기계학습 기말대체과제 보고서

- 제품 review 데이터의 감성 분석 -

과목명	기계학습(8585)
전공	소프트웨어전공
학번	2020925
이름	서조은
제출일	2024.06.20

목차

1. 감성 분석의 기본 이해 3

2. 데이터 전처리 4

2-1 데이터 정리

2-2 텍스트 토큰화 및 불용어 제거

2-3 텍스트 정규화

3. 모델 선택 및 학습 10

4. 모델 평가 16

4-1 지표를 사용한 모델 평가

4-2 오버피팅과 언더피팅

5. 시각화 및 해석 19

5-1 혼동행렬 결과 해석

5-2 ROC 곡선 결과 해석

1. 감성 분석의 기본 이해

감성 분석(Sentiment Analysis)이란 자연어 처리(Natural Language Processing) 기술 중 하나이다. 감성분석은 텍스트에 들어있는 평가나 태도 등의 주관적인 정보를 머신러닝 알고리즘을 이용하여 감정을 분류하거나 긍정/부정의 정도를 점수화 하는 기술이다.

감성분석에는 어휘 기반 감성분석, 사전 기반 감성 분석, 말뭉치 기반 감성분석이 있다.

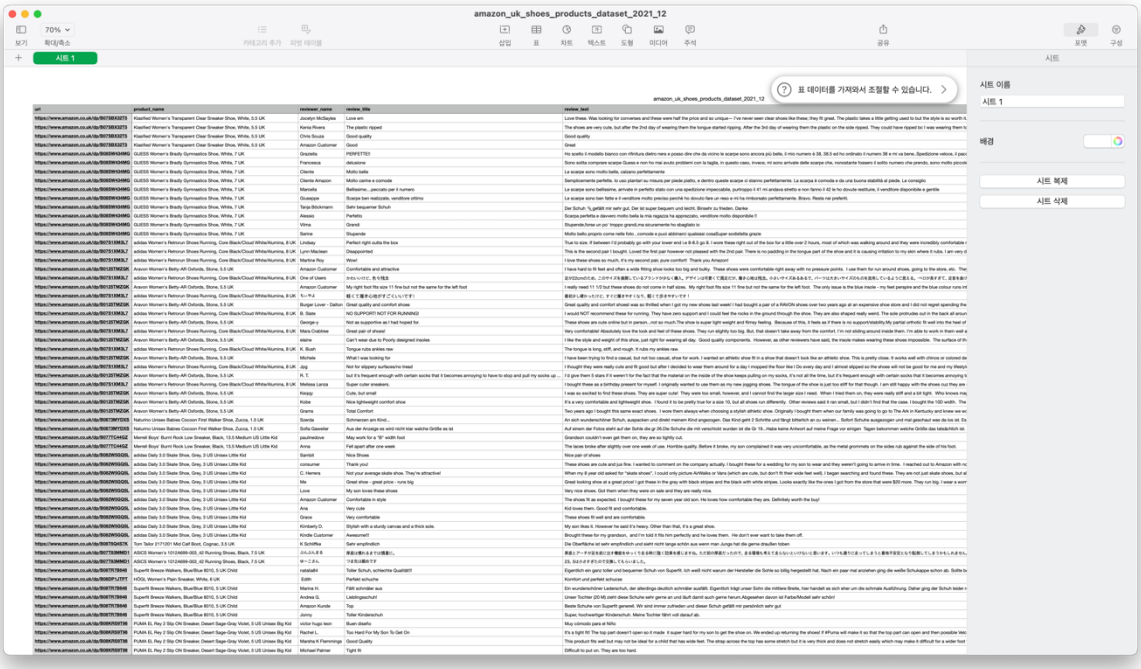
머신러닝 기반 감성 분석은 데이터를 가지고 우선 모델을 학습시킨다. 이후 학습된 모델에 대하여 신규 텍스트의 감정을 예측한다. 최근에는 머신러닝 기반의 감성분석이 많이 수행되고 있다. 그 중 회귀 분석을 통해 사전을 구축하여 진행하는 감성분석이 있다. 회귀 분석을 이용하는 감성분석은 라벨링된 데이터에 회귀 분석 모델을 적용하여 각 단어에 대한 감성 사전을 구축 후 교차 검증을 통해 타당성을 평가한다. 이후 새로운 데이터에 대한 감성 분석을 수행한다.

이 과제에서는 NLTK 의 VADER 을 사용하여 감성분석을 진행한 뒤 Logistic Regression 알고리즘을 사용해 모델을 학습 시키고 리뷰에 대한 감정을 예측한 뒤 성능평가를 진행할 예정이다.

2. 데이터 전처리

2-1. 데이터 정리

데이터 전처리를 위해서는 우선 데이터가 어떻게 구성되어 있는지 확인할 필요가 있다. 'amazon_uk_shoes_products_dataset_2021_12.csv' 데이터는 신발 상품에 대한 리뷰 데이터이다. 해당 데이터의 열은 url : 상품 URL 주소, product_name : 상품명, reviewer_name : 리뷰 작성자 이름, review_title : 리뷰 제목, review_text : 리뷰 내용, _rating : 리뷰 별점, verified_purchase : 구매 확인 여부, review_date : 리뷰 작성 날짜, helpful_count : 리뷰가 도움이 된 사람의 수, uniq_id : 리뷰 ID, scraped_at : 리뷰를 가져온 시점으로 구성되어 있다.



이에 따라 데이터를 어떻게 전처리 할 것인지 생각해 보았다. 우선 결측치를 확인한 뒤 이를 어떻게 처리할 것인지 확인한다. 그리고 실제 구매자가 리뷰를 작성한 리뷰만 가지고 감성분석을 진행할 것이다. 또한 다양한 언어로 리뷰가 작성되어 있는데 VADER 감성분석기에 적용하기 위해 영어리뷰만 남기도록 전처리할 예정이다. 이후 노이즈 제거로 리뷰를 모두 소문자로 변환하고 숫자를 제거한 뒤 불필요한 공백의 경우 제거할 예정이다. 참고로 VADER 의 경우에는 이모티콘, 비속어, 축약어, 감정 기호 등에 감정 분석이 가능함으로 이에 대한 것은 남겨두기로 결정했다.

