

Wine review 데이터의 감성 분석

문헌정보학과 202120622 박주홍

목차	
1. 감성 분석의 기본 이해	5. 모델 평가
2. 데이터 전처리	5.1 모델평가
2.1 데이터 정리	5.2 오버피팅과 해결방안
2.2 텍스트 토큰화 및 불용어 제거	6. 시각화 및 보고
2.3 텍스트 정규화	6.1 결과 시각화
3. 모델 구현	6.2 혼동 행렬
4. 모델 학습 및 튜닝	6.3 ROC 곡선
4.1 데이터셋 분할	6.4 결과 분석
4.2 모델학습 및 하이퍼파라미터 튜닝, 교차검증	

1. 감성 분석의 기본 이해

감성분석이란 대량의 텍스트를 분석하여 해당 내용이 긍정/부정/중립적인지에 대한 감정을 판단하는 분석이다. 자연어 처리(NLP)와 머신러닝(ML)기술을 활용한다.

2. 데이터 전처리

2.1. 데이터 정리

실행 번호	입력 코드
1	<pre>import pandas as pd df = pd.read_csv('./wine_review.csv')</pre>

csv 파일을 불러오는 pandas를 호출한 뒤 'wine\_review.csv'을 불러온다.

실행 번호	입력 코드
2	<pre>print("The shape of the data (rows, cols) is " + str(df.shape))</pre>
	출력 결과
	The shape of the data (rows, cols) is (2890, 32)

'wine\_review.csv' 파일의 행과 열의 개수를 확인한다. 결측치 처리 후의 결과 비교하여 사용할 데이터의 수를 확인한다.

실행 번호	입력 코드
3	df.isnull().sum()
	출력 결과
	<pre>id                0 asins             2020 brand             65 categories        0 dateAdded         0 dateUpdated       0 descriptions     152 dimension        1052 ean              724 flavors          2739 keys              0 manufacturer     2041 manufacturerNumber 433 name              0 reviews.date      338 reviews.dateAdded 0 reviews.dateSeen  0 reviews.didPurchase 841 reviews.doRecommend 979 reviews.id        1005 reviews.numHelpful 2264 reviews.rating     445 reviews.sourceURLs 0 reviews.text       1 reviews.title      44 ... sizes             2868 sourceURLs        0 upc               147 weight           1894 dtype: int64</pre> <p>Output is truncated. View as a <a href="#">scrollable element</a> or open in a <a href="#">text editor</a>. Adjust cell output <a href="#">settings</a>...</p>

결측치가 있는 열을 확인한다.

실행 번호	입력 코드
4	df = df[['name', 'reviews.rating','reviews.text']]

분석에 필요한 열인 'name'과 'reviews.rating', 'reviews.text'을 추출하여 df로 저장한다.

실행 번호	입력 코드
5	<pre>df = df.dropna(axis=0) print(df.isnull().sum()) print("The shape of the data (rows, cols) is " + str(df.shape))</pre>
	출력 결과
	<pre>name                0 reviews.rating      0 reviews.text        0 dtype: int64</pre> <p>The shape of the data (rows, cols) is (2444, 3)</p>

분석에 방해가 되는 결측치가 있는 행을 제거한다. 따라서 총 2444개의 데이터로 감정 분석을 진행한다. 2444개의 데이터도 충분히 분석할 수 있는 양의 데이터 개수라고 판단하였다.

실행 번호	입력 코드		
6	df.head()		
	출력 결과		
		name	reviews.rating
			reviews.text
	0	Ecco Domani174 Pinot Grigio - 750ml Bottle	5.0
	1	Fresh Craft174 Mango Citrus - 4pk / 250ml Bottle	5.0
2	1000 Stories174 Zinfandel - 750ml Bottle	5.0	
3	1000 Stories174 Zinfandel - 750ml Bottle	5.0	
4	Pink Moscato - 3l Bottle - Wine Cube153	5.0	

데이터 프레임을 확인하여, 열이 정확히 추출되었는지 확인한다.

