**기계학습 기말과제 보고서**

**담당교수 : 백우진 교수님**

**이름 : 장승훈**

**학번 : 201921037**

**전공 : 소프트웨어전공**

1. **감성분석의 기본이해**

감성분석(Sentiment Analysis)은 자연어 처리(Natural Language Processing, NLP) 분야에서 중요한 기술입니다. 이 기술은 텍스트 데이터의 감성과 의견을 자동으로 분류하는 데 사용됩니다. 감성 분석은 인공지능과 기계학습 알고리즘을 이용하여 텍스트로 표현된 감정 정보를 추출하고 긍정적, 부정적, 또는 중립적으로 분류합니다. 이를 통해 기업과 개인은 고객의 반응과 느낌을 파악하여 더 나은 비즈니스 전략을 수립하고 서비스 품질을 개선하는 데에 도움을 받습니다.

1. **데이터 전처리**

df = pd.read\_csv('/content/amazon\_uk\_shoes\_products\_dataset\_2021\_12.csv')

데이터를 Pandas 데이터프레임으로 로드하고

print(f"Total number of reviews: {df.shape[0]}")

print(f"Dataset Columns: {df.columns}")  
print(df.info())

print(df.head())

데이터프레임의 리뷰수, 열, 행, 등의 정보들을 확인합니다.

# 중복 데이터 확인

dup = df.duplicated(subset=['uniq\_id'], keep=False)

# 중복된 행의 개수 확인

count\_dup = dup.sum()

print(f"Number of duplicate rows: {count\_dup}")

중복된 행의 개수가 364개 임을 확인합니다.

# 중복 제거 후 행의 수 확인

df\_unique = df.drop\_duplicates(subset=['uniq\_id'], keep='first')

print(f"# of rows after removing duplicates: {len(df\_unique)}")

중복 제거 후 행의 수가 6641개임을 확인합니다.

# 각 열에서 결측값의 개수를 계산

print(f"Number of missing values in 'review\_title': {df\_unique['review\_title'].isnull().sum()}")

print(f"Number of missing values in 'review\_text': {df\_unique['review\_text'].isnull().sum()}")

print(f"Number of missing values in 'review\_rating': {df\_unique['review\_rating'].isnull().sum()}")

각 열의 결측값의 개수를 계산합니다.  
review\_title :1 , review\_text:8, review\_rating: 0 임을 확인합니다.

# 결측값 제거 후 남은 행의 개수 확인

df\_cleaned = df\_unique.dropna(subset=["review\_text", "review\_title"])

print(f"Number of rows after removing missing values: {len(df\_cleaned)}")

결측값 제거 후 남은 행의 개수가 6641-9 = 6632개임을 확인합니다.

# 데이터 전처리

nltk.download('stopwords')

nltk.download('wordnet')

stop\_words = set(stopwords.words('english'))

lemmatizer = WordNetLemmatizer()

special\_chars = r'!"#$%&\'()\*+,-./:;<=>?@[\\]^\_`{|}~'

# 텍스트 전처리 함수

def preprocess\_text(text):

    # 소문자로 변환

    text = text.lower()

    # 특수문자 제거

    text = re.sub(f'[{special\_chars}]', '', text)

    # 토큰화

    tokens = nltk.word\_tokenize(text)

    # 표제어 추출

    tokens = [lemmatizer.lemmatize(word) for word in tokens if word.isalpha() and word not in stop\_words]

    return ' '.join(tokens)

# 전처리

df\_cleaned['cleaned\_review\_text'] = df\_cleaned['review\_text'].apply(preprocess\_text)

앞서 결측값 처리를 하였으니 모델 성능 향상과 정확한 분석을 위해

노이즈 제거, 토큰화 및 불용어 제거, 정규화를 진행합니다.

