**기계학습 감성 분석 기말 과제**

**학번 : 201921030**

**이름 : 임재석**

**목차**

1. 감성 분석의 기본 이해

1.1. 감성 분석 소개

1.2. 감성 분석의 응용

2. 데이터 전처리

2.1. 데이터 수집

2.2. 데이터 전처리 과정

3. 모델 선택 및 학습

3.1. 사용된 모델 소개

3.2. 모델 학습 방법

3.3. 모델 강점과 약점

4. 평가

4.1. 성능 평가 지표 소개

4.2. 모델 성능 평가 결과

5. 시각화 및 해석

5.1. 결과 시각화

5.2. 결과 해석

**1-1**

**감성분석이란?**

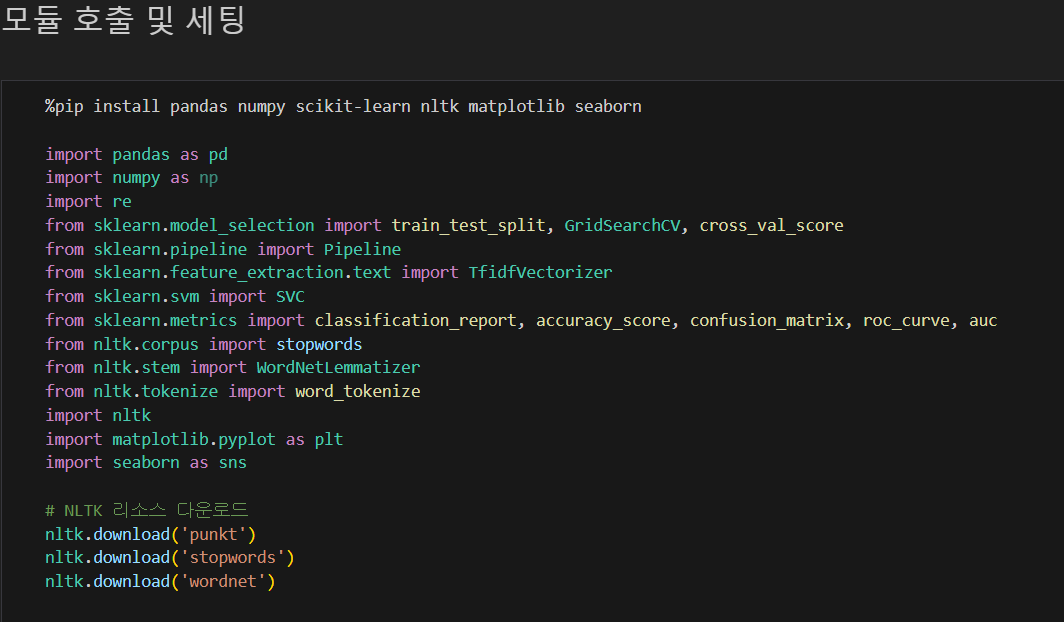
**감성 분석(Sentiment Analysis)**이란 텍스트에 들어있는 의견이나 감성, 평가, 태도 등의 주관적인 정보를 컴퓨터를 통해 분석하는 과정

**응용**

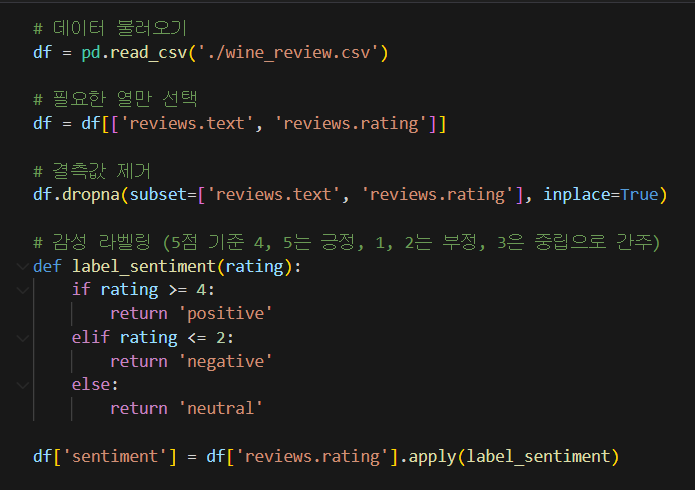
감성 분석은 마케팅에서 고객 서비스, 임상 의학에 이르기까지 다양한 애플리케이션을 위한 리뷰 및 설문 조사 응답, 온라인 및 소셜 미디어, 의료 자료 등 고객의 소리 자료에 널리 적용

