# CSE17040 - Apriori from scratch

August 18, 2020

## 1 MLDM Lab 4 - Apriori

(Without libraries)

Manojkumar V K - CB.EN.U4CSE17040

[1]:
```
import numpy as np
import pandas as pd
from itertools import combinations
```

[2]:
```
df = pd.read_csv('GroceryStoreDataSet.csv', sep=',',header=None,index_col=False)
df['I1'],df['I2'],df['I3'],df['I4'] = np.nan,np.nan,np.nan,np.nan
for r in range(df.shape[0]):
    l = df.iloc[r,0].split(',')
    n = len(l)
    for i in range(1,n+1):
        df.iloc[r,i] = l[i-1]
df.head()
```

[2]:

| | 0 | I1 | I2 | I3 | I4 |
|---|---|---|---|---|---|
| 0 | MILK,BREAD,BISCUIT | MILK | BREAD | BISCUIT | NaN |
| 1 | BREAD,MILK,BISCUIT,CORNFLAKES | BREAD | MILK | BISCUIT | CORNFLAKES |
| 2 | BREAD,TEA,BOURNVITA | BREAD | TEA | BOURNVITA | NaN |
| 3 | JAM,MAGGI,BREAD,MILK | JAM | MAGGI | BREAD | MILK |
| 4 | MAGGI,TEA,BISCUIT | MAGGI | TEA | BISCUIT | NaN |

Assumed Minimum support = 2, Minimum confidence = 75%

[3]:
```
min_sup, min_conf, records = 2, 75, []
for i in range(0,df.shape[0]):
    records.append([str(df.values[i,j]) for j in range(1,len(df.columns)) if␣
 ↪str(df.values[i,j]) != 'nan'])
itemlist = sorted([item for sublist in records for item in sublist if item !=␣
 ↪np.nan])
records[:5]
```

[3]:
```
[['MILK', 'BREAD', 'BISCUIT'],
 ['BREAD', 'MILK', 'BISCUIT', 'CORNFLAKES'],
```

```
['BREAD', 'TEA', 'BOURNVITA'],
['JAM', 'MAGGI', 'BREAD', 'MILK'],
['MAGGI', 'TEA', 'BISCUIT']]
```

### 1.1 1 - Item list

[4]:
```
def stage_1(itemlist,min_sup):
    c1 = {i: itemlist.count(i) for i in itemlist}
    l1 = {}
    for key,val in c1.items():
        if val >= min_sup:
            l1[key] = val
    return c1,l1
```

[5]:
```
c1,l1 = stage_1(itemlist,min_sup)
print(c1)
print(l1)
```

```
{'BISCUIT': 7, 'BOURNVITA': 4, 'BREAD': 13, 'COCK': 3, 'COFFEE': 8,
'CORNFLAKES': 6, 'JAM': 2, 'MAGGI': 5, 'MILK': 5, 'SUGER': 6, 'TEA': 7}
{'BISCUIT': 7, 'BOURNVITA': 4, 'BREAD': 13, 'COCK': 3, 'COFFEE': 8,
'CORNFLAKES': 6, 'JAM': 2, 'MAGGI': 5, 'MILK': 5, 'SUGER': 6, 'TEA': 7}
```

[6]:
```
df_stage1 = pd.DataFrame(l1,index=['sup_count']).T
df_stage1.head()
```

[6]:

| | sup_count |
|---|---|
| BISCUIT | 7 |
| BOURNVITA | 4 |
| BREAD | 13 |
| COCK | 3 |
| COFFEE | 8 |

### 1.2 2 - Item list

[7]:
```
def check_freq(curr,prev,n):
    if n > 1:
        subsets = list(combinations(curr,n))
    else:
        subsets = curr
    for item in subsets:
        if not item in prev:
            return False
        else:
            return True
```

[8]:
```
def sublist(i1,i2):
    return set(i1) <= set(i2)
```

[9]:
```
def stage_2(l1,records,min_sup):
    l1 = sorted(list(l1.keys()))
    L1 = list(combinations(l1,2))
    c2,l2 = {},{}
    for it1 in L1:
        count = 0
        for it2 in records:
            if sublist(it1,it2):
                count += 1
        c2[it1] = count
    for key,val in c2.items():
        if val >= min_sup:
            if check_freq(key,l1,1):
                l2[key] = val
    return c2,l2
```

[10]:
```
c2,l2 = stage_2(l1,records,min_sup)
l2 = {key: value for key,value in l2.items() if value != 0}
print("No. of itemsets = {}, No. of frequent itemsets = {}".
 ↪format(len(list(c2)),len(list(l2))))
```

```
No. of itemsets = 55, No. of frequent itemsets = 24
```

[11]:
```
df_stage2 = pd.DataFrame(l2,index=['sup_count']).T
df_stage2.head()
```

[11]:

| | | sup_count |
|---|---|---|
| BISCUIT | BREAD | 4 |
| | COCK | 2 |
| | COFFEE | 2 |
| | CORNFLAKES | 3 |
| | MAGGI | 2 |