❯ 결측치

```
# 결측치 확인
df.info()
```
✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6823 entries, 0 to 6822
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   url                6823 non-null   object
 1   product_name       6823 non-null   object
 2   reviewer_name      6823 non-null   object
 3   review_title       6822 non-null   object
 4   review_text        6814 non-null   object
 5   review_rating      6823 non-null   float64
 6   verified_purchase  6823 non-null   bool
 7   review_date        6823 non-null   object
 8   helpful_count      1953 non-null   object
 9   uniq_id            6823 non-null   object
 10  scraped_at         6823 non-null   object
dtypes: bool(1), float64(1), object(9)
memory usage: 539.8+ KB
```

결측치를 확인한 결과 review_title, review_text, helpful_count 열에 대해서 발생한 것을 알 수 있다. 리뷰 제목과 리뷰 내용이 없는 경우 감성분석을 진행할 수 없으므로 해당 행은 데이터프레임에서 제거하는 것으로 결정했다. helpful_count 열은 도움이 된다고 생각한 사람들의 숫자로, 감성분석에 영향을 미치는 열이 아니므로 신경 쓰지 않기로 했다.

```
# 결측치 제거
df = df.dropna(subset=['review_title','review_text'])
```

❯ 실 구매자의 리뷰

실 구매 여부가 False 인 경우의 리뷰를 제거하기 위해 verified_purchase 열의 데이터에 대하여 False 가 아닌 행들만 데이터프레임에 남기기로 했다. 이때 데이터 값에 대하여 문자열로 변환한 뒤 해당 코드를 수행했다.

```
# 실제 구매자가 작성하지 않은 리뷰 제외
df['verified_purchase'] = df['verified_purchase'].astype(str) #문자열로 변환
df = df[df.verified_purchase != 'False']
```

❯ 필요한 열만 선택

감성 분석에 필요한 열인 'review_title', 'review_text', 'review_rating'만 남기기로 결정했다.

```
# 감성분석에 필요한 'review_title', 'review_text', 'review_rating' 열만 선택
df = df[['review_rating', 'review_title', 'review_text']]
```

❯ 영어로 된 리뷰만 선택

영어로 된 리뷰만 남기기위해 langdetect 를 설치하고 detect 를 import 한다. 이후 함수를 정의하고 데이터 프레임에 적용한다.

```
# 영어로 된 리뷰만 남기기
%pip install langdetect
from langdetect import detect
# 'review_text'가 영어인 행만 남기고 나머지 제거 하는 함수 정의
def is_english(text):
    try:
        return detect(text) == 'en'
    except:
        return False
df = df[df['review_text'].apply(is_english)]
```

❯ 노이즈 제거

이후 노이즈 제거를 위해 소문자변환, 숫자제거, 공백처리를 수행하는 전처리 함수를 정의하고 적용한다.

```
# 텍스트 전처리 함수
def preprocess_text(text):
    # 소문자 변환
    text = text.lower()
    # 숫자 제거
    text = re.sub(r'\d+', '', text)
    # 연속된 공백을 하나의 공백으로 통일
    text = re.sub(r'\s+', ' ', text).strip()
    return text
# 'review_title'과 'review_text' 전처리
df['review_title'] = df['review_title'].apply(preprocess_text)
df['review_text'] = df['review_text'].apply(preprocess_text)
```

	review_rating	review_title	review_text
0	5.0	love em	love these. was looking for converses and thes...
1	2.0	the plastic ripped	the shoes are very cute, but after the nd day ...
2	5.0	good quality	good quality
3	5.0	good	great
14	5.0	perfect right outta the box	true to size. if between i'd probably go with ...
...	...	...	...
6813	5.0	great for early walkers	the only shoes (after many tries) that worked ...
6814	3.0	three stars	too narrow hard to get on for a toddler
6815	5.0	said they were very comfortable.	my son loves them. said they were very comfort...
6816	2.0	they are smaller than other shoes the same size	size but they are smaller than the size my son...
6817	4.0	these shoes are great for the price	these shoes are great for the price. been lovi...

3819 rows × 3 columns

<전처리 결과 데이터프레임>

2-2. 텍스트 토큰화 및 불용어 제거

NLTK(Natural Language Toolkit)는 자연어 처리 및 문서 분석용 파이썬 패키지이다. 이 패키지에는 토큰화와 불용어 제거 및 텍스트 정규화에 필요한 패키지들이 있다. 따라서 해당 패키지를 다운로드 받은 후 텍스트 처리에 필요한 패키지를 import 할 예정이다.

텍스트 토큰화(Tokenization)는 텍스트를 단어, 문장, 구문 등의 단위로 분리하는 과정이다. 이는 자연어 문서를 분석하기 위한 첫 단계라고 할 수 있다.

› 텍스트 토큰화

텍스트 토큰화를 위해 NLTK 의 tokenize 의 word_tokenize 를 사용했다. 감성 분석에 사용되는 review_title 과 review_text 열에 대해서 토큰화를 진행했다. 이의 결과를 새로운 열을 만들어 저장했다.

```
# 텍스트 토큰화
df['review_title_token'] = df['review_title'].apply(word_tokenize)
df['review_text_token'] = df['review_text'].apply(word_tokenize)
```

	review_rating	review_title	review_text	review_title_token	review_text_token
0	5.0	love em	love these. was looking for converses and thes...	[love, em]	[love, these, ., was, looking, for, converses,...
1	2.0	the plastic ripped	the shoes are very cute, but after the nd day ...	[the, plastic, ripped]	[the, shoes, are, very, cute, ,, but, after, t...
2	5.0	good quality	good quality	[good, quality]	[good, quality]
3	5.0	good	great	[good]	[great]
14	5.0	perfect right outta the box	true to size. if between i'd probably go with ...	[perfect, right, outta, the, box]	[true, to, size, ., if, between, i, 'd, probab...
...	...	...	...	...	...
6813	5.0	great for early walkers	the only shoes (after many tries) that worked ...	[great, for, early, walkers]	[the, only, shoes, (, after, many, tries,), t...
6814	3.0	three stars	too narrow hard to get on for a toddler	[three, stars]	[too, narrow, hard, to, get, on, for, a, toddler]
6815	5.0	said they were very comfortable.	my son loves them. said they were very comfort...	[said, they, were, very, comfortable, .]	[my, son, loves, them, ., said, they, were, ve...
6816	2.0	they are smaller than other shoes the same size	size but they are smaller than the size my son...	[they, are, smaller, than, other, shoes, the, ...	[size, but, they, are, smaller, than, the, siz...
6817	4.0	these shoes are great for the price	these shoes are great for the price. been lovi...	[these, shoes, are, great, for, the, price]	[these, shoes, are, great, for, the, price, .,...

3819 rows × 5 columns

<토큰화 결과 데이터프레임>

› 불용어 제거

불용어(Stopwords)는 텍스트에서 자주 등장하지만 의미가 없는 단어들을 말한다. 영어의 "the", "is", "in", "and" 등이 불용어에 해당한다. 불용어를 제거함으로써 텍스트의 주요 의미를 더 잘 파악할 수 있다.

이러한 불용어 제거를 위해 NLTK 의 stopwords 를 사용했다. 감성 분석에 사용되는 review_title 과 review_text 열에 대해서 불용어 제거를 진행했다. 이의 결과를 토큰화 한 열에 저장했다.

