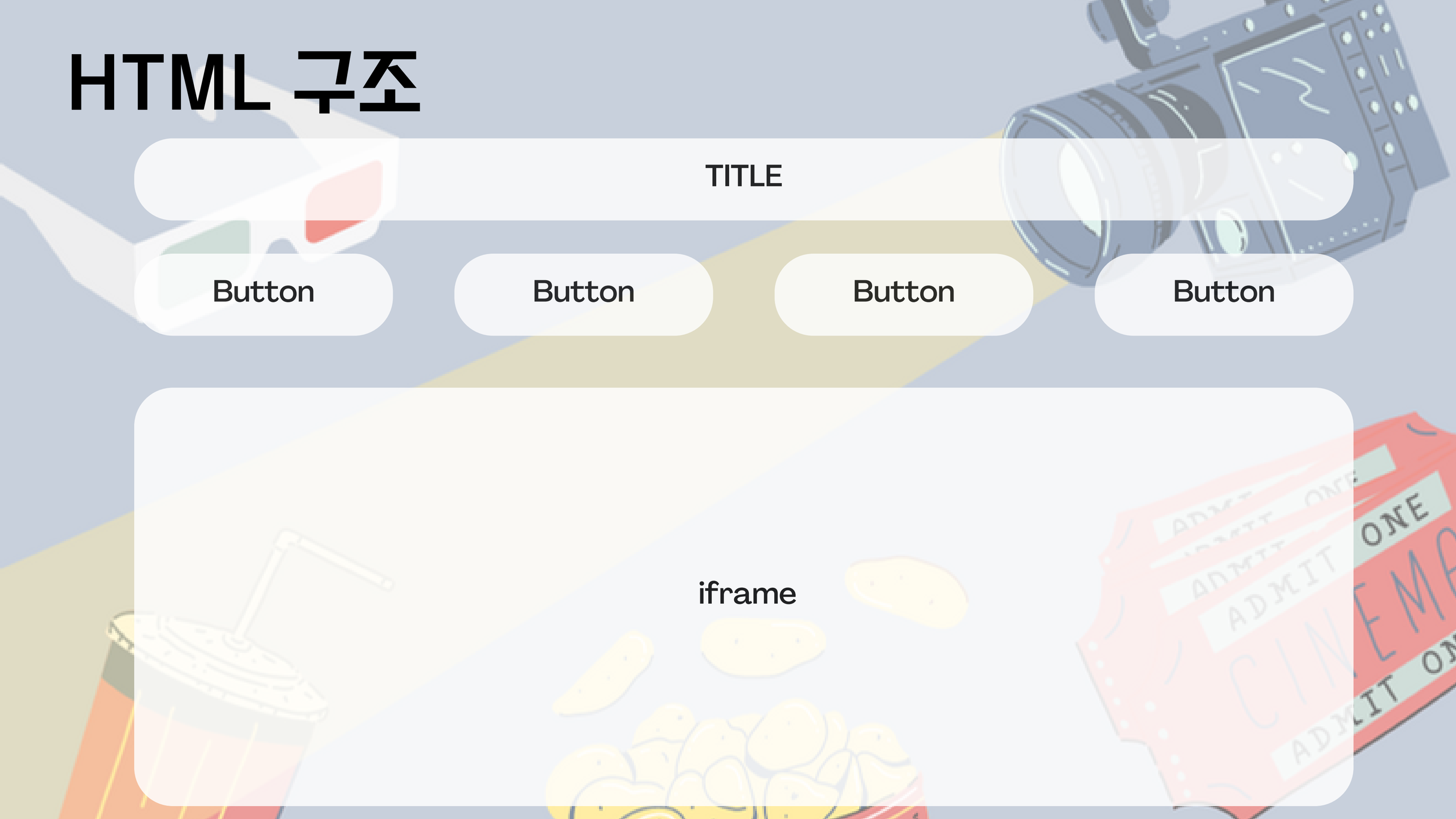
WEB

HTML 구조

TITLE

Button Button Button Button

iframe



.py

인스턴스 생성

```python
# 모듈 로딩
import cgi, sys, codecs, cgitb
import torch
import pandas as pd
from torch import nn as nn


# 웹페이지의 form 태그 내의 input 태그 입력값 가져와서
# 저장하고 있는 인스턴스
form = cgi.FieldStorage()
```

