모두 함께 힘내는 방법!

소상공인 살리기 캠페인

소상공인 화이팅 전문가가 도와드리겠습니다.

장진석, 이수진

박선경, 최성진

경북대학교 빅데이터·AI 전문가 양성과정 4기

목차

1. 팀원별 파트분배
2. 주제 선정 이유
3. 데이터수집 및 전처리
4. 모델링
5. 시연 및 결론

팀원별 파트분배

팀원별 파트분배

★ 박선경

- 데이터수집(의류,음식)
- 모델링
- 워드클라우드

팀원별 파트분배

★ 장진석

- 데이터수집(의류)
- 모델링
- PPT제작

팀원별 파트분배

★ 이수진

- 주제 선정
- 데이터수집(의류,음식)
- 모델링

팀원별 파트분배

★ 최성진

- 주제 선정
- 데이터수집(의류분야)
- 모델링

[소화전] 요구사항 정의서

개발환경	O/S	window				2023. 10. 05.
	언어	python				

대분류	중분류	소분류	기능설명	우선순위	담당자	라이브러리
데이터 처리	데이터 수집 및 정제	데이터 수집	데이터 수집 (Naver Smart Store) 장진석 : 바람막이 박선경 : 레깅스, 밀키트, 수산물 최성진 : 트레이닝 바지, 상하의 이수진 : 런닝화, 과일, 축산물	1	전원	-
데이터 전처리	자연어 처리	정규화	다양한 형태의 단어들을 표준화	2	전원	Okt
		TF-IDF 변환	단어의 중요도 결정	2	최성진	TfidfVectorizer
모델링	회귀 모델 구축	모델 선구축 및 데이터 학습	Linear Regression 초기 모델 구축	3	장진석	sklearn
		데이터 반복 학습 및 모델 반복 검증	다양한 모델 적용, 평가 모델 교차 검증	3	이수진	sklearn
					최성진	sklearn
					박선경	sklearn
키워드 추출 프로그램	빈도수 높은 단어 추출	워드클라우드를 이용한 단어 추출	높은 빈도값을 가진 핵심 키워드들을 유저에게 제공하는 서비스	4	박선경	wordcloud
PPT	PPT 구성	PPT 구성 및 디자인	PPT 초기 템플릿 제작	5	장진석	-
PPT	PPT 제작	PPT 제작	PPT 슬라이다 수정 및 추가 제작	5	전원	-

주제선정이유

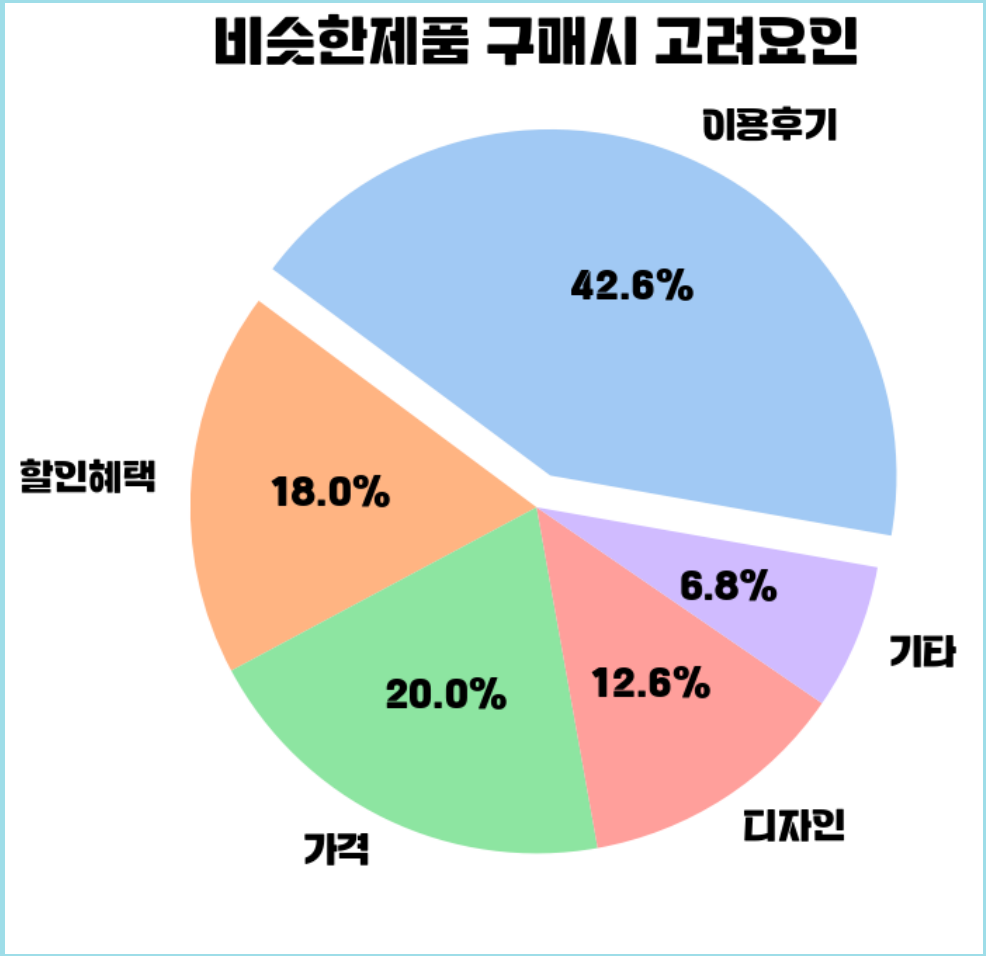
“ ”

