머신러닝 기반 외식 서비스(식당) 리뷰 감성 분석 및 만족도 지수 개발

AI융합학부 20243305 하정훈

1. 문제 정의

가. 최종 목표

본 프로젝트는 네이버 지도 등 지도 서비스에서 카페, 식당 등의 외식 장소를 선택할 때 사용자가 겪는 현실적인 불편함에서 시작한다.

1) 샘플링 편향

일본에서 방문한 별점 4.3의 소고기 식당에서 불친절한 서비스를 경험한 적이 있다. 구글 리뷰를 살펴보니 서비스에 불만을 토로하는 외국인의 후기가 있었다. 한편, 3,000여 개의 후기를 보유한 경기도의 한 식당은 상위 노출된 몇몇의 긍정적인 후기를 제외하고 부정적인 리뷰가 많은 것을 확인할 수 있다.

이처럼 사용자는 평균 별점과 상위에 노출된 몇 개의 리뷰만을 확인하게 되므로 예상과 다른 부정적인 경험을 하는 경우가 있다. 이는 전체적인 문맥을 반영하지 못하는 샘플링 시스템의 문제를 보여준다.

2) 정보 과부하

사용자는 수천~수만 개의 리뷰 텍스트를 모두 읽고 장소의 실제 만족도를 파악할 수 없다.

3) 객관성 확보의 어려움

기존 시스템은 단순 입력한 별점의 평균을 보여준다. 평균에는 반어법 등의 사용자의 주관, 클릭 실수 등의 사용자의 실수가 반영될 수 있다. 또한, 긍정적인 후기에도 부정적인 문맥은 숨어있을 수 있으며, 부정적인 후기에도 긍정적인 문맥이 숨어있을 수 있다. 긍정(1)과 부정(0)의 이진 분류 상황에서, 텍스트의 문맥을 확률 기반으로 파악하여 실제 문맥이 담고 있는 긍/부정 정도를 파악할 수 있는 척도가 필요하다.

이러한 별점과 실제 경험의 괴리 문제를 머신러닝과 자연어처리로 해결하는 것을 목표로 한다. 단순 별점 입력식 시스템의 평균 별점이 놓칠 수 있는 텍스트 문맥을 재평가 및 정량화하여 사용자의 의사결정을 돕는 신뢰할 수 있는 지표를 제시할 수 있다.

나. 연구 가설

1) 별점은 일부 노이즈를 포함하지만, 대규모 데이터 관점에서 리뷰 텍스트 본문의 긍/부정 문맥과는 매우 강한 양의 상관관계를 가질 것이다. 리뷰 텍스트 본문의 긍/부정 레이블 데이터로서 별점을 활용할 수 있다.
2) 단순 단어 빈도보다 문맥 정보를 포함했을 때, 감성 분석의 성능이 향상될 것이다.
3) TF-IDF와 같은 방식보다 BERT와 같은 딥러닝 임베딩을 특징 추출기를 사용하고 머신러닝 분류기에 결합한 하이브리드 모델이 텍스트 감성 분류에서 더 높은 성능을 보일 것이다.
4) 특정 도메인(외식 서비스, 인도, 영어 사용)에서 학습된 모델은 타 플랫폼(Yelp)과 타 언어(일본어) 데이터에서도 유의미한 일반화 성능을 보일 것이다.

다. 연구 문제 설정

1) Kaggle의 대규모 다국어(영어/일본어) 텍스트를 머신러닝이 학습할 수 있도록 어떤 정제·전처리 전략을 세울 것인가?(인코딩 문제, 별점 형식 통일 문제, 데이터 병합 문제 등)
2) 별점 3점과 같은 모호한 데이터는 어떻게 처리할 것인가?

3) 텍스트와 별점이 일치하지 않는 노이즈 데이터는 어떻게 처리할 것인가?
4) 텍스트를 벡터로 변환하는 주요 방식(TF-IDF, 딥러닝 임베딩 등) 간 성능 차이는 어떠한가?
5) 다양한 머신러닝 분류기 중 어떤 모델이 텍스트 기반 감성 분류에 가장 적합한가?
6) 최종 모델을 활용하여 별점보다 객관적인 텍스트 기반 만족도 지수를 산출할 수 있는가?

라. 머신러닝 Task

1) 지도학습 기반의 이진 분류

리뷰 텍스트 데이터를 입력받아 긍정(1) 또는 부정(0)일 확률을 예측하고, 이를 1~5점 척도로 변환하여 보정된 점수를 산출한다.

2. 데이터 정의

가. 데이터 확보 전략

1) 영어권 외식 서비스 리뷰 감성 분석

Zomato Bangalore Restaurants 데이터셋을 메인 분석 대상으로 선정하였다. 해당 데이터셋은 영어권 외식 서비스 리뷰 모델의 학습용 데이터로 사용되어, 약 5만 건 이상의 풍부한 데이터셋을 기반으로 모델 학습에 최적화되었다.

기존에 Yelp Restaurant Reviews 데이터셋을 사용하기로 계획하였다. 데이터셋의 데이터 양은 충분해 보였으나, 식당 이름을 기준으로 식당의 개수를 세어본 결과 49개의 식당만이 포함된 데이터였다. 따라서, 해당 데이터셋이 아닌 Yelp Dataset을 활용하기로 하였다. Yelp 사이트의 모든 리뷰 데이터를 담고 있어, 식당 리뷰만 필터링하여 사용하였다. 언어 인코딩은 영어로 한정하였다. Yelp Dataset을 검증용 데이터로 활용하여 모델의 신뢰성을 확인하였다.

2) 일어권 외식 서비스 리뷰 감성 분석

Tokyo Restaurant Reviews on Tabelog를 활용할 예정이었으나, 해당 데이터셋은 텍스트 리뷰가 포함되지 않는다. 따라서 Yelp Dataset의 일본어 리뷰만을 선별하여 일본어 모델을 구축하고자 하였다.

일본어 모델로 확장하기 위하여 사용한 확장용 데이터는 히라가나, 가타카나 포함 문장을 선별하는 방식으로 인코딩하였다.

나. 데이터 상세 명세

1) 학습용 데이터 : Zomato Bangalore Restaurants(Kaggle)

칼럼 명	데이터 타입	설명
name	object	식당 이름
reviews_list	object	개별 후기의 평점과 텍스트 본문

* reviews_list는 개별 후기의 평점(int64)와 텍스트 본문(object)이 들어 있는 튜플이다.