## 2.2. 텍스트 토큰화 및 불용어 제거

실행 번호	입력 코드
7	%pip install nltk
8	import nltk from nltk.tokenize import word_tokenize from nltk.corpus import stopwords, wordnet from nltk.stem import PorterStemmer, WordNetLemmatizer from nltk import pos_tag from nltk.sentiment.vader import SentimentIntensityAnalyzer
9	nltk.download('punkt') nltk.download('stopwords') nltk.download('wordnet') nltk.download('averaged_perceptron_tagger') nltk.download('vader_lexicon')

텍스트 전처리 및 감정 분석에 필요한 nltk 라이브러리를 설치한 뒤 호출한다. 불용어 등의 감성 분석에 필요한 nltk의 내장 자료를 다운로드한다.

실행 번호	입력 코드
10	def tokenize_reviews(text): return word_tokenize(text)
11	def remove_stopwords(text): stop_words = set(stopwords.words('english')) filtered_tokens = [word for word in text if word.lower() not in stop_words] return filtered_tokens

	<pre>from nltk import pos_tag from nltk.sentiment.vader import SentimentIntensityAnalyzer</pre>
--	---

텍스트 토큰화와 불용어를 제거한다. 다운로드받은 nltk라이브러리의 불용어 리스트를 사용하고 소문자로 변환하여 제거한다.

2.3. 텍스트 정규화

실행 번호	입력 코드
12	<pre>def lemmatize(tokens):     lemmatizer = WordNetLemmatizer()     pos_tags = pos_tag(tokens)     lemmatized_tokens = [lemmatizer.lemmatize(word.lower(), get_wordnet_pos(tag)) for word, tag in pos_tags]     return lemmatized_tokens</pre>
13	<pre>def get_wordnet_pos(treebank_tag):     if treebank_tag.startswith('J'):         return wordnet.ADJ     elif treebank_tag.startswith('V'):         return wordnet.VERB     elif treebank_tag.startswith('N'):         return wordnet.NOUN     elif treebank_tag.startswith('R'):         return wordnet.ADV     else:         return wordnet.NOUN  def stem(tokens):     stemmer = PorterStemmer()     stemmed_tokens = [stemmer.stem(word.lower()) for word in tokens]     return stemmed_tokens</pre>

텍스트 스테밍(어간 추출)과 표제어를 추출한다.

3. 모델 구현

실행 번호	입력 코드
14	<pre>sia = SentimentIntensityAnalyzer()  def analyze_sentiment(text):     sentiment_scores = sia.polarity_scores(text)</pre>

	return sentiment_scores
15	df['tokenized_text'] = df['reviews.text'].apply(tokenize_reviews) df['filtered_text'] = df['tokenized_text'].apply(remove_stopwords) df['stemmed_text'] = df['filtered_text'].apply(stem) df['lemmatized_text'] = df['filtered_text'].apply(lemmatize) df['sentiment'] = df['lemmatized_text'].apply(lambda x: analyze_sentiment(' '.join(x)))

review.text의 내용을 통해 감성을 분석하여 점수로 반환한다. 데이터 전처리 과정인 텍스트 토큰화, 불용어 제거, 어간 추출, 표제어 추출을 순서로 전처리 된 내용을 새로운 열에 저장하여 결과를 확인한다.

실행 번호	입력 코드
16	print(df[['stemmed_text', 'lemmatized_text']])
	출력 결과
	<pre>stemmed_text \ 0          [fantast, white, wine, occas, !] 1          [tart, ,, sweet, ..., refresh, delici, !] 2  [given, wine, delight, surpris, find, flavor, ... 3          [phenomen, wine, new, favorit, red, .] 4  [4, 750ml, bottl, price, two, way, less, packa... ... 2885 [like, sweet, wine, ., skeptic, order, without... 2886 [order, 3, bottl, set, meritag, &amp;, moscata, wi... 2887 [order, white, zinfadel, 3, bottl, select, arr... 2888 [first, purchas, wine, ., tri, moscato, like, ... 2889 [n't, know, neg, review, said, wine, tast, hor...  lemmatized_text 0          [fantastic, white, wine, occasion, !] 1          [tart, ,, sweet, ..., refresh, delicious, !] 2  [give, wine, delightful, surprise, find, flavo... 3          [phenomenal, wine, new, favorite, red, .] 4  [4, 750ml, bottle, price, two, way, le, packag... ... 2885 [like, sweet, wine, ., skeptical, order, witho... 2886 [order, 3, bottle, set, meritage, &amp;, moscata, ... 2887 [order, white, zinfadel, 3, bottle, selection,... 2888 [first, purchase, wine, ., tried, moscato, lik... 2889 [n't, know, negative, review, say, wine, taste...  [2444 rows x 2 columns]</pre>

실행 번호	입력 코드
17	<pre>def sentiment_label(sentiment_scores):     if sentiment_scores['compound'] &gt;= 0.05:         return 1     else:         return 0  df['sentiment_label'] = df['sentiment'].apply(sentiment_label)</pre>

텍스트의 전체적인 감성을 나타내는 compound의 기준을 0.05로 정한다. 0.05 이상은 긍정인 1로 나타내며, 0.05 미만은 부정으로 0을 반환하도록 한다.

