제품 리뷰 데이터의 감성분석 보고서

202020863 김주현

**보고서 목차**

1. **서론**
   1. 프로젝트 개요
   2. 감성 분석의 중요성 및 응용
2. **데이터 전처리**
   1. 데이터 소개
   2. 데이터 정리 및 전처리 단계
   3. 텍스트 정규화 과정
3. **모델 선택 및 학습**
   1. 사용한 모델 소개
   2. 데이터 벡터화 과정
   3. 모델 학습 과정
4. **모델 평가**
   1. 평가 지표 설명
   2. 모델 성능 결과
5. **결과 시각화 및 해석**
   1. 혼동 행렬
   2. ROC 곡선
   3. 결과 해석
6. **결론**

**1. 서론**

**1) 프로젝트 개요**

본 프로젝트에서는 텍스트 데이터를 감성으로 분류하는 감성 분석 시스템을 개발합니다. 이 프로젝트는 데이터 전처리, 모델 선택, 학습, 평가 및 결과 시각화를 포함합니다.

**2) 감성 분석의 중요성 및 응용**

감성 분석은 텍스트 데이터를 긍정, 부정 또는 중립과 같은 감성으로 분류하는 기술로, 제품 리뷰, 소셜 미디어 게시물 등에서 사람들의 감정을 이해하는 데 널리 사용됩니다. 이는 마케팅 전략 수립, 고객 만족도 분석, 제품 개선 등에 중요한 역할을 합니다.[[1]](#footnote-1)

**2. 데이터 전처리**

**1) 데이터 소개**

본 프로젝트에서는 amazon\_uk\_shoes\_products\_dataset\_2021\_12.csv 데이터를 사용하였습니다. 주요 컬럼으로는 리뷰 텍스트와 리뷰 평점이 있습니다.

**2) 데이터 정리 및 전처리 단계**

데이터 전처리를 통해 데이터를 일관성 있게 하여 모델이 더 잘 학습될 수 있게 했습니다..

1. **데이터 로드 및 확인**: 데이터의 구조와 내용물을 확인하여 전처리 및 분석에 필요한 정보를 파악하기 위해서 CSV 파일을 pands 데이터 프레임으로 불러와 데이터의 구조를 확인하였습니다.
2. **결측값 처리**: 분석의 정확성을 높이고, 불완전한 데이터로 인한 오류를 방지하기 위해서리뷰 텍스트에 결측값이 있는 행을 제거하였습니다.
3. **텍스트 정규화**: 텍스트 데이터를 일관성 있게 만들어 분석을 용이하게 하고, 불필요한 정보를 제거하여 모델 학습 성능을 향상시키기 위해서 텍스트 데이터를 소문자로 변환하고 특수 문자를 제거하여 정리하였습니다.

**3) 감성 레이블 생성**

# 데이터 로드

file\_path = '/mnt/data/amazon\_uk\_shoes\_products\_dataset\_2021\_12.csv'

data = pd.read\_csv(file\_path)

# 결측값 제거

data = data.dropna(subset=['review\_text'])

# 텍스트 정리 함수

def clean\_text(text):

text = re.sub(r'[^a-zA-Z\s]', '', text, re.I|re.A) # 특수 문자 제거

text = text.lower() # 소문자 변환

text = text.strip() # 양쪽 공백 제거

return text

# 리뷰 텍스트 정리

data['cleaned\_review\_text'] = data['review\_text'].apply(clean\_text)

평점 5점만 긍정으로 할까 고민을 했으나 4점까지는 리뷰가 긍정적인 내용이 대부분이었기에 리뷰 평점을 기반으로 평점이 4이상이면 긍정(1), 그렇지 않으면 부정(0)으로 분류했습니다.

data['sentiment'] = data['review\_rating'].apply(lambda x: 1 if x >= 4 else 0)

**3. 모델 선택 및 학습**

**1) 사용한 모델 소개**

로지스틱 회귀(Logistic Regression) 모델을 사용하여 감성 분석을 수행하였습니다. 이 모델은 간단하지만 이진 분류 문제에 적합하며, 텍스트 데이터 분류 문제에서 자주 사용되고 모델의 해석도 용이하기 때문입니다.

**2) 데이터 벡터화 과정**

TF-IDF(Term Frequency-Inverse Document Frequency) 벡터화는 단어의 빈도와 중요도를 모두 고려하여 텍스트 데이터를 효과적으로 수치화할 수 있기에 벡터라이저를 사용하여 텍스트 데이터를 숫자 벡터로 변환하였습니다

# TF-IDF 벡터화

vectorizer = TfidfVectorizer()

X\_train\_tfidf = vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = vectorizer.transform(X\_test)

**3) 모델 학습 과정**

로지스틱 회귀 모델을 사용하여 새로운 데이터를 입력받았을 때 적절한 예측을 할 수 있도록 학습 데이터를 기반으로 모델을 학습시켰습니다.