1. **모델 선택 및 학습**

# VADER 감성 분석기 초기화

vader\_sentiment = SentimentIntensityAnalyzer()

# 감성 점수 계산 함수 선언

def calc\_sentiment(review):

    return vader\_sentiment.polarity\_scores(review)["compound"]

# 데이터 전처리 및 감성 점수 추가

df\_unique = df.drop\_duplicates(subset=['uniq\_id'], keep='first')

df\_cleaned = df\_unique.dropna(subset=["review\_text", "review\_title"])

df\_cleaned['cleaned\_review\_text'] = df\_cleaned['review\_text'].apply(preprocess\_text)

df\_cleaned["text\_compound"] = df\_cleaned["cleaned\_review\_text"].apply(calc\_sentiment)

VADER 감성 분석기를 초기화하고, 감성 점수를 계산하는 함수를 선언한 후

텍스트의 감성을 분석하여 compound 점수를 반환합니다.

# 리뷰 평점을 이진 감성 레이블로 변환 (긍정/부정, 임계값: 4)

df\_cleaned['sentiment'] = df\_cleaned['review\_rating'].apply(lambda x: 'positive' if x >= 4 else 'negative')

y = df\_cleaned['sentiment']

평점이 4 이상인 경우 긍정, 그렇지 않은 경우 부정으로 분류합니다.

# TF-IDF 벡터화

vectorizer = TfidfVectorizer(max\_features=2000)

X\_tfidf = vectorizer.fit\_transform(df\_cleaned['cleaned\_review\_text']).toarray()

# 감성 점수와 TF-IDF 결합

X\_sentiment = df\_cleaned[['text\_compound']].values

X\_combined = np.hstack((X\_tfidf, X\_sentiment))

전처리된 텍스트 데이터를 TF-IDF 벡터화하여 수치 벡터로 변환하고 각 단어의 중요도를 반영합니다.

TF-IDF와 감성점수를 결합하여 행렬을 생성하고 모델이 텍스트 내용과 감성을 동시에 학습할 수 있게 합니다.

# 데이터를 학습 세트와 테스트 세트로 분할

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 로지스틱 회귀와 그리드 서치

model = LogisticRegression()

parameters = {'C': [0.1, 1, 10], 'penalty': ['l2'], 'solver': ['lbfgs']}

grid\_search = GridSearchCV(model, parameters, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

데이터셋을 학습 및 테스트 세트로 분할하고 그리드 서치를 이용하여 하피퍼파라미터 튜닝을 하며 5겹 교차 검증을 통해 각 하이퍼파라미터의 성능을 평가하고 최적의 하이퍼파라미터를 선택합니다.

1. **모델 평가**

Accuracy: 0.8568198944988696

Precision: 0.8877455565949486

Recall: 0.9313052011776252

F1 Score: 0.9090038314176245

ROC-AUC: 0.8850652536864508

정확도 (Accuracy): 약 0.85

대부분의 경우에 대해 올바르게 예측했지만 살짝 아쉬운 수치입니다.

정밀도 (Precision) : 약 0.88

대부분의 경우에 대해 올바르게 예측 했음을 알 수 있습니다.

재현율 (Recall) : 약 0.93

90% 이상의 높은 수치로, 대부분의 경우에 대해 올바르게 예측 했음을 알 수 있습니다.

F1 점수 (F1 Score) : 약 0.90

이 역시 높은 수치로, 우수한 성능을 보였음을 의미합니다.

ROC-AUC 점수 (ROC-AUC) : 약 0.88

꽤 높은 수치로 양성과 음성을 잘 구분하는것으로 보입니다.

**시각화**

텍스트, 스크린샷, 도표, 직사각형이(가) 표시된 사진

자동 생성된 설명

True Negative (TN): 188

모델이 실제 부정 리뷰를 부정으로 정확히 예측한 수

False Positive (FP): 120

모델이 실제 부정 리뷰를 긍정으로 잘못 예측한 수

False Negative (FN): 70

모델이 실제 긍정 리뷰를 부정으로 잘못 예측한 수

True Positive (TP): 949

모델이 실제 긍정을 긍정으로 정확히 예측한 수

텍스트, 라인, 그래프, 도표이(가) 표시된 사진

자동 생성된 설명

ROC 곡선이 대각선 기준선보다 위에 위치할수록 더 우수한 모델임을 뜻하는데, 이 모델이 대각선보다 위에 위치하여좋은 성능을 낸다는 것을 알 수 있습니다. AUC는 0.89로, 모델이 양성과 음성을 잘 구분하고 있음을 뜻합니다.