**기계학습\_기말고사 대체과제**

**(제품 review 데이터의 감성 분석)**



|  |  |
| --- | --- |
| **과목명:** | 기계학습(8585) |
| **학과:** | 소프트웨어전공 |
| **학번:** | 201920848 |
| **이름:** | 김석원 |
| **제출일자:** | 2024-06-21 |

**프로젝트 개요**

본 프로젝트는 파이썬을 사용하여 SVM (Support Vector Machine) 모델과 SVR (Support Vector Regression) 모델을 구현하고, wine\_review.csv 데이터를 전처리하여 감성 분석을 수행하는 것을 목적으로 한다. 이 데이터셋은 와인 리뷰와 그에 따른 점수(평점)로 구성되어 있으며, 각 리뷰의 감성(긍정적 또는 부정적)을 예측하는 것이 이 프로젝트의 핵심 과제이다.

**데이터 전처리**

wine\_review.csv 데이터를 로드하고, 결측치 처리 및 텍스트 데이터의 특성 추출을 통해 학습 가능한 형태로 변환하였으며, SVM모델과 SVR모델의 사용할 데이터를 각각 전처리 하였다.

**데이터 전처리(공통부분) :**

**SVM과 SVR 모델의 사용할 데이터 전처리가 공통 적용된 부분이다.**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

위 코드는 wine\_review.csv 데이터를 SVM 분류 모델과 SVR 회귀 모델에 공통으로 적용한 데이터 전처리 과정이다. 여기서 각 단계를 설명하면 다음과 같다:

**1. 결측값 처리 :**

* **df.dropna(subset=['reviews.text', 'reviews.rating'], inplace=True) :** 리뷰 텍스트**(reviews.text)**와 평점**(reviews.rating)**이 없는 행을 제거한다. 이는 모델이 텍스트와 평점이 있는 데이터에서 학습할 수 있도록 한다.

**2. 불필요한 열 제거 :**

* **columns\_to\_drop :** 분석에 불필요한 열들을 리스트로 지정한다.
* **df = df.drop(columns=columns\_to\_drop) :** 지정된 열들을 데이터프레임에서 삭제한다

**3. 리뷰 텍스트 합치기 :**

* **df['reviews'] = df['reviews.text'].fillna('') + ' ' + df['reviews.title'].fillna('') :** 리뷰의 본문**(reviews.text)**과 제목**(reviews.title)**을 합쳐서 새로운 **(reviews)**열을 생성한다. 결측값은 빈 문자열**('')**로 대체한다.
* **df.drop(columns=['reviews.text', 'reviews.title'], inplace=True) :** 기존의 리뷰 텍스트와 제목 열을. 삭제한다.

**4. 평점이 3.0인 행 제거 :**

* **df = df[df['reviews.rating'] != 3.0] :** 평점이 3.0인 리뷰를 제거한다. SVM 분류 모델에서는 이진.분류를 수행하기 때문에 중립적인 평점(3점)을 제외하는 것이 일반적이다.

**5. 레이블 생성 :**

* **df['label'] = np.where(df['reviews.rating'] > 3, 1, 0) :** 평점이 4 이상인 경우에는 긍정적인 레이블**(1)**, 2 이하인 경우에는 부정적인 레이블**(0)**을 부여하여 label 열에 저장한다. 이는 감성 분류를 위한 타겟 변수로 사용된다.

**데이터 전처리(SVM 분류 모델)**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위 코드는 SVM 분류 모델에서 사용할 텍스트 데이터를 전처리하는 함수인 **preprocess\_text**를 정의한 부분이다. 각 단계를 설명하면 다음과 같다:

**1. 소문자 변환 (text.lower()) :**

* 입력된 텍스트를 모두 소문자로 변환한다. 이는 단어의 대소문자 구분을 없애기 위한 일반적인 전처리 단계이다

**2. 구두점 제거 (text.translate(str.maketrans('', '', string.punctuation))) :**

* **string.punctuation**에 정의된 모든 구두점을 제거한다. 구두점이 텍스트의 의미 분석에 불필요하므로 제거한다.

**3. 숫자 제거 (re.sub(r'\d+', '', text)) :**

* 정규 표현식을 사용하여 텍스트에서 모든 숫자를 제거한다. 텍스트 마이닝에서는 일반적으로 숫자는 의미 있는 정보를 제공하지 않기 때문에 삭제한다.

