제품 review 데이터의 감성 분석

- 2024학년도 기말고사 대체과제 -

|  |  |
| --- | --- |
| **제출일** | **2024. 6. 10. 토요일** |
| **과목명** | **기계학습(8585)** |
| **교수명** | **백우진 교수님** |
| **제출자** | **문헌정보학과 202120608 김고은** |

**프로젝트 개요**

|  |  |  |  |
| --- | --- | --- | --- |
| | 1 | 감성 분석의 기본 이해 | --------------------------------- | 0 3 |
| | 2 | 데이터 전처리 | --------------------------------- | 0 3 |
|  | 2.1 필요 Library 및 Data 로드 | --------------------------------- | 0 4 |
|  | 2.2 영문 텍스트 데이터 전처리 | --------------------------------- | 0 5 |
|  | 2.3 영문 형태소 분석 | --------------------------------- | 0 5 |
|  | 2.4 영문 불용어 제거 | --------------------------------- | 0 6 |
|  | 2.5 Word Count 데이터 빈도 분석 | --------------------------------- | 0 7 |
|  | 2.6 TF-IDF 적용 | --------------------------------- | 0 8 |
| | 3 | 모델 선택 및 학습 | --------------------------------- | 0 9 |
|  | 3.1 데이터셋 생성 | --------------------------------- | 0 9 |
|  | 3.2 Training set / Test set 나누기 | --------------------------------- | 1 0 |
|  | 3.3 모델 학습 | --------------------------------- | 1 0 |
| | 4 | 모델 평가 | --------------------------------- | 1 0 |
|  | 4.1 샘플링 재조정 | --------------------------------- | 1 1 |
|  | 4.2 모델 재학습 | --------------------------------- | 1 2 |
| | 5 | 시각화 및 해석 | --------------------------------- | 1 2 |
|  | 5.1 ROC 곡선 생성 | --------------------------------- | 1 4 |

**1. 감성 분석의 기본 이해**

감성 분석은 텍스트 데이터를 분석하여 데이터 속 감정적 어조가 긍정인지 부정인지 또는 부정인지 확인하는 분석 기법이다. 오피니언 마이닝이라고도 하며 감성 분석 도구는 이 텍스트를 스캔하여 주제에 대한 글쓴이의 태도를 자동으로 확인할 수 있다. 기업이 감성 분석을 활용하는 이유는 실제적이고 구체적인 고객 피드백을 기반으로 제품과 서비스를 개선하기 위함이다. 또한, 이메일 및 설문조사, 제품 피드백과 같은 대규모 데이터에서 클라우드 기반 감정 분석 도구를 통해 고객의 감정을 파악할 수 있기 때문에 고객 기반 개선방향을 잡기에도 도움이 된다.

이번 프로젝트는 이러한 감성분석을 활용하여 와인 리뷰 데이터에서 감성을 분류하는 시스템을 개발하는 것에 초첨을 둔다. 전체 프로젝트는 데이터 전처리 및 모델 선택, 학습, 평가 및 결과 시각화를 포함한다.

2. 데이터 전처리

데이터 전처리는 데이터를 분석 및 처리하기 전, 적합한 형태로 만드는 과정이다. 데이터 분석을 하기에 전처리는 다음과 같은 단계를 거친다.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 데이터 수집 | → | 데이터 정제 | → | 데이터 통합 | → | 데이터 축소 | → | 데이터 변환 |

전처리 예시로는 중복값 제거, 결측값 보정, 데이터 연계/통합, 노이즈 제거, 데이터 구조 변경(차원 변경), 데이터 벡터화, outlier detection, feature Engineering 등이 있다.

이러한 예시를 바탕으로 ‘wine\_review.csv’를 활용하여 감성 분류 예측모델을 작성한다.

