**Data Mining Assignment 4**

**Name : Venkatesh Suvarna**

**Student ID : 1001390986**

**Aim :** To implement the Apriori Algorithm using Python

**Code :**

**Apriori.py (Comments included in the program):**

#Importing the necessary basic modules for the code to run successfully

import sys

import csv

import operator

#The below function calculates powerset - all the possible subsets with set size = k

def powerset(s, k):

x = len(s)

powerset = []

list = None

for i in range(1, 1 << x):

list = [s[j] for j in range(x) if (i & (1 << j))]

if len(list) == k:

powerset.append(list)

return powerset

#The below function checks for each subset whether it is a frequent subset

def has\_frequent\_subset(c, L, k):

subsets = powerset(c, k)

for subset in subsets:

frequent\_subset = False

for item in L:

if set(subset) == set(item[0].split(",")):

frequent\_subset = True

break

if frequent\_subset == False:

return False

return True

#The below function calculates the frequent itemsets

#Which includes the calculation of candidate and frequent itemsets

def calculate\_freq\_itemsets(Dataset, minimum\_support, row\_count):

dataset = None

itemset = {}

#The below code calculates candidate itemsets

with open(Dataset, 'r') as f:

dataset = csv.reader(f)

for i in dataset:

for j in i:

if j in itemset:

itemset[j.strip()] += 1

else:

itemset[j.strip()] = 1

#The below code calculates the frequent itemsets

for item in itemset.copy():

if itemset[item]/float(row\_count) < minimum\_support:

itemset.pop(item, None)

return sorted(itemset.items(), key=operator.itemgetter(0))

#The below function refines the association rules already generated by the generate\_apriori\_rules(L,k,row\_count)

def generate\_association\_rules(itemset, minimum\_confidence, row\_count):

if len(itemset) < 2:

print("There are no association rules present for the given dataset retail")

else:

Dataset = str(sys.argv[1])

print("\nThe minimum confidence threshold as given by user : ", minimum\_confidence\*100, "%\n")

print("The association rules are as follows : \n")

for k in range(1, len(itemset)):

for pair in itemset[k]:

for i in range(1, len(itemset[k][0][0].split(','))):

for item in powerset(pair[0].split(','), i):

item\_sup = None

for j in itemset[i-1]:

if j[0] == ",".join(item):

item\_sup = int(j[1])

if item\_sup is not None and pair[1]/float(item\_sup) >= minimum\_confidence: #This line checks if the particular association rule is above the minimum confidence level

print(",".join(item), "=>", ",".join(list(set(pair[0].split(',')) - set(item))), "Support: ", float("{0:.2f}".format(float(item\_sup)/row\_count))\*100, "%", "Confidence: ", float("{0:.2f}".format(pair[1]/float(item\_sup)\*100)), "%")

#The below function generates the association rules

def generate\_apriori\_rules(L, k, row\_count):

C = []

for l1 in L:

for l2 in L:

first\_itemlist = l1[0].split(",")

second\_itemlist = l2[0].split(",")

i = 0

flag = True

while i <= k-2-1:

if first\_itemlist[i] != second\_itemlist[i]:

flag = False

break

i += 1

if not first\_itemlist[k-1-1] < second\_itemlist[k-1-1]:

flag = False

if flag == True:

c = sorted(set(first\_itemlist) | set(second\_itemlist))

if has\_frequent\_subset(list(c), L, k-1):

C.append(",".join(list(c)))

return C

def main():

Dataset = str(sys.argv[1])

minimum\_support = float(sys.argv[2])

minimum\_confidence = float(sys.argv[3])

with open(sys.argv[1], 'r') as input\_file:

lines = input\_file.readlines()

newLines = []

for line in lines:

newLine = line.strip(' ').split()

newLines.append(newLine)

with open("retail.csv", 'w') as output\_file:

file\_writer = csv.writer(output\_file)

file\_writer.writerows(newLines)

Dataset = "retail.csv"

print("The dataset is :")

print(Dataset)

print("The minimum support is :")

print(minimum\_support)

print("The Minimum confidence is :")

print(minimum\_confidence)

row\_count = 0

with open(Dataset, 'r') as f:

dataset = csv.reader(f)

row\_count = sum(1 for row in dataset)

#This line generates the frequent itemsets, which includes generating the candidate and frequent itemsets

L1 = calculate\_freq\_itemsets(Dataset, minimum\_support, row\_count)

itemset = [L1]

k = 2

while True:

if not itemset[k-2]:

break

#This line generates all the possible apriori rules

C = generate\_apriori\_rules(itemset[k-2], k, row\_count)