**4. 불용어 제거 :**

* **stopwords.words('english')**를 사용하여 영어 불용어 목록을 불러온다. 이 목록에 있는 단어들을 제거한다.
* **word\_tokenize(text)**를 통해 텍스트를 단어 단위로 토큰화한 후, 불용어 목록에 포함되지 않은 단어만 남기고 이를 다시 공백을 이용하여 연결한다.

**5. 어간 추출 (PorterStemmer() 사용) :**

* **PorterStemmer()**를 사용하여 단어의 어간을 추출한다. 어간 추출은 단어의 원형을 추출하는 과정으로, 단어의 다양한 형태를 통일시켜 단어의 의미를 파악하기 쉽게 한다.
* 각 단어를 어간 추출기를 사용하여 변환한 후, 다시 공백을 기준으로 단어를 연결하여 최종적으로 전처리된 텍스트를 반환한다.

이 함수는 입력된 텍스트 데이터에 대해 소문자 변환, 구두점 제거, 숫자 제거, 불용어 제거, 어간 추출을 수행하여 SVM 분류 모델에서 사용할 수 있는 형태로 변환한다. 이러한 전처리 과정은 텍스트 데이터의 정제와 의미를 보존하는데 도움을 주며, SVM 모델 학습 전에 이 함수를 사용하여 각 리뷰 텍스트를 처리할 수 있다..

**데이터 전처리(SVR 회귀 모델)**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위 코드는 SVR (Support Vector Regression) 회귀 모델에서 사용할 텍스트 데이터를 간단히 전처리하는 함수인 **preprocess\_text**를 정의한 부분이다. 각 단계를 설명하면 다음과 같다:

**1. 소문자 변환 (text.lower()) :**

* 입력된 텍스트를 모두 소문자로 변환한다. 이는 단어의 대소문자 구분을 없애기 위한 일반적인 전처리 단계이다.

**2. 정규 표현식을 사용한 문자 제거 (re.sub(r'[^a-zA-Z\s]', '', text) ) :**

* **[^a-zA-Z\s]** 패턴을 사용하여 영문 알파벳과 공백을 제외한 모든 문자를 제거한다.
* **re.sub()** 함수를 사용하여 해당 패턴에 매칭되는 문자를 공백으로 대체한다. 따라서 텍스트에서 알파벳과 공백만 남게 된다.

**3. 불필요한 공백 처리 (' '.join(text.split())) :**

* **text.split()**을 사용하여 텍스트를 공백을 기준으로 단어 단위로 나눈 후,
* 다시 **' '.join()**을 사용하여 나눠진 단어들을 공백을 기준으로 연결한다. 이 과정에서 중복된 공백을 제거하여 최종적으로 정제된 텍스트를 반환한다.

이 함수는 입력된 텍스트 데이터에 대해 대소문자를 통일하고, 영문 알파벳과 공백 이외의 문자를 제거하여 SVR 회귀 모델에서 사용할 수 있는 형태로 변환한다. SVR 모델은 텍스트 데이터를 직접 사용하는 것이 아니라, 텍스트를 숫자로 변환하여 회귀 분석을 수행하므로 이러한 간단한 전처리 단계로도 충분히 효과적인 데이터 준비가 가능하다.

**모델 구현, 학습 및 튜닝**

**SVM 분류모델**

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

위 코드는 SVM 분류 모델을 구현하고, 데이터를 학습 및 튜닝하여 최적의 모델을 찾고, 이를 사용하여 테스트 데이터를 예측하는 과정을 담고 있다. 각 단계를 설명하면 다음과 같다:

**1. 전처리된 데이터 불러오기 :**

* **df=pd.read\_csv("./wine\_review\_filtered\_svm.csv") :** 미리 전처리된 데이터 파일**(wine\_review\_filtered\_svm.csv)**을 **pandas**의 데이터프레임으로 불러온다.

**2. 특성과 레이블 분리하기 :**

* **X = df['cleaned\_reviews'] :** 전처리된 리뷰 데이터를 X 변수에 저장한다.
* **y = df['label'] :** 감성 레이블(0 또는 1)을 y 변수에 저장한다.

**3. 학습 데이터와 테스트 데이터로 나누기 :**

* **train\_test\_split()** 함수를 사용하여 X와 y를 학습 데이터**(X\_train, y\_train)**와 테스트 데이터**(X\_test, y\_test)**로 나눈다. 여기서 **test\_size=0.2**는 전체 데이터의 20%를 테스트 데이터로 사용하겠다는 의미이며, **random\_state=42**는 재현성을 위한 랜덤 시드이다.

