Association Rule (Apriori Algorithm)

import itertools

support = int(input("Please enter support value in %: "))

confidence = int(input("Please enter confidence value in %: "))

C1 = {}

transactions = 0

D = []

T = []

with open("DataSet5.txt", "r") as f:

for line in f:

T = []

transactions += 1

for word in line.split():

T.append(word)

if word not in C1.keys():

C1[word] = 1

else:

count = C1[word]

C1[word] = count + 1

D.append(T)

print ("-------------------------TEST DATASET----------------------------")

print (D)

print ("-----------------------------------------------------------------")

L1 = []

for key in C1:

if (100 \* C1[key]/transactions) >= support:

list = []

list.append(key)

L1.append(list)

print ("----------------------FREQUENT 1-ITEMSET-------------------------")

print (L1)

print ("-----------------------------------------------------------------")

def apriori\_gen(Lk\_1, k):

length = k

Ck = []

for list1 in Lk\_1:

for list2 in Lk\_1:

count = 0

c = []

if list1 != list2:

while count < length-1:

if list1[count] != list2[count]:

break

else:

count += 1

else:

if list1[length-1] < list2[length-1]:

for item in list1:

c.append(item)

c.append(list2[length-1])

if not has\_infrequent\_subset(c, Lk\_1, k):

Ck.append(c)

c = []

return Ck

def findsubsets(S,m):

return set(itertools.combinations(S, m))

def has\_infrequent\_subset(c, Lk\_1, k):

list = []

list = findsubsets(c,k)

for item in list:

s = []

for l in item:

s.append(l)

s.sort()

if s not in Lk\_1:

return True

return False

def frequent\_itemsets():

k = 2

Lk\_1 = []

Lk = []

L = []

count = 0

transactions = 0

for item in L1:

Lk\_1.append(item)

while Lk\_1 != []:

Ck = []

Lk = []

Ck = apriori\_gen(Lk\_1, k-1)

for c in Ck:

count = 0

transactions = 0

s = set(c)

for T in D:

transactions += 1

t = set(T)

if s.issubset(t) == True:

count += 1

if (100 \* count/transactions) >= support:

c.sort()

Lk.append(c)

Lk\_1 = []

print ("-----------------------FREQUENT %d-ITEMSET------------------------" % k)

print (Lk)

print ("------------------------------------------------------------------")

for l in Lk:

Lk\_1.append(l)

k += 1

if Lk != []:

L.append(Lk)

return L

def generate\_association\_rules():

s = []

r = []

length = 0

count = 1

inc1 = 0

inc2 = 0

num = 1

m = []

L= frequent\_itemsets()

print ("---------------------ASSOCIATION RULES------------------")

print ("RULES \t SUPPORT \t CONFIDENCE")

print ("--------------------------------------------------------")

for list in L:

for l in list:

length = len(l)

count = 1

while count < length:

s = []

r = findsubsets(l,count)

count += 1

for item in r:

inc1 = 0

inc2 = 0

s = []

m = []

for i in item:

s.append(i)

for T in D:

if set(s).issubset(set(T)) == True:

inc1 += 1

if set(l).issubset(set(T)) == True:

inc2 += 1

if 100\*inc2/inc1 >= confidence:

for index in l:

if index not in s:

m.append(index)

print ("Rule# %d : %s ==> %s %d %d" %(num, s, m, 100\*inc2/len(D), 100\*inc2/inc1))

num += 1

generate\_association\_rules()

print ("--------------------------------------------------------")