#### 4. 모델 학습 및 튜닝

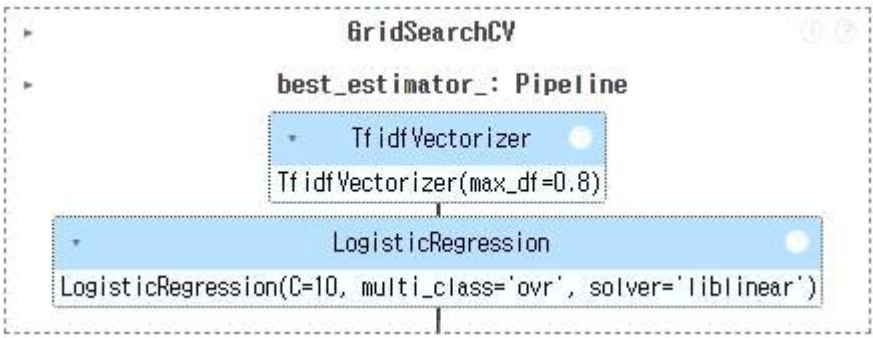
##### 4.1. 데이터셋 분할

실행 번호	입력 코드
18	<pre>from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.model_selection import train_test_split, GridSearchCV from sklearn.linear_model import LogisticRegression from sklearn.pipeline import Pipeline from sklearn.metrics import classification_report, accuracy_score, roc_auc_score, roc_curve, auc, confusion_matrix</pre>
19	<pre>X = df['reviews.text'] y = df['sentiment_label']  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)</pre>

모델 학습에 필요한 라이브러리를 호출한다. 입력 데이터 x와 레이블 y를 설정한 뒤 학습데이터와 테스트 데이터로 분할한다.

##### 4.2. 모델학습 및 하이퍼파라미터 튜닝, 교차검증

실행 번호	입력 코드
20	<pre>pipeline = Pipeline([     ('tfidf', TfidfVectorizer()),     ('logreg', LogisticRegression(solver='liblinear', multi_class='ovr')) ])</pre>
21	<pre>param_grid = {     'tfidf__max_df': [0.8, 0.9, 1.0],     'tfidf__ngram_range': [(1, 1), (1, 2)],     'logreg__C': [0.1, 1, 10]</pre>

	}
22	<code>grid_search = GridSearchCV(pipeline, param_grid, cv=5, n_jobs=-1, verbose=1)</code>
23	<code>grid_search.fit(X_train, y_train)</code>
	<div>출력 결과</div> 

로지스틱 회귀 분류 모델을 사용하기 전 TF-IDF 벡터화를 위한 파이프라인을 정의한다. 이후 그리드 서치를 진행하기 위해 하이퍼파라미터 그리드를 정의한다. 긍정/부정과 같은 두 가지 감정을 분류할 수 있기에 이진 분류 문제에 적합한 로지스틱 회귀 모델을 사용한다.

그리드 서치 시, 5-fold 교차 검증을 적용하여 최적의 하이퍼파라미터를 찾아 모델을 학습시킨다.

결과로 나온 'max\_df = 0.8'은 문서에서 80% 이상 나타나는 즉, 빈도가 높은 단어는 TF-IDF 벡터화에서 무시한다는 의미를 가지고 있다. 로지스틱 회귀에서는 libliner라는 회귀 모델 최적화에 사용되는 알고리즘으로 libliner 알고리즘을 사용한다.

실행 번호	입력 코드
24	<code>print("Best parameters found: ", grid_search.best_params_)</code>
	<code>best_model = grid_search.best_estimator_ y_pred = best_model.predict(X_test) y_pred_proba = best_model.predict_proba(X_test)</code>
	<div>출력 결과</div> <code>Best parameters found: {'logreg__C': 10, 'tfidf__max_df': 0.8, 'tfidf__ngram_range': (1, 1)}</code>

앞서 정의된 그리드 서치를 통해 모델을 학습한다.

