**<page 15>**

- 신체 치수 정보를 이용해서 체형을 분류하는 서비스

- 세 가지 알고리즘을 사용하여 결과를 비교/분석

- 랜덤포레스트

- 특징:

1. 앙상블(ensemble): 뛰어난 분류 성능 + 과대적합에 안정적

2. 원리: 여러 개의 (깊은) 결정 트리의 평균을 내고, 다수결 투표를 통해 클레스 레이블을 선정하는 방법

3. 장점: 하이퍼파라미터 튜닝에 많은 노력X + 가지치기를 통한 과대 적합을 조정할 필요X

- 알고리즘 선택 이유

1. 다양한 신체 측정 정보처럼 수치형 특성과 범주형 특성이 혼합된 데이터를 다룰 때 뛰어난 성능

2. 각각의 결정 트리가 데이터의 다양한 부분집합과 특성 조합에 대해 학습하기 때문에, 신체 지표 간 복잡한 비선형 관계를 효과적으로 포착

3. 앙상블 방식으로 여러 결정 트리를 평균내는 구조 덕분에, 과대적합에 매우 강함

4. 특히, 체형 분류처럼 클래스 간 경계가 애매한 문제에서도 모델이 세밀하게 다양한 케이스를 학습하면서도 전체적인 일반화 성능을 유지

5. 하이퍼파라미터 튜닝이나 가지치기 과정 없이도 신체 지표만으로 바디 타입을 안정적으로 분류할 수 있다는 점

**<page 16>**

- K-Means

- 특징

1. 군집 알고리즘: 비슷한 객체로 이루어진 그룹을 찾는 기법

2. 같은 그룹 안에 객체들과의 연관성이 다른 그룹 안의 객체들보다 연관성이 높음

3. 구현하기 쉽고 다른 군집 알고리즘에 비해 계산 효율성이 높음

4. 군집의 목표: 특성의 유사도에 기초하여 샘플을 그룹으로 모으는 것

- 알고리즘 선택 이유

1. 사전에 레이블(label) 없이 신체 데이터(허리, 엉덩이, 가슴 등)만으로 사람들을 그룹화(clustering)

2. 각 사람을 가장 가까운 중심(centroid)으로 묶기 때문에, 비슷한 체형을 가진 사람들끼리 자연스럽게 하나의 군집(cluster) 으로 분류

3. 대용량 신체 데이터(수천~수만명)를 빠르게 클러스터링

4. 신체 치수들은 연속형(continuous) 수치 데이터라서, K-평균이 가정하는 "유클리드 거리 기반의 비슷함" 이 이 문제에 잘 맞음

5. 각 군집의 중심값(대표 체형)을 통해 "내 체형 vs 군집 대표 체형" 비교 분석이 직관적으로 가능해, 개인화 추천이나 특이체형 탐지에도 활용

- 랜덤포레스트와 k-means의 비교 이유:

1. 랜덤포레스트는 라벨을 기반으로 한 정확한 예측을 위해

2. K-means는 타겟 없는 군집화로 데이터 내 자연스러운 패턴을 파악하는 데 유리

--> 기존 공식으로 계산한 바디 타입과 군집화 결과의 일치를 평가하여 모델 성능을 검증하려는 목적

**<page 17>**

- 딥러닝 Multilayer Perceptron, MLP

- 특징

1. 딥러닝 기반 모델 (Feedforward Neural Network): 머신러닝의 하위 분야

2. 사람의 뇌가 어떻게 복잡한 문제를 푸는 지에 대한 가설과 모델 기반

3. 대표적인 피드포워드 인공 신경망: 각 층에서 입력을 순환시키지 않고 다음 층으로 전달한다는 의미

4. 입력 → 은닉층(hidden layer) → 출력층(output layer) 구조.

5. 복잡하고 비선형적인 데이터 관계를 스스로 학습

- 알고리즘 선택 이유

1. Weight, Waist, Bust, Height, BMI 같은 피처들은 단순한 선형관계가 아님.

2. MLP는 다양한 조합과 복잡한 패턴을 유연하게 학습할 수 있어.

3. 특히, 피처 수가 적당히 많고, 관계가 얽혀 있는 데이터셋에 아주 적합.