**4. TF-IDF 벡터화를 사용하여 텍스트 데이터 변환하기 :**

* **TfidfVectorizer**를 사용하여 텍스트 데이터를 **TF-IDF (Term Frequency-Inverse Document Frequency)** 형태로 벡터화한다. **max\_features=5000**은 최대 5000개의 단어 특성만 사용하도록 설정한다.
* **fit\_transform()** 메서드로 학습 데이터**(X\_train)**를 벡터화하고**, transform()** 메서드로 테스트 데이터**(X\_test)**를 벡터화한다.

**5. SVM 모델 초기화 및 하이퍼파라미터 그리드 정의 :**

* **SVC(probability=True)**를 사용하여 SVM 분류기를 초기화한다. **probability=True**는 클래스 확률을 계산할 수 있도록 설정한다.
* **param\_grid** 변수에 SVM 모델의 하이퍼파라미터 그리드를 정의한다. 여기서는 다양한 **커널(linear, rbf, poly), 규제 매개변수 C, 커널 계수 gamma, 다항식 커널의 차수 degree**를 조정할 수 있도록 설정한다.

**6. GridSearchCV 초기화 및 하이퍼파라미터 튜닝 :**

* **GridSearchCV**를 사용하여 SVM 모델에 대한 그리드 서치를 수행한다. **cv=5**는 **5-fold cross-validation**을 사용하여 모델을 평가한다**. n\_jobs=-1**은 모든 CPU 코어를 사용하여 병렬 처리를 수행하도록 설정한다. **scoring='accuracy'**는 정확도를 평가 지표로 사용함을 의미한다.
* **fit()** 메서드를 호출하여 그리드 서치를 통해 최적의 하이퍼파라미터를 탐색하고, 학습 데이터**(X\_train\_vectorized, y\_train)**에 대해 모델을 학습시킨다.

**7. 최적의 모델로 테스트 데이터 예측 :**

* **grid\_search.best\_estimator\_**에서 반환된 최적의 모델을 **best\_model** 변수에 저장한다.
* **best\_model.predict()** 메서드를 사용하여 테스트 데이터**(X\_test\_vectorized)**에 대한 예측값을 계산하고, 이를 **y\_pred** 변수에 저장한다. 예측값은 각 데이터 포인트의 클래스 **레이블(0 또는 1)**이다.
* **best\_model.predict\_proba()** 메서드를 사용하여 각 클래스에 대한 예측 확률을 계산하고, 이 중 **긍정 클래스(레이블 1)**에 대한 확률을 **y\_pred\_proba** 변수에 저장한다.

**SVR 회귀모델**

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

위 코드는 SVR (Support Vector Regression) 회귀 모델을 구현하고, 데이터를 학습 및 튜닝하여 최적의 모델을 찾고, 이를 사용하여 테스트 데이터를 예측하는 과정을 담고 있다. 각 단계를 설명하면 다음과 같다:

**1. 전처리된 데이터 불러오기 :**

* **df=pd.read\_csv("./wine\_review\_filtered\_svr.csv") :** 미리 전처리된 데이터 파일**(wine\_review\_filtered\_svr.csv)**을 **pandas**의 데이터프레임으로 불러온다.

**2. 특성과 레이블 분리하기 :**

* **X = df['cleaned\_reviews'] :** 전처리된 리뷰 데이터를 X 변수에 저장한다.
* **y = df['reviews.rating'] :** 각 리뷰의 평점을 y 변수에 저장한다. 이 경우 SVR 모델에서는 텍스트가 아닌 숫자 형태의 평점을 예측할 수 있도록 한다.

**3. 학습 데이터와 테스트 데이터로 나누기 :**

* **train\_test\_split()** 함수를 사용하여 X와 y를 학습 데이터**(X\_train, y\_train)**와 테스트 데이터**(X\_test, y\_test)**로 나눈다. 여기서 **test\_size=0.2**는 전체 데이터의 20%를 테스트 데이터로 사용하겠다는 의미이며, **random\_state=42**는 재현성을 위한 랜덤 시드이다.