```
# 불용어 제거 함수
stop_words = set(stopwords.words('english'))

def remove_stopwords(tokens):
    return [word for word in tokens if word.lower() not in stop_words]

# 불용어 제거 적용
df['review_title_token'] = df['review_title_token'].apply(remove_stopwords)
df['review_text_token'] = df['review_text_token'].apply(remove_stopwords)
```

	review_rating	review_title	review_text	review_title_token	review_text_token
0	5.0	love em	love these. was looking for converses and thes...	[love, em]	[love, ., looking, converses, half, price, uni...
1	2.0	the plastic ripped	the shoes are very cute, but after the nd day ...	[plastic, ripped]	[shoes, cute, ,, nd, day, wearing, tongue, sta...
2	5.0	good quality	good quality	[good, quality]	[good, quality]
3	5.0	good	great	[good]	[great]
14	5.0	perfect right outta the box	true to size. if between i'd probably go with ...	[perfect, right, outta, box]	[true, size, ., 'd, probably, go, lower, end, ...
...	...	...	...	...	...
6813	5.0	great for early walkers	the only shoes (after many tries) that worked ...	[great, early, walkers]	[shoes, (, many, tries,), worked, early, walk...
6814	3.0	three stars	too narrow hard to get on for a toddler	[three, stars]	[narrow, hard, get, toddler]
6815	5.0	said they were very comfortable.	my son loves them. said they were very comfort...	[said, comfortable, .]	[son, loves, ., said, comfortable, .]
6816	2.0	they are smaller than other shoes the same size	size but they are smaller than the size my son...	[smaller, shoes, size]	[size, smaller, size, son, outgrowing, ., disa...
6817	4.0	these shoes are great for the price	these shoes are great for the price. been lovi...	[shoes, great, price]	[shoes, great, price, ., loving, skechers, sho...

3819 rows × 5 columns

<불용어 제거 결과 데이터 프레임>

결과를 확인해 보면 are 과 같은 것들이 삭제된 것을 확인할 수 있다.

2-3. 텍스트 정규화

텍스트 정규화 (Text Normalization)는 단어의 형태를 표준화 하는 과정으로, 주로 스테밍(Stemming)과 표제어 추출(Lemmatization)을 말한다.

스테밍(Stemming)이란 단어의 어근(stem)을 추출하는 과정이다. 단순히 단어의 끝을 자르는 방식으로 구현되며, 결과는 항상 실제 단어가 아닐 수도 있다. 표제어 추출(Lemmatization)은 단어를 그 기본 형태(표제어)로 변환하는 과정이다. 표제어 추출은 스테밍보다 더 정교하며, 단어의 품사(part of speech)를 고려한다.

따라서 더 정교한 표제어 추출로 텍스트를 정규화 하기로 결정했다. 이에 NLTK 의 WordNetLemmatizer 를 import 했다.

```
# 표제어 추출 함수
lemmatizer = WordNetLemmatizer()

def lemmas(tokens):
    return [lemmatizer.lemmatize(word, pos='v') for word in tokens] #'v'는 동사(verb)라는 뜻

# 표제어 추출 적용
df['review_title_token'] = df['review_title_token'].apply(lemmas)
df['review_text_token'] = df['review_text_token'].apply(lemmas)
```

	review_rating	review_title	review_text	review_title_token	review_text_token
0	5.0	love em	love these. was looking for converses and thes...	[love, em]	[love, ., look, converse, half, price, unique—...
1	2.0	the plastic ripped	the shoes are very cute, but after the nd day ...	[plastic, rip]	[shoe, cute, ,, nd, day, wear, tongue, start, ...
2	5.0	good quality	good quality	[good, quality]	[good, quality]
3	5.0	good	great	[good]	[great]
14	5.0	perfect right outta the box	true to size. if between i'd probably go with ...	[perfect, right, outta, box]	[true, size, ., 'd, probably, go, lower, end, ...
...	...	...	...	...	...
6813	5.0	great for early walkers	the only shoes (after many tries) that worked ...	[great, early, walkers]	[shoe, (, many, try,), work, early, walker, b...
6814	3.0	three stars	too narrow hard to get on for a toddler	[three, star]	[narrow, hard, get, toddler]
6815	5.0	said they were very comfortable.	my son loves them. said they were very comfort...	[say, comfortable, .]	[son, love, ., say, comfortable, .]
6816	2.0	they are smaller than other shoes the same size	size but they are smaller than the size my son...	[smaller, shoe, size]	[size, smaller, size, son, outgrow, ., disappo...
6817	4.0	these shoes are great for the price	these shoes are great for the price. been lovi...	[shoe, great, price]	[shoe, great, price, ., love, skechers, shoe, ...

3819 rows × 5 columns

<표제어 추출 결과 데이터 프레임>

looking 이 look 으로 변하는 등의 결과를 확인할 수 있다.

3. 모델 선택 및 학습

› VADER 을 통한 감성 분석

접근이 용이한 NTLK 의 VADER 을 사용하여 감성분석을 진행했다.