**<page 18>**

- 샘플 개수는 이만개

- 캐글에서 가져온 csv에서 기본적으로 제공하는 특성은 성별/몸무게/바디 타입 인덱스/허리/엉덩이/가슴/키/컵사이즈

- 데이터 전처리 1

1. 바디 쉐입 인덱스의 구성이 숫자, 즉 인덱스로 되어 있고 각 인덱스에 대한 타입 설명이 매핑이 되어 있지 않아 사용할 수 없으며, 0~4까지 구성되어 있는데 우리의 바디 타입은 총 7개라서 맞지 않아 새로운 클래스 레이블을 생성하는 방향으로 감

2. BMI는 기존에 존재하는 몸무게와 키의 특성을 이용해서 계산해서 새로운 컬럼으로 추가

**<page 19>**

- 데이터 전처리 2

3. 새로운 타겟인 바디 타입은 옆의 공식으로 만들어짐, 허리 수치,엉덩이 수치, 가슴 수치, 키 컬럼을 이용해서 계산하는데, 이는 현업에서 바디 타입을 계산하는 데 사용하고 있고, 논문을 참고함

4. 학습을 위해 데이터 타입을 object에서 숫자형으로 모두 인코딩, 성별은 여성은 0, 남성은 1/컵 사이즈는 순서가 존재하는 범주형 데이터라고 간주하고 매핑함

**<page 20>**

- 순서: 성별/몸무게/허리/엉덩이/가슴/키/컵사이즈/BMI/바디 타입(숫자로 인코딩)

- 옆에는 각 숫자로 인코딩된 바디 타입의 문자열 매핑 정보임

- 시각화 이유: 데이터 분포를 확인

**<page 21>**

- boxplot 간단 설명("중앙값", "범위", "이상치")

1. 중앙값 (Median): 데이터 정렬했을 때 중간값 (박스 안 가로선)

2. 1사분위수 (Q1): 하위 25% 지점

3. 3사분위수 (Q3): 상위 75% 지점

4. 박스 (Box): Q1 ~ Q3 사이 (데이터의 50%가 여기 들어있어)

5. 수염 (Whisker): 박스 밖의 데이터 중 정상 범위까지 선을 긋는 부분

6. 이상치 (Outlier): 수염 밖에 튀어나온 점들 (극단적인 값들)

- 순서: 성별/몸무게/허리/엉덩이

- 성별: 여성 데이터를 0으로 인코딩했고, 여성 데이터만 존재하므로 박스가 중앙에 위치됨을 볼 수 있음

- 나머지: 앞서 설명한 것처럼 수염 밖으로 튀어 나온 점들이 이상치임

- 이상치 개수

1. Gender : 0 : 없음

2. Weight : 178 : full bust(79), apple(33) 등

3. Waist : 327 : apple(171), pear(112) 등

4. Hips : 2 : apple(2)만

**<page 22>**

- 순서: 가슴/키/컵사이즈/BMI

- 이상치 개수

1. Bust/Chest : 560 : full bust(520), athletic(28) 등

2. Height : 0 : 없음

3. Cup Size : 0 : 없음

4. BMI : 233 : full bust(126), apple(43) 등

- 결론: 가슴둘레와 BMI 쪽에 이상치가 많고, 키나 컵사이즈는 거의 깨끗

- 비지도

1. 비지도 학습에서는 타겟(label)이 존재하지 않기 때문에, 각 클래스 레이블별로 이상치를 구분하거2. 나 제거할 수 없음.

3. 따라서, 비지도에서는 이 이상치 분석 결과를 가지고 \*\*특성 선택(feature selection)\*\*이나 \*\*이상치 전처리(outlier removal)\*\*를 따로 진행하지 않음.

- 지도

1. 지도 학습에서는 현재 분석한 이상치들을 중복 없이 모두 합쳤을 경우 약 1300개로, 전체 데이터 20000개 중 \*\*약 6.5%\*\*를 차지함.

2. 6.5%는 약간 신경 써야 하는 수준의 비율이지만, 데이터를 모두 삭제하기에는 20000개라는 데이터 크기가 충분히 크지 않다고 판단했음.

3. 이상치에 덜 민감한 랜덤포레스트(Random Forest)를 사용하여 이상치의 영향을 최소화하고,

이상치에 민감한 MLP(Multilayer Perceptron)도 함께 사용해 복잡한 피처 관계를 더 정밀하게 학습하고자 함.