2.1 필요 Library 및 Data 로드

먼저, 사용할 라이브러리와 데이터를 로드한다.

|  |
| --- |
| *# 사용할 Library 로드*  **import** pandas **as** pd  **import** numpy **as** np  **import** re  **from** nltk.corpus **import** stopwords  **from** nltk.tokenize **import** word\_tokenize  **from** nltk.stem **import** WordNetLemmatizer  **from** sklearn.model\_selection **import** train\_test\_split  **from** sklearn.feature\_extraction.text **import** TfidfVectorizer  **from** sklearn.naive\_bayes **import** MultinomialNB  **from** sklearn.svm **import** SVC  **from** sklearn.linear\_model **import** LogisticRegression  **from** sklearn.ensemble **import** RandomForestClassifier  **from** sklearn.metrics **import** accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score, confusion\_matrix, roc\_curve, auc  **import** matplotlib.pyplot **as** plt  **import** seaborn **as** sns  **import** nltk |
| *# 데이터 로드*  df **=** pd**.**read\_csv("wine\_review.csv") |
| *# 데이터 확인*  df**.**head() |
| df**.**info() |

데이터를 확인했을 때, **rating**가 1에서 5로 이루어진 이용자 리뷰의 평가 점수라는 것과, **review\_text**가 이용자 리뷰 평가 내용이라는 것을 확인할수 있다. 다음으로 데이터셋을 확인한다.

|  |
| --- |
| *# dimension*  df**.**shape |
| *# 결측치 확인*  df**.**isnull()**.**sum() |

데이터 프레임의 크기를 확인하고 결측치를 확인한다.

2.2 영문 텍스트 데이터 전처리

감성 분석에 필요없는 단어가 존재하는 텍스트 데이터를 제거하기 위해 단어 단위로 분리하는 전처리 과정 필요하다. 리뷰 평가 내용 단어화하여 형태소 추출 및 TF-IDF 변환한다.

|  |
| --- |
| *# text 변수 확인*  df['reviews.text'][0] |
| *# 정규 표현식*  **import** re  **import** pandas **as** pd  **def** apply\_regular\_expression(text):  *# 영어 단어와 공백만을 추출하는 정규 표현식*  pattern **=** re**.**compile(r'[a-zA-Z\s]+')  *# 매칭된 부분만 추출하여 소문자로 변환*  matches **=** pattern**.**findall(text)  cleaned\_text **=** ' '**.**join(matches)**.**lower()  **return** cleaned\_text  *# NaN 값을 빈 문자열로 대체*  df['reviews.text'] **=** df['reviews.text']**.**fillna("") |
| *#결측값 처리*  df **=** df**.**dropna(subset**=**['reviews.text', 'reviews.rating']) |
| apply\_regular\_expression(df['reviews.text'][0]) |
| *# 전체 데이터프레임에 대해 함수를 적용*  df['reviews.text'] **=** df['reviews.text']**.**apply(apply\_regular\_expression) |

2.3 영문 형태소 분석

정규 표현식을 적용한 후 필요 없는 부분을 제거한 것을 확인했다면 형태소를 분석한다. 형태소를 분석하기 위해 영문 자연어 처리 라이브러리인 NLTK를 활용한다.