```
# 필요 라이브러리 import
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# VADER 감정 분석기 초기화
vader_sentiment = SentimentIntensityAnalyzer()

# 감정 분석 함수 정의
def calc_sentiment(review_tokens):
    if isinstance(review_tokens, list):
        review_text = ' '.join(review_tokens)  #토큰 리스트를 문자열로 변환
        return vader_sentiment.polarity_scores(review_text) #감성 분석
    else:
        return {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}  #기본값 반환

# 데이터 프레임의 각 리뷰에 대해 감정 분석 수행
df["review_title_sentiment_score"] = df['review_title_token'].apply(calc_sentiment)
df["review_text_sentiment_score"] = df['review_text_token'].apply(calc_sentiment)

# compound 점수만 추출하여 새로운 열 생성
df['review_title_compound'] = df['review_title_sentiment_score'].apply(lambda x:
x['compound'])
df['review_text_compound'] = df['review_text_sentiment_score'].apply(lambda x:
x['compound'])
```

	review_rating	review_title	review_text	review_title_token	review_text_token	review_title_sentiment_score	review_text_sentiment_score	review_title_compound	review_text_compound
0	5.0	love em	love these. was looking for converses and thes...	[love, em]	[love, ., look, converse, half, price, unique—...	{'neg': 0.0, 'neu': 0.192, 'pos': 0.808, 'comp...	{'neg': 0.07, 'neu': 0.446, 'pos': 0.484, 'com...	0.6369	0.9188
1	2.0	the plastic ripped	the shoes are very cute, but after the nd day ...	[plastic, rip]	[shoe, cute, ,, nd, day, wear, tongue, start, ...	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	{'neg': 0.0, 'neu': 0.841, 'pos': 0.159, 'comp...	0.0000	0.6705
2	5.0	good quality	good quality	[good, quality]	[good, quality]	{'neg': 0.0, 'neu': 0.256, 'pos': 0.744, 'comp...	{'neg': 0.0, 'neu': 0.256, 'pos': 0.744, 'comp...	0.4404	0.4404
3	5.0	good	great	[good]	[great]	{'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound...	{'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound...	0.4404	0.6249
14	5.0	perfect right outta the box	true to size. if between i'd probably go with ...	[perfect, right, outta, box]	[true, size, ., 'd, probably, go, lower, end, ...	{'neg': 0.0, 'neu': 0.448, 'pos': 0.552, 'comp...	{'neg': 0.07, 'neu': 0.728, 'pos': 0.202, 'com...	0.5719	0.6361
...	...	...	...	...	...	...	...	...	...
6813	5.0	great for early walkers	the only shoes (after many tries) that worked ...	[great, early, walkers]	[shoe, (, many, try,), work, early, walker, b...	{'neg': 0.0, 'neu': 0.328, 'pos': 0.672, 'comp...	{'neg': 0.0, 'neu': 0.623, 'pos': 0.377, 'comp...	0.6249	0.8658
6814	3.0	three stars	too narrow hard to get on for a toddler	[three, star]	[narrow, hard, get, toddler]	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	{'neg': 0.318, 'neu': 0.682, 'pos': 0.0, 'comp...	0.0000	-0.1027
6815	5.0	said they were very comfortable.	my son loves them. said they were very comfort...	[say, comfortable, .]	[son, love, ., say, comfortable, .]	{'neg': 0.0, 'neu': 0.233, 'pos': 0.767, 'comp...	{'neg': 0.0, 'neu': 0.211, 'pos': 0.789, 'comp...	0.5106	0.8176
6816	2.0	they are smaller than other shoes the same size	size but they are smaller than the size my son...	[smaller, shoe, size]	[size, smaller, size, son, outgrow, ., disappo...	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	{'neg': 0.351, 'neu': 0.649, 'pos': 0.0, 'comp...	0.0000	-0.4019
6817	4.0	these shoes are great for the price	these shoes are great for the price. been lovi...	[shoe, great, price]	[shoe, great, price, ., love, skechers, shoe, ...	{'neg': 0.0, 'neu': 0.328, 'pos': 0.672, 'comp...	{'neg': 0.0, 'neu': 0.52, 'pos': 0.48, 'compou...	0.6249	0.8519

3818 rows × 9 columns

<감성분석 결과 데이터 프레임>

VADER 감정 분석기에서 각 점수(neg, neu, pos, compound)는 서로 다른 방식으로 계산된다. Negative(부정)은 텍스트에서 부정적인 단어의 비율이다. Neutral(중립)은 텍스트에서 중립적인 단어의 비율이다. Positive(긍정)은 텍스트에서 긍정적인 단어의 비율이다. compound: 텍스트의 전체적인 감정을 나타내는 종합 점수로 -1(매우 부정적)에서 1(매우 긍정적) 사이의 값이다. 이때 compound 점수는 텍스트의 전체적인 감정을 나타내는 종합 점수로, neg, neu, pos 점수를 바탕으로 계산된다.

› 결과 확인

첫번째 리뷰를 보면 제목에 대하여 "love em"으로 이 제품이 좋다는 뜻을 가지고 있다. 이에 대하여 감성 분석은 'neu'가 0.192 'pos'가 0.808 로 compound 는 0.6369 라는 결과를 도출했다. 0 보다 1 에 가까운 것으로, 높은 긍정을 띄게 되었다. 본래의 긍정적인 뜻과 맞게 나온 것을 알 수 있다. 마지막에서 두번째 리뷰의 내용을 보면 해당 신발의 사이즈가 작다며 disappoint 로 실망스럽다는 뜻을 보이고 있다. 이에 대하여 감성분석 결과 'neg'는 0.351, 'neu'는 0.649 로 compound 는 -0.4019 로 부정의 감정을 나타내고 있음을 알 수 있다.

하지만 감성분석이 제대로 안되는 경우도 있다. 별점은 1.0, "i would not recommend these for running. they have zero support and i could feel the rocks in the ground through the shoe. they are also shaped really weird. the sole protrudes out in the back all around the heel and makes your foot appear much larger than it should."로 부정적 리뷰를 작성한 것에 대해 {'neg': 0.071, 'neu': 0.745, 'pos': 0.184, 'compound': 0.4951}라는 결과를 도출해 긍정적으로 분석한 내용도 있었다.

› VADER 의 강점과 약점

이를 통해 VADER 의 강점과 약점에 대하여 알 수 있었다. VADER 의 강점은 사전 훈련된 모델로서, 별도의 훈련 없이 즉시 사용할 수 있다는 점과 감정 점수(positive, negative, neutral, compound)를 모두 제공하여 다양한 측면에서 감정을 분석할 수 있다는 점이 있다. 반대로 VADER 의 약점은 미리 정의된 단어 사전을 사용하므로, 새로운 단어나 신조어를 인식하지 못할 수 있다는 점과 "not bad"와 같은 부정어를 적절히 처리하지만, 이보다 더 복잡한 부정 표현에서는 한계가 있을 수 있다는 점이 있다.

따라서 개요에서 설명했듯이 Logistic Regression 을 통한 모델의 성능을 향상시킬 필요성을 느꼈다.

› Logistic Regression

Logistic Regression 이란 입력 데이터를 해당 클래스 레이블에 할당하는 확률을 예측하는 분류 알고리즘이다. 확률값은 0 에서 1 사이이며 임계값(threshold)를 설정하여 클래스를 구분한다.

이에 따라 감성분석한 내용에 대하여 라벨링할 필요가 있다.