주제선정이유

“이거 제가 잘못한건가요?”

건강한 리뷰 문화의 필요성

주제선정이유



“상품 구매에 있어 상당부분 영향을 끼치는 리뷰”

주제선정이유

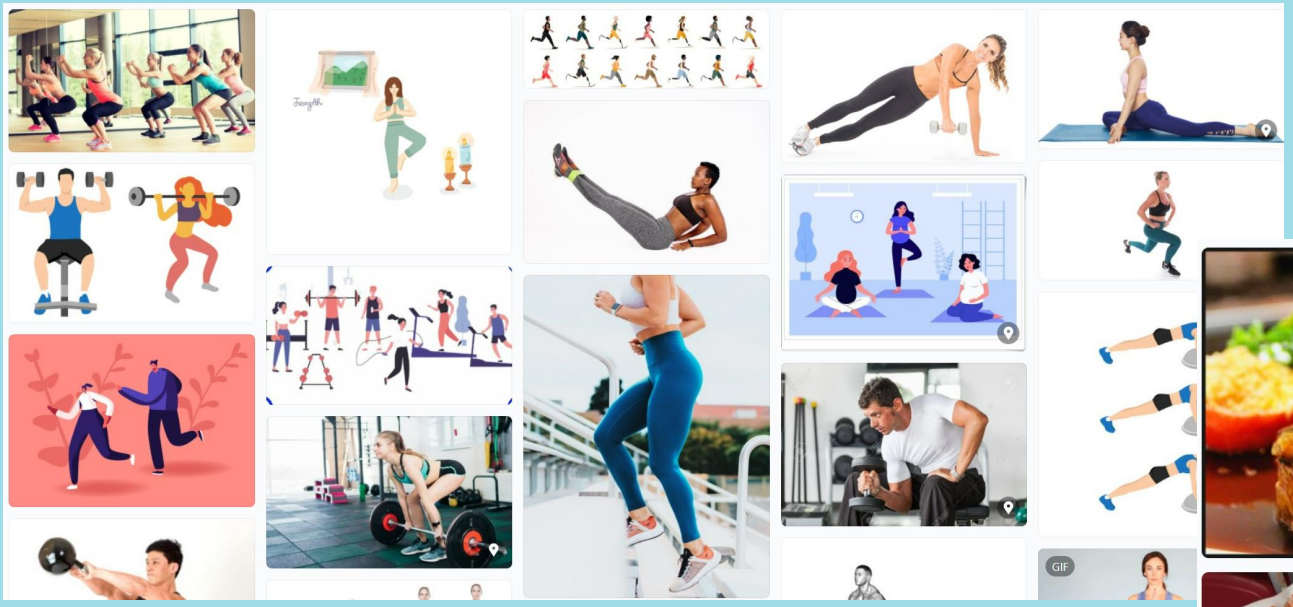
“초보 셀러들을 위한 핵심 키워드 제공”

“의류 및 음식분야 크롤링”

데이터 수집,처리

“의류 및 음식분야 크롤링”