**NLTK** : 자연어 처리 라이브러리로, 형태소 분석을 위해 품사 태깅 사용 가능 NLTK를 사용하여 영문 텍스트에서 명사, 동사, 형용사, 부사 추출

|  |
| --- |
| pip install nltk |
| **import** nltk  **from** nltk.tokenize **import** word\_tokenize  **import** string |
| *#nltk 자원 다운로드*  **import** nltk  nltk**.**download('punkt')  nltk**.**download('averaged\_perceptron\_tagger')  nltk**.**download('stopwords') |
| *# 명사, 동사, 형용사, 부사 형태소 추출을 위한 함수*  **def** extract\_words\_with\_tags(text):  *# 텍스트를 토큰화하고 품사 태깅*  tokens **=** word\_tokenize(text)  tagged\_tokens **=** nltk**.**pos\_tag(tokens)  *# 명사, 동사, 형용사, 부사 추출*  words **=** [word **for** word, pos **in** tagged\_tokens **if** pos**.**startswith('N') **or** pos**.**startswith('V') **or** pos**.**startswith('J') **or** pos**.**startswith('R')]  **return** words |
| *# 예시 데이터의 첫 번째 리뷰 텍스트에 대해 전처리 및 필터링된 단어 추출*  cleaned\_text **=** apply\_regular\_expression(df['reviews.text'][0])  words **=** extract\_words\_with\_tags(cleaned\_text)  print(words)  *# 전체 리뷰 텍스트 말뭉치(corpus) 생성*  corpus **=** " "**.**join(df['reviews.text']**.**tolist())  cleaned\_corpus **=** apply\_regular\_expression(corpus) |
| *# 정규 표현식 적용*  corpus **=** apply\_regular\_expression(corpus) |
| *# 전체 말뭉치에서 명사, 동사, 형용사, 부사 추출*  words\_corpus **=** extract\_words\_with\_tags(corpus) |
| *# 빈도탐색*  **from** collections **import** Counter  *# 단어 빈도 계산*  counter **=** Counter(words\_corpus) |
| counter**.**most\_common(10) |
| *#한 글자 문자 제거*  available\_counter **=** Counter({x: counter[x] **for** x **in** counter **if** len(x) **>** 1})  available\_counter**.**most\_common(10) |

2.4 영문 불용어 제거

영문 텍스트의 형태소를 분석했다면 분석에 큰 의미가 없거나 정확도를 낮출 수 있는 단어를 NLTK의 stopwords 모듈을 사용하여 제거한다.

|  |
| --- |
| **import** nltk  **from** nltk.corpus **import** stopwords  **from** nltk.tokenize **import** word\_tokenize  *# 필요한 NLTK 데이터를 다운로드*  nltk**.**download('punkt') *# 토큰화를 위해 필요*  nltk**.**download('stopwords') *# 불용어를 제거하기 위한 데이터* |
| *# 영어 불용어 리스트*  stop\_words **=** set(stopwords**.**words('english')) |
| *# 불용어를 제거하는 함수*  **def** remove\_stopwords(text):  stop\_words **=** set(stopwords**.**words('english')) *# NLTK의 영어 불용어 리스트*  tokens **=** word\_tokenize(text) *# 토큰화*  filtered\_tokens **=** [word **for** word **in** tokens **if** word**.**lower() **not** **in** stop\_words]  filtered\_text **=** ' '**.**join(filtered\_tokens)  **return** filtered\_text |
| *# 전체 리뷰 텍스트에 정규 표현식 적용*  df['cleaned\_text'] **=** df['reviews.text']**.**apply(apply\_regular\_expression)  *# 불용어 제거*  df['cleaned\_text'] **=** df['cleaned\_text']**.**apply(remove\_stopwords)  *# 전체 말뭉치에서 명사, 동사, 형용사, 부사 추출*  df['filtered\_words'] **=** df['cleaned\_text']**.**apply(extract\_words\_with\_tags)  *# 결과 확인*  df['filtered\_words'] |

2.5 Word Count 데이터 빈도 분석

추출된 텍스트 데이터를 데이터 빈도 분석 한다.

|  |
| --- |
| **import** pandas **as** pd  **from** sklearn.feature\_extraction.text **import** CountVectorizer  **from** nltk.corpus **import** stopwords  **from** nltk.tokenize **import** word\_tokenize |
| *# 불용어를 제거하는 함수*  **def** remove\_stopwords(text):  stop\_words **=** set(stopwords**.**words('english')) *# NLTK의 영어 불용어 리스트*  tokens **=** word\_tokenize(text) *# 토큰화*  filtered\_tokens **=** [word **for** word **in** tokens **if** word**.**lower() **not** **in** stop\_words]  filtered\_text **=** ' '**.**join(filtered\_tokens)  **return** filtered\_text |
| *# 전체 리뷰 텍스트에 정규 표현식 적용*  df['cleaned\_text'] **=** df['reviews.text']**.**apply(apply\_regular\_expression) |
| *# 불용어 제거*  df['cleaned\_text'] **=** df['cleaned\_text']**.**apply(remove\_stopwords) |
| *# CountVectorizer를 사용하여 BoW 벡터 생성*  vect **=** CountVectorizer(stop\_words**=**'english') *# 'english' 문자열을 사용하여 사전 정의된 영어 불용어 리스트 사용*  bow\_vect **=** vect**.**fit\_transform(df['cleaned\_text'])  word\_list **=** vect**.**get\_feature\_names\_out()  count\_list **=** bow\_vect**.**toarray()**.**sum(axis**=**0)  *# 결과 확인*  print("Word list:", word\_list)  print("BoW vector:\n", bow\_vect**.**toarray()) |
| *# 단어 리스트*  word\_list |
| *# 각 단어가 전체 리뷰중에 등장한 총 횟수*  count\_list |
| *# 각 단어의 리뷰별 등장 횟수*  bow\_vect**.**toarray() |
| *#CountVectorizer로 변환된 BoW벡터의 형태(shape)를 확인*  bow\_vect**.**shape |
| *# "단어" - "총 등장 횟수" Matching*  word\_count\_dict **=** dict(zip(word\_list, count\_list)) |

