연관 규칙 분석 (Association Rule Mining)

- 대규모 데이터 항목 집합(e.g., 마트 거래 기록, 웹사이트 방문 기록) 속에서 항목들 간의 유용한 관계, 즉 '연관 규칙'을 발견하는 비지도 학습 방법
- 가장 대표적인 예시는 "장바구니 분석"으로, "만약 고객이 '기저귀'를 구매했다면, '맥주'도 함께 구매할 가능성이 높다"와 같은 규칙을 찾아내는 것

연관 규칙은 **'만약 {A}이면, {B}이다' ($A \rightarrow B$)** 형태를 가집니다.

- **A**: 선행 항목 (Antecedent)
- **B**: 후행 항목 (Consequent)

이러한 규칙의 유용성을 평가하기 위해 다음 세 가지 핵심 지표가 사용됩니다.

1. **지지도 (Support)**:
    - **개념**: 전체 거래 중, 항목 A와 항목 B를 **모두 포함**하는 거래의 비율.
    - **수식**: $Support(A \rightarrow B) = \frac{P(A \cap B)}{전체 거래 수}$
    - **의미**: 규칙이 데이터에서 얼마나 자주 발생하는지를 나타냅니다. 지지도가 너무 낮으면 우연히 발생한 규칙일 가능성이 높습니다.
2. **신뢰도 (Confidence)**:
    - **개념**: 항목 A를 포함하는 거래 중에서, 항목 B도 **함께 포함**하는 거래의 비율 (조건부 확률).
    - **수식**: $Confidence(A \rightarrow B) = \frac{P(A \cap B)}{P(A)} = \frac{Support(A, B)}{Support(A)}$
    - **의미**: A를 구매했을 때 B를 구매할 조건부 확률을 나타냅니다. 규칙의 신뢰성을 측정합니다.
3. **향상도 (Lift)**:
    - **개념**: 항목 B를 단독으로 구매할 확률 대비, 항목 A를 구매했을 때 항목 B를 구매할 확률의 증가 비율.
    - **수식**: $Lift(A \rightarrow B) = \frac{Confidence(A \rightarrow B)}{Support(B)} = \frac{P(B|A)}{P(B)}$
    - **의미**:
        - `Lift > 1`: A와 B는 양의 상관관계를 가집니다. (A를 구매하면 B를 구매할 확률이 높아짐)
        - `Lift = 1`: A와 B는 서로 독립적입니다. (A 구매가 B 구매에 영향을 주지 않음)
        - `Lift < 1`: A와 B는 음의 상관관계를 가집니다. (A를 구매하면 B를 구매할 확률이 낮아짐)

예제 데이터 생성

`scikit-learn`에는 Apriori, FP-growth 알고리즘이 직접 포함되어 있지 않아, 외부 라이브러리인 `mlxtend`를 사용하는 것이 일반적이다.

```
# !pip install mlxtend
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

# 1. 데이터 준비 (리스트의 리스트 형태)
dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
```

```
            ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
            ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
            ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
            ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

# 2. 데이터를 트랜잭션 형태로 변환
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
print("--- Transaction Data ---")
print(df)
'''
   Apple   Corn   Dill   Eggs  Ice cream  Kidney Beans   Milk  Nutmeg  Onion
Unicorn  Yogurt
0  False  False  False   True      False          True   True    True   True
False    True
1  False  False   True   True      False          True  False    True   True
False    True
2   True  False  False   True      False          True   True   False  False
False   False
3  False   True  False  False      False          True   True   False  False
True    True
4  False   True  False   True       True          True  False   False   True
False   False
'''
```

주의사항

- **데이터 형태**
  - Apriori, FP-growth 알고리즘을 사용하려면 입력 데이터를 **트랜잭션(Transaction) 형태**로 변환 필수
  - 각 행은 하나의 거래(transaction)를, 각 열은 하나의 항목(item)을 나타냄
  - 해당 거래에 항목이 포함되었는지 여부를 `True/False` 또는 `1/0`으로 표시하는 형태
- **최소 지지도/신뢰도 설정**
  - `min_support`와 `min_confidence` 값을 어떻게 설정하느냐에 따라 결과의 양과 질이 크게 달라짐
  - 너무 낮게 설정하면 의미 없는 규칙이 너무 많이 생성되고, 너무 높게 설정하면 유용한 규칙을 놓칠 수 있음

Apriori 알고리즘

- `Apriori`는 연관 규칙 분석을 위한 가장 대표적인 알고리즘
- 모든 가능한 항목 집합을 탐색하는 것은 계산적으로 불가능하므로, `Apriori`는 **'빈번하지 않은 항목 집합의 부분 집합 또한 빈번하지 않다'**는 원리를 사용하여 탐색 범위를 효율적으로 줄임
- **동작 방식**:

  1. 사용자가 지정한 **최소 지지도(minimum support)**를 설정합니다.
  2. 최소 지지도를 넘는 개별 항목들(1-itemset)만 찾습니다.
  3. 1단계에서 찾은 항목들을 조합하여 2-itemset을 만들고, 이 중에서 최소 지지도를 넘는 것들만 찾습니다.