**<page 23>**

(2) 결과 해석:

1. body\_type\_full bust, body\_type\_apple, body\_type\_pear, body\_type\_athletic, body\_type\_petite → precision, recall, f1-score 다 높다. (0.94~0.98) → 이 클래스들은 모델이 잘 맞췄다는 뜻!

2. body\_type\_hourglass, body\_type\_straight & narrow → precision, recall, f1-score가 0.00... → 이 클래스들은 거의 못 맞췄다는 의미야. (샘플 수가 매우 적거나 구분이 어려운 케이스)

- Accuracy (전체 정확도): 97%로 꽤 높음. (하지만 일부 소수 클래스는 무시된 효과일 수 있음)

- macro avg: 모든 클래스를 동일 가중치로 평균낸 것. (클래스 imbalance 고려)

- macro f1-score가 0.68 → 일부 클래스를 잘 못 맞춘 영향.

- weighted avg: 클래스별 sample 수를 고려해서 평균낸 것. (전체 평가 느낌)

- weighted f1-score가 0.96 → 데이터셋 전체적으로 보면 매우 잘 맞춘 것.

(1) diagonal (대각선):

- 대각선 숫자가 크면 클수록 좋은 모델!

- 대각선에 많은 숫자가 몰려 있으면 예측을 잘 했다는 의미야.

(2) off-diagonal (대각선 외):

- 대각선 바깥쪽에 숫자가 크면 → 모델이 헷갈려서 틀렸다는 뜻.

(3) 네 시각화에서 볼 수 있는 점:

- full bust, apple, pear 같은 클래스는 대각선 숫자가 큼 → 잘 맞춘 거야.

- hourglass, straight & narrow는 대각선에 거의 숫자가 없음 → 거의 예측 실패.

- 특히 hourglass는 샘플이 19개밖에 없었는데 아예 못 맞췄어.

3. 전체 요약

- 데이터 imbalance(특정 클래스 개수 차이)가 심함. → 그래서 일부 소수 클래스들은 예측이 거의 안 됨.

- 다수 클래스를 중심으로는 예측 성능이 매우 좋음.

- confusion matrix 시각화를 통해 어떤 클래스에서 성능이 좋은지/나쁜지 직관적으로 볼 수 있음.

- 따라서: 다수 클래스에는 강하고, 소수 클래스는 추가 보완 필요!

**<page 24>**

1. Silhouette Score: 0.1584

- Silhouette Score는 군집(clustering)이 얼마나 잘 되었는지를 평가하는 지표야.

- 값 범위: -1 ~ 1

- 1에 가까울수록 → 군집이 잘 구분되어 있다는 뜻

- 0에 가까울수록 → 군집이 애매하게 겹쳐 있다는 뜻

- 음수 → 군집을 아예 잘못 나눈 거

- 여기선 0.1584 → 군집이 어느 정도 형성되긴 했지만, 겹침이 꽤 많다는 의미(완벽하게 나뉘지는 않았다는 뜻)

2. 전체 데이터 기준 일치율 (Accuracy): 0.6262

- "k-means로 만든 군집"과 "기존에 주어진 body\_type"이 얼마나 일치하는지를 본 거야.

- 62.62% 일치 → 완벽하진 않지만, 절반 이상은 기존 바디타입 분류와 비슷하게 묶였다는 의미야.

- 비지도 학습(K-means)인데도 60% 이상 맞췄다는 건, 원래 데이터 안에 body\_type별로 어느 정도 구조적인 패턴이 존재했다는 걸 보여줘.

(1) 대표 body type 설명

- 0번 클러스터: full bust가 97%나 해당 → 군집 안에서 매우 순수한(homogeneous) 그룹.

- 4번 클러스터: full bust가 87% → 굉장히 일관성 있는 군집.

- 2번 클러스터: apple이 52% → 절반 정도는 apple인데, 나머지는 다른 타입도 섞임.

- 3번 클러스터: pear가 39% → pear 비율이 비교적 낮음, 많이 섞인 군집.

- 5번 클러스터: athletic 46% → 거의 절반 정도 athletic.

(2) 해석 요약

- 일부 클러스터는 특정 body\_type 중심으로 잘 뭉쳐졌고 일부는 다양한 body\_type이 섞여서 덜 뚜렷한 상태

- 특히 full bust는 여러 클러스터를 차지했는데, 아마 데이터셋 내 full bust 비율이 워낙 높아서 그럴 가능성 존재

- K-means로 군집화했을 때, 전체 데이터의 약 63%는 기존 body\_type과 어느 정도 일치했음.