2.6 TF-IDF 적용

TF-IDF 변환과 Bow 벡터 TF-IDF 변환을 진행한다.

|  |
| --- |
| from sklearn.feature\_extraction.text import TfidfTransformer  tfidf\_vectorizer = TfidfTransformer()  tf\_idf\_vect = tfidf\_vectorizer.fit\_transform(bow\_vect) |
| print(tf\_idf\_vect.shape)  #변환 수 2445\*4749 matrix 출력 |
| # 첫 번째 리뷰에서의 단어 중요도(TF-IDF 값) -- 0이 아닌 것만 출력  print(tf\_idf\_vect[0]) |
| # 첫 번째 리뷰에서 모든 단어의 중요도 -- 0인 값까지 포함  print(tf\_idf\_vect[0].toarray().shape)  print(tf\_idf\_vect[0].toarray()) |

벡터-단어를 mapping 시킨다.

|  |
| --- |
| vect.vocabulary\_=vect.vocabulary\_ |
| invert\_index\_vectorizer = {v: k for k, v in vect.vocabulary\_.items()}  print(str(invert\_index\_vectorizer)[:100]+'...') |

3. 모델 선택 및 학습

전처리한 리뷰 데이터 활용하여 감성 분류 예측 모델 제작한다.

X 값(feature 값): 이용자의 리뷰 평가 내용 Y값(label 값) 이용자의 긍/부정 감성을 분류하도록 한다.

3.1 데이터셋 생성

rating 값에 있는1~5의 지표를 활용하여 긍정/부정 점수 value 설정한다.

|  |
| --- |
| df**.**head() |
| df **=** df[['reviews.rating','reviews.text']]  df**.**head(30) |
| df['reviews.rating']**.**hist() |
| **def** rating\_to\_label(rating):  **if** rating **>** 3:  **return** 1  **else**:  **return** 0    df['y'] **=** df['reviews.rating']**.**apply(**lambda** x: rating\_to\_label(x)) |
| df**.**head() |
| df["y"]**.**value\_counts()  *#모델의 Feature 변수는 리뷰에서 추출된 형태소와 그들의 중요도를 나타나는 tf\_idf\_vect로 대체* |

3.2 Training set / Test set 나누기

|  |
| --- |
| **from** sklearn.model\_selection **import** train\_test\_split  x **=** tf\_idf\_vect  y **=** df['y']  x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size **=** 0.3, random\_state**=**1) |
| x\_train**.**shape, y\_train**.**shape |
| x\_test**.**shape, y\_test**.**shape |

3.3 모델 학습

Logistic Regression 모델을 학습 학습시킨다.

|  |
| --- |
| **from** sklearn.linear\_model **import** LogisticRegression  **from** sklearn.metrics **import** accuracy\_score, precision\_score, recall\_score, f1\_score  *# fit in training set*  lr **=** LogisticRegression(random\_state **=** 0)  lr**.**fit(x\_train, y\_train)  *# predict in test set*  y\_pred **=** lr**.**predict(x\_test) |