```
# logistic regression 적용을 위한 라벨링 함수 정의
def change_to_binary(sentiment_label):
    if sentiment_label >= 0.001: #0.001 이상이면 긍정, 나머지는 부정
        return 1
    else:
        return 0

# 데이터 프레임의 각 리뷰에 대해 라벨링
df["review_title_sentiment_label"] =
df.review_title_compound.apply(change_to_binary)
df["review_text_sentiment_label"] = df.review_text_compound.apply(change_to_binary)
```

	review_rating	review_title	review_text	review_title_token	review_text_token	review_title_sentiment_score	review_text_sentiment_score	review_title_compound	review_text_compound	review_title_sentiment_label	review_text_sentiment_label
0	5.0	love em	love these. was looking for converses and thes...	[love, em]	[love, ., look, converse, half, price, unique—...	{'neg': 0.0, 'neu': 0.192, 'pos': 0.808, 'comp...	{'neg': 0.07, 'neu': 0.446, 'pos': 0.484, 'com...	0.6369	0.9188	1	1
1	2.0	the plastic ripped	the shoes are very cute, but after the nd day ...	[plastic, rip]	[shoe, cute, ., nd, day, wear, tongue, start, ...	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	{'neg': 0.0, 'neu': 0.841, 'pos': 0.159, 'comp...	0.0000	0.6705	0	1
2	5.0	good quality	good quality	[good, quality]	[good, quality]	{'neg': 0.0, 'neu': 0.256, 'pos': 0.744, 'comp...	{'neg': 0.0, 'neu': 0.256, 'pos': 0.744, 'comp...	0.4404	0.4404	1	1
3	5.0	good	great	[good]	[great]	{'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound...	{'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound...	0.4404	0.6249	1	1
14	5.0	perfect right outta the box	true to size. if between i'd probably go with ...	[perfect, right, outta, box]	[true, size, ., 'd, probably, go, lower, end, ...	{'neg': 0.0, 'neu': 0.448, 'pos': 0.552, 'comp...	{'neg': 0.07, 'neu': 0.728, 'pos': 0.202, 'com...	0.5719	0.6361	1	1
...	...	...	...	...	...	...	...	...	...	...	...
6813	5.0	great for early walkers	the only shoes (after many tries) that worked ...	[great, early, walkers]	[shoe, (, many, try,), work, early, walker, b...	{'neg': 0.0, 'neu': 0.328, 'pos': 0.672, 'comp...	{'neg': 0.0, 'neu': 0.623, 'pos': 0.377, 'comp...	0.6249	0.8658	1	1
6814	3.0	three stars	too narrow hard to get on for a toddler	[three, star]	[narrow, hard, get, toddler]	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	{'neg': 0.318, 'neu': 0.682, 'pos': 0.0, 'comp...	0.0000	-0.1027	0	0
6815	5.0	said they were very comfortable.	my son loves them. said they were very comfort...	[say, comfortable, .]	[son, love, ., say, comfortable, .]	{'neg': 0.0, 'neu': 0.233, 'pos': 0.767, 'comp...	{'neg': 0.0, 'neu': 0.211, 'pos': 0.789, 'comp...	0.5106	0.8176	1	1

<라벨링 결과 데이터 프레임>

❯ 라이브러리 설명

```
# 필요 라이브러리 다운
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score
```

train_test_split 함수는 데이터를 학습용 데이터와 테스트용 데이터로 분할하는 데 사용한다. 이 함수는 데이터를 무작위로 섞어서 분할한다. GridSearchCV 클래스는 하이퍼파라미터 튜닝을 자동화하는 데 사용한다. 여러 하이퍼파라미터 조합을 시도하여 최적의 성능을 내는 파라미터를 찾는다. sklearn.metrics 에서 제공하는 각 평가 지표 함수는 정확도, 정밀도, 재현율, F1, ROC_AUC 가 있다.

정확도 (Accuracy)는 전체 데이터 중 올바르게 예측된 샘플의 비율을 말한다. 단순하지만, 클래스 불균형이 심한 경우에는 부적절하다. 공식은 Accuracy = (True Positives + True Negatives) / Total Samples 이다.

정밀도 (Precision)는 모델이 양성이라고 예측한 것 중 실제 양성의 비율을 말한다. 양성 예측이 정확해야 할 때 (예: 스팸 필터링, 의료 진단) 사용된다. 공식은 Precision = True Positives / (True Positives + False Positives)이다.

재현율 (Recall)은 실제 양성 중 모델이 양성이라고 예측한 비율을 말한다. 실제 양성을 최대한 많이 찾아야 할 때 (예: 질병 검출) 사용된다. 공식은 Recall = True Positives / (True Positives + False Negatives)이다.

F1 점수 (F1 Score)은 정밀도와 재현율의 조화 평균이다. 불균형 클래스 문제에서 유용하다. 공식은 F1 Score = 2 · (Precision·Recall / (Precision + Recall))이다.

ROC-AUC (Receiver Operating Characteristic - Area Under Curve)은 모델의 분류 임계값을 바꾸면서, 참 긍정 비율(TPR)과 거짓 긍정 비율(FPR)을 비교한 곡선의 아래 면적을 구한다. 0.5 는 랜덤 예측, 1.0 은 완벽한 예측을 의미한다.

› 모델 학습

먼저 특성과 라벨을 분리하였다. 제목과 내용에 대한 감성분석 결과를 특성으로 저장하였고, 이는 2 차원 배열로 진행되어야 한다. 라벨은 모델이 예측하려고 하는 출력 데이터로, 모델이 학습하는 동안 실제 값과 비교하여 모델의 성능을 평가하는 데 사용되는 데이터이다. 따라서 리뷰 내용에 대한 감성분석을 진행한 것에 대한 데이터를 라벨로 저장하였다. 왜 특성과 라벨로 분리해야 하나면 모델은 특성을 입력으로 받아 라벨을 예측하는 방식으로 학습하기 때문이다.

이후 데이터를 7(학습):3(테스트)으로 분리하였다. 이때 시드값을 77 로 정하여 여러 번 코드를 실행하여도 동일한 값을 갖도록 하였다.

모델을 초기화 한 뒤 하이퍼파라미터를 튜닝했다. 하이퍼파라미터 튜닝은 모델의 성능을 최적화하는 데 매우 중요하다. 잘못 설정된 하이퍼파라미터는 모델의 성능을 저하시키거나 오버피팅과 언더피팅을 일으킬 수 있다. GridSearchCV 는 다양한 하이퍼파라미터 조합을 자동으로 테스트하고 최적의 조합을 선택해줌으로써 튜닝을 효율적으로 수행할 수 있게 한다. 이때 Logistic Regression 하이퍼파라미터 C 는 규제 강도를 제어하는 역수이다. 값이 작을수록 규제가 강해진다. 모델이 복잡한 패턴을 덜 학습하게 하여 과적합을 방지하려는 목적을 가지고 규제한다. 반대로 값이 크면 규제가 약하므로 모델이 데이터를 더 세밀하게 학습할 수 있다.

정의한 하이퍼파라미터 그리드와 함께 GridSearchCV 객체를 초기화한다. 이때, 교차 검증의 폴드 수(cv)는 5 로, 평가 지표(scoring)는 정확도도 지정한다. 이후 모델 학습을 진행한다. GridSearchCV 는 정의된 각 하이퍼파라미터 조합에 대해 교차 검증을 수행하며, 각 조합의 성능을 평가한다. 모든 하이퍼파라미터 조합에 대한 교차 검증이 완료되면, GridSearchCV 는 최적의 하이퍼파라미터 조합을 선택한다. 이를 출력한다.