**1-2**

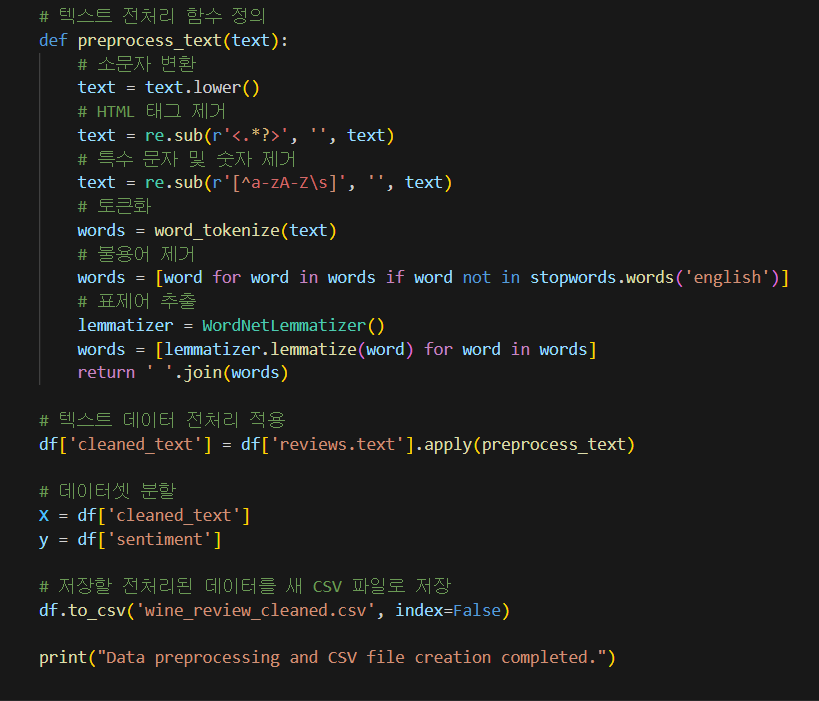
**전처리**



우선 ML에 필요한 모듈 호출과 감성 분석에 필요한 NLTK 소스를 호출

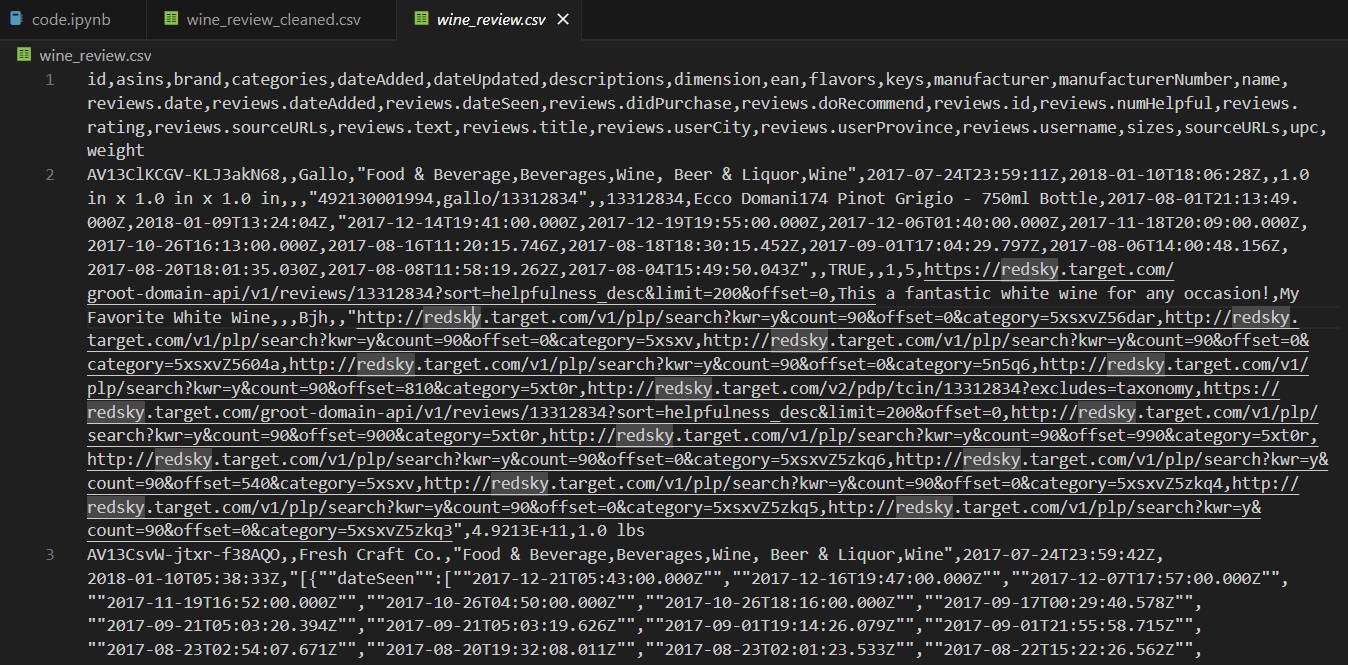