데이터 수집,처리

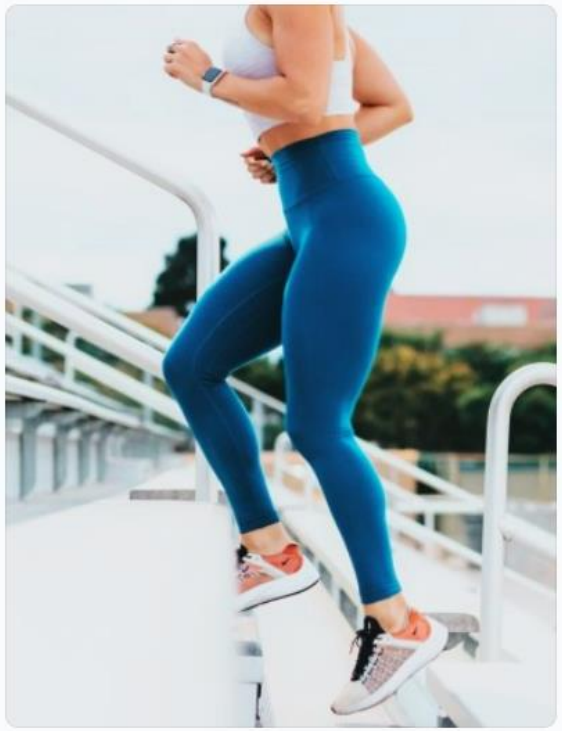
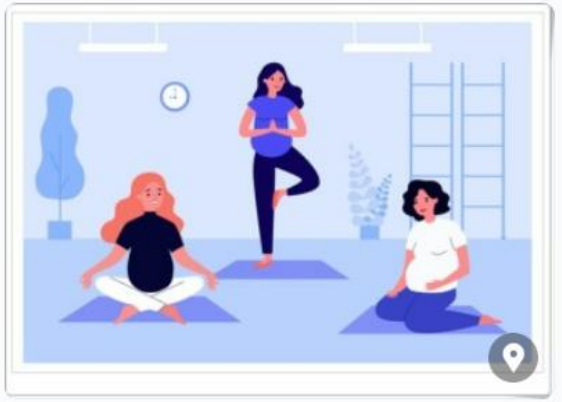






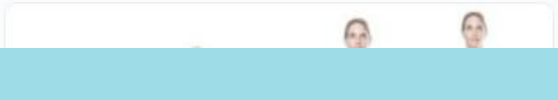









GIF

“네이버

스마트스토어,,

하이루하이 관심고객수 4,173 알림받기

나이키 트레이닝복세트 조거팬츠 맨투맨 셋업 기모 BEST

스는톨판

“네이버 스마트스토어,,

리뷰 및 평점

“네이버 스마트스토어,, 리뷰 및 평점

- -상위 키워드 20개
- -리뷰로 판단하는 예상별점

“”


5
선물
구매하기

의류

트레이닝복

키워드 추출

모델링

평점 예측

Step 01

Step 02

Step 03

키워드 추출

의류

트레이닝복

1

[('사이즈', 1774), ('구매', 1576), ('만족', 1309), ('운동', 1131), ('가격', 1065), ('바지', 967), ('성비', 885), ('감사', 876), ('착용', 609), ('여름', 584), ('생각', 547), ('제품', 493), ('재질', 493), ('주문', 490), ('대비', 483), ('아들', 402), ('허리', 334), ('상품', 318), ('최고', 311), ('기장', 310)]

키워드 추출

트레이닝복 구매 리뷰 댓글

정규화 후

상위 20개 단어 추출

2

워드클라우드 생성

리뷰 댓글 빈도 수

상위 20개의 명사 단어들로

워드클라우드 생성

모델링

의류

트레이닝복

1

```
# 텍스트 데이터를 TF-IDF로 변환
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# 선형 회귀 모델 생성과 훈련
model = LinearRegression()
model.fit(X_train_tfidf, y_train)
```

모델링 1

텍스트 데이터를 TF-IDF로 변환

선형 회귀 모델 생성과 훈련

2

```
# 테스트 세트로 예측
y_pred = model.predict(X_test_tfidf)

# 모델 평가
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 0.5068662973525954
```