2) 검증용/확장용 데이터 : Yelp Dataset(Kaggle)

칼럼 명	데이터 타입	설명
business_id	object	식당 이름
rating	int64	개별 후기의 평점
review_text	object	텍스트 본문

3. 데이터 처리 과정

가. 데이터 전처리

- Zomato Bangalore Restaurants

1) 필요 칼럼 추출

```
zomato_filtered = zomato_copy[['name', 'reviews_list']].dropna()
```

2) 리뷰 파싱 및 Explode

reviews_list 칼럼의 문자열을 실제 리스트로 변환하고, (평점, 리뷰 본문) 형태로 추출하였다. 식당 내 리뷰들을 개개의 행으로 분리하였다(explode).

```
def parse_reviews_column(row):
    try:
        reviews = ast.literal_eval(row)

        if not reviews:
            return []

        return reviews
    except:
        return []

zomato_filtered['parsed_reviews'] = zomato_filtered['reviews_list'].apply(parse_reviews_column)
zomato_filtered_exploded = zomato_filtered.explode('parsed_reviews')
```

3) 평점 및 리뷰 텍스트 정제 및 공백 검사

평점 부분의 'Rated '와 텍스트 리뷰 부분의 'RATED₩n '을 제거하였다. 공백의 유무를 검사하고 제거하였다.

```
def clean_rating_and_text(review_tuple):
    # 튜플/리스트가 아니거나 길이가 부족하면 None 반환
    if not isinstance(review_tuple, (list, tuple)) or len(review_tuple) < 2:
        return None, None

    rating, text = review_tuple

    # 평점(Rating) 정제
    if pd.isna(rating):
        cleaned_rating = None
```

```
    else:
        try:
            rating_str = str(rating).replace("Rated", "").strip()
            if not rating_str:
                cleaned_rating = None
            else:
                cleaned_rating = float(rating_str)
        except (ValueError, TypeError):
            cleaned_rating = None

    # 리뷰 텍스트(Review Text) 정제
    cleaned_text = None
    if isinstance(text, str):
        temp_text = text.replace("RATED₩n", "")

        if not re.match(r'^₩s*$', temp_text):
            cleaned_text = temp_text.strip()

            if not cleaned_text:
                cleaned_text = None
        else:
            cleaned_text = None

    return cleaned_rating, cleaned_text
```

4) 학습 데이터셋 최종 정제

name 칼럼은 그대로 취하고, reviews_list를 파싱, explode, 공백 검사, 불필요 텍스트를 제거한 값으로 생성한 rating, review_text 칼럼을 생성하였다.

```
zomato_filtered_exploded['rating'], zomato_filtered_exploded['review_text'] = zip(*zomato_filtered_exploded['parsed_reviews'].apply(clean_rating_and_text))
zomato_df = zomato_filtered_exploded.dropna(subset=['rating', 'review_text']).loc[:, ['name', 'rating', 'review_text']]

display(zomato_df.head())
```

	name	rating	review_text
0	Jalsa	4.0	A beautiful place to dine in.The interiors tak...
0	Jalsa	4.0	I was here for dinner with my family on a week...
0	Jalsa	2.0	Its a restaurant near to Banashankari BDA. Me ...
0	Jalsa	4.0	We went here on a weekend and one of us had th...
0	Jalsa	5.0	The best thing about the place is itÃ□□Ã□□Ã□□Ã...

5) 특수문자 제거 및 불용어 처리, 표제어 추출

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    if not isinstance(text, str):
        return ""

    text = text.lower()
    text = re.sub(r'[^a-zA-Z]', ' ', text)

    words = text.split()

    clean_words = [
        lemmatizer.lemmatize(word)
        for word in words
        if word not in stop_words and len(word) > 1
    ]