## 5. 모델 평가

### 5.1. 모델 평가

실행 번호	입력 코드
25	<code>print("Accuracy:", accuracy_score(y_test, y_pred)) print("Classification Report:\n", classification_report(y_test, y_pred))</code>
	<div>출력 결과</div>

	Accuracy: 0.8957055214723927				
	Classification Report:				
		precision	recall	f1-score	support
	0	0.77	0.48	0.59	77
	1	0.91	0.97	0.94	412
	accuracy			0.90	489
	macro avg	0.84	0.73	0.77	489
	weighted avg	0.89	0.90	0.89	489

정확도, 정밀도, 재현율, F1점수를 통해 모델을 평가한다. 평가 결과는 다음과 같다.

- 1) 정확도(accuracy)는 0.8957로 89.57%의 정확도로 예측할 수 있다.
- 2) class 0에서는 비교적 낮은 재현율과 낮은 f1점수를 가지고 있다.
- 3) class 1에서는 높은 정밀도와 재현율, fq점수를 가지고 있다.
- 4) 즉, 긍정적인 반응(class 1)에 대해서는 잘 예측하고 있으나, 부정적인 반응(class 2)에 대해서는 모델 개선이 필요하다.

실행 번호	입력 코드
26	y_pred_proba = best_model.predict_proba(X_test)[:, 1]
	roc_auc = roc_auc_score(y_test, y_pred_proba)
	print("ROC-AUC Score:", roc_auc)
	출력 결과
	ROC-AUC Score: 0.9031647963686799

ROC-AUC는 10에 가까울 수록 분류 성능이 높다는 것을 의미한다. 따라서 약 0.9032의 결과는 높은 분류 성능을 가진다는 것을 알 수 있다.

5.2. 오버피팅과 해결방안

실행 번호	입력 코드
27	param_grid = { 'tfidf__max_df': [0.8, 0.9, 1.0], 'tfidf__ngram_range': [(1, 1), (1, 2)], 'logreg__C': [0.01, 0.1, 5] #c값을 10에서 5로 재설정 }
	grid_search = GridSearchCV(pipeline, param_grid, cv=5, n_jobs=-1, verbose=1)



	<pre>grid_search.fit(X_train, y_train)  print("Best parameters found: ", grid_search.best_params_)  best_model = grid_search.best_estimator_ y_pred = best_model.predict(X_test)</pre>
	출력 결과
	Best parameters found: {'logreg__C': 5, 'tfidf__max_df': 0.8, 'tfidf__ngram_range': (1, 1)}

평가 결과를 통해 모델이 class 1을 주로 학습하는, 훈련 데이터에 너무 맞추어져 있기에 오버피팅을 의심해 볼 수 있다. 이를 해결하기 위해 로지스틱 회귀의 c값을 작게 설정하여 일 반화할 수 있도록 조정한다.

실행 번호	입력 코드
28	<pre>print("Accuracy:", accuracy_score(y_test, y_pred)) print("Classification Report:\n", classification_report(y_test, y_pred))</pre>
	출력 결과
	Accuracy: 0.8957055214723927
	Classification Report:
	precision recall f1-score support
	0 0.93 0.36 0.52 77
	1 0.89 1.00 0.94 412
	accuracy 0.90 489
	macro avg 0.91 0.68 0.73 489
	weighted avg 0.90 0.90 0.88 489

오버피팅을 조정할 결과는 다음과 같다.

- 1) class 0에 대한 정밀도가 증가했으나, 재현율과 F1-score은 감소했다.
- 2) 이는 클래스 0을 더 적게 예측하고 예측하더라도 정확하지 않는다는 것을 의미한다.
- 3) 오버피팅을 해결하는 방안은 부정적 반응에 대한 데이터의 수를 늘리거나(기존의 결측치가 있는 값을 제거하지 않는 방법이 있다. 또한 사용한 방법과 같이 로지스틱 회귀 값의 강도를 의미하는 c값을 줄인다. 마지막으로 regulaization이라는 모델의 계수를 0에 가깝게 하여 계수의 절대값을 줄이는 방안(Ridge, Lasso)을 사용한다.

6. 시각화 및 보고

6.1. 결과 시각화

실행 번호	입력 코드
29	<pre>import matplotlib.pyplot as plt import seaborn as sns import matplotlib.pyplot as plt import numpy as np</pre>

시각화를 위한 Matplotlib과 Seaborn 라이브러리를 호출한다.