모델링 2

테스트 세트로 예측

평균 제곱 오차 : 0.50

텍스트 데이터로 만든 모델이라

괜찮은 성능이라고 판단

의류

트레이닝복

```
new_review = ["""가격이 싸요.
              핏이 딱맞아요.
              예뻐요"""]
print(f"예측된 리뷰평점: {predicted_rating}")
```

```
예측된 리뷰평점: [4.82072081]
```

긍정적인 텍스트로
리뷰작성 시
4점 이상 리뷰평점 부여

```
new_review =["""핏이 괜찮은데 비싸요.
              배송은 늦었지만 편해요.
              재질이 얇지만 가격은 착해요"""]
print(f"예측된 리뷰평점: {predicted_rating}")
```

```
예측된 리뷰평점: [2.70150107]
```

긍정과 부정이 혼합된
리뷰작성 시
2~3점 사이의 리뷰평점 부여

```
new_review =["""두께가 너무 두꺼워요.
              별로에요.
              옛날것같아요"""]
print(f"예측된 리뷰평점: {predicted_rating}")
```

```
예측된 리뷰평점: [0.70528751]
```

부정적인 텍스트로
리뷰작성 시
2점 이하 리뷰평점 부여

평점 예측

의류

일반의류

키워드 추출

모델링

평점 예측

Step 01

Step 02

Step 03

키워드 추출

“ 의류 ”

일반의류

1

[('구매', 8574), ('배송', 7040), ('사이즈', 6711), ('만족', 5532), ('가격', 5132), ('성비', 3380), ('감사', 3272), ('여름', 2880), ('바지', 2794), ('생각', 2584), ('재질', 2537), ('색상', 2506), ('주문', 2265), ('대비', 2186), ('제품', 2117), ('길이', 2104), ('최고', 1445), ('구입', 1417), ('오버', 1285), ('상품', 1250), ('마음', 1243), ('느낌', 1199), ('세탁', 1186), ('원단', 1108), ('기장', 1090), ('착용', 1078), ('운동', 1073), ('티셔츠', 1030), ('품질', 967), ('정도', 917), ('추천', 906), ('남편', 901), ('허리', 883), ('건조기', 834), ('편안', 799), ('스판', 775), ('너', 773), ('소재', 765), ('두께', 745), ('팔', 674), ('부분', 659), ('처수', 658), ('아들', 655), ('기본', 632), ('색깔', 606), ('추가', 601), ('디자인', 593), ('사진', 585), ('다음', 578), ('색감', 578)]

키워드 추출

트레이닝복 구매 리뷰 댓글

정규화 후

상위 50개 단어 추출

2

워드클라우드 생성

리뷰 댓글 빈도 수

상위 50개의 키워드들을

판매자들에게 정보 제공

모델링

“ ”

의류

일반의류

1

```
# 텍스트 데이터를 TF-IDF로 변환
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# 선형 회귀 모델 생성과 훈련
model = LinearRegression()
model.fit(X_train_tfidf, y_train)
```

모델링 1

텍스트 데이터를 **TF-IDF로 변환**

선형 회귀 모델 생성과 훈련

2

```
# 테스트 세트로 예측
y_pred = model.predict(X_test_tfidf)

# 모델 평가
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 0.501067492549371
```

모델링 2

테스트 세트로 예측

평균 제곱 오차 : 0.50

텍스트 데이터로 만든 모델이라

괜찮은 성능이라고 판단

의류

일반의류

```
new_review = ["""가격이 싸요.
              핏이 딱맞아요.
              예뻐요"""]
print(f"예측된 리뷰평점: {predicted_rating}")
```

```
예측된 리뷰평점: [4.17094398]
```

긍정적인 텍스트로
리뷰작성 시
4점 이상 리뷰평점 부여

```
new_review =["""좋은데 배달이 느려요.
괜찮은데 사진보다 좀 작네요.
예쁜데 오래입진 못하겠네요"""]
print(f"예측된 리뷰평점: {predicted_rating}")
```

```
예측된 리뷰평점: [3.15300123]
```

긍정과 부정이 혼합된
리뷰작성 시
2~4점 사이의 리뷰평점 부여

```
new_review =["""너무 칙칙해요.
              두꺼워요.
              비싸요.
              별로에요."""]
print(f"예측된 리뷰평점: {predicted_rating}")
```

```
예측된 리뷰평점: [1.73952851]
```