    return ' '.join(clean_words)
```

나. 모호한 데이터(별점 3점) 처리 전략

별점 3점은 긍/부정이 혼재된 리뷰라고 할 수 있다. 이를 학습 데이터에 포함할 경우 모델의 결정 경계를 흐리는 노이즈가 될 수 있다.

학습 단계에서는 명확한 기준 확립을 위하여 3점 데이터를 제외하였다. 1~2점 데이터는 부정(0)으로, 4~5점 데이터는 긍정(1)로 레이블링하여 모델을 학습시켜 명확한 감성 기준을 확립하였다.

적용 단계에서는 모델 학습에서 제외하였던 3점 데이터를 재평가하여 모호함 속의 문맥을 긍/부정의 이원값 정도(확률)을 추출하여 최종 지수에 반영하였다.

3점대 리뷰는 2.5점~3.5점으로 한정하여 –1 수치를 할당하였다. 2.5점~3.5점의 범위는 1~5점 값의 중간에 해당하는 범위이다. 긍정(1) 데이터는 888,947건(79.44%), 부정(0) 데이터는 230,114건(20.56%)으로 불균형 정도가 심하지 않아 학습에 유의미한 수치라고 판단하였다;. 다시 말해, 데이터가 100만 건으로 확보된 상황이므로 성능 파악 시 정확도를 활용해도 무리가 없다고 판단하였다.

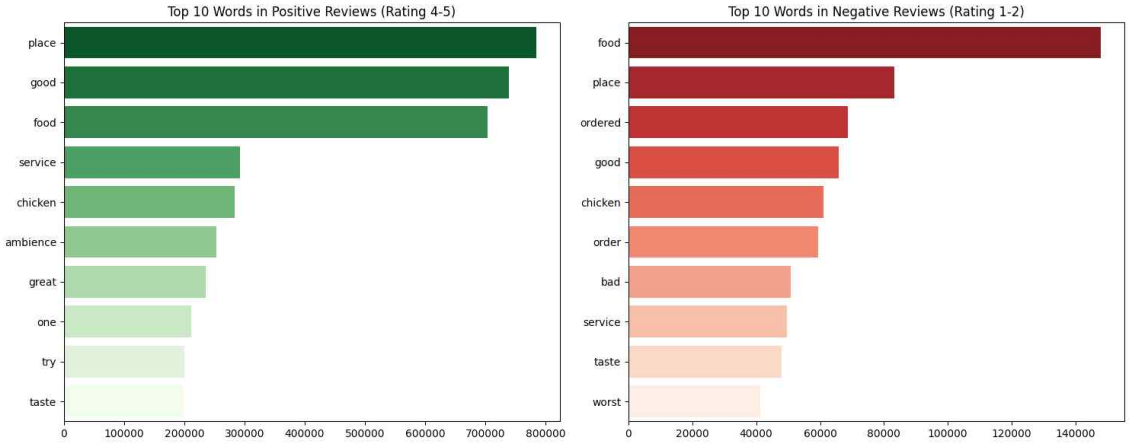
```
def assign_sentiment_label(rating):
    if rating >= 3.5:
        return 1
    elif rating < 2.5:
        return 0
    else:
        return -1
```

4. 머신러닝 모델의 튜닝 과정

가. 단순 단어 빈도수 방식의 상위 출현 빈도 단어 확인

긍정과 부정 리뷰 모두에서 'place', 'food', 'chicken', 'service', 'taste' 등의 단어가 최상위권에 등장한다. 이는 단순 빈도수 방식만으로는 긍/부정의 문맥을 명확히 구분하기 어려운 한계가 발견된다.

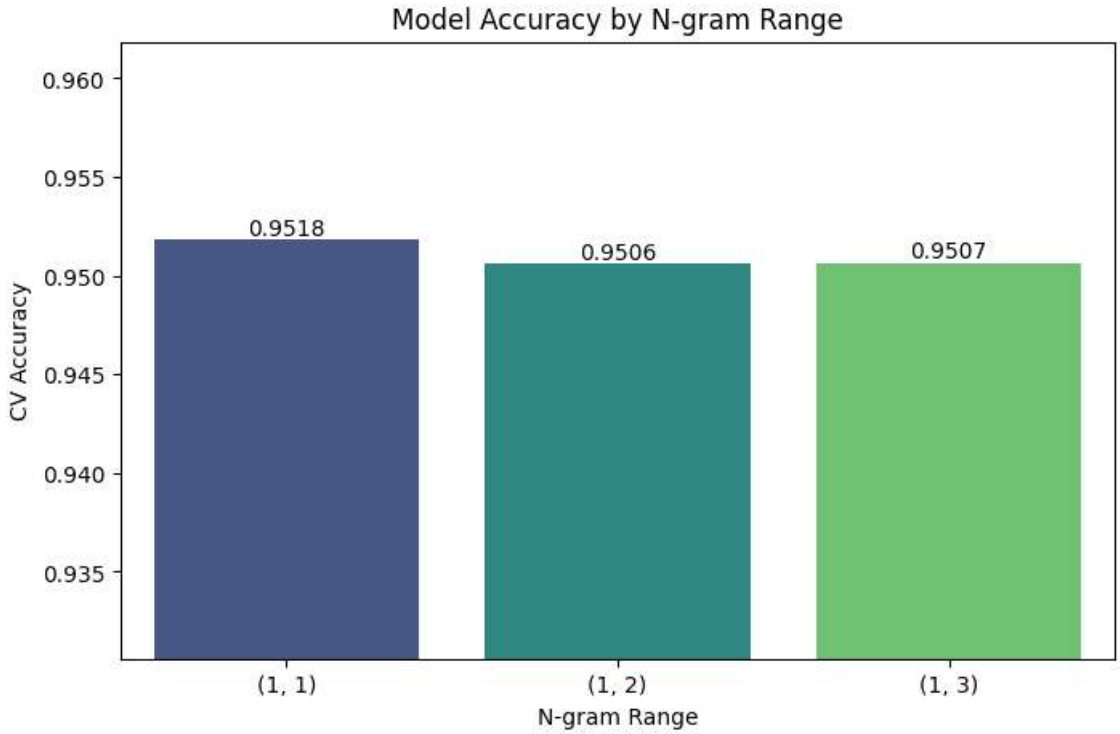
```
def get_top_words(texts, n=10):
    all_words = ' '.join(texts).split()
    return Counter(all_words).most_common(n)
```



나. N-gram 방식 도입(1-gram, 1,2-gram, 1,3-gram 성능 비교)

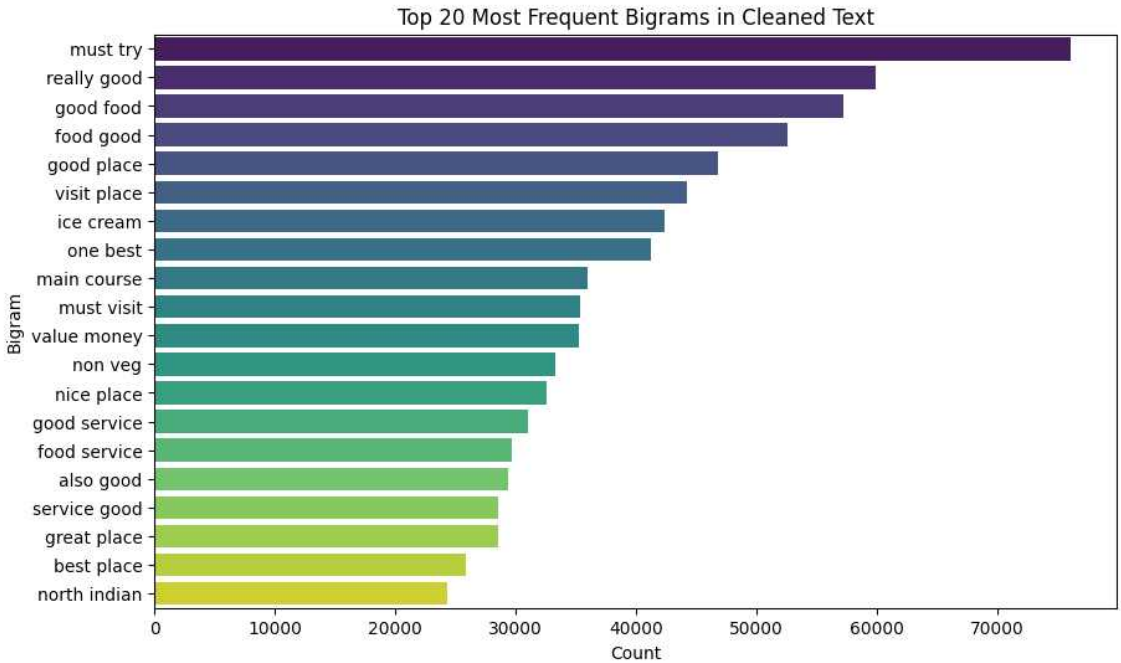
앞서 확인한, 긍정과 부정의 상위권에 공통으로 있는 단어들을 제외하기 위하여 N-gram의 범위를 (1, 3)까지 확장하여 실험하였으나, 예상과 달리 단일 단어인 Unigram(1-gram) 기반의 모델이 95.18%로 가장 높은 성능을 보였다.