불러온 데이터를 호출하고 기초적인 결측치 제거와 라벨링을 진행



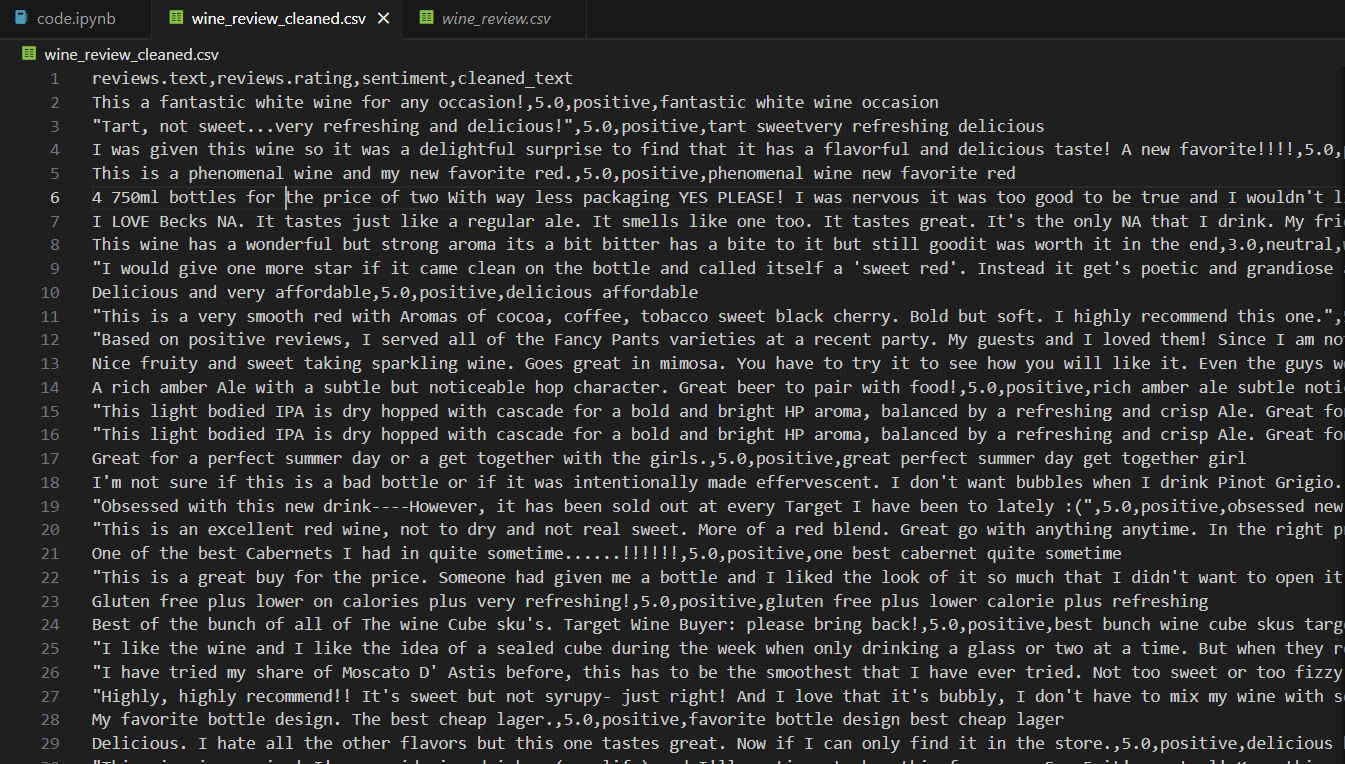
요구사항에 맞춘 토큰화와 각 문자들을 전처리하고 새로운 데이터셋을 csv로 저장