부정적인 텍스트로
리뷰작성 시
2점 이하 리뷰평점 부여

평점 예측

축산물

워드 클라우드

신선, 상태, 냄새 : 신선도, 품질

주문, 배송, 구매 : 빠른 배송 서비스

밀키트

워드 클라우드

주문, 배송, 구매 : 빠른 배송 서비스

간편,소스 : 조리 과정 단순화

곱창, 대창, 캠핑: 캠퍼 타겟팅

“과일”

워드 클라우드

크기,당도,흠집,껍질 : 특징 강조
부모님, 감사, 선물 : 선물용 타겟팅
박스, 포장 : 포장 품질 강조

“ 수산물 / 건어물 ”

워드 클라우드

신선, 상태 : 신선도, 품질

가시, 냉동, 손질 : 제품 선택사항 강조

모델 선정 및 시연

” 모델링 과정:

- 모델 : RandomForestRegressor
-데이터 전처리 : TF-IDF 변환, 정규화
- 타겟 예측 : 리뷰 평점

모델 테스트

- 긍정 리뷰 :

"좋네요.

앞으로 종종 이용할게요"

- 평점 : 4.8

```
new_review = ["좋네요. 앞으로 종종 이용할게요"]
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = rf.predict(new_review_tfidf)
rf_rating=rf.predict(new_review_tfidf)

print(f"예측된 리뷰평점: {rf_rating}")
```

✓ 0.0s

```
예측된 리뷰평점: [4.8287525]
```

모델 테스트

- 부정 리뷰 :

""배달이 너무 늦고,

신선도가 떨어져요..

최악이에요"

- 평점 : 4.8

```
new_review = ["배달이 너무 늦고, 신선도가 떨어져요.. 최악이에요"]
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = rf.predict(new_review_tfidf)
rf_rating=rf.predict(new_review_tfidf)

print(f"예측된 리뷰평점: {rf_rating}")
```

✓ 0.0s

```
예측된 리뷰평점: [4.85040478]
```

모델링 문제

- 심한 과적합
- 데이터 불균형 :

긍정 리뷰 많고, 부정 리뷰 데이터 적음

=>부정 리뷰의 예측 정확도 낮아짐

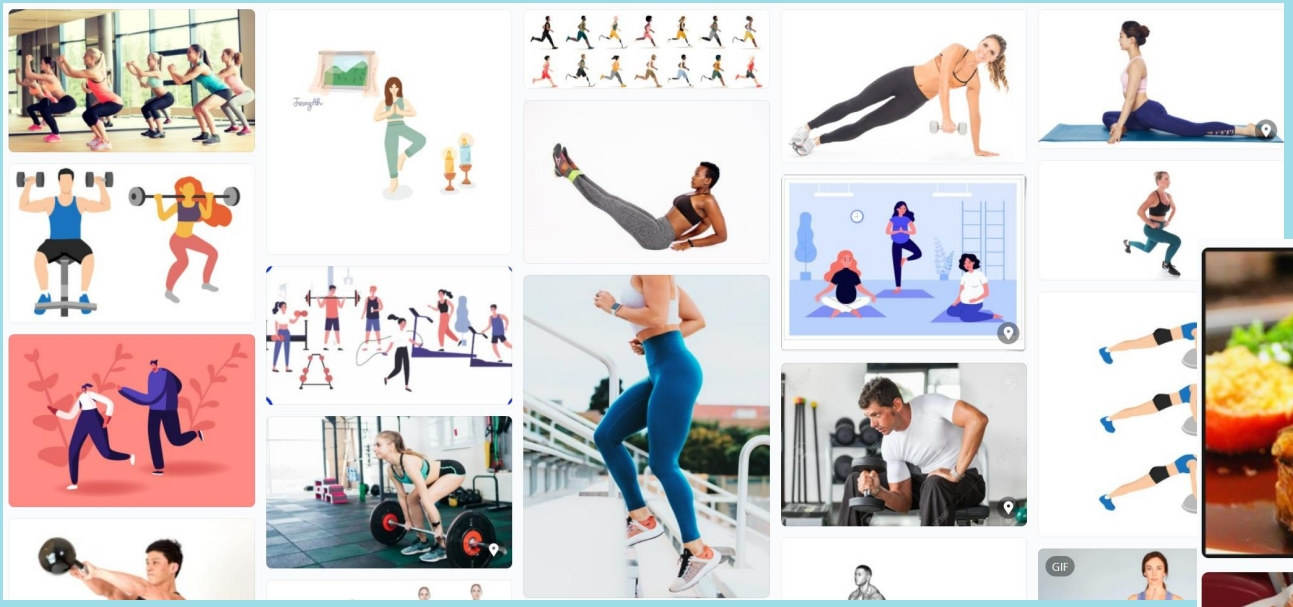
```
# 모델 평가
mse = mean_squared_error(test_y, y_pred)
print(f'Mean Squared Error: {mse}')
```