### 1.3 3 - Item list

[12]:
```
def stage_3(l2,records,min_sup):
    l2 = list(l2.keys())
    L2 = sorted(list(set([item for temp in l2 for item in temp])))
    L2 = list(combinations(L2,3))
    c3,l3 = {},{}
    for it1 in L2:
        count = 0
        for it2 in records:
```

```
            if sublist(it1,it2):
                count += 1
        c3[it1] = count
    for key,val in c3.items():
        if val >= min_sup:
            if check_freq(key,l2,2):
                l3[key] = val
    return c3,l3
```

[13]:
```
c3,l3 = stage_3(l2,records,min_sup)
l3 = {key: value for key,value in l3.items() if value != 0}
print("No. of itemsets = {}, No. of frequent itemsets = {}".
 ↪format(len(list(c3)),len(list(l3))))
```

```
No. of itemsets = 165, No. of frequent itemsets = 10
```

[14]:
```
df_stage3 = pd.DataFrame(l3,index=['sup_count']).T
df_stage3
```

[14]:

| | | | sup_count |
|---|---|---|---|
| BISCUIT | BREAD | MILK | 2 |
| | COCK | COFFEE | 2 |
| | | CORNFLAKES | 2 |
| | COFFEE | CORNFLAKES | 2 |
| | MAGGI | TEA | 2 |
| BOURNVITA | BREAD | TEA | 2 |
| BREAD | COFFEE | SUGER | 2 |
| | JAM | MAGGI | 2 |
| | MAGGI | TEA | 2 |
| COCK | COFFEE | CORNFLAKES | 2 |

### 1.4 4 - Item list

[15]:
```
def stage_4(l3,records,min_sup):
    l3 = list(l3.keys())
    L3 = sorted(list(set([item for temp in l3 for item in temp])))
    L3 = list(combinations(L3,4))
    c4,l4 = {},{}
    for it1 in L3:
        count = 0
        for it2 in records:
            if sublist(it1,it2):
                count += 1
        c4[it1] = count
        for key,val in c4.items():
            if val >= min_sup:
```

```
                if check_freq(key,l3,3):
                    l4[key] = val
    return c4,l4
```

[16]:
```
c4,l4 = stage_4(l3,records,min_sup)
l4 = {key: value for key,value in l4.items() if value != 0}
print("No. of itemsets = {}, No. of frequent itemsets = {}".
 ↪format(len(list(c4)),len(list(l4))))
```

```
No. of itemsets = 330, No. of frequent itemsets = 1
```

[17]:
```
df_stage4 = pd.DataFrame(l4,index=['sup_count']).T
df_stage4
```

[17]:

| | sup_count |
|---|---|
| BISCUIT COCK COFFEE CORNFLAKES | 2 |

### 1.5 Building Association Rules

[18]:
```
items = {**l1,**l2,**l3,**l4}
items
```

[18]:
```
{'BISCUIT': 7,
 'BOURNVITA': 4,
 'BREAD': 13,
 'COCK': 3,
 'COFFEE': 8,
 'CORNFLAKES': 6,
 'JAM': 2,
 'MAGGI': 5,
 'MILK': 5,
 'SUGER': 6,
 'TEA': 7,
 ('BISCUIT', 'BREAD'): 4,
 ('BISCUIT', 'COCK'): 2,
 ('BISCUIT', 'COFFEE'): 2,
 ('BISCUIT', 'CORNFLAKES'): 3,
 ('BISCUIT', 'MAGGI'): 2,
 ('BISCUIT', 'MILK'): 2,
 ('BISCUIT', 'TEA'): 2,
 ('BOURNVITA', 'BREAD'): 3,
 ('BOURNVITA', 'SUGER'): 2,
 ('BOURNVITA', 'TEA'): 2,
 ('BREAD', 'COFFEE'): 3,
 ('BREAD', 'JAM'): 2,
 ('BREAD', 'MAGGI'): 3,
```

```
 ('BREAD', 'MILK'): 4,
 ('BREAD', 'SUGER'): 4,
 ('BREAD', 'TEA'): 4,
 ('COCK', 'COFFEE'): 3,
 ('COCK', 'CORNFLAKES'): 2,
 ('COFFEE', 'CORNFLAKES'): 4,
 ('COFFEE', 'SUGER'): 4,
 ('CORNFLAKES', 'MILK'): 2,
 ('CORNFLAKES', 'TEA'): 2,
 ('JAM', 'MAGGI'): 2,
 ('MAGGI', 'TEA'): 4,
 ('BISCUIT', 'BREAD', 'MILK'): 2,
 ('BISCUIT', 'COCK', 'COFFEE'): 2,
 ('BISCUIT', 'COCK', 'CORNFLAKES'): 2,
 ('BISCUIT', 'COFFEE', 'CORNFLAKES'): 2,
 ('BISCUIT', 'MAGGI', 'TEA'): 2,
 ('BOURNVITA', 'BREAD', 'TEA'): 2,
 ('BREAD', 'COFFEE', 'SUGER'): 2,
 ('BREAD', 'JAM', 'MAGGI'): 2,
 ('BREAD', 'MAGGI', 'TEA'): 2,
 ('COCK', 'COFFEE', 'CORNFLAKES'): 2,
 ('BISCUIT', 'COCK', 'COFFEE', 'CORNFLAKES'): 2}
```