**4. TF-IDF 벡터화를 사용하여 텍스트 데이터 변환하기 :**

* **TfidfVectorizer**를 사용하여 텍스트 데이터를 **TF-IDF (Term Frequency-Inverse Document Frequency)** 형태로 벡터화한다. **max\_features=5000**은 최대 5000개의 단어 특성만 사용하도록 설정한다.
* **fit\_transform()** 메서드로 학습 데이터**(X\_train)**를 벡터화하고, **transform()** 메서드로 테스트 데이터**(X\_test)**를 벡터화 한다.

**5. SVR 모델 초기화 및 하이퍼파라미터 그리드 정의 :**

* **SVR()**를 사용하여 SVR 회귀 모델을 초기화한다.
* **param\_grid** 변수에 SVR 모델의 하이퍼파라미터 그리드를 정의한다. 여기서는 다양한 **커널(linear, rbf, poly), 규제 매개변수 C, 커널 계수 gamma, 다항식 커널의 차수 degree**를 조정할 수 있도록 설정한다..

**6. GridSearchCV 초기화 및 하이퍼파라미터 튜닝 :**

* **GridSearchCV**를 사용하여 SVR 모델에 대한 그리드 서치를 수행한다. **cv=5**는 **5-fold cross-validation**을 사용하여 모델을 평가한다. **n\_jobs=-1**은 모든 CPU 코어를 사용하여 병렬 처리를 수행하도록 설정한다**. scoring='neg\_mean\_squared\_error'**는 평균 제곱 오차의 반대값을 최소화하는 방향으로 모델을 평가함을 의미한다.

**7. 최적의 모델로 테스트 데이터 예측 :**

* **grid\_search.best\_estimator**\_에서 반환된 최적의 모델을 **best\_model** 변수에 저장한다.
* **best\_model.predict()** 메서드를 사용하여 테스트 데이터**(X\_test\_vectorized)**에 대한 평점 예측을 수행하고, 예측 결과를 **y\_pred** 변수에 저장한다.

**결과 및 분석**

**SVM 분류 모델 결과 및 분석 :**

**1. 하이퍼파라미터 튜닝 결과**

****

* **C: 10** - 모델의 정규화 파라미터로, 큰 값일수록 모델이 더 많은 데이터를 설명하려고 한다. 이는 높은 값을 설정해 학습 데이터에 더 잘 맞추도록 하였다

.

* **degree: 2** - 다항식 커널을 사용할 경우 차수를 결정한다. 하지만 여기서는 선형 커널을 사용하므로 실제로 의미는 없다..
* **gamma: 'scale'** - 데이터의 특성(피처)의 수와 분산을 사용하여 gamma 값을 자동으로 설정한다.
* **kernel: 'linear'** - 선형 커널을 사용하여 데이터 간의 선형 관계를 모델링한다.

**2. 교차 검증 결과**

****

* **최적의 교차 검증 정확도** - 5겹 교차 검증 동안 평균적으로 96%의 정확도를 보여준다. 이는 모델이 학습 데이터에 대해 매우 일관되게 성능을 보여준다는 것을 의미한다.

**3. 테스트 데이터 평가 결과**

**텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

자동 생성된 설명**

* **정확도 (Accuracy)** - 전체 예측 중 정확하게 맞춘 비율로 96%를 기록하였다. 이는 모델이 테스트 데이터에서 매우 좋은 성능을 보였음을 나타낸다.
* **정밀도 (Precision)** - 양성으로 예측한 샘플 중 실제로 양성인 샘플의 비율로, 97%이다. 이는 모델이 양성 리뷰(긍정적 리뷰)를 예측할 때 매우 정확하다는 것을 의미한다.
* **재현율 (Recall)** - 실제 양성 샘플 중에서 모델이 양성으로 정확히 예측한 비율로, 99%이다. 이는 모델이 거의 모든 양성 샘플을 잘 잡아낸다는 것을 나타낸다.
* **F1 점수 (F1-score)** - 정밀도와 재현율의 조화 평균으로, 98%이다. 이는 모델이 정밀도와 재현율에서 모두 균형 잡힌 성능을 보이고 있음을 나타낸다
* **ROC-AUC (Receiver Operating Characteristic - Area Under Curve)** - 모델의 분류 능력을 나타내는 지표로, 0.91이다. 1에 가까울수록 모델이 우수하다는 것을 의미합니다. 이 값은 모델이 양성과 음성을 얼마나 잘 구분하는지를 나타낸다.