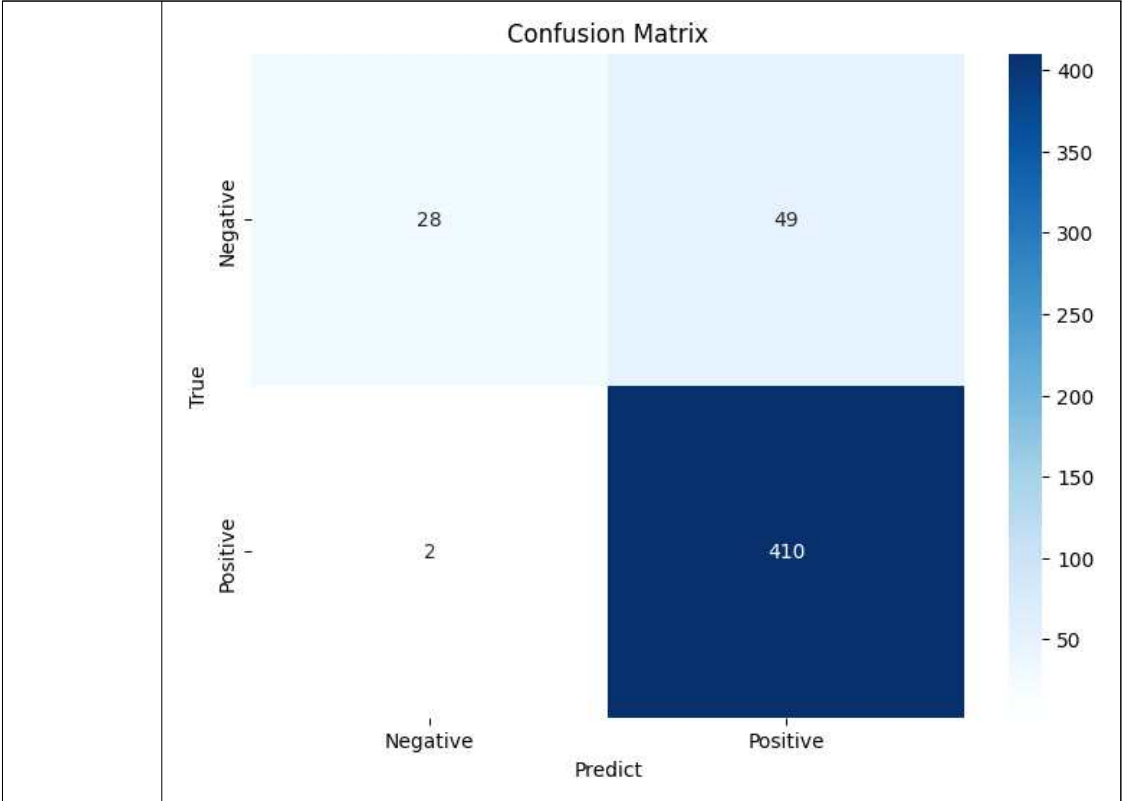
실행 번호	입력 코드
30	<pre>feature_names = best_model.named_steps['tfidf'].get_feature_names_out() #TF-IDF 벡터 화에서 추출된 단어를 가져온다. coefficients = best_model.named_steps['logreg'].coef_[0] #로지스틱 회 귀 모델의 계수를 가져온다.  top_positive_coefficients = np.argsort(coefficients)[-10:] #계수가 가장 큰 10개의 단어 추출 top_negative_coefficients = np.argsort(coefficients)[:10] #계수가 가장 작은 10개의 단어 추출  pos_top_word = [] pos_top_score = [] for coef in top_positive_coefficients:     pos_top_word.append(feature_names[coef]) #단어를 리스트에 추가     pos_top_score.append(coefficients[coef]) #계수를 리스트에 추가  neg_top_word = [] neg_top_score = [] for coef in top_negative_coefficients:     neg_top_word.append(feature_names[coef]) #단어를 리스트에 추가     neg_top_score.append(coefficients[coef]) #계수를 리스트에 추가</pre>

로지스틱 회귀 모델에서 사용된 단어들의 계수를 통해 긍정/부정 단어를 추출하고 시각화하는 과정이다. TF-IDF 벡터화에서 추출된 단어와 계수를 가져와 각각 다른 변수로 저장한다. 그 뒤 긍정적/부정적 감정에 기여하는 상위 10개 단어를 각각 추출하여 단어와 점수를 저장한다.