✓ 0.0s

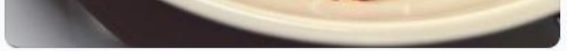
```
Mean Squared Error: 0.3566126120245767
```

“의류 및 음식분야 크롤링”

데이터 수집,처리





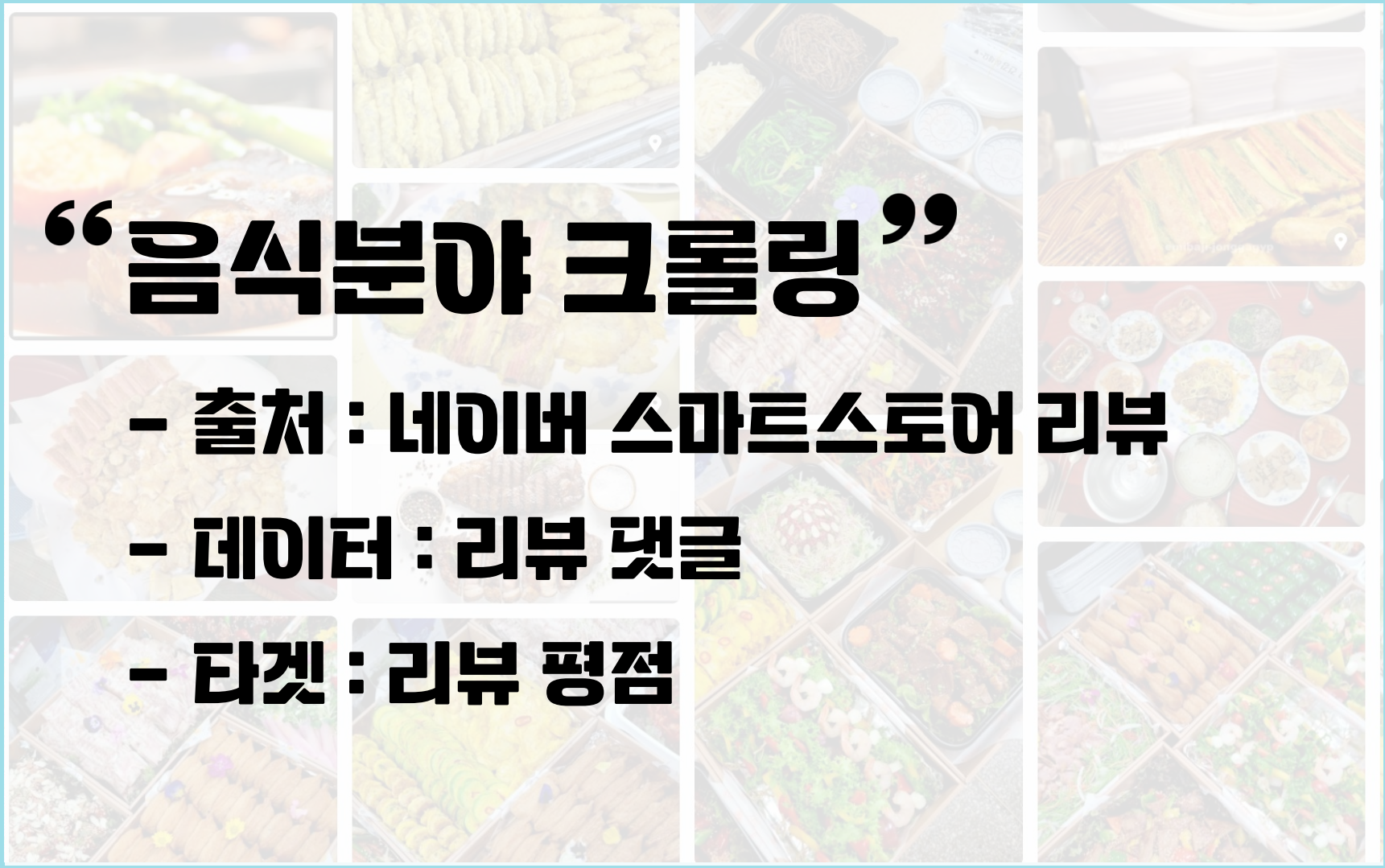










데이터 수집 및 처리



“음식분야 크롤링”