**BEFORE Data Set**



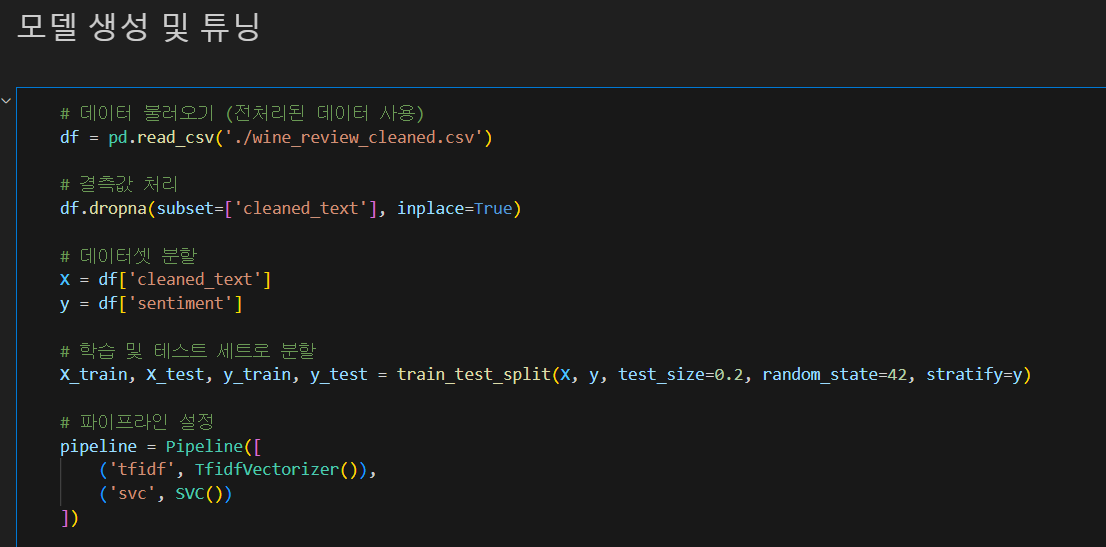
전처리 전 데이터는 각 컬럼과 필요없는 날짜, 링크 등 데이터가 존재

**AFTER Data Set**



토큰 제거와 필요한 평가, 감성 텍스트만 남은 데이터셋을 확인가능

**3 -1 모델**

****

위 코드를 통해 전처리를 거친 데이터를 호출하고 학습을 위해 분할.