4. 모델 평가

|  |
| --- |
| *# 테스트 세트의 분류 결과*  print('accuracy: %.2f' **%** accuracy\_score(y\_test, y\_pred))  print('precision: %.2f' **%** precision\_score(y\_test, y\_pred))  print('recall: %.2f' **%** recall\_score(y\_test, y\_pred))  print('F1: %.2f' **%** f1\_score(y\_test, y\_pred)) |
| *# confusion matrix*  conf\_matrix **=** confusion\_matrix(y\_test, y\_pred)  plt**.**figure(figsize**=**(10, 7))  sns**.**heatmap(conf\_matrix, annot**=True**, fmt**=**'d', cmap**=**'Blues')  plt**.**xlabel('Predicted')  plt**.**ylabel('Actual')  plt**.**title('Confusion Matrix')  plt**.**show() |

텍스트, 스크린샷, 직사각형, 도표이(가) 표시된 사진

자동 생성된 설명

* 실제 클래스 0, 예측 클래스 0: 0 (True Negative, TN)
* 실제 클래스 0, 예측 클래스 1: 53 (False Positive, FP)
* 실제 클래스 1, 예측 클래스 0: 0 (False Negative, FN)
* 실제 클래스 1, 예측 클래스 1: 681 (True Positive, TP)

모델 평과 결과를 살펴볼 때 지나치게 긍정으로면 예측하는 경향이 있는 것을 볼 수 있다. 부정 감정에 대한 예측 정확도가 매우 낮기 때문에 '클래스 불균형 조정'을 시도하고자 한다.

4.1 샘플링 재조정

|  |
| --- |
| **import** pandas **as** pd  **from** sklearn.feature\_extraction.text **import** TfidfVectorizer  **from** sklearn.model\_selection **import** train\_test\_split |
| *# tf-idf 벡터 생성 (실제 tf\_idf\_vect가 이미 존재한다고 가정)*  tfidf\_vect **=** TfidfVectorizer()  tf\_idf\_vect **=** tfidf\_vect**.**fit\_transform(df['reviews.text']) |
| *# 'y' 값이 1인 인덱스와 0인 인덱스를 적절히 샘플링*  positive\_random\_idx **=** df[df['y'] **==** 1]**.**sample(min(len(df[df['y'] **==** 1]), 100), random\_state**=**12)**.**index**.**tolist()  negative\_random\_idx **=** df[df['y'] **==** 0]**.**sample(min(len(df[df['y'] **==** 0]), 100), random\_state**=**12)**.**index**.**tolist() |
| *# 샘플링한 인덱스들을 합침*  random\_idx **=** positive\_random\_idx **+** negative\_random\_idx |
| *# 인덱스가 범위 내에 있는지 확인*  print(f"random\_idx: {random\_idx}")  print(f"tf\_idf\_vect.shape: {tf\_idf\_vect**.**shape}") |
| *# 범위 내의 인덱스만 선택*  valid\_random\_idx **=** [idx **for** idx **in** random\_idx **if** idx **<** tf\_idf\_vect**.**shape[0]] |
| *# x와 y를 인덱스를 기준으로 선택*  x **=** tf\_idf\_vect[valid\_random\_idx]  y **=** df['y']**.**iloc[valid\_random\_idx]**.**values |
| *# train/test 데이터셋 분리*  x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size**=**0.25, random\_state**=**1) |
| print(f"x\_train shape: {x\_train**.**shape}")  print(f"x\_test shape: {x\_test**.**shape}")  print(f"y\_train shape: {y\_train**.**shape}")  print(f"y\_test shape: {y\_test**.**shape}") |