- 출처 : 네이버 스마트스토어 리뷰
- 데이터 : 리뷰 댓글
- 타겟 : 리뷰 평점

모델 선정 및 시연

“ 첫번째 모델링 과정:

- 모델 : LSTM (장단기기억망)
- 데이터 전처리 : TF-IDF 변환, 정규화
- 목적 : 리뷰를 통한 판매량 예측

모델1 문제 예시

```
# 모델 학습 (두 데이터셋을 합쳐서 사용)
model.fit(np.vstack((X_high, X_low)), np.hstack((y_high, y_low)), epochs=5, batch_size=32, validation_split=0.2)
```

[25] ✓ 187m 52.2s

```
Epoch 1/5
2549/2549 [==============================] - 2222s 866ms/step - loss: 3.2228 - accuracy: 0.6218 - val_loss: 1.3302 - val_accuracy: 0.0000e+00
Epoch 2/5
2549/2549 [==============================] - 2375s 932ms/step - loss: 1.0702 - accuracy: 0.6220 - val_loss: 1.3849 - val_accuracy: 0.0000e+00
Epoch 3/5
2549/2549 [==============================] - 2208s 866ms/step - loss: 1.0690 - accuracy: 0.6220 - val_loss: 1.3486 - val_accuracy: 0.0000e+00
Epoch 4/5
2549/2549 [==============================] - 2263s 888ms/step - loss: 1.0686 - accuracy: 0.6220 - val_loss: 1.4135 - val_accuracy: 0.0000e+00
Epoch 5/5
2549/2549 [==============================] - 2202s 864ms/step - loss: 1.0685 - accuracy: 0.6220 - val_loss: 1.3469 - val_accuracy: 0.0000e+00

<keras.src.callbacks.History at 0x29d829aca60>
```

첫번째 모델링 문제:

- 심한 과적합
- 피쳐 중요도 선정 문제
- 귀무가설 채택 :

"댓글만으로 판매량의 영향을 알 수 없다"

모델 선정 및 시연

모델 선정 및 시연

두번째 모델링 과정:

- 모델 : LinearRegression (선형회귀)
- 데이터 전처리 : TF-IDF 변환, 정규화
- 목적: 리뷰를 통한 평점 예측

모델 테스트

- 긍정 리뷰 :

"배달이 빠르고 괜찮네요"

- 평점 : 4.7

```
new_review = ["배달이 빠르고 괜찮네요"]  # 긍정적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

if predicted_rating > 5.0:
    print (f"예측된 리뷰평점: 5.0")
else:
    print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

✓ 0.0s

```
예측된 리뷰평점: 4.7
```

모델 테스트

- 긍정 리뷰 :

"신선하고 맛있어요!

아이들이 좋아하네요"

- 평점 : 4.9

```
new_review = ["신선하고 맛있어요! 아이들이 좋아하네요"]  # 긍정적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

if predicted_rating > 5.0:
    print (f"예측된 리뷰평점: 5.0")
else:
    print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

✓ 0.0s

```
예측된 리뷰평점: 4.9
```

모델 테스트

- 긍정 리뷰 :

"최고에요! 남편이 맛있다고 좋아해요!"

- 평점 : 5.0

```
new_review = ["최고에요! 남편이 맛있다고 좋아해요!"]  # 긍정적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

if predicted_rating > 5.0:
    print (f"예측된 리뷰평점: 5.0")
else:
    print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

2] ✓ 0.0s

```
예측된 리뷰평점: 5.0
```

모델 테스트

- 긍정 리뷰 :

"와이프가 맛있다고 계속 사자고 하네요"

- 평점 : 4.7

```
new_review = ["와이프가 맛있다고 계속 사자고 하네요"]  # 긍정적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

if predicted_rating > 5.0:
    print (f"예측된 리뷰평점: 5.0")
else:
    print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