L = {}

#The below code

with open(Dataset, 'r') as f:

dataset = csv.reader(f)

for t in dataset:

for c in C:

if set(c.split(",")).issubset(set(t)):

if c in L:

L[c] += 1

else:

L[c] = 1

#This removes the items from the list whose support fall below the minimum\_support\_line

for item in L.copy():

if L[item]/float(row\_count) < minimum\_support:

L.pop(item, None)

itemset.append(sorted(L.items(), key=operator.itemgetter(0)))

k += 1

itemset.pop()

generate\_association\_rules(itemset, minimum\_confidence, row\_count)

print("\nThe resultant itemsets are : ")

for k in range(1, len(itemset)):

print("Total Number of Itemsets are : ",k)

print("\n")

for item in itemset[k]:

print(item[0])

print("with support")

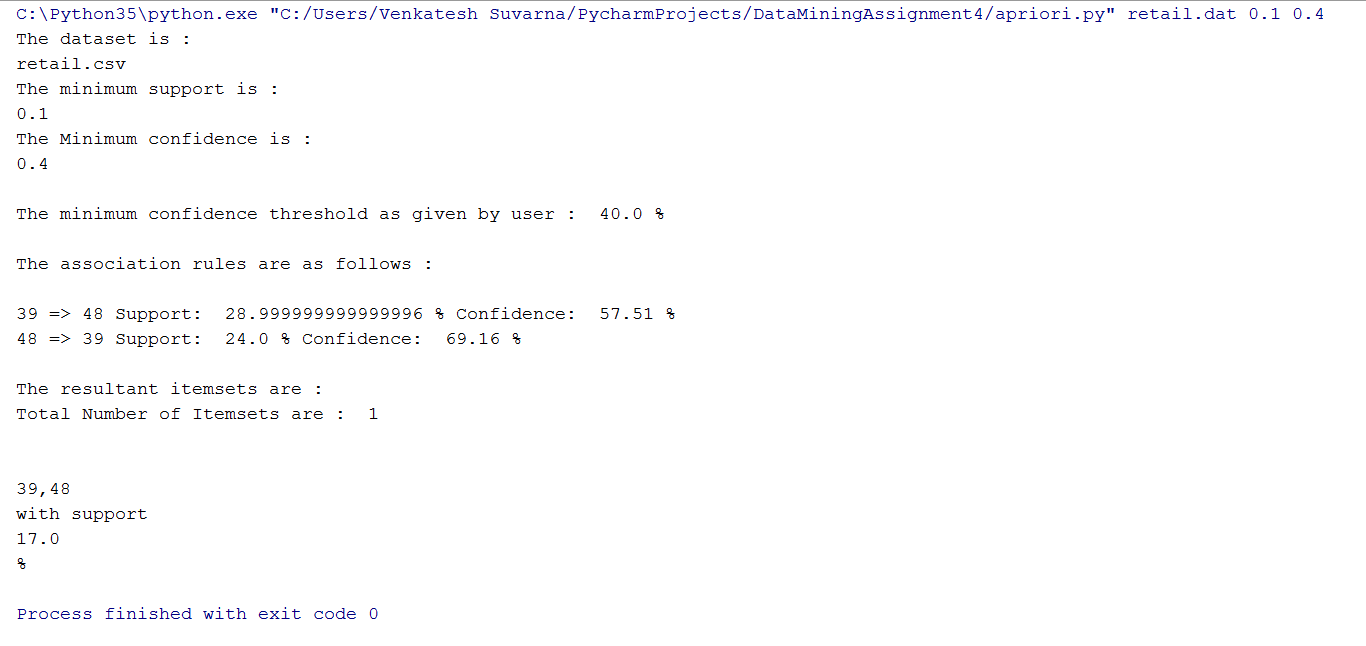
print(float("{0:.2f}".format(item[1]/float(row\_count)))\*100)

print("%")

if \_\_name\_\_ == "\_\_main\_\_":

main()

***Output :***



**Working Explanation :**

From the main method, we get all the input parameters such as the retail.dat file, minimum support, and the minimum confidence. From there it calls the find\_frequent\_item\_sets\_1, where the itemset is set in an array. We then generate the candidate itemsets and check their support count, all the items less than the item count are removed from the itemset array. And then we calculate the confidence of all the association rules, and only those rules that are greater than or equal to the confidence value defined by the user are displayed as the final association rules.

**References :**

<https://github.com/asaini/Apriori>

<https://github.com/timothyasp/apriori-python>

<https://github.com/jiteshjha/Frequent-item-set-mining>

<https://github.com/abarmat/python-apriori>

<https://gist.github.com/Stiivi/4730288>