```python
# 특성과 라벨 분리
X = df[['review_title_compound', 'review_text_compound']]
y = df['review_text_sentiment_label']

# 학습 및 테스트 데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=77)

# 로지스틱 회귀 모델 초기화
model = LogisticRegression()

# 하이퍼파라미터 그리드 정의
param_grid = {'C': [0.01, 0.1, 1, 10, 100]}

# 그리드 서치 초기화
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy')

# 모델 학습
grid_search.fit(X_train, y_train)

# 최적의 하이퍼파라미터 출력
print("Best hyperparameters: ", grid_search.best_params_)
```

✓ 0.2s

```
Best hyperparameters:  {'C': 100}
```

<모델 학습 화면>

› 모델 선택 및 예측

최적의 하이퍼파라미터로 학습된 모델을 최적의 모델로 선택하며 이를 활용해 예측을 수행한다.

```python
# 최적의 모델 선택
best_model = grid_search.best_estimator_

# 예측
y_pred = best_model.predict(X_test)
```

4. 모델 평가

4-1. 지표를 사용한 모델 평가

› 평가 지표 계산

정확도, 정밀도, 재현율, F1, ROC_AUC 를 계산한다.

```
# 최적의 모델 선택
best_model = grid_search.best_estimator_

# 예측
y_pred = best_model.predict(X_test)

# 평가 지표 계산
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"ROC-AUC: {roc_auc}")
```

✓ 0.0s

```
Accuracy: 0.9973821989528796
Precision: 1.0
Recall: 0.9969418960244648
F1 Score: 0.998468606431853
ROC-AUC: 0.9984709480122325
```

<결과>

정확도는 0.997...로 올바르게 예측된 샘플의 비율이 매우 높은 것을 알 수 있다.

정밀도는 1.0 으로 모델이 양성이라 예측한 것들은 실제로 모두 양성인 것을 알 수 있다.

재현율은 0.996 으로 실제 양성 중 모델이 양성으로 예측한 비율이 매우 높은 것을 알 수 있다.

F1 점수는 0.998...로 조화 평균이 1 에 가까우므로 모델의 성능이 매우 좋은 것을 알 수 있다.

ROC_AUC 점수 또한 1.0 에 매우 가까운 0.998...로 거의 완벽한 예측을 진행하고 있음을 알 수 있다.

4-2. 오버피팅과 언더피팅

오버피팅(과적합) 모델이 학습 데이터에 과적합되어 테스트 데이터에서 성능이 저하되는 현상을 말한다. 교차 검증, 정규화, 더 많은 데이터 수집으로 해결할 수 있다.

언더피팅(과소적합) 모델이 학습 데이터의 패턴을 제대로 학습하지 못한 상태를 말한다. 모델의 복잡성 증가, 더 많은 특징 사용으로 해결할 수 있다.

오버피팅과 언더피팅에 대한 확인을 위해 교차 검증을 진행한다.

› 교차 검증

각 지표들에 대하여 교차 검증을 실시하고 교차검증에 대한 평균과 표준편차를 확인한다.

```
# 교차 검증
cv_scores_accuracy = cross_val_score(best_model, X, y, cv=5, scoring='accuracy')
print(f"Cross-validation accuracy scores: {cv_scores_accuracy}")
print(f"Mean cross-validation accuracy: {np.mean(cv_scores_accuracy)}") #평균
print(f"Standard deviation of cross-validation accuracy: {np.std(cv_scores_accuracy)}")
#표준편차
print('\n')

cv_scores_precision = cross_val_score(best_model, X, y, cv=5, scoring='precision')
print(f"Cross-validation precision scores: {cv_scores_precision}")
print(f"Mean cross-validation precision: {np.mean(cv_scores_precision)}") #평균
print(f"Standard deviation of cross-validation precision: {np.std(cv_scores_precision)}")
#표준편차
print('\n')

cv_scores_recall = cross_val_score(best_model, X, y, cv=5, scoring='recall')
print(f"Cross-validation recall scores: {cv_scores_recall}")
print(f"Mean cross-validation recall: {np.mean(cv_scores_recall)}") #평균
print(f"Standard deviation of cross-validation recall: {np.std(cv_scores_recall)}") #표준편차
print('\n')

cv_scores_f1 = cross_val_score(best_model, X, y, cv=5, scoring='f1')
print(f"Cross-validation f1 scores: {cv_scores_f1}")
print(f"Mean cross-validation f1: {np.mean(cv_scores_f1)}") #평균
print(f"Standard deviation of cross-validation f1: {np.std(cv_scores_f1)}") #표준편차
print('\n')

cv_scores_roc_auc = cross_val_score(best_model, X, y, cv=5, scoring='roc_auc')
print(f"Cross-validation roc_auc scores: {cv_scores_roc_auc}")
print(f"Mean cross-validation roc_auc: {np.mean(cv_scores_roc_auc)}") #평균
print(f"Standard deviation of cross-validation roc_auc: {np.std(cv_scores_roc_auc)}") #표준편차
```