실행 번호	입력 코드																																																														
31	<pre>plt.figure(figsize=(10, 8)) plt.barh(neg_top_word, neg_top_score, label='negative', color='r') plt.barh(pos_top_word, pos_top_score, label='positive', color='g') plt.legend() plt.xlabel('keyword score') plt.ylabel('Top 10 keywords') plt.title('Positive and Negative keywords analysis ') plt.show()</pre>																																																														
	<div>출력 결과</div> <table><tr><th>Keyword</th><th>Score (approx.)</th><th>Polarity</th></tr><tr><td>great</td><td>9.0</td><td>positive</td></tr><tr><td>love</td><td>8.5</td><td>positive</td></tr><tr><td>best</td><td>6.0</td><td>positive</td></tr><tr><td>good</td><td>4.8</td><td>positive</td></tr><tr><td>better</td><td>4.5</td><td>positive</td></tr><tr><td>favorite</td><td>4.0</td><td>positive</td></tr><tr><td>perfect</td><td>3.5</td><td>positive</td></tr><tr><td>delicious</td><td>3.5</td><td>positive</td></tr><tr><td>helps</td><td>3.5</td><td>positive</td></tr><tr><td>original</td><td>3.0</td><td>positive</td></tr><tr><td>suffered</td><td>-2.5</td><td>negative</td></tr><tr><td>works</td><td>-2.5</td><td>negative</td></tr><tr><td>sores</td><td>-2.5</td><td>negative</td></tr><tr><td>bomb</td><td>-2.5</td><td>negative</td></tr><tr><td>beat</td><td>-2.5</td><td>negative</td></tr><tr><td>years</td><td>-2.5</td><td>negative</td></tr><tr><td>without</td><td>-2.5</td><td>negative</td></tr><tr><td>manhattens</td><td>-2.5</td><td>negative</td></tr><tr><td>dirty</td><td>-3.0</td><td>negative</td></tr><tr><td>bad</td><td>-3.5</td><td>negative</td></tr></table>	Keyword	Score (approx.)	Polarity	great	9.0	positive	love	8.5	positive	best	6.0	positive	good	4.8	positive	better	4.5	positive	favorite	4.0	positive	perfect	3.5	positive	delicious	3.5	positive	helps	3.5	positive	original	3.0	positive	suffered	-2.5	negative	works	-2.5	negative	sores	-2.5	negative	bomb	-2.5	negative	beat	-2.5	negative	years	-2.5	negative	without	-2.5	negative	manhattens	-2.5	negative	dirty	-3.0	negative	bad	-3.5
Keyword	Score (approx.)	Polarity																																																													
great	9.0	positive																																																													
love	8.5	positive																																																													
best	6.0	positive																																																													
good	4.8	positive																																																													
better	4.5	positive																																																													
favorite	4.0	positive																																																													
perfect	3.5	positive																																																													
delicious	3.5	positive																																																													
helps	3.5	positive																																																													
original	3.0	positive																																																													
suffered	-2.5	negative																																																													
works	-2.5	negative																																																													
sores	-2.5	negative																																																													
bomb	-2.5	negative																																																													
beat	-2.5	negative																																																													
years	-2.5	negative																																																													
without	-2.5	negative																																																													
manhattens	-2.5	negative																																																													
dirty	-3.0	negative																																																													
bad	-3.5	negative																																																													

