IT 414

Assignment 1 - Apriori Algorithm

Name: Niraj Nandish

Roll no.: 191IT234

```python
from collections import defaultdict
from itertools import chain, combinations
import pandas as pd

```

```python
def powerset(s):
    return chain.from_iterable(combinations(s, r) for r in range(1, len(s)))


def getAboveMinSup(itemSet, itemSetList, minSup, globalItemSetWithSup):
    freqItemSet = set()
    localItemSetWithSup = defaultdict(int)

    for item in itemSet:
        for itemSet in itemSetList:
            if item.issubset(itemSet):
                globalItemSetWithSup[item] += 1
                localItemSetWithSup[item] += 1

    for item, supCount in localItemSetWithSup.items():
        support = float(supCount / len(itemSetList))
        if (support >= minSup):
            freqItemSet.add(item)

    return freqItemSet


def getUnion(itemSet, length):
    return set([i.union(j) for i in itemSet for j in itemSet if len(i.union(j)) == length])


def pruning(candidateSet, prevFreqSet, length):
    tempCandidateSet = candidateSet.copy()
    for item in candidateSet:
        subsets = combinations(item, length)
        for subset in subsets:
            if (frozenset(subset) not in prevFreqSet):
                tempCandidateSet.remove(item)
                break
    return tempCandidateSet


def associationRule(freqItemSet, itemSetWithSup, minConf):
    rules = []
    for k, itemSet in freqItemSet.items():
        for item in itemSet:
            subsets = powerset(item)
            for s in subsets:
                confidence = float(
                    itemSetWithSup[item] / itemSetWithSup[frozenset(s)])
                if (confidence > minConf):
                    rules.append([set(s), set(item.difference(s)), confidence])
    return rules


def getItemSetFromList(itemSetList):
    tempItemSet = set()

    for itemSet in itemSetList:
        for item in itemSet:
            tempItemSet.add(frozenset([item]))

    return tempItemSet
```

```python
def apriori(itemSetList, minSup, minConf):
    C1ItemSet = getItemSetFromList(itemSetList)
    globalFreqItemSet = {}
    globalItemSetWithSup = defaultdict(int)

    L1ItemSet = getAboveMinSup(C1ItemSet, itemSetList, minSup, globalItemSetWithSup)
    currentLSet = L1ItemSet
    print(1)
    print(C1ItemSet)
    print(currentLSet)
    k = 2

    while (currentLSet):
        globalFreqItemSet[k-1] = currentLSet
        candidateSet = getUnion(currentLSet, k)
        candidateSet = pruning(candidateSet, currentLSet, k-1)
        currentLSet = getAboveMinSup(candidateSet, itemSetList, minSup, globalItemSetWithSup)
        print(k)
        print(candidateSet)
        print(currentLSet)
        k += 1

    rules = associationRule(globalFreqItemSet, globalItemSetWithSup, minConf)
    rules.sort(key=lambda x: x[2])

    return globalFreqItemSet, rules

```

Q1

Support count = 2

Minimum confidence threshold = 70%

```python
data = [
    ['I1', 'I2', 'I5'],
    ['I2', 'I4'],
    ['I2', 'I3'],
    ['I1', 'I2', 'I4'],
    ['I1', 'I3'],
    ['I2', 'I3'],
    ['I1', 'I3'],
    ['I1', 'I2', 'I3', 'I5'],
    ['I1', 'I2', 'I3']
]

MIN_SUP = 2
MIN_CONF = 0.7

apriori(data, MIN_SUP/len(data), MIN_CONF)
```

```
1
{frozenset({'I4'}), frozenset({'I5'}), frozenset({'I1'}), frozenset({'I2'}), frozenset({'I3'})}
{frozenset({'I5'}), frozenset({'I4'}), frozenset({'I1'}), frozenset({'I2'}), frozenset({'I3'})}
2
{frozenset({'I2', 'I3'}), frozenset({'I5', 'I3'}), frozenset({'I3', 'I4'}), frozenset({'I1', 'I3'}), frozenset({'I1',
{frozenset({'I2', 'I3'}), frozenset({'I1', 'I3'}), frozenset({'I1', 'I2'}), frozenset({'I5', 'I2'}), frozenset({'I2',
3
{frozenset({'I1', 'I5', 'I2'}), frozenset({'I1', 'I2', 'I3'})}
{frozenset({'I1', 'I5', 'I2'}), frozenset({'I1', 'I2', 'I3'})}
4
set()
set()
({1: {frozenset({'I5'}),
   frozenset({'I4'}),
   frozenset({'I1'}),
   frozenset({'I2'}),
   frozenset({'I3'})},
  2: {frozenset({'I2', 'I3'}),
   frozenset({'I1', 'I3'}),
   frozenset({'I1', 'I2'}),
   frozenset({'I2', 'I5'}),
   frozenset({'I2', 'I4'}),
   frozenset({'I1', 'I5'})},
  3: {frozenset({'I1', 'I2', 'I5'}), frozenset({'I1', 'I2', 'I3'})}},
 [[{'I5'}, {'I2'}, 1.0],
  [{'I4'}, {'I2'}, 1.0],
  [{'I5'}, {'I1'}, 1.0],
  [{'I5'}, {'I1', 'I2'}, 1.0],
  [{'I1', 'I5'}, {'I2'}, 1.0],
  [{'I2', 'I5'}, {'I1'}, 1.0]])
```