# 로지스틱 회귀 모델 학습

model = LogisticRegression()

model.fit(X\_train\_tfidf, y\_train)

**4. 모델 평가**

**평가 지표 설명**

모델의 성능을 평가하기 위해 다음 지표를 사용하였습니다:

* **Accuracy (정확도)**: 전체 예측에서 맞게 예측한 비율로써 모델의 전반적인 성능을 평가하는 기본적인 지표로 사용하였습니다. 정확도가 높을 수록 모델이 전체적으로 잘 작동한다는 것입니다.
* **Precision (정밀도)**: 긍정 예측 사례 중 실제 긍정 사례의 비율로써 모델의 예측 정확성을 측정하기 위해 사용하였습니다. 정밀도가 높을 수록 모델이 불필요한 긍정 예측을 적게 한다는 것입니다.
* **Recall (재현율)**: 정밀도와는 반대로 실제 긍정 사례 중 긍정으로 예측한 비율로써 실제 긍정 사례를 찾아내는 모델의 탐지 능력 측정을 위해 사용하였습니다.
* **F1 Score**: 정밀도와 재현율의 균형을 조화 평균을 사용해 평가하여 전반적인 성능을 종합적으로 측정하시 위해 사용하였습니다.
* **ROC-AUC**: ROC 곡선 아래 면적으로 다양한 임계값에서의 모델의 분류 성능을 평가하기 위해 사용하였습니다. ROC-AUC 값이 1에 가까울수록 모델의 분류 능력이 우수하다는 것입니다.

# 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\_prob)

# 평가 결과 출력

print(f'Accuracy: {accuracy}')

print(f'Precision: {precision}')

print(f'Recall: {recall}')

print(f'F1 Score: {f1}')

print(f'ROC-AUC: {roc\_auc}')

모델 평가 결과는 다음과 같이 나왔습니다.

모든 부분이 80%가 넘게 나와 대체적으로 정확하게 예측하고 있으며 재현율이 굉장히 높게 나와 모델이 긍정 클래스를 잘 탐지하고 있다는 것을 알 수 있습니다. 또한 F1 Score와 ROC-AUC도 90%에 가까운 결과가 나옴으로 정밀도와 재현율 사이의 균형도 잘 맞추고 분류 성능도 우수함을 알 수 있었습니다. 솔직히 이렇게 모델 평가가 매우 높게 나온 이유는 이 데이터에 특화된 모델이기 떄문이라는 생각이 듭니다.

Accuracy: 0.8378576669112252

Precision: 0.8402718776550552

Recall: 0.9677103718199609

F1 Score: 0.89949977262392

ROC-AUC: 0.8970364589012402

**5. 결과 시각화 및 해석**

**1) 혼동 행렬**

혼동 행렬을 통해 모델의 예측 결과를 한눈에 파악할 수 있게 하여, 모델의 강점과 약점을 시각적으로 평가하였습니다.

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.savefig('confusion\_matrix.png') // 현재 작업 디렉토리에 저장

또한 plt.show()를 했을 떄 시각화 자료가 보이지 않아 현재 작업 디렉토리에 시각화 자료를 저장하도록 작성하였습니다.

**텍스트, 스크린샷, 직사각형, 도표이(가) 표시된 사진

자동 생성된 설명**

행은 실제 클래스를, 열은 예측된 클래스를 나타냅니다.

* **True Positive (TP)**: 실제 클래스 1, 모델 1인 경우가 989.
* **True Negative (TN)**: 실제 클래스 0, 모델 0인 경우가 153.
* **False Positive (FP)**: 실제 클래스 0, 모델 1인 경우가 188.
* **False Negative (FN)**: 실제 클래스 1, 모델 0인 경우가 33.

이 결과를 토대로 성능 지표를 계산하면

1. 정확도:
2. 정밀도:
3. 재현율:
4. F1 스코어:

모델 평가지표와 동일한 결과를 확인할 수 있습니다. 계산은 손으로 하기 힘들어서 핸드폰 공학계산기를 사용했습니다. 또한 계산하지 않더라도 예측결과가 매우 확실하고 정확함을 시각화한 자료색의 선명함과 숫자로 한눈에 확인가능한 부분이 매력적으로 느껴졌습니다.

**2) ROC 곡선**

ROC 곡선을 통해 다양한 임계값에서의 모델 성능을 평가할 수 있게 하여, 모델의 분류 능력을 종합적으로 평가하였습니다.

마찬가지로 시각화 자료가 현재 작업 디렉토리에 저장될 수 있도록 조치하였습니다.

from sklearn.metrics import roc\_curve

fpr, tpr, \_ = roc\_curve(y\_test, y\_prob)

plt.plot(fpr, tpr, marker='.')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.savefig('roc\_curve.png') // 현재 작업 디렉토리에 저장

텍스트, 스크린샷, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

Y축은 성공적인 긍정리뷰 분류 비율 (True Positive Rate, TPR), X축은 실패한 긍정리뷰 분류 비율 (False Positive Rate, FPR)입니다.

식을 써보면

* 1. 성공적인 예측 분류 비율:
  2. 실패한 예측 분류 비율:

위와 같고 계산은 생략하겠습니다. 시각화한 자료를 보면 ROC곡선의 아래면적 값이 1에 가까울 수록 모델 성능이 우수하다는 것인데 눈으로 봐도 거의 1에 가깝고 실제 모델 평가에서 값이 0. 8970364589012402로 나왔기에 대부분의 경우에서 대체적으로 좋은 분류 성능을 보인다는 걸 알 수 있습니다.

**6. 결론**

본 프로젝트에서는 amazon\_uk\_shoes\_products\_dataset\_2021\_12.csv를 사용하여 감성 분석을 수행하였습니다. 구현한 모델은 모든 부분에서 대체적으로 좋은 결과가 나왔으며 해당 모델을 통해 제품 리뷰의 긍정적인 사례들을 분류하여 비율을 예측해볼 수 있습니다.

1. Cambria, E., Schuller, B., Xia, Y., & Havasi, C. (2013). New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, 28(2), 15-21. [↑](#footnote-ref-1)