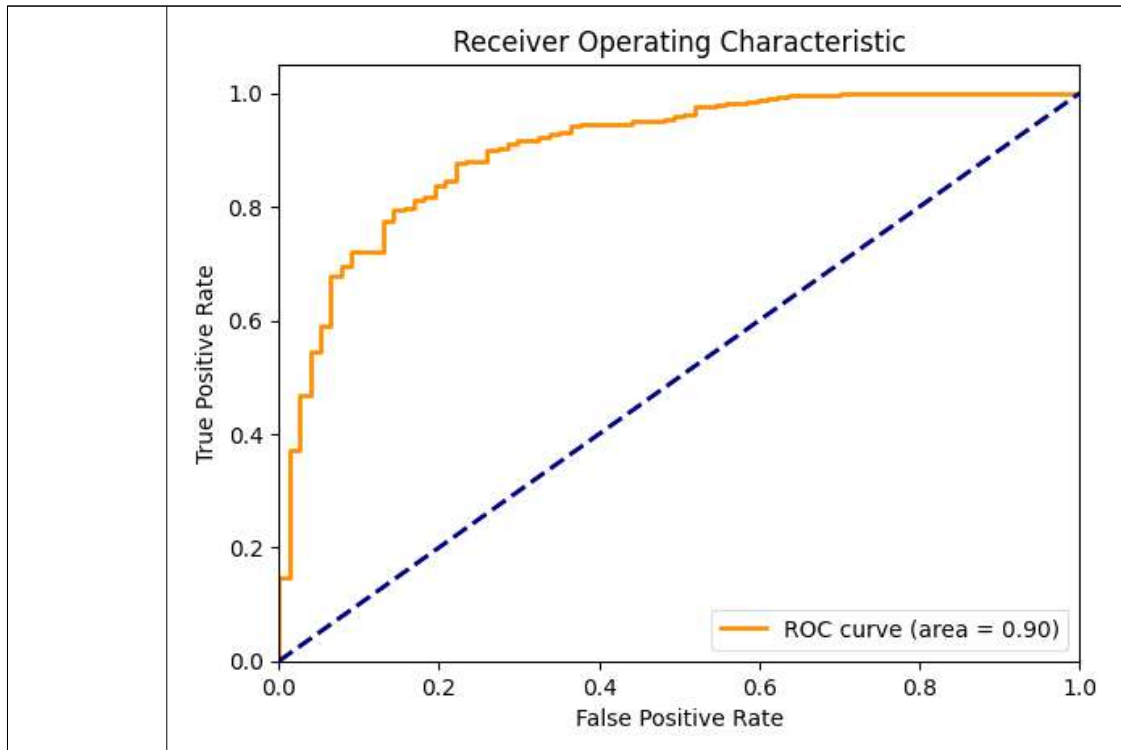
6.2. 혼동행렬

실행 번호	입력 코드
32	<pre>conf_matrix = confusion_matrix(y_test, y_pred) plt.figure(figsize=(8, 6)) sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive']) plt.xlabel('Predict') plt.ylabel('True') plt.title('Confusion Matrix') plt.show()</pre>
	<div>출력 결과</div>



6.3. ROC 곡선

실행 번호	입력 코드
33	<pre>fpr, tpr, _ = roc_curve(y_test, y_pred_proba) roc_auc = auc(fpr, tpr)  plt.figure() plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc) plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title('Receiver Operating Characteristic') plt.legend(loc="lower right") plt.show()</pre>
	출력 결과



#### 6.4. 결과 분석

- 긍정적 키워드는 'great', 'love', 'best', 'good', 'better'등의 표현을 사용하는 것을 알 수 있다.
- 부정적 키워드로는 'bad'가 가장 높았다. 부정적인 반응을 나타내는 키워드(형용사)는 긍정적인 반응에 대한 키워드 보다 적다.
- 주목해야할 부정적 키워드 중 'dirty'는 단어 그대로 '더러운'이라는 뜻이 아닌, 실제 리뷰를 살펴본 결과 주로 'dirty martini'의 'dirty'를 지칭하는 것으로 나타남. 'dirt martini'는 칵테일의 한 종류이다.
- 긍정적인 반응임에도 불구하고 'dirty'라는 키워드가 나온 것을 통해, 고유 명사에 나오는 단어를 해결하지 못하여 부정적인 반응에 대한 예측이 정확하지 않다는 가능성 또한 알 수 있다.
- 혼동 행렬 결과 positive 모델을 실제로 positive로 예측한 결과가 410으로 가장 높게 나왔다.
- 이는 positive, 긍정적인 반응에 대한 예측은 정확히 할 수 있다는 것을 의미한다.
- 가장 낮은 헛수인 상단 왼쪽의 모델이 Negative를 올바르게 예측한 경우, 즉 부정적 반응에 대해 부정적인 반응이라고 예측한 결과는 28으로 매우 낮게 나왔다.
- 따라서, 해당 모델은 positive 클래스에 편향되었다고 볼 수 있다. 이를 해결하기 위해 앞서 설명한 것과 같이 부정적인 사례에 대한 데이터를 추가하거나 오버피팅을 해결하여야 한다.