- 특히 full bust 관련 클러스터는 매우 순수하게 묶였지만, pear나 athletic 같은 타입은 다소 섞이는 경향을 보임.

- Silhouette Score가 낮은 걸 보면, 바디타입 간 경계가 완전히 명확하지 않은 특징도 있음 → 즉, 데이터 안에 패턴은 존재하지만 완벽히 분리되지는 않는다는 걸 알 수 있음

**<page 25>**

1. Silhouette Score: 0.2934

- 군집화 품질 지표

- 값 범위는 -1 ~ 1 사이

- 1에 가까울수록 → 군집이 잘 분리됨

- 0에 가까울수록 → 군집이 겹침

- 0.2934는 이전(0.1584)보다 확실히 높아짐 → 즉, 군집들이 전보다 덜 겹치고 더 명확하게 구분됐다는 뜻

2. 전체 데이터 기준 일치율 (Accuracy): 0.6962

- k-means 군집 결과와 원래 body\_type 라벨이 얼마나 일치하는지 측정

- 69.62% 일치 →꽤 높은 편(70% 가까우면 비지도 군집으로는 매우 양호한 수준이라고 볼 수 있음)

- k-means로 라벨 없이 군집을 나눴는데도 약 70%는 기존 body type 분류와 일치했다는 뜻.

(1) 대표 body type 정리

- 3번 클러스터: full bust 비율이 93.4%, 거의 완벽하게 full bust로 구성된 군집.

- 0번, 1번 클러스터: 둘 다 full bust 주도, 각각 87.6%, 76.4%로 매우 높음.

- 4번 클러스터: apple 60% → 절반 이상은 apple이지만, 나머지는 다른 타입도 조금 섞여 있음.

- 5번 클러스터: pear가 55%로 중간 수준 일치.

- 6번 클러스터: athletic 43.8% → athletic이 대표지만, 섞여 있음.

- 2번 클러스터: apple인데 일치율 33%로 다소 낮음, 이 군집은 다양한 타입들이 더 많이 섞였을 가능성이 있음.

4. 최종 요약

- K-means 군집화 결과, 기존 바디타입과 약 70% 일치했으며, 특히 full bust 타입은 매우 깔끔하게 군집화

- 전체적으로 Silhouette Score도 이전보다 향상되어, 군집들이 더 명확해졌음을 알 수 있음

- apple, pear, athletic 타입은 어느 정도 섞이는 경향이 있었지만, 주요 바디타입 패턴은 데이터 안에 분명히 존재함을 확인할 수 있음

**<page 26>**

1. 전체적인 모델 구조

- MLP(다층 퍼셉트론) 모델

- Dense(128) → relu → Dropout(30%)

- Dense(64) → relu → Dropout(30%)

- Dense(7) → softmax

- 손실 함수: sparse\_categorical\_crossentropy (클래스 레이블을 숫자로 직접 줬을 때 사용

- 최적화 방법: Adam

- 조기 종료: EarlyStopping으로 과적합 방지 (validation loss가 5 epoch 동안 안 줄어들면 종료)

2. 모델 학습 결과

- Validation Set (Test Set) 기준 평가

- accuracy: 0.99

- weighted f1-score: 0.98

Confusion Matrix (혼동 행렬 시각화)

- 거의 모든 클래스가 대각선에 몰려 있음 → 즉, 모델이 정확하게 예측하고 있다는 뜻!

- 특히 큰 클래스들은 잘 맞췄고, 소수 클래스(3, 6번)는 조금 어려워했음.

Loss, Accuracy 그래프

- loss는 학습/검증 모두 부드럽게 감소했고,

- accuracy는 꾸준히 증가 → 오버피팅 없이 학습이 잘 된 것으로 보임.

3. Classification Report 해석

- 요약하면: 주요 클래스 (full bust, apple, pear, athletic)는 매우 잘 예측.

- 소수 클래스인 hourglass(3번)와 straight & narrow(6번)는 데이터가 적어서 recall이 낮음.

- 특히 hourglass는 19개밖에 없는데, recall 0.47 → 실제 hourglass를 절반 정도밖에 맞추지 못했어.