**4. 분류보고서**

**텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명**

**Class 0 (부정적 리뷰) :**

* **정밀도: 64%** - 부정적 리뷰로 예측한 샘플 중 실제로 부정적인 비율이 64%이다.
* **재현율: 30%** - 실제 부정적 리뷰 중 모델이 부정적으로 예측한 비율이 30%이다..
* **F1 점수: 41%** - 낮은 재현율과 정밀도로 인해 F1 점수가 낮다.
* **지원 (Support):** **23** - 테스트 데이터에서 실제 부정적 리뷰의 수는 23개이다.

**Class 1 (긍정적 리뷰) :**

* **정밀도: 97%** - 긍정적 리뷰로 예측한 샘플 중 실제로 긍정적인 비율이 97%이다.
* **재현율: 99%** - 실제 긍정적 리뷰 중 모델이 긍정적으로 예측한 비율이 99%이다.
* **F1 점수: 98%** - 정밀도와 재현율 모두 높아 F1 점수도 매우 높다.
* **지원 (Support): 453** - 테스트 데이터에서 실제 긍정적 리뷰의 수는 453개이다

.

**전체:**

* **정확도: 96%** - 전체 데이터에서 정확하게 예측한 비율이다.
* **매크로 평균 (Macro Average) :** 각 클래스별 지표의 단순 평균입니다. 클래스 0의 낮은 재현율로 인해 매크로 평균 값이 다소 낮다.
* **가중 평균 (Weighted Average) :** 클래스별 지원 수에 비례한 평균이다. 이는 데이터셋에서 클래스 1이 압도적으로 많기 때문에, 가중 평균이 클래스 1의 성능을 더 잘 반영한다.

**5. 혼동 행렬**

**텍스트, 스크린샷, 도표, 번호이(가) 표시된 사진

자동 생성된 설명**

**6. ROC 곡선**

**텍스트, 라인, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명**

**ROC 곡선**은 왼쪽 위 모서리 쪽으로 급격히 상승한 후, 오른쪽으로 평평하게 이동합니다. 이는. 모델이 매우 높은 True Positive Rate(TPR)을 유지하면서 False Positive Rate(FPR)을 낮게 유지하고 있다는 것을 보여준다. 즉, 대부분의 임계값에서 모델이 양성과 음성을 효과적으로 구분하고 있음을 보여준다.

**AUC (Area Under Curve)** 값이 **0.91**로 나타났습니다. 이 값은 매우 높은 값으로, 모델이 우수한 성능을 발휘하고 있음을 나타낸다. 특히, 모델은 양성과 음성을 91%의 정확도로 구별할 수 있음을 보여준다.

**7. 총평**

모델은 전체적으로 매우 높은 정확도를 보이며, 특히 긍정적 리뷰에 대해 매우 우수한 성능을 보여준다.

부정적 리뷰에 대한 성능은 상대적으로 낮으며, 이는 데이터셋의 클래스 불균형으로 인한 것으로 보인다. 데이터셋의 대부분의 평점이 4~5점으로 상대적으로 1~2점의 데이터가 부족했다.

오버피팅이나 언더피팅의 증거는 보이지 않지만, 클래스 불균형 문제를 해결하기 위해 평점이 낮은 데이터를 추가하는 방향으로 개선할 수 있을 것 같다.

**SVR 회귀 모델 결과 및 분석 :**

**1. 이진 분류 레이블로 변환**

**텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명**

* 위 코드는 SVR회귀 모델의 예측 결과를 이진 분류 문제로 변환하여, 이진 분류 성능 평가 지표를 계산하기 위해 사용한 코드이다.
* **y\_pred\_binary**는 예측된 평점이 3.5 이상이면 1 (긍정적 평가), 그렇지 않으면 0 (부정적 평가)으로 변환된 이진 레이블이 된다. **y\_test\_binary**는 실제 평점이 3.5 이상이면 1, 그렇지 않으면 0으로 변환된 이진 레이블이 된다.
* 결과적으로 평점 3.5를 기준으로 이진 분류 문제로 변환함으로써, 예측 성능을 이진 분류 성능 평가 지표로 평가할 수 있게 된다.

