**<page 14>**

- 신체 치수 정보를 이용해서 체형을 분류하는 서비스: 세 가지 알고리즘을 사용하여 결과를 비교/분석

- 랜덤포레스트 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 15>**

- K-Means 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 16>**

- 딥러닝 Multilayer Perceptron, MLP 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 17>**

- 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 18>**

- 데이터 전처리 2

3. 새로운 타겟인 바디 타입은 옆의 공식으로 만들어짐, 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 19>**

- 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 20>**

- 순서: 성별/몸무게/허리/엉덩이

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

**<page 21>**

- 순서: 가슴/키/컵사이즈/BMI

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

- 비지도

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

2. 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 22>**

(2) 결과 해석

- 전체 요약

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

2. 다수 클래스를 중심으로는 예측 성능이 매우 좋음.

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

4. 따라서: 다수 클래스에는 강하고, 소수 클래스는 추가 보완 필요

**<page 23>**

1. 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 24>**

1. 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 25>**

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)

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 26>**

✅ 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 27>**

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

- DistilBERT

- 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

- 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 28>**

1. 입력 문장: 자연어 문장 하나를 입력으로 넣는 거야.

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

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

- 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

6. 결과 선택: 확률이 더 높은 쪽을 최종 예측으로 선택

**<page 29>**

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

**<page 30>**

✅ 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.