```
ngram_ranges = [(1, 1), (1, 2), (1, 3)]
cv_scores = []
ngram_labels = []

for n_range in ngram_ranges:
    vectorizer_test = TfidfVectorizer(max_features=2000, ngram_range=n_range)
    X_test = vectorizer_test.fit_transform(final_data['cleaned_text'])
    y_test = final_data['label']

    scores = cross_val_score(LogisticRegression(max_iter=500), X_test, y_test, cv=3, scoring='accuracy')
    avg_score = scores.mean()

    cv_scores.append(avg_score)
    ngram_labels.append(str(n_range))
    print(f"  N-gram {n_range}: 정확도 {avg_score:.4f}")
```



불용어를 제외했음에도 불구하고 위와 같은 결과가 나온 원인, 즉 Bigram을 섞었는데 성능이 오르지 않은 원인을 조사하였다. 아래는 Bigram 방식, 즉 두 단어 조합의 모델에서 상위 20개의 단어를 확인한 것이다. max_features는 동일하게 2,000이다.

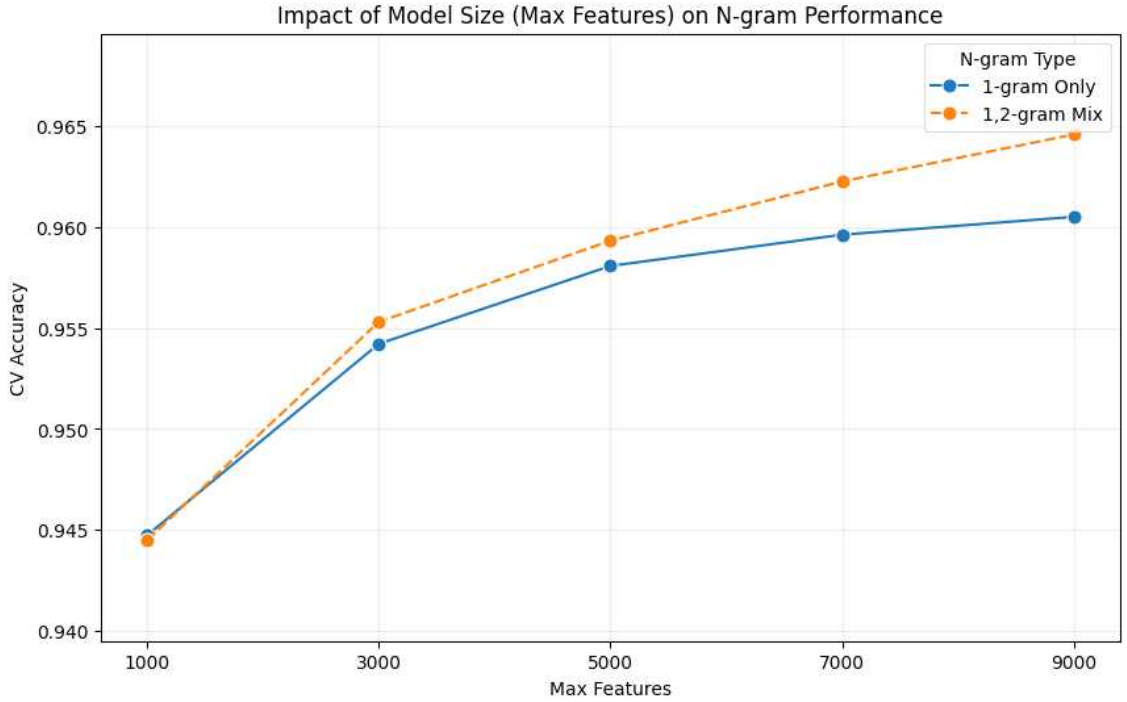
두 단어의 조합 결과는 매우 유의미한 단어들의 조합을 확인할 수 있었다.



1-gram의 성능보다 1,2-gram, 즉 Bigram에 Unigram(1-gram)을 혼합한 모델의 결과가 성능이 낮게 나온 이유를 확인하고자 하였다. 각각 처리해야 할 데이터 수는 1-gram 사용 시 48,153개 어휘를 학습해야 했고, 1,2-gram 사용 시 1,006,164개 어휘를 학습해야 했다. 즉, 1,2-gram 도입 시 모델이 처리해야 할 차원이 약 20.9배 폭등하는 차원의 저주를 확인할 수 있었다.

그러나, 2,2-gram 사용 시 상위권 단어 조합의 유의미성을 확인한바, 1,2-gram 도입의 가능성을 확인하고자 한다. 우선, 1,2-gram의 불용어는 완벽히 제거되었다. 반면, 1,2-gram은 기존의 1-gram을 활용하므로, 1,2-gram 성능 변화에 기여하는 것으로 확인되었다. 예를 들어, good food는 good과 food로 나뉠 수 있는데, good 자체로 긍정으로서 매우 유의미하다.

따라서, 1-gram과 1,2-gram의 각각의 성능에 영향을 주는 피쳐 수를 조절함으로써 2,2-gram에서 확인한 유의미한 단어 조합을 성능에 기여하도록 max_features 수를 1,000부터 9,000까지 2,000 간격으로 성능을 확인하였다. 1-gram과 1,2-gram의 성능 우위 관계를 확인하였다. 결론적으로 9,000개의 최대 피쳐에서도 성능이 향상된 것을 확인하였다.



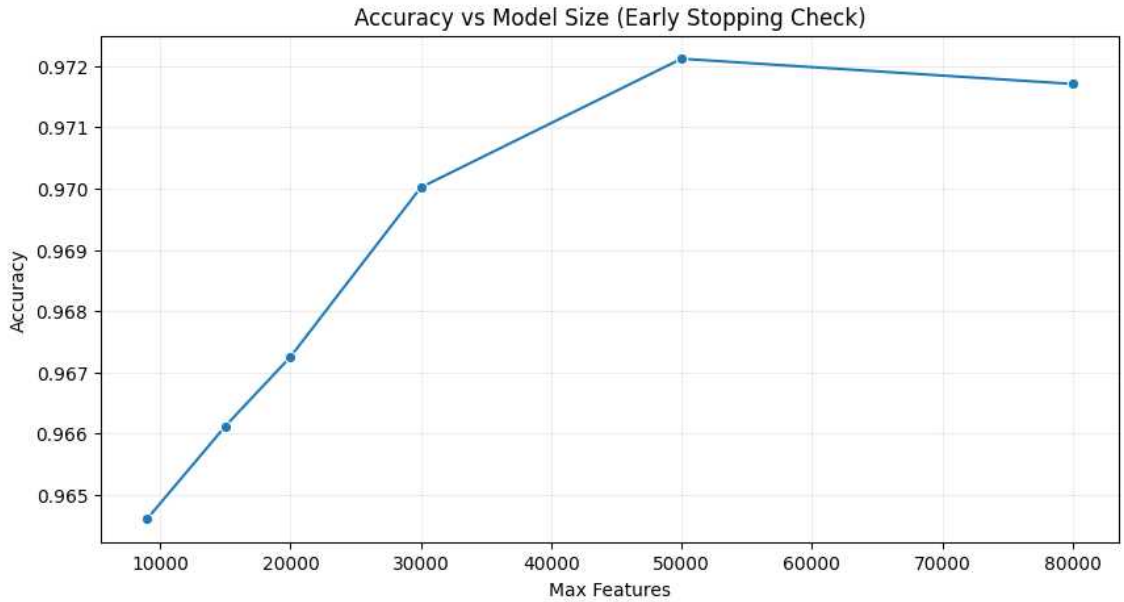
위에서 설정한 피쳐 수의 범위의 최댓값인 9,000개의 피쳐를 선택하는 것이 유의미한지 알아보았다. 1,2-gram의 개수는 1,006,164개로, 선택한 9,000개는 상위 0.89%로 파악되었다. 20,000개로 급격히 늘려도 성능 향상이 일어나는지 확인해 보았다.

20,000개에서의 정확도는 0.9673으로, 9,000개 대비 향상 폭은 0.0027이다. 성능 향상이 있으므로, 80,000개까지 max_features를 다르게 하여 성능을 측정해 보았으며, 메모리와 시간 복잡도를 함께 비교하여 효율적인 피쳐 수를 확인하였다.

전체 데이터 내의 단어 등장 총횟수는 64,235,599회로 파악되었다. 상위 80%를 커버하려면 필요한 피쳐 수는 79,233개, 상위 90%는 223,873개, 상위 95%는 392,309개, 상위 98%는 595,699개가 필요하다.