Macro 평균 (단순 평균):

- precision: 0.97, recall: 0.87, f1-score: 0.90 → 소수 클래스 영향 때문에 recall이 좀 낮음.

Weighted 평균 (데이터 수 고려):

- precision: 0.99, recall: 0.99, f1-score: 0.98 → 전체 데이터 기준 매우 훌륭!

4. 최종 해석 정리 ✏️

- MLP 기반 신경망 모델은 99%에 가까운 높은 정확도와 F1-score를 기록했으며, 주요 클래스 예측 성능이 매우 뛰어남

- 다만 데이터가 적은 소수 클래스(hourglass, straight & narrow)에서는 recall이 상대적으로 낮았으며, 이는 향후 데이터 증강이나 클래스 재분류 등의 보완이 필요할 수 있음

- 전체적으로 신경망 모델은 복잡한 패턴을 잘 학습했고, body type 분류 작업에 적합한 것으로 나타남

**<page 27>**

🎯 Random Forest - 지도학습 모델

- 데이터의 불균형 상황에서도 다수 클래스에 대해 높은 예측 성능을 보임

- 전체 정확도 약 97%, 주요 클래스에 대해 안정적인 Precision/Recall

- 이상치에 강한 특성을 갖고 있어 복잡한 전처리 없이도 견고한 결과

- 소수 클래스에 대한 성능은 낮음 → 향후 데이터 보완 필요

- Confusion Matrix를 통해 어떤 클래스가 약한지 명확히 시각화 가능

🌀 K-Means - 비지도 군집화

- 기존 바디타입 라벨 없이 데이터를 클러스터링

- 전체 일치율 약 70%, 특히 full bust 타입은 군집화 성능이 매우 우수

- Silhouette Score 0.29로 군집 간 경계가 비교적 명확

- 일부 apple, pear, athletic 유형은 혼재되는 경향

- ✅ 데이터에 내재된 바디타입 패턴이 존재함을 확인

🧠 MLP - 딥러닝 기반 분류

- 복잡한 피처 간 관계를 학습하며 가장 높은 성능 (정확도 99%)

- 주요 클래스에 대해 Precision, Recall, F1-score 모두 우수

- 신경망은 이상치에 민감하므로 적절한 전처리 또는 이상치 제거 필요

- 소수 클래스에 대한 성능(Recall)이 낮은 부분은 추후 보완 가능성 있음

- 📈 학습 곡선을 통해 과적합 여부와 학습 안정성도 시각적으로 확인

✅ 최종 요약

- Random Forest: 다수 클래스에 강하고, 이상치에 robust함

- K-Means: 실제 라벨 없이도 군집이 잘 형성됨 → 데이터 구조 확인

- MLP: 전체 성능 최상, 소수 클래스 대응력 향후 개선 여지 있음

- 전체적으로 바디타입 분류는 가능성이 매우 높은 과제이며, 다양한 모델을 통해 데이터 구조와 학습 방향성을 확인함

**<page 28>**

- 별점과 상품 리뷰를 가지고 리뷰를 긍정/부정 리뷰인지 분석하고, 리뷰에 맞게 사용자가 별점을 얼마나 일치하게 줬는지 판단하는 서비스

- DistilBERT

1. Distilled BERT의 줄임말

2. BERT 모델을 지식 증류(Knowledge Distillation) 기법을 이용해 경량화한 버전

- Distillation이란?

- 큰 모델(teacher)의 출력을 보고 작은 모델(student)이 그걸 모방해서 학습하는 방식

- DistilBERT는 BERT 모델을 teacher로 삼고, 다음 세 가지 loss를 동시에 학습:

1. 언어 모델링 loss (MLM) — 마스킹된 단어 예측

2. Teacher-Student의 soft label 차이 loss

3. Hidden state 유사성 유지 loss

- 장점

1. 가볍고 빠르면서도 성능 손실이 거의 없음

2. 모바일, 웹 등 리소스 제한 환경에 적합

3. Hugging Face Transformers에서 바로 사용 가능

- 고성능 Pretrained 모델 사용 : DistilBERT는 BERT의 축소판이지만, 여전히 매우 높은 자연어 이해 성능을 유지함.

