First Name - Sai Prathyusha

Last Name - Devarapalli

NJIT UCID – sd894

Email address- sd894@njit.edu

Part 1

Creating Database with 20 transactions

*D4 - Notepad

File Edit Format View Help

```
Sony, Realme,Nokia,Samsung
LG, Sony Ericsson, Motorola, Blackberry
HTC, Razer Phone 2, Honor, Lava
Intex, MI, Micromax, Poco x4
RedMI, iQOO 9 5G,Jio Phone, Alcatel
Yota, Lenovo, Apple, Samsung
Apple, Samsung, Xiaomi,Oppo
Sony Ericsson, Motorola, HTC, Razer Phone 2
MI, Micromax, iQOO 9 5G, Jio Phone
Micromax, Lenovo, Google pixel
Huawei, Sony Ericsson, Apple, Lava
OnePlus, Huawei, Xiaomi
Vivo, Intex, MI, Micromax
Poco x4, Lava, Samsung, Xiaomi
iQOO 9 5G,Oppo, Sony Ericsson, Motorola
Huawei, Sony Ericsson, Google pixel, Apple
Lava, , Xiaomi, Oppo
MI, HTC,Yota, Jio Phone
```

Creating four additional databases

D3 - Notepad

File Edit Format View Help

```
Refrigerator, T.V, Micro oven, Bulb
Chargers, Geyser, Toaster, Laptop
Washing Machine, Hair dryers, Smart Watches, PC
Tablets, Headphones, Speakers, Air Conditioner
Heating Pads, Power Bank, Vacuum Cleaner, Induction Stove
Kettle, Blenders, CC Camera, MIC
Rice Cooker, Iron box, Extension Cord,
Stabilizer, Dish Washer, Steamer
Bulb, Chargers, Geyser, Vacuum Cleaner
Toaster, Blenders, Extension Cord, Laptop
Speakers, Air Conditioner, Heating Pads, Power
Micro oven, Bulb, Rice Cooker, Iron box
Tablets, Headphones, Geyser, Vacuum Cleaner
Iron box, Washing Machine, Hair dryers, Extension Cord
Induction Stove, CC Camera, Power Bank, Vacuum Cleaner
Kettle, Extension Cord, Iron box, Stabilizer
Headphones, Speakers, Blenders, Bulb
Extension Cord, Laptop, Air Conditioner, Heating Pads
Dish Washer, Bulb, Chargers, Steamer
Bulb, Toaster, Blenders Chargers, Geyser
```

D2 - Notepad

File Edit Format View Help

```
Shirts, Phants, Watches, Jeans
Socks, Sarees, Jackets, Head Cap
Muffler, Rain Coats, Gum Boots, Dresses
Sweater, Hand Gloves, Bath Towels, Shoes
Themals, Frocks, Bed sheets, Kurtas
Phants, Sarees, Rain Coats, Hand Gloves
Shirts, Socks, Muffler, Sweater, Themals
Watches, Paper towel, Cloth Bag, Bath Towels
Jeans, Head Cap, Dresses
Hand Gloves, Bath Towels, Bed sheets, Kurtas
Rain Coats, Watches, Themals, Shirts
Gum Boots, Dresses, Sweater, Hand Gloves
Muffler, Head Cap, Jeans, Jackets
Socks, Sarees, Bed sheets
Kurtas, Hand Gloves, Thermals, Bath Towels
Hand Gloves, Frocks, Sarees, Watches
Napkins, Comforters
Bed sheets, Kurtas, Phants, Sarees
Bath Towels, Hand Gloves, Frocks, Sarees
T-shirts, Tops, Rain Coats , Sarees
```

D5 - Notepad

File Edit Format View Help

```
Chips, Cakes, Cool Drinks, Pastries
Rolls, Coffee, Chocolates, Muffins
Cupcakes, Cookies, Bread, Bun
Donuts, Biscuits, Burger, Pizza
Milkshakes, Croissant, Noodles, Taccos
Puffs, Pancakes, Browines, Puddings
Desserts, Hot Dogs, Sweets, Pies
Sponge Cake, Chocolate Chips Cakes, Cool Drinks
Donuts, Biscuits, Bun, Pizza
Burger, Croissant, Pies, Milkshakes
Pancakes, Puddings, Sweets
Desserts, Pies, Browines, Hot Dogs
Biscuits, Puffs, Burger, Pizza
Hot Dogs, Chocolate Chips, Cakes
Sponge Cake, Sweets, Pies
Browines, Puddings, Milkshakes, Hot Dogs,
Biscuits, Sponge Cake, Donuts, Pancakes
Sweets, Muffins, Bun, Pies
Cookies, Milkshakes, Cool Drinks, Browines
Browines, Cool Drinks, Puddings, Donuts
```