```
print("₩n=== 'Early Stopping' 전략을 적용한 최적 Max Features 탐색 ===")
candidates = [9000, 15000, 20000, 30000, 50000, 80000]
threshold = 0.001

best_score = 0
best_feat = 0
prev_score = 0

history = []
```

```
for n_feats in candidates:
    print(f"Testing Max Features: {n_feats}...", end=" ")

    tfidf = TfidfVectorizer(max_features=n_feats, ngram_range=(1, 2))
    X_temp = tfidf.fit_transform(final_data['cleaned_text'])
    y_temp = final_data['label']

    score = cross_val_score(LogisticRegression(max_iter=500), X_temp, y_temp, cv=3).mean()

    improvement = score - prev_score
    print(f"Accuracy: {score:.4f} (Improvement: +{improvement:.4f})")

    history.append((n_feats, score))

    if prev_score > 0 and improvement < threshold:
        print(f"₩n[STOP] 성능 향상 폭({improvement:.4f})이 기준치({threshold}) 미만입니다.")
        print(f"-> 경제적 효율성을 위해 직전 단계인 {prev_feat}개 (또는 현재 {n_feats}개)에서 멈추는 것을 권장합니다.")
        best_feat = n_feats
        best_score = score
        break

    prev_score = score
    prev_feat = n_feats
    best_feat = n_feats
```



Early Stopping 전략을 적용하여 피처 수를 9,000개부터 80,000개까지 확장하며 성능, 시간, 메모리 효율성을 종합적으로 분석한 결과, 다음과 같은 결론을 얻었다.

첫째, 정확도는 50,000개(0.9721)에서 정점을 찍고, 80,000개(0.9717)에서는 오히려 하락했다. 이는 과

도한 피처가 노이즈로 작용하여 모델의 일반화 성능을 저해함을 증명한다(과적합).

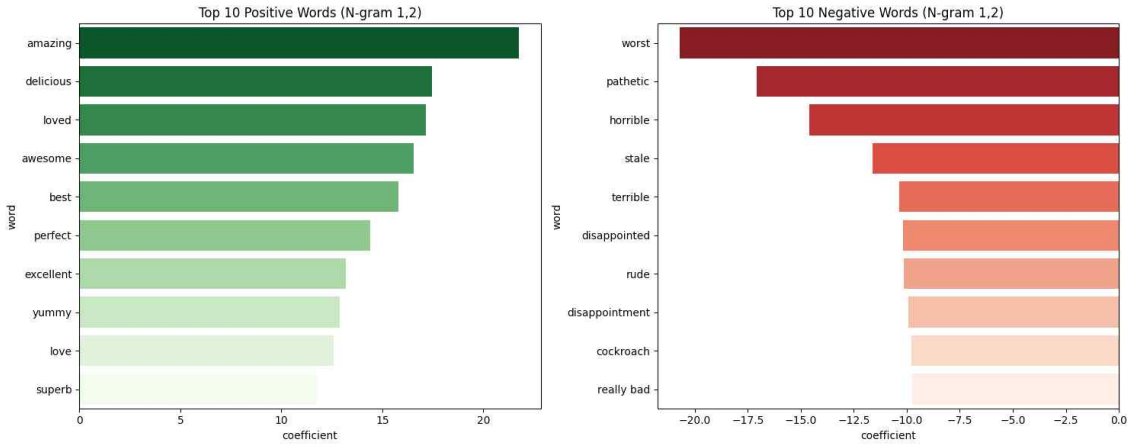
둘째, 50,000개가 수치로 최고였으나, 피처의 수가 66% 증가한 30,000개 대비 성능 향상 폭은 0.2%(0.9700->0.9721)에 불과했다. 반면, 메모리 점유율은 약 30MB 증가하고 연산 시간 또한 소폭 상승하여 성능 효율이 저하됨을 확인할 수 있었다.

따라서, 정확도 97% 달성과 과적합 방지 및 연산 효율성을 모두 만족하는 max_features=30000, ngram_range=(1, 2)를 최종 모델의 파라미터로 확정했다.

Max Features	Accuracy	Time(sec)	Memory(MB)
9,000	0.9646	64.92	414.3
15,000	0.9661	67.36	443.0
20,000	0.9673	68.29	459.2
30,000	0.9700	72.34	482.5
50,000	0.9721	73.03	513.0
80,000	0.9717	74.34	542.9

다. TF-IDF 수행

따라서 max_features=30,000과 1,2-gram을 적용하여 TF-IDF를 수행하였다.



로지스틱 회귀로 예측을 수행한 결과 모델의 성능은 다음과 같았다.

	precision	recall	f1-score	support
0	0.96	0.93	0.95	45,973
1	0.98	0.99	0.99	177,489

	precision	recall	f1-score	support
accuracy			0.98	223,462
macro avg	0.97	0.93	0.95	223,462
weighted avg	0.98	0.99	0.99	223,462

테스트 결과 예시는 다음과 같다.

[실전 테스트 결과]
리뷰: The food was amazing and the service was great!
예측: 긍정(1) (확률: 1.00)

리뷰: Absolutely terrible. The food was cold and staff was rude.

```
예측: 부정(0) (확률: 1.00)
------------------------------
리뷰: It was okay, not bad but not good either.
예측: 부정(0) (확률: 0.99)
------------------------------
```

긍정 확률 평균을 확인하였다. cross_val_predict를 사용하여 모델이 예측한 긍정 확률의 평균(긍정 레이블일 확률값의 평균)을 구하여 ai_prob 칼럼을 생성한다.