```
Cross-validation accuracy scores: [0.9986911  0.9973822  0.99737877 0.99868938 0.99606815]
Mean cross-validation accuracy: 0.9976419204984458
Standard deviation of cross-validation accuracy: 0.000980959533543142


Cross-validation precision scores: [1. 1. 1. 1. 1.]
Mean cross-validation precision: 1.0
Standard deviation of cross-validation precision: 0.0


Cross-validation recall scores: [0.99847793 0.99695586 0.99695586 0.99847793 0.99542683]
Mean cross-validation recall: 0.9972588818353936
Standard deviation of cross-validation recall: 0.0011412465020327774


Cross-validation f1 scores: [0.99923839 0.99847561 0.99847561 0.99923839 0.99770817]
Mean cross-validation f1: 0.9986272328889912
Standard deviation of cross-validation f1: 0.0005723046022707091


Cross-validation roc_auc scores: [0.9995448 1.        1.        1.        1.       ]
Mean cross-validation roc_auc: 0.999908960298155
Standard deviation of cross-validation roc_auc: 0.000182079403689972
```

<결과>

cross_val_score 함수를 사용하여 5-겹 교차 검증을 수행하고, scoring 매개변수로 각각의 성능 지표를 지정하여, 해당 지표에 대한 교차 검증 성능을 평가한다.

평균은 데이터의 중앙값을 나타내며, 교차 검증을 통해 얻은 성능 지표들의 평균은 모델의 전반적인 성능을 나타낸다. 표준편차는 데이터가 평균값 주위에 얼마나 분포되어 있는지를 나타내며, 변동성 또는 분산의 척도이다. 모델의 성능이 얼마나 안정적인지를 나타낸다.

예를 들어 평균이 높고 표준편차가 낮다면 모델의 성능이 전반적으로 좋고 일관되게 유지된다는 의미이다. 이때는 새로운 데이터에 모델을 적용해도 좋은 성능을 보일 가능성이 높다. 평균과 표준편차가 모두 높다면 일부 교차 검증 폴드에서는 성능이 매우 좋지만 다른 폴드에서 성능이 떨어졌다는 의미이므로 모델의 성능이 좋지만 불안정하다는 뜻이다. 이때는 오버피팅을 의심해볼 수 있다.

결과를 살펴보면 전반적으로 평균은 1.0 에 가깝고 표준편차는 매우 낮은 것을 알 수 있다. 즉, 이 모델은 성능이 매우 좋은 것을 알 수 있다.

5. 시각화 및 해석

4-1. 혼동행렬 결과 해석

혼동행렬(Confusion Matrix)은 분류 모델의 성능을 평가하기 위해 예측 결과와 실제 라벨을 비교한 표이다.

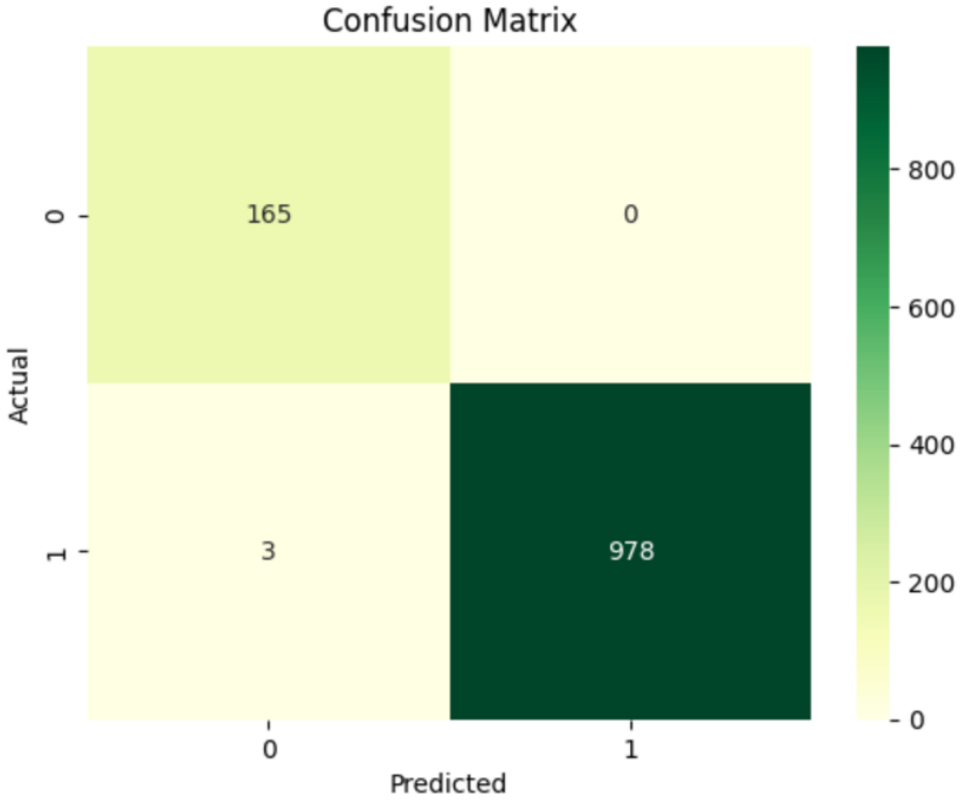
	예측: 부정	예측: 긍정
실제: 부정	TN	FP
실제: 긍정	FN	TP

True Negative(TN): 실제 값이 부정(Negative)이고, 모델이 부정으로 예측한 경우.
False Positive (FP): 실제 값이 부정(Negative)인데, 모델이 긍정으로 예측한 경우.
False Negative (FN): 실제 값이 긍정(Positive)인데, 모델이 부정으로 예측한 경우.
True Positive(TP): 실제 값이 긍정(Positive)이고, 모델이 긍정으로 예측한 경우.

혼동행렬은 위와 같은 방식으로 나타난다. 이를 seaborn 의 히트맵을 통해 시각화했다.