13] ✓ 0.0s

예측된 리뷰평점: 4.7

모델 테스트

- 긍정 리뷰 :

"앞으로 종종 이용할게요"

- 평점 : 5.0

```
new_review = ["앞으로 종종 이용할게요"]  # 긍정적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

if predicted_rating > 5.0:
    print (f"예측된 리뷰평점: 5.0")
else:
    print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

0.0s

```
예측된 리뷰평점: 5.0
```

```
new_review =["신선하긴 한데 배달이 너무 늦게 왔어요... 다음부터 조금 더 빨리
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

✓ 0.0s

예측된 리뷰평점: 4.4

모델 테스트

- 중립 리뷰 :

"신선하긴 한데 배달이 너무 늦게 왔어요... 다음부터 조금 더 빨리 와줬으면 좋겠습니다"

- 평점 : 4.4

```
new_review =["배달은 빠른데 뭔가 맛이 떨어져요"] # 중립적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

✓ 0.0s

```
예측된 리뷰평점: 3.7
```

모델 테스트

- 중립 리뷰 :

"배달은 빠른데 뭔가 맛이 떨어져요"

- 평점 : 3.7

```
new_review =["괜찮네요"] # 중립적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

117] ✓ 0.0s

```
예측된 리뷰평점: 4.4
```

모델 테스트

- 중립 리뷰 :

"괜찮네요"

- 평점 : 4.4

```python
new_review =["그저 그렇네요.. 배달은 빨라서 좋았습니다"] # 중립적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

✓ 0.0s

```
예측된 리뷰평점: 3.9
```

모델 테스트

- 중립 리뷰 :

"그저 그렇네요..

배달은 빨라서 좋았습니다"

- 평점 : 3.9

```
new_review =["맛있지만 배달이 조금 늦었어요"] # 중립적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

✓ 0.0s

```
예측된 리뷰평점: 4.3
```

모델 테스트

- 중립 리뷰 :

"맛있지만 배달이

조금 늦었어요"

- 평점 : 4.3

모델 테스트

- 부정 리뷰 :

"맛없어요."

- 평점 : 1.0

```
new_review =["맛없어요."] # 부정적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

if predicted_rating < 0:
    print (f"예측된 리뷰평점: 1.0")
else:
    print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

1] ✓ 0.0s

```
예측된 리뷰평점: 1.0
```

모델 테스트

- 부정 리뷰 :

"식감이 너무 질겨서

먹기가 힘들어요"

- 평점 : 3.5

```
new_review =["식감이 너무 질겨서 먹기가 힘들어요"] # 부정적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

if predicted_rating < 0:
    print (f"예측된 리뷰평점: 1.0")
else:
    print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

0.0s

```
예측된 리뷰평점: 3.5
```

모델 테스트

- 부정 리뷰 :

"배달이 너무 늦고,

신선도가 떨어져요..

최악이에요"

- 평점 : 2.4

```
new_review =["배달이 너무 늦고, 신선도가 떨어져요.. 최악이에요"] # 부정적인 리
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

if predicted_rating < 0:
    print (f"예측된 리뷰평점: 1.0")
else:
    print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

25] ✓ 0.0s

예측된 리뷰평점: 2.4

모델 테스트

- 부정 리뷰 :

"배달도 늦고

이상한 냄새가 나요"

- 평점 : 2.4

```
new_review =["배달도 늦고 이상한 냄새가 나요"] # 부정적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

if predicted_rating < 0:
    print (f"예측된 리뷰평점: 1.0")
else:
    print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

123] ✓ 0.0s

예측된 리뷰평점: 2.4

모델 테스트

- 부정 리뷰 :

"이거 먹고 배탈 났습니다.

절대 사지 마세요!!!!"

- 평점 : 1.4

```
new_review =["이거 먹고 배탈 났습니다. 여러분 이거 절대 사지 마세요!!!!"] # 부정적인 리뷰
new_review_tfidf = vectorizer.transform(new_review)

# 모델을 사용하여 예측
predicted_rating = model.predict(new_review_tfidf)

if predicted_rating < 0:
    print (f"예측된 리뷰평점: 1.0")
else:
    print(f"예측된 리뷰평점: {round(predicted_rating[0], 1)}")
```

0.0s

```
예측된 리뷰평점: 1.4
```

모델 시연