- 경량화 모델로 빠른 처리 : BERT보다 가볍고 속도는 60% 더 빠름 → 대량 리뷰 데이터 분석에 적합

- Fine-tuned 감성 분류기 활용 : SST-2 데이터셋으로 감정 분류에 특화된 모델로 이미 학습되어 있어, 직접 학습 없이 바로 활용 가능

- 실제 리뷰와 감정 비교 가능 : 예측 결과를 별점(rating)과 비교해 정확도 평가가 가능함

- 토큰 자르기 & 예외 처리 포함 : 너무 긴 텍스트는 자르고, 에러 발생 시 예외처리를 통해 안정성 확보

- 분포 및 시각화 기능 강화 : 감정 분포, 감정 점수, 별점 비교 등을 시각적으로 분석 가능

- 왜 BERT 대신 DistilBERT를 썼나요?

1. 모델 크기 | 크고 무거움 | 40% 더 작음

2. 처리 속도 | 느림 | 60% 더 빠름

3. 성능 | 기준점 역할 | BERT 성능의 97% 유지

4. 구조 | 12-layer Transformer | 6-layer Transformer (Knowledge Distillation 기반)

5. 활용성 | 파워풀하지만 느림 | 실시간 분석, 대량 데이터에 적합

- 그래서 DistilBERT를 선택한 이유는?

1. 속도와 정확도를 모두 고려한 균형잡힌 선택

2. 리뷰 수천~수만 건을 분석할 때 실용성이 훨씬 높음

3. 감성 분석이라는 단순한 이진 분류 작업에는 DistilBERT로도 충분한 성능

**<page 29>**

- [CLS] 토큰: 전체 문장을 요약한 하나의 벡터

- Linear Layer: 출력 차원은 2개 (긍정/부정)

- Softmax: 0~1 사이의 확률로 변환

- 결과: 높은 쪽 클래스를 선택 (ex: Positive 확률 0.93 → POSITIVE 예측)

- DistilBERT 인코딩

1. Transformer 인코더 구조를 기반

2. 입력 문장을 벡터 형태로 변환해주는 과정

3. 핵심적으로 다음과 같은 단계

(1) 입력 토크나이징 (Tokenization): 문장을 단어 단위가 아닌 WordPiece 단위로 분할

(2) 입력 임베딩 생성 (Input Embedding): 각 토큰은 아래 3가지를 더해서 하나의 벡터로 변환됨:

- Token Embedding: 단어 의미

- Position Embedding: 문장에서의 위치 정보

- Segment Embedding: 문장이 A인지 B인지 (문장이 2개 있을 경우)

(3) Transformer 인코더 (Self-Attention): 여러 층으로 구성된 인코더가 입력 임베딩을 처리

- 각 단어가 문장의 다른 단어들과 어떤 관계가 있는지를 Self-Attention으로 계산

- DistilBERT는 BERT보다 층 수가 적고, 중간 레이어 생략해서 속도를 빠르게 함

- 각 층은:

Multi-head Self-Attention

Feed Forward Layer

Residual Connection + Layer Normalization

(4) [CLS] 임베딩 추출: 문장의 요약 정보가 담긴 [CLS] 토큰의 최종 벡터를 사용

- 이 벡터가 문장의 전체 감성을 대표함

- 핵심 포인트: DistilBERT는 BERT의 약 60% 파라미터로 비슷한 성능을 냄

- 인코딩 결과: 문장당 하나의 벡터 ([CLS])\*\*로 요약 가능

- 이 벡터는 후속 레이어(Linear → Softmax 등)로 전달되어 최종 예측에 사용됨

1. 입력 문장: 자연어 문장 하나를 입력으로 넣는 거야.

- 영어 문장 기준, 단어들을 토큰화(tokenization) 해서 DistilBERT가 이해할 수 있는 형태로 바꿔줌

- ([CLS]는 문장 시작, [SEP]는 문장 끝 표시)

2. DistilBERT 인코딩

- 이 토큰들을 숫자 벡터(임베딩)로 바꾸고, 이 벡터를 여러 층의 Transformer 인코더에 넣음

- DistilBERT는 BERT보다 가볍고 빠르지만, 문맥 이해력은 거의 비슷

- 핵심 구조는:

Multi-Head Self-Attention → 각 단어가 다른 단어들과 얼마나 관련 있는지 계산

Feed Forward Layer → 비선형 변환

- 레이어가 쌓이면서 단어 간의 관계를 점점 더 깊이 이해함

3. [CLS] 임베딩

- DistilBERT의 출력에서, \*\*첫 번째 토큰인 [CLS]\*\*의 벡터는 전체 문장을 대표

- 이 [CLS] 임베딩 하나만 뽑아서 문장 전체의 요약 벡터로 사용

4. Linear Layer (분류기)

- [CLS] 벡터는 고차원(예: 768차원), 필요한 건 두 가지 감성(긍정/부정) 중 하나를 선택

- 이 벡터를 \*\*Linear Layer(선형 계층)\*\*에 통과시켜서 출력 차원을 2로 줄임

5. Softmax

- Linear Layer의 결과는 그냥 스코어(score)일 뿐이라, 이걸 확률로 바꾸기 위해 Softmax를 씌움

- Softmax 결과: [0.93, 0.07] → 두 클래스의 확률 합이 1이 되도록 정규화

- 첫 번째 값이 긍정, 두 번째가 부정일 수도 있어 (모델 설계에 따라 순서 다름)

6. 결과 선택: 확률이 더 높은 쪽을 최종 예측으로 선택

**<page 30>**

1. 감성 분석 모델 불러오기

- pipeline("sentiment-analysis"): 감성 분석을 위한 미리 학습된 파이프라인 생성.

- distilbert-base-uncased-finetuned-sst-2-english: 영화 리뷰로 유명한 SST-2 데이터셋으로 학습된

- DistilBERT 모델 사용. 긍정 / 부정 둘 중 하나로 분류

- framework="pt": PyTorch 기반 모델 사용.

- 이 모델은 영어 문장을 입력으로 받아 {"label": "POSITIVE" or "NEGATIVE", "score": 확신도} 형태로 반환

2. 리뷰 데이터 불러오기

- CSV 파일에서 리뷰 데이터를 불러옴

- review\_text가 없는 행(NaN)은 제거해서 감성 분석할 수 있는 데이터만 남김

3. 텍스트 전처리

- 혹시 모르게 숫자나 NaN이 남아 있는 경우를 대비

- 문자열(str) 타입이 아닌 값은 빈 문자열로 처리.

4. 감성 분석 안전 처리 함수 정의

- 텍스트가 비어 있거나 문자열이 아니면 분석을 건너뜀.

- 너무 긴 텍스트는 모델이 한 번에 처리할 수 있는 길이보다 많을 수 있으므로 1000자 제한을 걸어 앞부분만 사용.

- 모델 실행 중 오류가 날 수도 있으니 try-except로 안전하게 처리.

5. 전체 데이터에 감성 분석 적용

- review\_text 하나씩 감성 분석.

- 결과에서:

(1) label: 'POSITIVE', 'NEGATIVE' 중 하나

(2) score: 모델의 확신도

(3) 숫자 레이블도 추가: 1 = 긍정/0 = 부정/-1 = 알 수 없음 (예: 빈 텍스트)

6. 결과를 데이터프레임에 추가: 각 리뷰마다 감성 분석 결과를 새로운 컬럼으로 저장

- sentiment\_score: 확신도

- sentiment\_binary: 1 / 0 / -1

**<page 31>**

✅ 감성 분석 결과 요약 및 인사이트

- 총 20만 개의 리뷰 데이터를 대상으로 감성 분석을 수행한 결과, 리뷰에 포함된 문장의 감정과 사용자가 부여한 별점 사이의 일치 여부를 평가함.

- 분석 기준: 긍정 감정 → 별점 8점 이상/부정 감정 → 별점 8점 미만

🎯 분석 결과

- 감정 예측과 별점이 일치한 비율은 약 85.47% → 즉, 전체 리뷰 중 약 17만 건이 리뷰 내용과 별점이 잘 부합함.

📌 인사이트

- 사용자의 리뷰 텍스트와 별점은 높은 상관관계를 보이며, 이를 기반으로 신뢰할 수 있는 상품 평가가 가능함을 시사함.

- 따라서 별점 기준(예: 8점 이상)으로 상품을 추천하는 방식은 실제 사용자 만족도를 잘 반영하는 방법이라고 볼 수 있음.

🔍 결론

- 리뷰 감정과 별점 간의 높은 일치도를 바탕으로, 8점 이상인 상품을 우선적으로 추천하는 전략이 타당하며, 이는 사용자 경험 향상 및 추천 시스템의 정확도를 높이는 데 기여할 수 있음.