

Assignment #1

|  |  |
| --- | --- |
| 과목명. | 데이터사이언스 |
| 담 당. | 김 상 욱 교수님 |
| 제출일. | 2021년 03월 20일 |
| 공과대학 | 컴퓨터소프트웨어학부 |
| 학 번 | 2016025969 |
| 이 름. | 정지훈 |

****

**목차.**

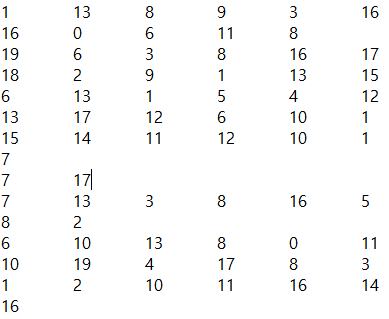
**1. Summary of algorithm**

**2. Detailed description of codes**

**3. Instructions for compiling source codes at TA's computer**

**4. Any other specification of implementation and testing**

**1. Summary of algorithm**

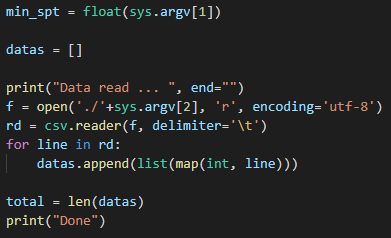


Input

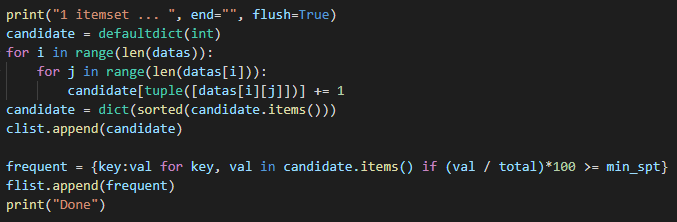
Output

연관된 구매 목록이 Index를 통해 Transaction으로 주어진 DB를 바탕으로 Frequent Pattern을 얻기 위해 사용자가 지정한 minimum support가 넘는 item set과 association set을 구하고 그것들에 대한 support와 confidence를 {item\_set} {association\_set} {support} {confidence}의 양식으로 output.csv 파일을 생성합니다. Support은 x -> y 일 때 x U y 합집합의 개수를 전체 Transaction 수로 나눈 값이며 confidence는 x -> y 일 때 x U y 합집합의 개수를 x가 포함된 Transaction 수로 나눈 값입니다. 이 때 Apriori 알고리즘에서 Candidate을 뽑고 난 뒤 이전 단계 itemset을 통해 Candidate을 pruning 함으로써 DB Scan의 시간을 줄일 수 있습니다.

**2. Detailed description of codes**

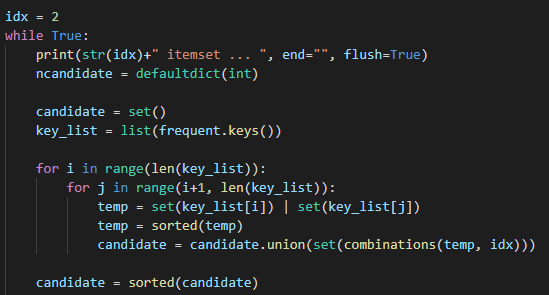
**- Data Read**

compile에서 주어지는 minimum support와 읽는 파일 쓰는 파일 명은 sys.argv 배열에 담기게 됩니다. 그렇게 해서 받은 input 파일명을 csv library를 통해 Transaction을 줄 단위로 받아 datas 배열에 넣게 됩니다. 이때 후에 계산할 support를 위해 전체 transaction수를 total 변수에 저장합니다.

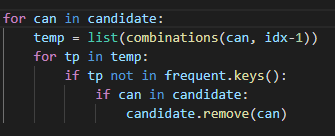
**- Init : 1 itemset Make** 

Apriori 알고리즘을 시작합니다. 우선 1단계에서는 이전 단계가 없어 후보군을 만들 수 없기에 DB를 scan해 모든 item의 개수를 구한 것으로 candidate를 만들고 그 개수가 minimum support를 넘지 않는 것을 제거해 frequent dictionary를 만듭니다. 후에 값들을 편하게 쓰기 위해 각각 clist와 flist에 dictionary를 담습니다.