```
restaurant_group = final_data.groupby('name').agg({
    'rating': 'mean',
    'ai_prob': 'mean',
    'review_text': 'count'
}).rename(columns={'review_text': 'review_count'})

target_restaurants = restaurant_group[restaurant_group['review_count'] >= 10].copy()
```

0과 1 사이의 값을 5점 만점(최소 1점)으로 환산한다.

```
target_restaurants['ai_score_5'] = 1 + target_restaurants['ai_prob'] * 4
```

거품 지수를 계산한다. (기존 별점 - AI 확률)이 클수록 별점은 높은데 텍스트는 부정적이다. 이는 거품을 의미한다. (기존 별점 - AI 확률)이 작을수록(음수) 별점은 낮은데 텍스트는 긍정적이다.

그래프(3점 제거)는 명백한 호불호를 바탕으로(긍정 1, 부정 0) 식당의 평균 만족도 지수를 역추적할 수 있음을 시사한다.

라. 머신러닝 알고리즘 비교(Logistic Regression, Random Forest, XGBoost)

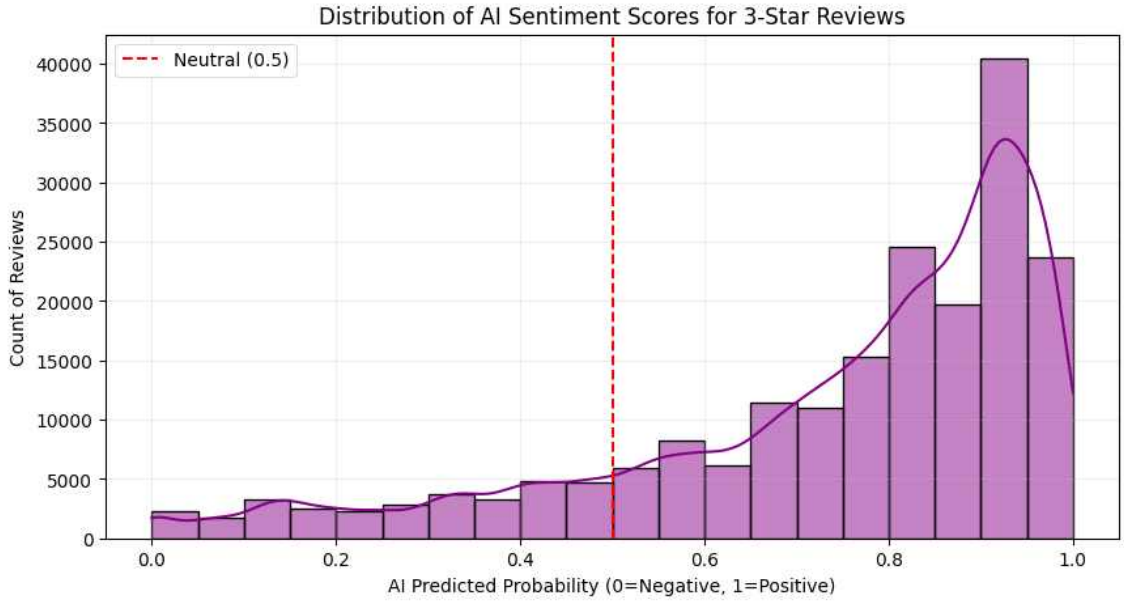
model	accuracy	f1-score
Random Forest	0.9935	0.9935
Logistic Regression	0.9790	0.9789
XGBoost	0.9638	0.9636

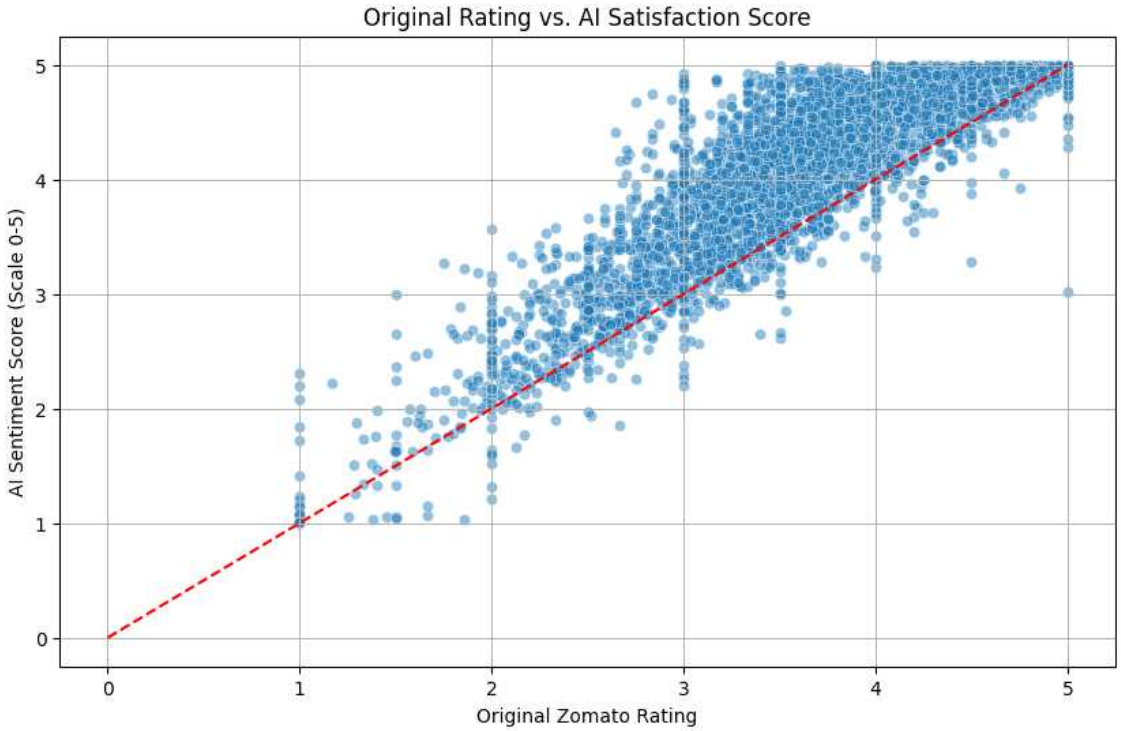
마. TF-IDF와 BERT 성능 비교(max_features=1,000)

BERT 성능 평가 시 데이터의 개수를 줄이는 것이 바람직하므로 max_features를 1,000으로 한정하였다. TF-IDF에는 Logistic Regression이 사용되었다. 결과적으로, TF-IDF의 정확도는 0.8900, BERT의 정확도는 0.9350으로 1,000개의 max_features에도 BERT 임베딩이 성능이 더 좋은 것을 알 수 있다.

바. 3점 리뷰 데이터를 예측

3점 리뷰는 텍스트의 긍/부정 맥락과 다르게 중간을 부여하는 모호한 평점으로, 중간 입장의 3점 리뷰도 긍/부정을 확률 기반으로 예측하여 재평가할 수 있다. 이를 통해 데이터의 손실 없이 지표에 반영하여 만족도 지수의 밀도를 향상할 수 있었다. 3점 리뷰의 개수는 198,152건이다. 이 중, 예측 결과 애매한 확률값(0.4~0.6)을 도출한 리뷰의 개수는 21,835건(11%), 확실한 긍/부정으로 나눌 수 있는 리뷰의 개수는 176,317건(89%)이었다.



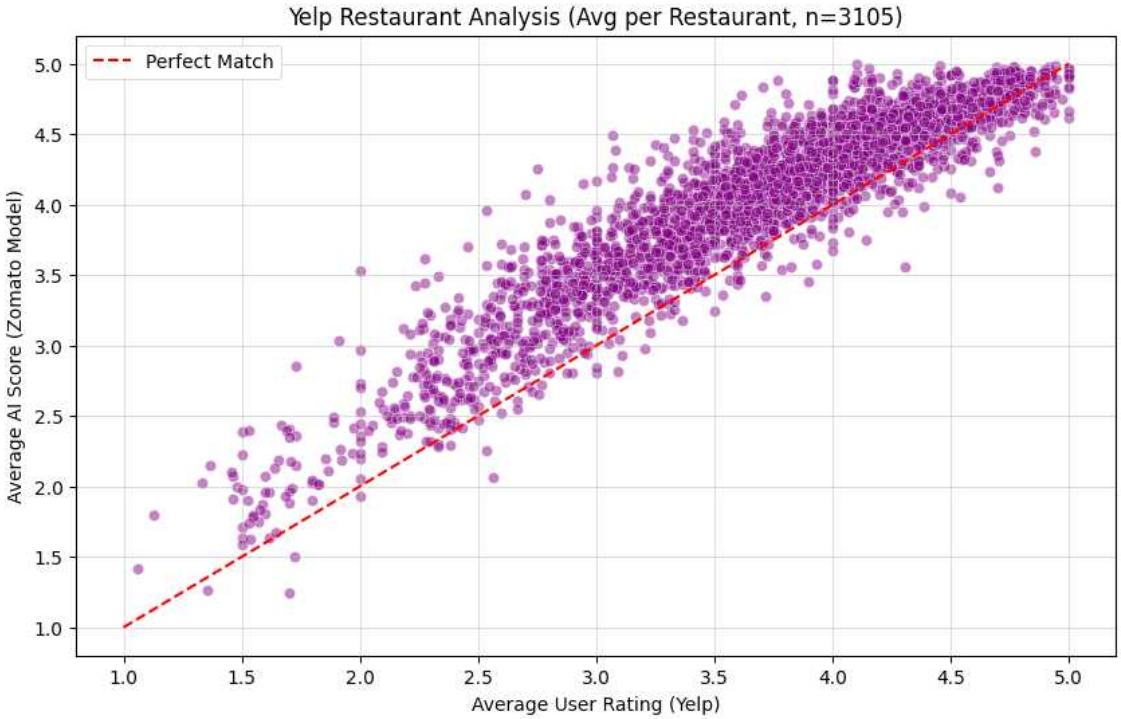


사. Yelp Dataset 적용

Yelp Dataset에서 식당 데이터(Restaurant/Food)를 추출한 결과, 64,616개의 식당이 추출되었다. 리뷰 데이터 중 200,000개를 분석 대상으로 무작위 선택하였다. 또한, 10개 이상의 리뷰를 가진 식당만을 선별하였는데, 최종적으로 3,105개의 식당이 추출되었다. 이를 바탕으로 예측을 수행하였다.