분할은 다음과 같음

X = df['cleaned\_text']: 텍스트 데이터를 특징 변수

y = df['sentiment']: 감성 데이터를 목표 변수

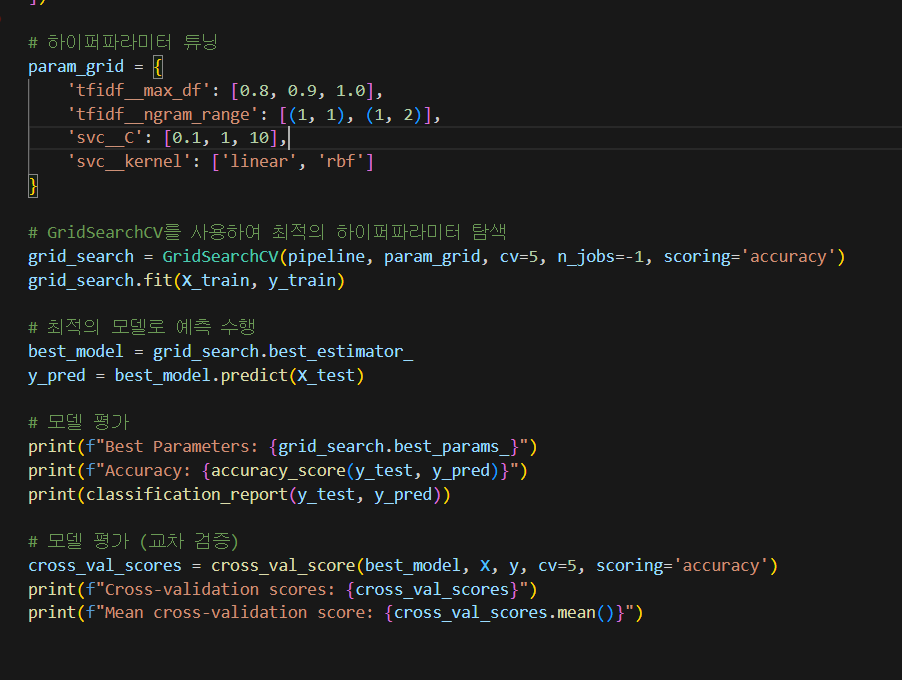
train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y): 데이터를 학습 세트와 테스트 세트로 80:20 비율로 분할 stratify=y는 원본 데이터의 클래스 비율을 유지하기 위해 사용

Pipeline: 여러 처리 단계를 묶어주는 역할

SVC(): 서포트 벡터 머신 분류기를 사용

TfidfVectorizer(): 텍스트 데이터를 TF-IDF(Term Frequency-Inverse Document Frequency) 벡터로 변환

**하이퍼파라미터 튜닝**

****

GridSearchCV에서 사용할 하이퍼파라미터 그리드 사용

GridSearchCV: 교차 검증을 통해 최적의 하이퍼파라미터를 탐색

grid\_search.best\_params\_: 최적의 하이퍼파라미터를 출력

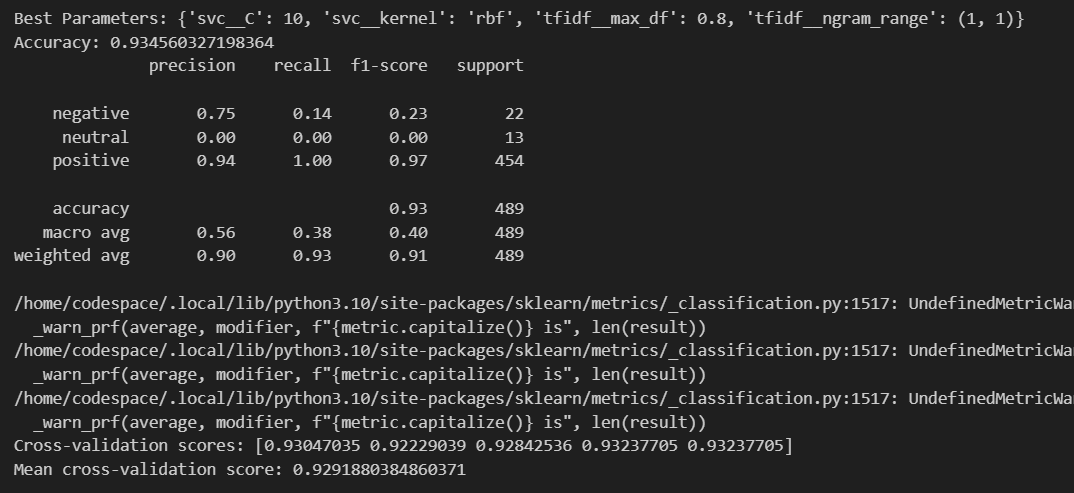
accuracy\_score(y\_test, y\_pred): 테스트 세트의 정확도를 계산

classification\_report(y\_test, y\_pred): 정밀도, 재현율, F1 점수를 포함한 상세한 분류 보고서를 출력

cross\_val\_score(best\_model, X, y, cv=5, scoring='accuracy'): 전체 데이터에 대해 5-폴드 교차 검증을 수행하여 정확도를 평가

cross\_val\_scores.mean(): 교차 검증의 평균 정확도를 계산

**튜닝 결과**



**최적의 하이퍼파라미터**

**svc\_\_C: 10**: SVM 분류기의 규제 매개변수 CCC가 10으로 설정 높은 CCC 값은 모델이 데이터에 더 잘 맞게 되지만, 과적합(overfitting)의 위험 가능

* **svc\_\_kernel: 'rbf'**: RBF (Radial Basis Function) 커널을 사용하여 비선형 경계를 학습
* **tfidf\_\_max\_df: 0.8**: TF-IDF 벡터라이저에서 단어가 전체 문서의 80% 이하에서 나타나면 포함
* **tfidf\_\_ngram\_range: (1, 1)**: Unigram (단일 단어)를 사용