모델 및 예측에 필요한 함수 로딩

```python
class LSTM_with_dropout(nn.Module) :…

mdl = torch.load('lstm.pth')
df = pd.read_csv('./encoded.csv')
vocabDF = pd.read_csv('./vocab.csv')

# 토큰화 함수 생성
def tokentext(text, df, vocaDF) :…

def predict(model, text):…

def print_browser(result=""):…
```

index.html

```html
<body>
    <center>
        <br>
        <h1 style="font-size: 50px;">
            KDT 시네마
        </h1>

        <!-- 희진 -->
        <button type="button" class="learn-more" onclick="changeIframeSrc('hj.html')">희진의 포스터로 장르 분류 모델</button>
        <!-- 동현-->
        <button type="button" class="learn-more" onclick="changeIframeSrc('dh.html')">동현의 포스터로 평점 예측 모델</button>
        <!-- 화은 -->
        <button type="button" class="learn-more" onclick="changeIframeSrc('he.html')">화은의 줄거리로 장르 분류 모델</button>
        <!-- 현우 -->
        <button type="button" class="learn-more" onclick="changeIframeSrc('hw.html')">현우의 줄거리로 평점 예측 모델</button>
        <br>
        <br>
        <br>
        <iframe id="myIframe" src="./image/model_bg.png" width="1200" height="672"></iframe>
    </center>
    <script>
        function changeIframeSrc(src) {
            var iframe = document.getElementById('myIframe');
            iframe.onload = function() {
                // iframe이 로드되면 아무 작업도 하지 않습니다.
            };
            iframe.onerror = function() {
                // iframe 로드가 실패하면 기본 이미지를 표시합니다.
                iframe.src = '../image/model_bg.png'; // 기본 이미지 파일 경로
            };
            iframe.src = src;
        }
```

아이프레임의 소스를 변경

.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>줄거리로 영화 장르 분류하기</title>
    <link href ='../css/my_style.css' rel = "stylesheet">
</head>
<body>
    <center>
    <h2>💔 영화 줄거리로 로맨스 영화 분류하기 💔</h2></center>

    <form action ="/cgi-bin/predgenre.py" method="get">
        <fieldset>
            <center>
                <label><h3> 줄거리 입력 </h3></label>
            <p>
            <textarea id = "plot" type = "text" name="plot" placeholder = "전달메시지"></textarea><br>
            <input type = "submit" value=" " class="submit-button"> <!--파일 제출-->
            </p>
        </fieldset>
        </center>
    </form>
</body>
</html>
```

.CSS

여백, 글자 색 및 배경 이미지 설정

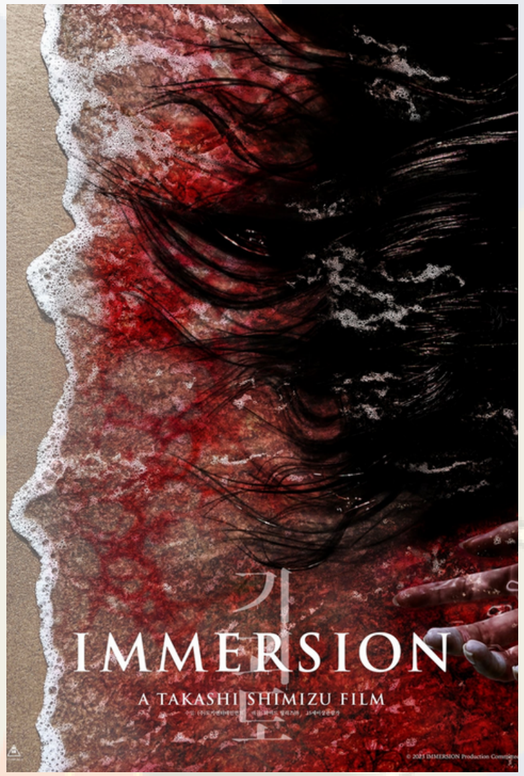
```css
h2 {
    /* 헤더의 위쪽 여백 설정하여 아래로 이동 */
    margin-top: 300px; /* 원하는 만큼의 여백 크기 지정 (예: 50px) */
    color: aliceblue;
}

h3{color: white}

body {
    /* 배경 이미지 설정 */
    background-image: url('bg_2.png'); /* 이미지 파일 경로 */
    background-size: cover;
    background-repeat: no-repeat;}
```

제출 버튼, fieldset에 스타일 적용