상관계수가 0.9264로, 모델의 일반화 성능이 매우 뛰어난 것을 확인하였다. 앞서 가설에서 대규모의 데이터가 정상의 데이터라면, 리뷰 텍스트와 별점의 평균은 강한 양의 상관관계를 가진다는 것을 가정한 바 있다. 이는 일반적으로 예상이 가능한 사실이지만, 학습된 모델로 이러한 경향성을 재확인해 보는 것은 유의미하다. 또한, Zomato Bengalore 데이터(인도 사용자의 영어 리뷰 데이터)로 학습한 모델을 Yelp 데이터(다국적 사용자의 영어 리뷰 데이터)로 검증하는 과정에서 위와 같은 상관관계를 보이는 것은 범용성 측면에서도 유의미하다.

상관계수 1과 이 모델의 상관계수 0.9264 사이의 값, 즉, 차이값 약 0.7은 노이즈 데이터의 예측치라고 해석할 수 있다. 리뷰 텍스트와 별점의 평균이 완벽하게 상관관계를 가진다면 1이라 할 수 있는데, 완벽하지 않은 노이즈 데이터이므로 그 차이 값은 노이즈 데이터의 모델 예측 시도 흔적이라고 확인할 수 있다. 이는 해당 모델의 만족도 지수가 필요함을 다른 측면에서 보여주는 대목이기도 하다.



아. 일본어 리뷰 확장 실험

먼저 일본어로 한정하여 언어 인코딩을 진행하였다. 일본어는 히라가나(3040~309F), 가타카나(30A0~30FF), 한자(4E00~9FFF)로 구성된 언어다. 그러나, 중국어는 한자로만 이루어져 있다. 따라서, 한자를 인코딩에서 포함하면 중국어 리뷰 텍스트도 포함이 되는 현상이 발생하여, 한자는 인코딩 조건에서 제외하였다. 따라서, 히라가나와 가타카나가 일정 비율(15%) 이상 포함된 데이터를 필터링하였다. 이러한 과정을 거쳐 총 순수 일본어 리뷰는 199개의 데이터를 확보할 수 있었다.

그러나, 앞선 모델은 100만개 리뷰 데이터를 대상으로 하였으므로, 199개의 데이터는 모델을 만들기도 어렵고, 수치적 차이로 비교의 의미도 크지 않으리라 판단하였다. 또한, 기존의 모델은 영어 리뷰 텍스트를 중점적으로 학습시킨 것으로, 일본어 리뷰에 적용하는 것은 언어적 측면과 문화적 측면에서 바람직하지 않다.

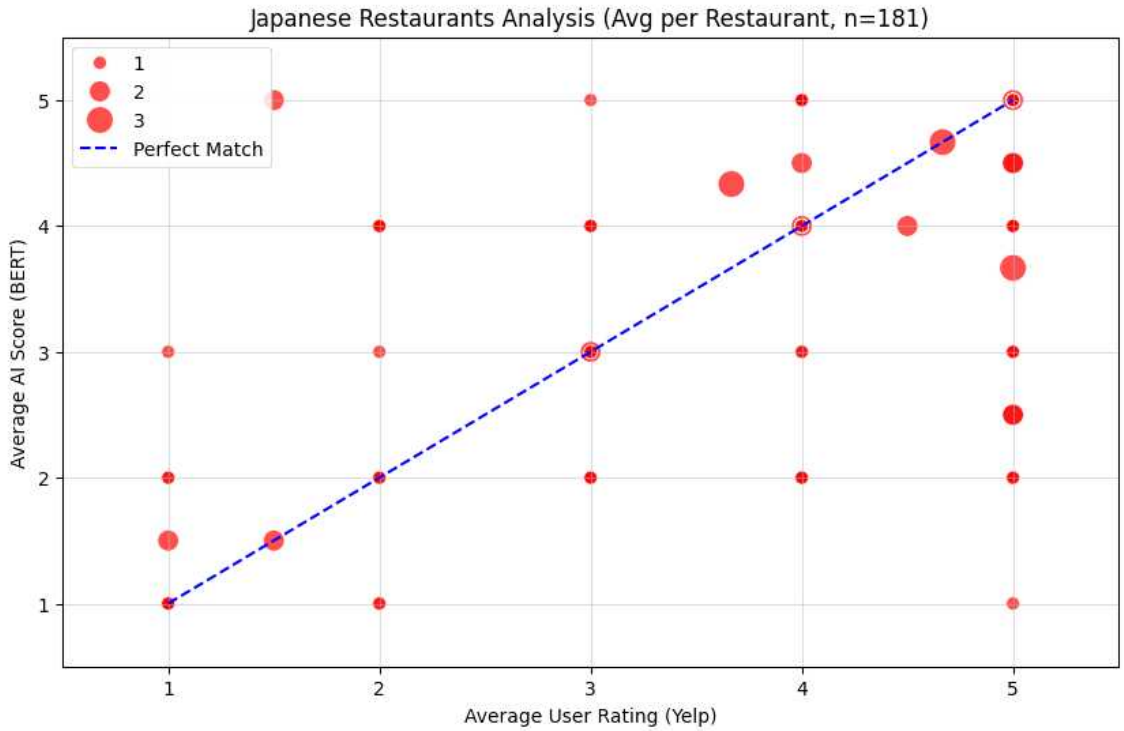
따라서 확장 프로젝트인 일본어 리뷰 대상으로는 어쩔 수 없이 BERT 모델을 사용하였다. 한편, 512 토큰으로 제한하여 분석 식당의 총개수는 181개다.

식당별 평균 별점과 BERT 예측 점수 간의 상관계수는 0.5866으로 나타났다. 통계적으로 이는 뚜렷한 양의 상관관계에 해당한다. 비록 영어 데이터 검증 성능보다는 낮지만, 별도의 추가 학습 없이 사전 학습된 모델을 바로 적용한 결과임을 감안할 때, 모델이 일본어 리뷰의 긍/부정 경향성을 유의미한 수준으로 파악하고 있음을 시사한다.

앞선 영어 모델은 Zomato 식당 리뷰 데이터로 학습되어 식당 용어에 특화되지만, 일본어 분석에 사용된 Multilingual BERT는 일반적인 텍스트로 학습된 범용 모델이다. 따라서, 식당 리뷰만의 특수한 문맥을 파악하는 데에는 한계가 있었다. 또한, 데이터의 모수의 차이에서도 기인하는데, 소규모 데이터이기 때문에 이상치의 영향을 더 크게 받은 것으로 보인다.

그러나, 최소한의 자원으로 0.6에 육박하는 상관관계를 확보할 수 있었고, 이는 보편적인 상관관계를 가정한 가설이 입증되는 대목이기도 하다. 또한, 이를 통해 다시 모델로 만족도 지수를 재산출할 수 있음을 확인할 수 있었다.

