# 범주형 데이터와 인코딩 기법

2025. 05. 12(월)

# 학습 목표

01 범주형 데이터의 개념과 유형을 이해합니다.

02 주요 범주형 데이터 인코딩 기법을 학습합니다.

03 각 인코딩 기법의 장단점과 적용 시점을 비교합니다.

# 목차

# 범주형 데이터 소개

# 범주형 데이터란?

- 유한한 수의 범주 또는 그룹으로 나눌 수 있는 데이터
- 수치형 데이터와 대조적
- 예시: 성별, 혈액형, 국가, 제품 카테고리, 설문 응답 등
- 머신러닝 적용 전 적절한 전처리 필수

# 범주형 데이터의 유형

# 주요 유형

- 명목형 (Nominal)
  - 범주 간 순서 없음
  - 예: 색상, 혈액형, 국가
- 순서형 (Ordinal)
  - 범주 간 의미 있는 순서 있음
  - 예: 학점 (A>B>C), 만족도 (매우 만족 > 만족)
- 수치형 (참고)
  - 구간형 (Interval): 순서 O, 간격 O, 절대 0점 X (예: 섭씨 온도)
  - 비율형 (Ratio): 순서 O, 간격 O, 절대 0점 O (예: 길이, 무게, 소득)

# 데이터 유형 비교

| 데이터 유형 | 순서 | 간격 | 절대 0점 | 예시 |
|---|---|---|---|---|
| 명목형 (Nominal) | X | X | X | 색상, 혈액형 |
| 순서형 (Ordinal) | O | X | X | 학점, 만족도 |
| 구간형 (Interval) | O | O | X | 섭씨 온도 |
| 비율형 (Ratio) | O | O | O | 길이, 무게, 나이 |

# 전처리 필요성

# 왜 전처리가 필요한가?

- 대부분의 머신러닝 알고리즘은 수치형 데이터를 입력으로 기대
- 범주형 데이터를 그대로 사용 시 오류 발생 또는 성능 저하
- 범주형 데이터 인코딩: 범주형 데이터를 수치화하는 과정

# 전처리의 목적

- 모델 입력 요구사항 충족: 알고리즘이 데이터를 처리 가능하게 함
- 모델 성능 향상: 데이터 정보를 효과적으로 학습하게 함
- 데이터 분석 용이성: 통계/시각화 등에 쉽게 활용

# 인코딩 기법 개요

# 주요 인코딩 기법

- 레이블 인코딩 (Label Encoding)
  - 각 범주에 고유한 정수 할당
- 원-핫 인코딩 (One-Hot Encoding)
  - 각 범주를 이진 벡터로 변환
- 순서형 인코딩 (Ordinal Encoding)
  - 범주 순서를 반영하여 정수 할당

# 레이블 인코딩

# 레이블 인코딩 (Label Encoding)

- 개념: 범주형 데이터를 순차적인 정수로 매핑
- 예시: `['사과', '바나나', '체리']` -> `[1, 0, 2]` (알파벳 순)
- 장점: 간단하고 구현 용이, 데이터 차원 축소
- 단점: 명목형 데이터에 적용 시 순서 정보 왜곡 가능성 높음

# 레이블 인코딩 예제 (Python)

```python
from sklearn.preprocessing import LabelEncoder

fruits = ["사과", "바나나", "체리", "사과", "바나나"]

encoder = LabelEncoder()
encoded_fruits = encoder.fit_transform(fruits)

print("원본:", fruits)
print("인코딩:", encoded_fruits)
print("매핑:", dict(zip(encoder.classes_, encoded_fruits)))

# 역변환
decoded_fruits = encoder.inverse_transform(encoded_fruits)
print("역변환:", decoded_fruits)
```

# 레이블 인코딩 적용 시점

- 순서형 데이터: 만족도, 학점 등 순서가 중요한 경우
- 트리 기반 모델: 의사결정나무, 랜덤 포레스트 등 순서 정보 왜곡에 덜 민감한 모델
- Target 변수 인코딩: 분류 문제의 라벨 변수

# 엎-핫 인코딩

# 원-핫 인코딩 (One-Hot Encoding)

- 개념: 각 범주를 새로운 이진 특성(열)으로 변환
- 해당 범주에 속하면 1, 아니면 0
- 예시: '빨강' -> `[1, 0, 0]`, '파랑' -> `[0, 1, 0]`
- 주로 명목형 데이터에 사용