[19]:
```
assc_sets = []
for it1 in list(l3.keys()):
    assc_subset = list(combinations(it1,2))
    assc_sets.append(assc_subset)
```

[20]:
```
def sup_calc(it,items):
    return items[it]
```

[21]:
```
l3_assc = list(l3.keys())
selected_assc = []
for i in range(len(l3_assc)):
    for it1 in assc_sets[i]:
        denom = it1
        num = set(l3_assc[i]) - set(it1)
        confidence = ((sup_calc(l3_assc[i],items))/(sup_calc(it1,items)))*100
        if confidence > min_conf:
            print("Confidence of the association rule {} --> {} = {:.2f}%".
 ↪format(denom,num,confidence))
            print("STATUS : SELECTED RULE\n")
        else:
            print("Confidence of the association rule {} --> {} = {:.2f}%".
 ↪format(denom,num,confidence))
            print("STATUS : REJECTED RULE\n")
```

```
Confidence of the association rule ('BISCUIT', 'BREAD') --> {'MILK'} = 50.00%
STATUS : REJECTED RULE

Confidence of the association rule ('BISCUIT', 'MILK') --> {'BREAD'} = 100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('BREAD', 'MILK') --> {'BISCUIT'} = 50.00%
STATUS : REJECTED RULE

Confidence of the association rule ('BISCUIT', 'COCK') --> {'COFFEE'} = 100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('BISCUIT', 'COFFEE') --> {'COCK'} = 100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('COCK', 'COFFEE') --> {'BISCUIT'} = 66.67%
STATUS : REJECTED RULE

Confidence of the association rule ('BISCUIT', 'COCK') --> {'CORNFLAKES'} =
100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('BISCUIT', 'CORNFLAKES') --> {'COCK'} =
66.67%
STATUS : REJECTED RULE

Confidence of the association rule ('COCK', 'CORNFLAKES') --> {'BISCUIT'} =
100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('BISCUIT', 'COFFEE') --> {'CORNFLAKES'} =
100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('BISCUIT', 'CORNFLAKES') --> {'COFFEE'} =
66.67%
STATUS : REJECTED RULE

Confidence of the association rule ('COFFEE', 'CORNFLAKES') --> {'BISCUIT'} =
50.00%
STATUS : REJECTED RULE

Confidence of the association rule ('BISCUIT', 'MAGGI') --> {'TEA'} = 100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('BISCUIT', 'TEA') --> {'MAGGI'} = 100.00%
STATUS : SELECTED RULE
```

```
Confidence of the association rule ('MAGGI', 'TEA') --> {'BISCUIT'} = 50.00%
STATUS : REJECTED RULE

Confidence of the association rule ('BOURNVITA', 'BREAD') --> {'TEA'} = 66.67%
STATUS : REJECTED RULE

Confidence of the association rule ('BOURNVITA', 'TEA') --> {'BREAD'} = 100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('BREAD', 'TEA') --> {'BOURNVITA'} = 50.00%
STATUS : REJECTED RULE

Confidence of the association rule ('BREAD', 'COFFEE') --> {'SUGER'} = 66.67%
STATUS : REJECTED RULE

Confidence of the association rule ('BREAD', 'SUGER') --> {'COFFEE'} = 50.00%
STATUS : REJECTED RULE

Confidence of the association rule ('COFFEE', 'SUGER') --> {'BREAD'} = 50.00%
STATUS : REJECTED RULE

Confidence of the association rule ('BREAD', 'JAM') --> {'MAGGI'} = 100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('BREAD', 'MAGGI') --> {'JAM'} = 66.67%
STATUS : REJECTED RULE

Confidence of the association rule ('JAM', 'MAGGI') --> {'BREAD'} = 100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('BREAD', 'MAGGI') --> {'TEA'} = 66.67%
STATUS : REJECTED RULE

Confidence of the association rule ('BREAD', 'TEA') --> {'MAGGI'} = 50.00%
STATUS : REJECTED RULE

Confidence of the association rule ('MAGGI', 'TEA') --> {'BREAD'} = 50.00%
STATUS : REJECTED RULE

Confidence of the association rule ('COCK', 'COFFEE') --> {'CORNFLAKES'} =
66.67%
STATUS : REJECTED RULE

Confidence of the association rule ('COCK', 'CORNFLAKES') --> {'COFFEE'} =
100.00%
STATUS : SELECTED RULE

Confidence of the association rule ('COFFEE', 'CORNFLAKES') --> {'COCK'} =
```

```
50.00%
STATUS : REJECTED RULE
```