```
sentiment_pipeline = pipeline(
    task="sentiment-analysis",
    model="nlptown/bert-base-multilingual-uncased-sentiment"
)

def predict_sentiment(text):
    if not isinstance(text, str): return 3
    # 길이 제한 (512 토큰)
    result = sentiment_pipeline(text[:512])[0]
    return int(result['label'].split()[0])
```



5. 결론

가. 연구 요약

본 연구는 사용자가 외식 장소를 선택할 때 직면하는 정보의 비대칭성과 별점의 주관성 문제를 해결하기 위하여 머신러닝 기반의 리뷰 감성 분석 시스템을 구축하였다. 100만 건 이상의 대규모 리뷰 데이터를 TF-IDF와 N-gram 방식으로 벡터화하고, Random Forest 모델을 통해 99% 이상의 높은 분류 정확도를 달성하였다. 이를 통해 텍스트 내면에 숨겨진 실제 만족도를 정량화한 AI 만족도 지수를 산출하는 데 성공하였다.

나. 연구 의의

객관적 지표를 제시하였다. 단순 평균 별점이 아닌 텍스트 문맥에 기반한 확률적 점수를 제공함으로

써 사용자의 의사결정 실패 비용을 줄일 수 있는 새로운 지표(Gap 지수)를 제시하였다.

데이터를 손실 없이 사용하여 모호한 데이터의 명확화를 이끌어냈다. 학습에서 제외했던 모호한 '3점 리뷰'를 재평가하여, 중립적인 의견 속에 숨겨진 긍/부정의 치우침을 판별했다.

일반화 성능이 높다. 특정 플랫폼(Zomato Bangalore, 인도) 데이터로 학습한 모델이 타 플랫폼(Yelp, 미국) 데이터에서도 0.92 이상의 높은 상관관계를 보임으로써, 리뷰 텍스트와 별점 간의 보편적인 연관성을 입증하고 모델의 상용화 가능성을 확인하였다.

다. 한계점 및 향후 과제

1) 언어적 뉘앙스의 한계 : TF-IDF와 같은 통계적 방식은 반어법이나 고도의 문맥적 뉘앙스를 완벽하게 파악하기에는 한계가 있다. BERT, GPT와 같은 트랜스포머 기반의 거대 언어모델을 전체 데이터에 적용한다면, 더욱 정교한 감성 분석이 가능할 것이다.

2) 다국어 확장성 : 일본어 리뷰 분석 시 데이터 모수 부족으로 인해 영어 모델만큼의 신뢰도를 확보하지 못했다. 충분한 다국어 데이터셋 확보와 전이 학습을 통해 언어에 구애받지 않는 글로벌 스코어링 모델로 고도화할 필요가 있다.

3) 최신성 반영 : 식당의 서비스 품질은 시간에 따라 변한다. 향후 연구에서는 리뷰 작성 시점에 따른 가중치를 적용하여 최근 트렌드를 반영한 동적인 만족도 지표를 개발하고자 한다.