# 원-핫 인코딩 작동 방식

- 데이터의 고유 범주 식별
- 각 고유 범주에 대한 새로운 이진 열 생성
- 해당 범주 값 위치에 1 할당, 나머지는 0

| 색상 | 빨강 | 파랑 | 초록 |
|---|---|---|---|
| 빨강 | 1 | 0 | 0 |
| 파랑 | 0 | 1 | 0 |
| 초록 | 0 | 0 | 1 |

# 원-핫 인코딩 예제 (pandas)

```
import pandas as pd

data = {'색상': ['빨강', '파랑', '초록', '빨강', '파랑'],
        '크기': ['S', 'M', 'L', 'M', 'S']}
df = pd.DataFrame(data)

print("원본 DataFrame:\n", df)

# One-Hot 인코딩 적용
df_encoded = pd.get_dummies(df, columns=['색상'])

print("\nOne-Hot 인코딩된 DataFrame:\n", df_encoded)
```

# 원-핫 인코딩 예제 (scikit-learn)

```
from sklearn.preprocessing import OneHotEncoder
import numpy as np

data = np.array(['빨강', '파랑', '노랑']).reshape(-1, 1)

encoder = OneHotEncoder(sparse_output=False)
encoded_data = encoder.fit_transform(data)

print("원본:\n", data)
print("\n인코딩:\n", encoded_data)
print("범주:\n", encoder.categories_)
```

# 원-핫 인코딩 장단점

- 장점:
  - 명목형 데이터 처리 용이
  - 범주 간 독립성 보장 (모델이 순서 오해 방지)
- 단점:
  - 차원 증가: 범주 수 많을 시 차원의 저주 발생 가능
  - 희소성 문제: 대부분 0인 데이터 생성 (메모리, 성능)
  - 다중공선성: (일부 모델에서) 완벽한 선형 종속성 발생 가능

# 원-핫 인코딩 활용 사례

- 머신러닝 모델 입력: 대부분의 모델에 명목형 Feature 입력 시 필수
- 텍스트 마이닝/NLP: BoW (Bag of Words) 등에서 단어 표현
- 추천 시스템: 사용자/상품 속성 인코딩
- 이미지 처리: 특정 속성(객체 종류) 인코딩

# 순서형 인코딩

# 순서형 인코딩 (Ordinal Encoding)

- 개념: 순서형 데이터의 순서 정보를 보존하며 정수 할당
- 예시: `['매우 불만족', '만족', '보통']` -> `[0, 3, 2]` (순서 정의 필요)
- 장점: 순서 정보 보존, 낮은 차원 유지
- 단점: 범주 간 간격이 동일하지 않음, 명목형에 잘못 적용 시 문제

# 순서형 인코딩 예제 (Python)

```
from sklearn.preprocessing import OrdinalEncoder
import numpy as np

data = np.array(['매우 불만족', '불만족', '보통']).reshape(-1, 1)

# 범주 순서 정의 (중요!)
categories_order = [['매우 불만족', '불만족', '보통', '만족', '매우 만족']]

encoder = OrdinalEncoder(categories=categories_order)
encoded_data = encoder.fit_transform(data)

print("원본:\n", data)
print("\n인코딩:\n", encoded_data)
print("범주 순서:\n", encoder.categories_)
```

*(Note: Example data '매우 불만족', '불만족', '보통' is subset of defined categories_order)*

# 인코딩 시 고려사항

# 중요 고려사항 (1)

- 데이터 유형: 명목형 vs 순서형 정확히 파악
- 모델 특성: 선형/신경망 vs 트리 기반 모델 (순서 민감도 다름)
- 범주의 수 (Cardinality): 범주 많을 시 차원 증가 문제 (원-핫)
- 데이터 희소성: 원-핫 인코딩 결과 확인 및 처리

# 중요 고려사항 (2)

- 정보 손실: 인코딩 과정에서 정보 손실 최소화 노력
- 데이터 누수 방지:
  - 훈련/테스트 데이터 분리 후 인코딩
  - `fit`은 훈련 데이터에만
  - `transform`은 훈련/테스트 모두에 적용

# 요약 정리

# 학습 요약 정리

- 범주형 데이터는 유한 범주 데이터이며, 머신러닝 적용 전 전처리 필요
- 레이블 인코딩은 정수 매핑 (순서 왜곡 주의)
- 원-핫 인코딩은 이진 벡터 변환 (명목형에 적합, 차원/희소성 주의)
- 순서형 인코딩은 순서 반영 정수 매핑 (순서형에 적합)