D1 - Notepad

File Edit Format View Help

```
Coffee Mug, Water bottles, Tea cups
Saucers, Spoons, Coffee Mug, Pan
Instant Pot, Pan ,Tea cups, Lunch boxes
Serving skimmer, Glass bowels, Juicer,
Spoons, Water bottles, Pan, Roti Maker
Mixer grinder, Tea cups, Oil Tin, Storage Boxes
Whisker, Lunch boxes, Instant Pot, Plates
Pan, Whisker, Plates, Coffee Mug, Juicer
Basket, Serving skimmer, Mixer grinder, Glass bowels
Juicer, Vegetable Chopper, Storage Boxes, Water bottles
Coffee Mug, Tava, Paper Cups, Table Cloth
Mixer grinder, Whisker, Basket, Juicer
Holders, Oil tin, Stainless Steel Products,Knife
Water bottles, Serving skimmer, Storage Boxes,
Lunch boxes, Tava, Jars
Basket,Vessels, Cutting Board, Saucers
Vegetable Chopper, Mixer grinder, Glass bowels, Oil tin
Jars, Tea cups, Whisker, Lunch boxes,
Storage Boxes, Stirring Mug, Basket, Roti Maker
Home Mats, Saucers, Spoons
```

Using Apriori algorithm ,generating the association rules

```
import itertools
from itertools import combinations
import time
File=input("Please Enter the path of the Dataset: ")
open_file=open(File,'r')
read_lines=open_file.readlines()
inps=[]
for i in range(len(read_lines)):
inps.append(read_lines[i].strip('\n'))
Data=[]
for i in inps:
split_inp=i.split(" ")
Data.append(split_inp)
print(len(Data))
print("Select the option whether you are reading in Percentage or Value")
print("1.Percentage")
print("2.Points")
option=int(input())
print("Enter the support")
min_support=float(input())
print("Enter the confidence")
min_confidence=float(input())
if option==1:
min_support=min_support/100
min_confidence=min_confidence/100
Start_time=time.time()
data_merge=[]
count_dict={}
for i in Data:
data_merge+=i
items_data=list(set(data_merge))
for i in items_data:
count_dict[i]=(data_merge.count(i))
```

```
First_itt_minsprt=[]
First_itt_minsprtt=[]
for x, y in count_dict.items():
if((y/len(Data))>=min_support):
a=[x,y]
First_itt_minsprt.append(x)
First_itt_minsprtt.append([x])
First_itt_minsprtt.append(y)
comb=list(itertools.combinations(First_itt_minsprt, 2))
def AssociationRules(freqSet):
associationRule = []
import itertools
for item in freqSet:
if isinstance(item, list):
if len(item) != 0:
length = len(item) - 1
while length > 0:
combinations = list(itertools.combinations(item, length))
temp = []
LHS = []
for RHS in combinations:
LHS = set(item) - set(RHS)
temp.append(list(LHS))
temp.append(list(RHS))
associationRule.append(temp)
temp = []
length = length - 1
return associationRule



def transac(comb,Data,min_support):
abc={}
for i in comb:
count=0
for j in Data:
if (set(i).issubset(j)):
count+=1
#print()
abc[i]=count
minii=[]
for x, y in abc.items():
if((y/len(Data))>=min_support):
First_itt_minsprtt.append(list(x))
First_itt_minsprtt.append(y)
minii.append(x)
are=minii
gf=[]
for i in range (len(are)):
for j in range(i+1,len(are)):
gf.append(tuple(set(are[i]+are[j])))
return gf,minii
sfg=[]
check=True
while check:
fnl,rdcd=transac(comb,Data,min_support)
if(len(fnl)!=0):
comb=fnl
sfg.append(rdcd)
print(fnl)
print(rdcd)
else:
if(len(rdcd)==0):
stp=sfg[-1]
else:
```

```
stp=rdcd
check=False
print("Dpo")
print(First_itt_minsprtt)
print("Frequent Item Sets")
print(stp)
def AssociationRules(freqSet):
associationRule = []
import itertools
for item in freqSet:
if isinstance(item, list):
if len(item) != 0:
length = len(item) - 1
while length > 0:
combinations = list(itertools.combinations(item, length))
temp = []
LHS = []
for RHS in combinations:
LHS = set(item) - set(RHS)
temp.append(list(LHS))
temp.append(list(RHS))
associationRule.append(temp)
temp = []
length = length - 1
return associationRule
associationRules = AssociationRules(First_itt_minsprtt)
def AprioriAlgorithm(rules, dataset, minSupport, minConfidence):
returnAprioriAlgorithm = []
for rule in rules:
supportOfX = 0
supportOfXinPercent = 0
supportOfXandY = 0
supportOfXandYinPercent = 0
for transaction in dataset:
if set(rule[0]).issubset(set(transaction)):
supportOfX = supportOfX + 1
if set(rule[0] + rule[1]).issubset(set(transaction)):
supportOfXandY = supportOfXandY + 1
supportOfXinPercent = (supportOfX * 1.0 / 20) * 100
supportOfXandYinPercent = (supportOfXandY * 1.0 / 20) * 100
confidence = (supportOfXandYinPercent / supportOfXinPercent) * 100
if confidence >= minConfidence:

returnAprioriAlgorithm.append(rule)

return returnAprioriAlgorithm


Apr_alg=AprioriAlgorithm(associationRules, Data , min_support, min_confidence)

counter = 1
if len(Apr_alg) == 0:
print("There are no association rules for this support and confidence.")
else:
for i in Apr_alg:
if counter == 4:
print("\n"+str(i[0]) + "------->" + str(i[1])+"\n")
counter = 0
else:
print(i, end='  ')
counter = counter + 1



print("Time for Aprior Algorithm is")
print(time.time()-Start_time)
```