```css
/* 제출 버튼에 스타일 적용 */
.submit-button {
    /* 버튼의 크기와 여백 설정 */
    width: 70px; /* 버튼 너비 */
    height: 70px; /* 버튼 높이 */
    padding: 20px; /* 내부 여백 */
    border: none; /* 테두리 없음 */
    cursor: pointer; /* 마우스 호버 시 커서 모양 변경 */
    background-image: url('popcorn.png'); /* 이미지 파일 경로 */
    background-size: cover; /* 이미지를 버튼에 꽉 차게 조정 */
    background-position: center; /* 이미지를 가운데 정렬 */
    background-repeat: no-repeat; /* 이미지 반복 없음 */
    /* 원하는 배경 색상 설정 (이미지가 로드되지 않았거나 버튼이 채워지지 않
    background-color: transparent; /* 배경 색상 투명으로 설정 */
}

fieldset {
    max-width: 600px;
    /* 투명한 검은색 배경 (RGBA 값의 마지막 값이 투명도) */
    background-color: rgba(255, 255, 255, 0.5);
    /* 필요에 따라 다른 스타일링 속성 추가 */
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 3px;
    margin: 0 auto;
}
```

Predict Data 미개봉작들

Horror



Animation

Romance

Action

Predict Data

드라마 # 로맨스 장르 영화 overview

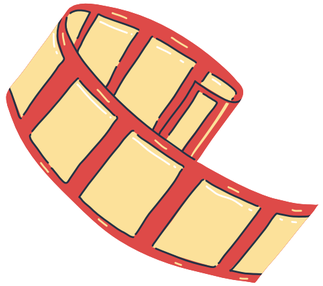
Nora and Hae Sung, two deeply connected childhood friends, are wrest apart after Nora's family emigrates from South Korea. Two decades later, they are reunited in New York for one fateful week as they confront notions of destiny, love, and the choices that make a life, in this heartrending modern romance.