테스트 세트에서 93.46%의 높은 정확도를 달성

**negative** 클래스:

* **precision: 0.75**: 모델이 negative 클래스로 예측한 것 중 75%가 실제로 negative
* **recall: 0.14**: 실제 negative 샘플 중 14%만이 정확히 예측
* **f1-score: 0.23**: 정밀도와 재현율의 조화 평균을 나타냄. 낮은 값은 모델이 negative 클래스에서 잘 작동하지 않음을 나타냈음

**neutral** 클래스:

* **precision: 0.00**, **recall: 0.00**, **f1-score: 0.00**: 모델이 neutral 클래스를 전혀 예측하지 못함. 이는 neutral 샘플이 너무 적어서 모델이 이 클래스를 제대로 학습하지 못한 결과를 도출함

**positive** 클래스:

* **precision: 0.94**: 모델이 positive 클래스로 예측한 것 중 94%가 실제로 positive
* **recall: 1.00**: 실제 positive 샘플 중 100%가 정확히 예측
* **f1-score: 0.97**: 매우 높은 값으로, 모델이 positive 클래스에서 매우 잘 작동하는 것을 알수있음

**전체 정확도**: 0.93으로, 모델이 전반적으로 높은 정확도를 달성

**macro avg**: 각 클래스의 평균 성능을 나타내며, 이 값이 낮은 이유는 neutral 클래스의 성능이 매우 저조함

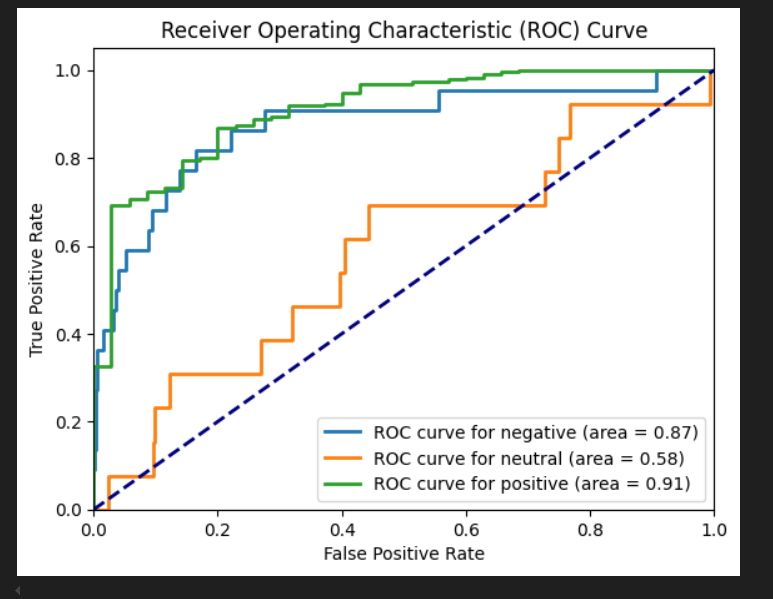
**weighted avg**: 클래스의 지원 수를 고려한 가중 평균입니다. 전체적인 성능을 더 잘 나타냅니다.

**교차 검증 정확도**: 각 폴드에서 높은 정확도를 유지하고 있으며, 평균 교차 검증 정확도가 약 92.92%입니다. 이는 모델이 일관되게 잘 작동함을 나타냄

**튜닝 결론**

* 모델은 positive 클래스에서 매우 잘 작동하며, 전반적인 정확도도 높음
* negative와 neutral 클래스의 성능은 낮으며, 특히 neutral 클래스는 전혀 예측되지 않았음
* 데이터 불균형 문제가 있으며, 이를 해결하기 위해 데이터 증강이나 클래스 가중치를 조정하는 방법을 고려해야함
* 모델의 전반적인 성능은 우수하지만, 모든 클래스에서 균형 잡힌 성능을 얻기 위해 추가적인 조정이 필요하다 판단됨