**2. 하이퍼파라미터 튜닝 결과**



**3. 회귀 모델 평가 지표**

텍스트, 폰트, 화이트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* **최적의 교차 검증 MSE : -0.46** - 교차 검증 과정에서 계산된 평균 제곱 오차이다. 음수 값으로 표시되지만, 이는 단순히 계산상의 부호 문제로 실제 MSE는 0.46에 가까울 것으로 보인다. 값이 낮을수록 모델의 예측이 정확하다.
* **평균 제곱 오차** **: 0.48** - 모델의 예측 값과 실제 값 간의 평균 제곱 차이이다. 이 값이 낮을수록 모델의 예측이 실제 값에 가깝다.
* **R² 점수: 0.28** - 모델이 데이터를 얼마나 잘 설명하는지를 나타내는 지표이다. 0.28은 모델이 데이터 변동의 28%를 설명하고 있음을 의미한다. 1에 가까울수록 모델이 잘 설명하고 있는 것이며, 이 경우 0.28은 상대적으로 낮은 설명력을 의미한다.

**4. 분류 모델 평가 지표**

**텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명**

* **정확도 (Accuracy): 0.96 -** 전체 예측 중 올바르게 예측한 비율이다. 여기서는 96%로, 모델이 대부분의 데이터를 정확히 예측했음을 나타낸다.
* **정밀도 (Precision): 0.96** - 모델이 긍정 클래스(Positive Class)로 예측한 샘플 중 실제로 긍정인 샘플의 비율이다. 96%의 정밀도는 모델의 긍정 예측이 매우 정확함을 의미한다.
* **재현율 (Recall): 1.00** - 실제로 긍정 클래스인 샘플 중 모델이 긍정으로 정확하게 예측한 비율이다. 100%의 재현율은 모델이 실제 긍정 샘플을 모두 잡아냈음을 의미한다.
* **F1 점수: 0.98** - 정밀도와 재현율의 조화 평균이다. 정밀도와 재현율 사이의 균형을 평가하는 데 사용된다. 0.98은 매우 높은 값으로, 모델이 정밀도와 재현율에서 모두 우수함을 나타낸다.
* **ROC-AUC 점수: 0.61** - ROC 곡선 아래의 면적이다. 이 값은 모델의 분류 성능을 평가하는데 사용되며, 0.5는 무작위 예측, 1은 완벽한 예측을 의미한다. 0.61은 모델이 분류 작업에서 다소 우수하지만, 탁월하지는 않다는 것을 의미한다.

**5. 혼돈행렬**

텍스트, 스크린샷, 직사각형, 도표이(가) 표시된 사진

자동 생성된 설명

**6. ROC 곡선**

텍스트, 스크린샷, 라인, 도표이(가) 표시된 사진

자동 생성된 설명

ROC 곡선이 왼쪽 상단 코너에 가까울수록 모델의 성능이 더 좋다. 이 그래프에서는 ROC 곡선이 대각선과 비교적 가까운 위치에 있으므로, 성능이 뛰어나지는 않음을 알 수 있다.

AUC 값이 0.61이므로, 모델은 무작위 예측보다는 성능이 약간 우수하다. 하지만, 더 높은 성능을 달성하려면 모델의 개선이 필요하다.

**7. 총평**

평균 제곱 오차가 0.48로 낮은 편이지만, R² 점수가 0.28로 상대적으로 낮다. 이는 모델이 예측하는 값이 실제 값과 어느 정도 가까울 수 있지만, 데이터의 총 변동성을 충분히 설명하지 못하고 있다는 의미이며 이는 언더피팅의 가능성을 시사한다.

분류 모델의 경우, 정밀도와 재현율, F1 점수가 매우 높아 학습 데이터에 대해서는 우수한 성능을 보이고 있다. 그러나 ROC-AUC 점수가 0.61로 낮은 편이다. 이는 임계값을 변화시켜도 모델의 성능이 안정적이지 않을 수 있음을 나타낸다. 높은 정확도와 정밀도, 재현율은 모델이 학습 데이터에 대해 매우 잘 맞추고 있음을 나타내며, 이로 인해 오버피팅의 가능성을 시사한다.

회귀 모델은 언더피팅의 가능성이 크므로, 모델 복잡도를 증가시키고, 추가적인 특징을 사용하거나, 더 많은 데이터를 수집하여 성능을 개선해야 할 것 같다.

분류 모델은 학습 데이터에 대한 성능이 매우 높지만, ROC-AUC 점수가 상대적으로 낮아 오버피팅의 가능성이 있다. 모델의 복잡도를 줄이고, 규제를 추가하거나, 더 많은 데이터로 모델을 학습시키는 방법으로 해결할 수 있을 것 같다.