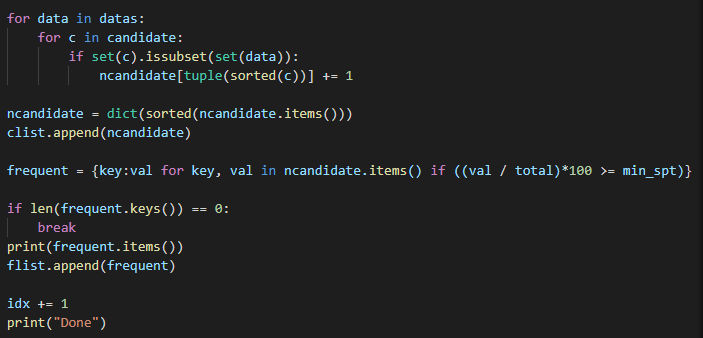
**- Self-joining**



2 itemset 부터 이전 단계의 frequent itemset을 self-joining 해서 candidate를 만듭니다. Candidate를 만들 때 이전 item set들을 합치는데 합집합 원소의 중복을 제거하기 위해 Set 자료형을 이용한 합집합 연산 | 을 수행합니다. 이 후 만들어진 합집합도 중복이 있을 수 있기에 candidate set에 union 함수를 통해 중복이 제거되도록 구현했습니다.

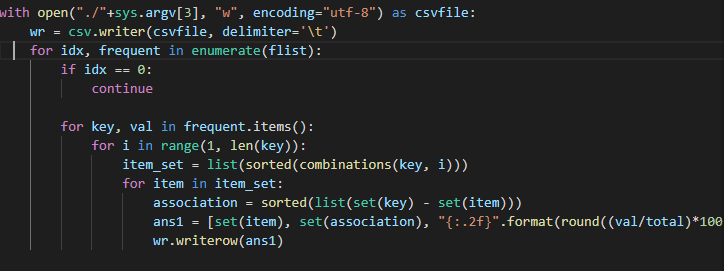
**- Pruning**

join을 통해 만들어진 Candidate의 요소 중에서 만약에 이전 단계에서 frequent pattern이 아니었던 것이 포함되었다면 이 후보군은 pruning 됩니다. 후보군의 조합(combinations)을 이용해서 하나 적은 모든 요소들을 추출, 각 요소들이 이전 frequent에 있었는지 판별합니다.

**- Frequent – pattern with minimum support**

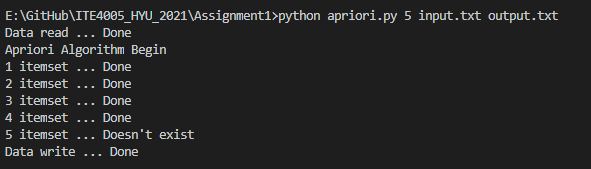
Self-joining과 pruning을 통해 얻은 Candidate를 다시 DB Scan을 통해 모든 Transaction 안에 있는 각 candidate의 개수를 얻습니다. 이후 다시 주어진 minimum support를 적용해 frequent-pattern을 얻습니다. 이 때 frequent-pattern의 개수가 없다면 이 후 frequent-pattern도 자명하게 없는 것이므로 Apriori 함수를 종료합니다.

**- Data Write**



구한 모든 frequent-pattern에 대해 속해 있는 모든 가능성에 대한 조합 결과인 item\_set과 association\_set을 구분하고 이들의 support와 confidence를 기존에 구한 flist에서 정보를 얻어 최종 결과를 csv 파일로 만듭니다.

**3. Instructions for compiling source codes at TA's computer**



주어진 apiriori.py 파일을 python을 통해 compile 하면서 이때 minimum support의 크기와 input, output 파일의 이름을 입력하면 됩니다. 파일의 이름은 상대경로를 통해 받기에 주어진 코드 파일과 input 파일이 같은 위치에 있어야 합니다.

**4. Any other specification of implementation and testing**

Python version은 3.9.1이며 내부 library만 썼기에 별도의 패키지 install은 없습니다.