**5 시각화**

****

분석

각 클래스별 ROC 곡선 및 AUC 값

부정 클래스 (Negative)

ROC 곡선: 파란색

AUC: 0.87

분석: AUC 값이 0.87로 높아 부정 클래스에 대한 모델의 성능이 좋음을 나타냄

중립 클래스 (Neutral)

ROC 곡선: 주황색

AUC: 0.58

분석: AUC 값이 0.58로, 중립 클래스에 대한 모델의 성능이 낮음. 이 클래스에 대한 예측 성능이 저조함을 알수있음.

긍정 클래스 (Positive)

ROC 곡선: 초록색

AUC: 0.91

분석: AUC 값이 0.91로 매우 높아 긍정 클래스에 대한 모델의 성능이 매우 좋음을 나타냅니다.

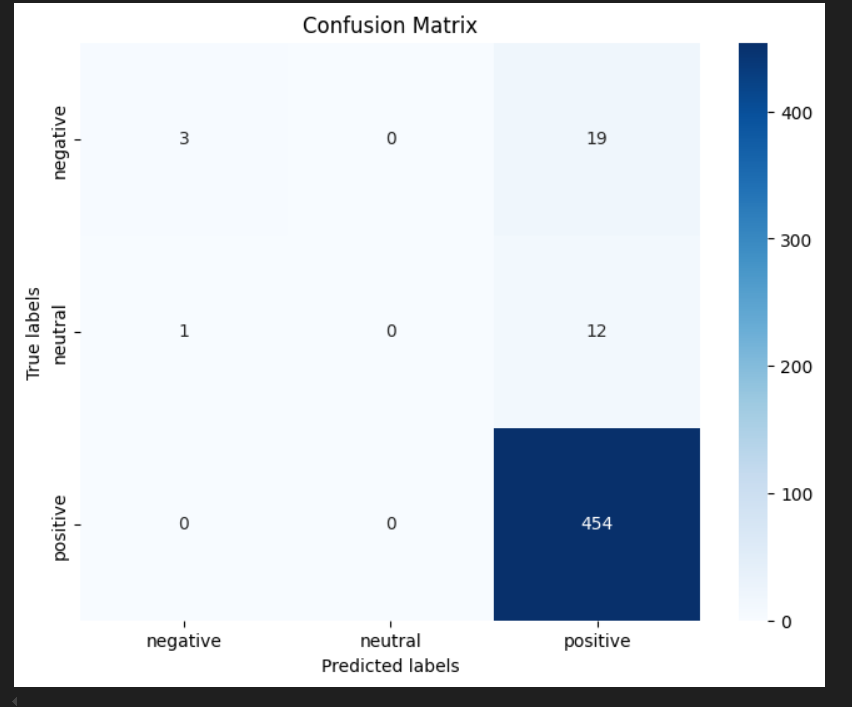
요약 및 결론

긍정 클래스는 AUC 값이 0.91로 가장 높아, 이 클래스에 대한 예측 성능이 매우 뛰어남

부정 클래스도 AUC 값이 0.87로 높아, 모델이 부정 클래스를 비교적 잘 예측하고 있음

중립 클래스는 AUC 값이 0.58로, 모델이 중립 클래스를 잘 예측하지 못지만. 이는 앞서 혼동행렬 분석에서 확인한 바와 같이 중립 클래스를 거의 긍정 클래스로 예측하는 경향과 일치

혼돈 행렬 분석

****

Negative (부정) 클래스:

총 22개 (3 + 0 + 19) 중 실제 부정인 22개 데이터 중 3개는 부정으로 정확하게 예측되었고, 19개는 긍정으로 잘못 예측.

단, 중립으로 예측된 경우는 없음

Neutral (중립) 클래스:

총 13개 (1 + 0 + 12) 중 실제 중립인 13개 데이터 중 1개는 부정으로, 12개는 긍정으로 잘못 예측.

중립으로 예측된 경우는 없음

Positive (긍정) 클래스:

총 454개 (0 + 0 + 454) 중 실제 긍정인 454개 데이터는 모두 긍정으로 정확하게 예측.