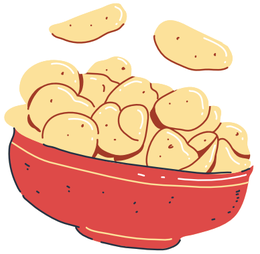
호러 # 스릴러 # 미스터리 장르 영화 overview

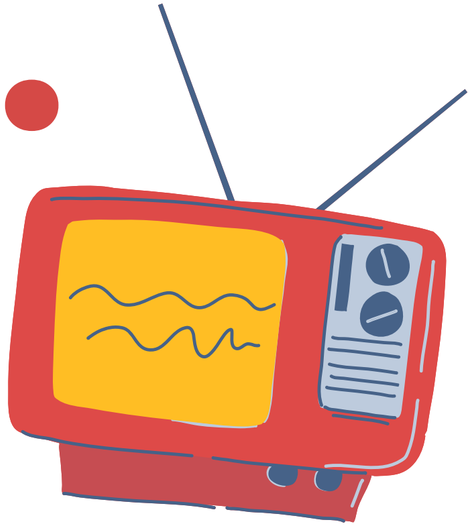
When a renowned shaman (KIM Go-Eun) and her protégé (Lee Do-hyun) are hired by a wealthy, enigmatic family, they begin investigating the cause of a disturbing supernatural illness that affects only the first-born children of each generation. With the help of a knowledgeable mortician (YOO Hai-jin) and the country's most revered geomancer (CHOI Min-sik), they soon trace the affliction's origin to a long-hidden family grave located on sacred ground.

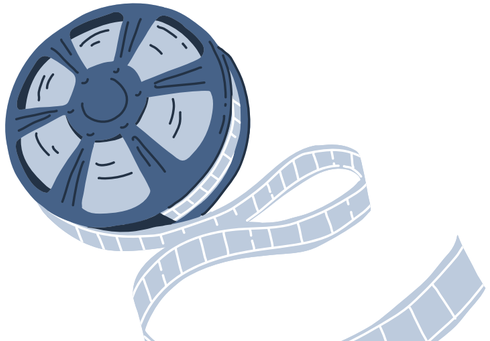
감사합니다.

ANY QUESION?

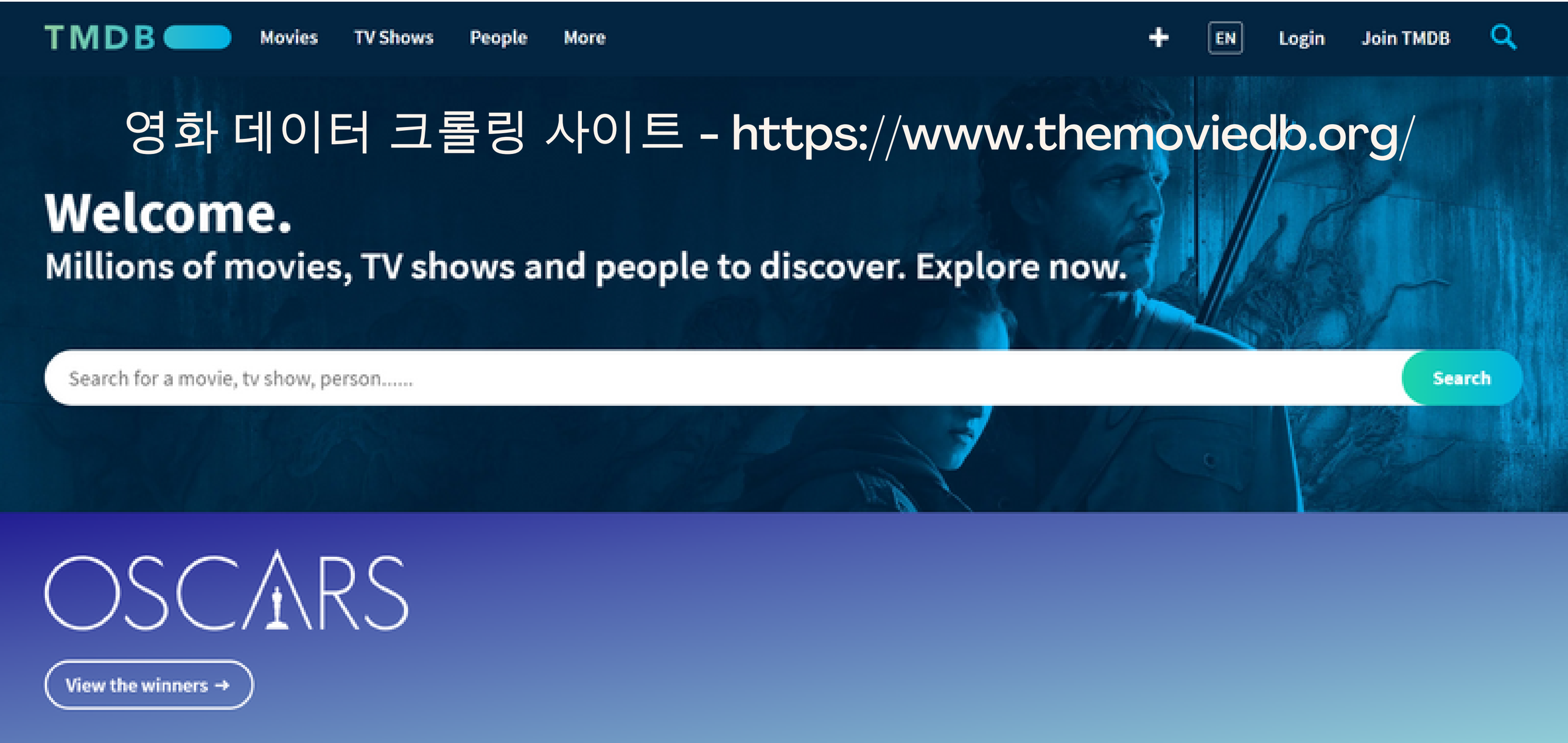








REFERENCE


TMDB
Movies
TV Shows
People
More
EN
Login
Join TMDB
영화 데이터 크롤링 사이트 - https://www.themoviedb.org/
Welcome.
Millions of movies, TV shows and people to discover. Explore now.
Search for a movie, tv show, person.....
Search
OSCARS
View the winners →