```
# 혼동 행렬 생성
conf_matrix = confusion_matrix(y_test, y_pred)

# 혼동 행렬 시각화
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='YlGn') #혼동행렬에 대하여 히트맵
작성
plt.title('Confusion Matrix') #그래프 제목 지정
plt.xlabel('Predicted') #x 축 제목 지정
plt.ylabel('Actual') #y 축 제목 지정
plt.show()
```



<혼동행렬 시각화 결과>

실제 값이 부정인데, 모델이 긍정으로 예측한 경우는 한 번도 없었고, 실제 값이 긍정인데, 모델이 부정으로 예측한 경우는 3 번 있었다.

실제 값이 긍정이고, 모델이 긍정으로 예측한 경우는 978 번, 실제 값이 부정이고, 모델이 부정으로 예측한 경우는 165 번이다.

즉, 이 모델은 예측을 매우 잘 한다는 것을 알 수 있다. 성능이 좋은 것을 확인할 수 있다.

4-2. ROC 곡선 결과 해석

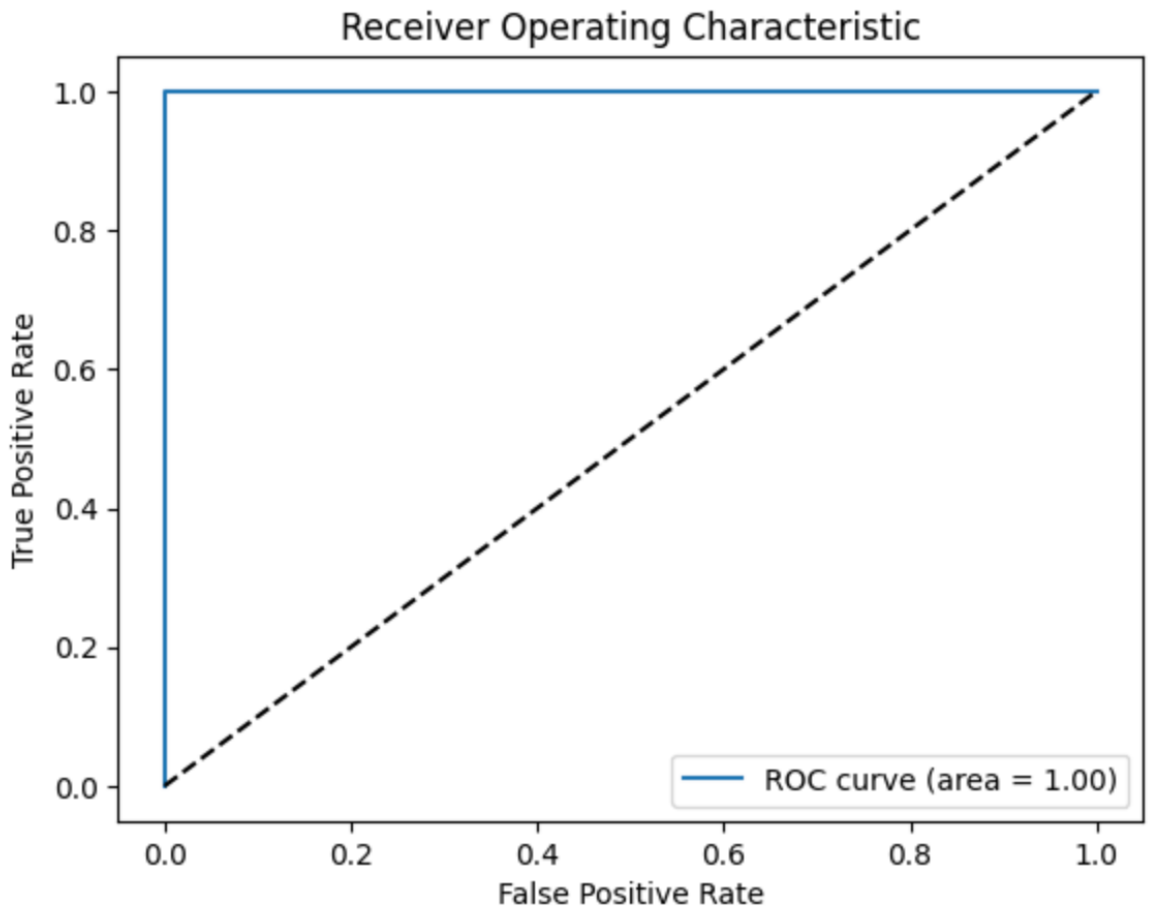
predict_proba 는 각 클래스에 대한 예측 확률을 반환한다. 이에 대하여 긍정 클래스(1)의 확률만 선택하도록 하였다. 즉, 테스트 데이터(X_test)에 대한 각 샘플의 긍정 클래스(1)일 확률을 반환한 데이터 값을 y_prob 에 저장한다.

roc_curve 함수는 다양한 임계값(threshold)에 대해 FPR 과 TPR 을 계산하여 반환한다. 이때 임계값은 모델이 긍정 클래스와 부정 클래스를 구분하는 기준이 되는 값이다. 예를 들어, 임계값을 0.5 로 설정하면, 예측 확률이 0.5 이상인 경우 긍정 클래스로, 그렇지 않은 경우 부정 클래스로 예측한다. roc_curve 함수는 세 가지 값을 반환한다. 다양한 임계값에 대한 FPR 값의 리스트, 다양한 임계값에 대한 TPR 값의 리스트, 계산할 때 사용된 임계값의 리스트이다.

이 반환된 값을 가지고 X 축을 FPR(실제 부정을 긍정으로 예측한 비율), Y 축을 TPR(실제 긍정을 긍정으로 올바르게 예측비율)로 두고 그래프를 그린다.

```
#ROC 곡선 생성
y_prob = best_model.predict_proba(X_test)[:, 1] #테스트 데이터에 대한 예측 확률을 반환.
[:, 1]은 긍정 클래스(1)의 확률을 선택.
fpr, tpr, _ = roc_curve(y_test, y_prob) #다양한 임계값에 대한 FPR(False Positive Rate),
TPR(True Positive Rate) 계산

plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc) #FPR 를 x 축, TPR 를
y 축으로 그래프 생성
plt.plot([0, 1], [0, 1], 'k--') #[0, 1]에서 [0, 1]로 이어지는 대각선 점선 그래프를 추가
plt.title('Receiver Operating Characteristic') #그래프 제목 지정
plt.xlabel('False Positive Rate') #x 축 제목 지정
plt.ylabel('True Positive Rate') #y 축 제목 지정
plt.legend(loc="lower right") #우하단 범례 위치 지정
plt.show()
```



<ROC 곡선 시각화 결과>

결과가 1.00 으로 모델이 모든 긍정 샘플을 긍정으로, 모든 부정 샘플을 부정으로 정확히 예측을 했음을 알 수 있다. 이런 경우는 흔치 않은데 예상으로 데이터의 크기가 매우 작아서 그렇다고 생각한다. 전처리 과정에서 절반의 리뷰가 미사용되었다. 이 점이 영향이 된 것 같다.