Q2

Minimum support threshold: 60%

Minimum confidence threshold: 80%

```python
data = [
    ['M', 'O', 'N', 'K', 'E', 'Y'],
    ['D', 'O', 'N', 'K', 'E', 'Y'],
    ['M', 'A', 'K', 'E'],
    ['M', 'U', 'C', 'K', 'Y'],
    ['C', 'O', 'O', 'K', 'I', 'E']
]

MIN_SUP = 0.6
MIN_CONF = 0.8

apriori(data, MIN_SUP, MIN_CONF)
```

```
({1: {frozenset({'E'}),
      frozenset({'K'}),
      frozenset({'Y'}),
      frozenset({'M'}),
      frozenset({'O'})},
     2: {frozenset({'K', 'O'}),
      frozenset({'E', 'K'}),
      frozenset({'E', 'O'}),
      frozenset({'K', 'M'}),
      frozenset({'K', 'Y'})},
     3: {frozenset({'E', 'K', 'O'})}},
    [[{'O'}, {'K'}, 1.0],
     [{'E'}, {'K'}, 1.0],
     [{'O'}, {'E'}, 1.0],
     [{'M'}, {'K'}, 1.0],
     [{'Y'}, {'K'}, 1.0],
     [{'O'}, {'E', 'K'}, 1.0],
     [{'E', 'O'}, {'K'}, 1.0],
     [{'K', 'O'}, {'E'}, 1.0]])
```

Q3

Minimum support threshold: 60%

Minimum confidence threshold: 80%

```
df = pd.read_csv('SPECTF_test.csv')
df = df.iloc[:20, :5]
df
```

	Attr_1	Attr_2	Attr_3	Attr_4	Attr_5
0	32	41	76	34	65
1	76	65	60	40	32
2	60	51	75	60	65
3	64	60	71	69	71
4	65	69	66	76	58
5	71	76	74	79	71
6	55	66	58	75	71
7	76	77	69	70	64
8	70	72	65	64	71
9	64	72	70	74	63
10	76	59	82	76	80
11	54	53	73	68	77
12	68	59	77	69	77
13	64	63	62	59	60
14	40	73	61	74	61

```
apriori(df.values.tolist(), MIN_SUP, MIN_CONF)
```

```
({}, [])
```

```
itemSetCount = {}
for x in df.values.tolist():
    for y in x:
        itemSetCount[y] = itemSetCount.get(y, 0) + 1

print("Min Count required:", MIN_SUP*len(df.values.tolist()))
itemSetCount
```

```
Min Count required: 12.0
{32: 2,
 41: 1,
 76: 8,
 34: 1,
 65: 7,
 60: 5,
 40: 2,
 51: 1,
 75: 2,
 64: 8,
 71: 7,
 69: 5,
 66: 5,
 58: 4,
 74: 4,
 79: 1,
 55: 1,
 77: 4,
 70: 8,
 72: 3,
 63: 3,
 59: 3,
 82: 1,
 80: 2,
 54: 1,
 53: 1,
 73: 2,
 68: 3,
 62: 1,
 61: 2,
 67: 1,
 78: 1}
```