4. 이 과정을 반복하여 최소 지지도를 만족하는 모든 **빈번한 항목 집합(frequent itemsets)**을 찾습니다.
5. 찾아낸 빈번한 항목 집합들을 기반으로, 사용자가 지정한 **최소 신뢰도(minimum confidence)**를 넘는 연관 규칙들을 생성합니다.

적용 가능한 상황

- 마트, 온라인 쇼핑몰 등에서 고객의 구매 패턴을 분석하여 교차 판매(cross-selling) 전략을 수립할 때.
- 웹사이트 로그 분석을 통해 사용자의 페이지 이동 경로를 파악하고 웹사이트 구조를 개선할 때.
- 의료 데이터에서 특정 질병과 증상 간의 연관성을 분석할 때.

```
# Apriori 알고리즘을 이용해 빈번한 항목 집합 찾기
# min_support: 최소 지지도.
# use_colnames=True: 항목 이름을 컬럼명으로 사용.
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)
print("\n--- Frequent Itemsets (min_support >= 0.6) ---")
print(frequent_itemsets)
'''
    support                      itemsets
0       0.8                       (Eggs)
1       1.0                (Kidney Beans)
2       0.6                       (Milk)
3       0.6                      (Onion)
4       0.6                     (Yogurt)
5       0.8          (Eggs, Kidney Beans)
6       0.6                 (Onion, Eggs)
7       0.6          (Milk, Kidney Beans)
8       0.6         (Onion, Kidney Beans)
9       0.6        (Yogurt, Kidney Beans)
10      0.6  (Onion, Eggs, Kidney Beans)
'''

# 연관 규칙 생성
# metric: 규칙을 필터링할 기준 ('confidence', 'lift' 등).
# min_threshold: metric의 최소 임계값.
rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=0.7)
print("\n--- Association Rules (min_confidence >= 0.7) ---")
# antecedents: 선행 항목, consequents: 후행 항목
# antecedent support, consequent support: 각 항목 집합의 지지도
# support, confidence, lift: 규칙의 지지도, 신뢰도, 향상도
# leverage, conviction: 다른 연관성 척도
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
'''
             antecedents            consequents  support  confidence  lift
0                 (Eggs)         (Kidney Beans)      0.8        1.00  1.00
1          (Kidney Beans)                 (Eggs)      0.8        0.80  1.00
2                (Onion)                 (Eggs)      0.6        1.00  1.25
3                 (Eggs)                (Onion)      0.6        0.75  1.25
4                 (Milk)         (Kidney Beans)      0.6        1.00  1.00
5                (Onion)         (Kidney Beans)      0.6        1.00  1.00
6               (Yogurt)         (Kidney Beans)      0.6        1.00  1.00
```

```
7          (Onion, Eggs)         (Kidney Beans)      0.6        1.00  1.00
8   (Onion, Kidney Beans)                 (Eggs)      0.6        1.00  1.25
9    (Eggs, Kidney Beans)                (Onion)      0.6        0.75  1.25
10               (Onion)   (Eggs, Kidney Beans)      0.6        1.00  1.25
11                (Eggs)  (Onion, Kidney Beans)      0.6        0.75  1.25
'''
```

결과 해석

- `frequent_itemsets`를 통해 최소 지지도가 0.6 이상인, 함께 자주 팔리는 품목을 산출
  - `(Eggs) support 0.8` : 전체 거래의 80%에서 달걀이 포함됨
  - `(Kidney Beans) support 1.0` : 모든 거래(100%)에 강낭콩가 포함됨
  - `(Onion, Eggs, Kidney Beans) support 0.6` : 전체 거래의 60%에서 달걀과 양파, 강낭콩이 함께 구매됨
- `rules`는 `association_rules`를 통해, 신뢰도(`antecedents` 구매 시 `consequents`를 구매할 조건부 확률)가 0.7 이상인 연관 규칙만 추출한 데이터
  - `(Eggs) → (Kidney Beans)` : 달걀 구매 시 강낭콩 구매율 100%
    - 향상도(lift)가 1이므로 서로 독립이다. (연관 없을 가능성도 높다.)
  - `(Onion) → (Eggs)` : 양파 구매 시 달걀 구매율 100%
    - 향상도(lift)가 1.25이므로, 일반적인 경우보다 25% 더 자주 함께 구매됨

장단점 및 대안

- **장점**:
  - 결과가 'If-Then' 형태의 규칙으로 나와 해석이 매우 직관적이고 비즈니스에 적용하기 쉽습니다.
  - 비지도 학습으로, 데이터의 숨겨진 패턴을 발견하는 데 유용합니다.
- **단점**:
  - 데이터셋이 크거나 항목의 종류가 많아지면 가능한 모든 조합을 탐색해야 하므로 계산량이 기하급수적으로 증가합니다.
  - `min_support` 설정에 따라 성능과 결과가 민감하게 변합니다.