4.2 모델 재학습

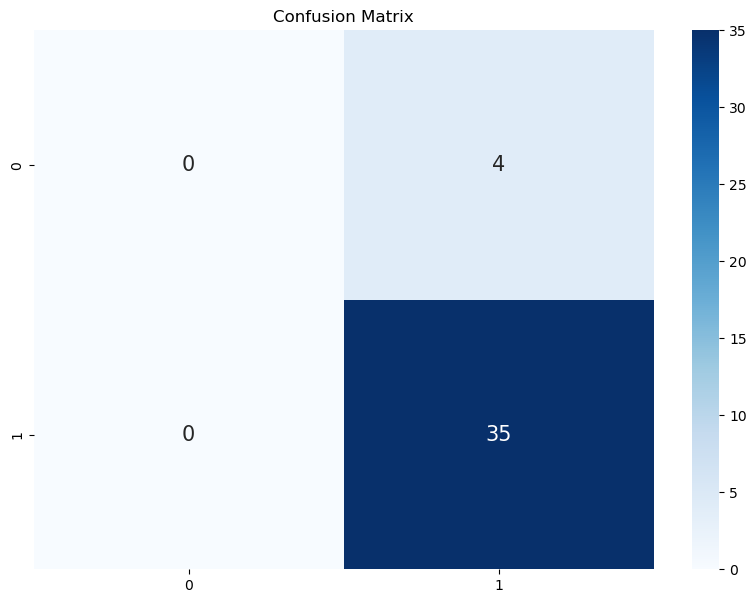
샘플링 재조정한 후 모델을 재학습 시킨다.

|  |
| --- |
| lr2 **=** LogisticRegression(random\_state **=** 0)  lr2**.**fit(x\_train, y\_train)  y\_pred **=** lr2**.**predict(x\_test) |
| *# classification result for test set*  print('accuracy: %.2f' **%** accuracy\_score(y\_test, y\_pred))  print('precision: %.2f' **%** precision\_score(y\_test, y\_pred))  print('recall: %.2f' **%** recall\_score(y\_test, y\_pred))  print('F1: %.2f' **%** f1\_score(y\_test, y\_pred)) |

5. 시각화 및 해석

학습한 모델을 시각화 한다.

|  |
| --- |
| *# confusion matrix*  **from** sklearn.metrics **import** confusion\_matrix  confu **=** confusion\_matrix(y\_true **=** y\_test, y\_pred **=** y\_pred)  plt**.**figure(figsize**=**(10, 7))  sns**.**heatmap(confu, annot**=True**, annot\_kws**=**{'size':15}, cmap**=**'Blues', fmt**=**'.10g')  plt**.**title('Confusion Matrix')  plt**.**show() |



* 실제 클래스 0, 예측 클래스 0: 0 (True Negative, TN)
* 실제 클래스 0, 예측 클래스 1: 4 (False Positive, FP)
* 실제 클래스 1, 예측 클래스 0: 0 (False Negative, FN)
* 실제 클래스 1, 예측 클래스 1: 35 (True Positive, TP)

혼동 행렬의 기존 모델 평가 결과와 샘플링 재조정 후의 변화가 생긴 것을 볼 수 있다.

샘플링 전의 모델은 실제 음성 사례를 전혀 감지하지 못하고 모든 사례를 양성으로 예측하는 경향이 보였다. 이로 인해 모든 실제 양성 사례는 정확하게 예측되었지만, 음성 예측 사례는 잘못 예측 되었다.

샘플링 재조정을 통해 균형잡힌 모들의 예측을 볼 수 있다. FP가 53에서 4로 감소 되었고, 양성 사례도 대부분 정확하게 예측 된 것을 볼 수 있다.

5.1 ROC 곡선 생성

|  |
| --- |
| *# 모델 학습*  model **=** LogisticRegression()  model**.**fit(x\_train, y\_train) |
| *# 예측 확률 계산*  y\_probs **=** model**.**predict\_proba(x\_test)[:, 1] |
| *# ROC 곡선 계산*  fpr, tpr, \_ **=** roc\_curve(y\_test, y\_probs)  roc\_auc **=** auc(fpr, tpr) |
| *# ROC 곡선 시각화*  plt**.**figure()  plt**.**plot(fpr, tpr, color**=**'darkorange', lw**=**2, label**=**f'ROC curve (area = {roc\_auc:.2f})')  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 (ROC)')  plt**.**legend(loc**=**'lower right')  plt**.**show() |

텍스트, 라인, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명

ROC곡선은 이진 분류 모델의 성능을 평가하는 데 사용된다. 그래프를 통해 이진 분류 모델의 AUC가 0.80으로 비교적 좋은 성능을 나타내고 있음을 확인할 수 있다.