FP-Growth 알고리즘

- `FP-Growth`는 연관 규칙 분석에서 `Apriori`의 계산 복잡도를 개선하기 위해 제안된 알고리즘
- `Apriori`는 후보 항목 집합(candidate itemsets)을 반복적으로 생성해야 하지만,
  `FP-Growth`는 이를 생략하고 **트랜잭션 데이터를 트리 형태로 압축(FP-Tree)** 하여 효율적으로 빈번항목집합을 탐색
- 결과적으로, **Apriori보다 훨씬 빠른 수행 속도**와 **적은 메모리 사용량**을 보임
- **동작 방식**:
  1. **트랜잭션 데이터를 FP-Tree로 변환**
     - 각 거래(Transaction)의 항목들을 지지도 순으로 정렬하여 트리 형태로 저장합니다.
     - 공통 부분이 있는 거래는 트리에서 경로를 공유함으로써 중복을 최소화합니다.
  2. **빈번한 패턴 탐색 (Frequent Pattern Mining)**
     - 트리의 각 항목에 대해 조건부 패턴 트리(Conditional Pattern Tree)를 생성하고, 그 안에서 재귀적으로 빈번항목집합을 추출합니다.

3. **연관 규칙 생성**
    - `Apriori`와 동일하게, 최소 신뢰도(`min_confidence`) 기준을 만족하는 규칙만 필터링합니다.

적용 가능한 상황

- 거래 데이터가 크거나 항목의 종류가 많은 경우 (`Apriori`의 연산량이 폭발하는 경우)
- 고객 구매 이력, 웹 로그, 의료 데이터 등에서 **대규모 데이터의 연관 패턴 탐색**이 필요한 경우
- 데이터가 메모리에 올릴 수 있을 정도로 크지 않고, **트리 구조로 압축할 수 있을 때**

```
# FP-Growth 알고리즘으로 빈번항목집합 찾기
frequent_itemsets = fpgrowth(df, min_support=0.6, use_colnames=True)
print("\n--- Frequent Itemsets (min_support >= 0.6) ---")
print(frequent_itemsets)
'''
    support                      itemsets
0       1.0                 (Kidney Beans)
1       0.8                         (Eggs)
2       0.6                       (Yogurt)
3       0.6                        (Onion)
4       0.6                         (Milk)
5       0.8           (Eggs, Kidney Beans)
6       0.6         (Yogurt, Kidney Beans)
7       0.6                  (Onion, Eggs)
8       0.6          (Onion, Kidney Beans)
9       0.6    (Onion, Eggs, Kidney Beans)
10      0.6           (Milk, Kidney Beans)
'''

# 연관 규칙 생성
rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=0.7)
print("\n--- Association Rules (min_confidence >= 0.7) ---")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
'''
              antecedents            consequents  support  confidence  lift
0                  (Eggs)          (Kidney Beans)      0.8        1.00  1.00
1           (Kidney Beans)                  (Eggs)      0.8        0.80  1.00
2                 (Yogurt)          (Kidney Beans)      0.6        1.00  1.00
3                  (Onion)                  (Eggs)      0.6        1.00  1.25
4                  (Eggs)                 (Onion)      0.6        0.75  1.25
5                  (Onion)          (Kidney Beans)      0.6        1.00  1.00
6            (Onion, Eggs)          (Kidney Beans)      0.6        1.00  1.00
7   (Onion, Kidney Beans)                  (Eggs)      0.6        1.00  1.25
8    (Eggs, Kidney Beans)                 (Onion)      0.6        0.75  1.25
9                  (Onion)    (Eggs, Kidney Beans)      0.6        1.00  1.25
10                 (Eggs)   (Onion, Kidney Beans)      0.6        0.75  1.25
11                 (Milk)          (Kidney Beans)      0.6        1.00  1.00
'''
```

결과 해석

- `frequent_itemsets`를 통해 최소 지지도가 0.6 이상인, 함께 자주 팔리는 품목을 산출
- `rules`는 `association_rules`를 통해, 신뢰도(`antecedents` 구매 시 `consequents`를 구매할 조건부 확률)가 0.7 이상인 연관 규칙만 추출한 데이터

주의사항

- FP-Growth는 **FP-Tree를 메모리에 저장**하므로, 데이터가 매우 크면 메모리 한계에 부딪힐 수 있습니다.
- `min_support`를 너무 낮게 설정하면 FP-Tree 크기가 커져 속도가 느려질 수 있습니다.
- 데이터가 희소하고 항목 종류가 너무 많으면 Apriori보다 큰 이점을 보지 못할 수 있습니다.

장단점 및 비교

구분	Apriori	FP-Growth
핵심 아이디어	후보 항목집합을 생성하고 검증	FP-Tree로 거래 데이터를 압축하여 탐색
속도	느림 (조합 폭발 문제)	빠름 (트리 구조로 압축 탐색)
메모리 사용	상대적으로 많음	중복 거래를 공유하여 절약
결과 동일성	동일한 최소지지도를 사용하면 결과는 같음	동일한 결과 산출
적합한 경우	소규모, 직관적 분석	대규모 거래 데이터, 고속 분석 필요 시