Output:

```
Sup(x): 10.0
Sup(X & Y:) 10
Confidence of List: 100
['Motorola,', 'Ericsson,']---->['Sony']

Sup(x): 25.0
Sup(X & Y:) 10
Confidence of List: 40
['Sony', 'Ericsson,']---->['Motorola,']

Sup(x): 10.0
Sup(X & Y:) 10
Confidence of List: 100
['Sony', 'Motorola,']---->['Ericsson,']

Time for Aprior Algorithm is
0.007231712341308594
```

Apriori Algorithm – 0.007231712341308594

BruteForce Algorithm

```
import sys
import time
import tkinter as tk
from tkinter import filedialog


print("Select the option whether you are reading in Percentage or Value")
print("1.Percentage")
print("2.Points")
option=int(input())
print("Enetr the support")
min_support=float(input())
print("Enter the confidence:")
min_confidence=float(input())
if option==1:
min_support=min_support/100
min_confidence=min_confidence/100
t_file = input("Enter the file path: ")
with open("C:/Users/HP/Desktop/datasets/items.txt") as f:
items = f.read().replace("\n", "").split(",")
items.sort()

print(items)

with open(t_file) as f:
db = [l.replace("\n", "").split(" ") for l in f]
a=[]
print("------------------------------------------------------------ INPUT TRANSACTIONS:")
for transaction in db:
print(transaction)
for i in transaction:
a.append(i)
#items=list(set(a))
```

```
def generate_k(items, k):

if k == 1:
return [[x] for x in items]

all_res = []
for i in range(len(items)-(k-1)):
for sub in generate_k(items[i+1:], k-1):
tmp = [items[i]]
tmp.extend(sub)
all_res.append(tmp)
return all_res


def scan(db, s):
count = 0
for t in db:
if set(s).issubset(t):
count += 1
return count


def generate_frequent_and_support():
frequent = []
support = {}
for k in range(1, len(items)+1):
current = []
for comb in generate_k(items, k):
count = scan(db, comb)
if count/len(db) >= min_support:
support[frozenset(comb)] = count/len(db)
current.append(comb)
if len(current) == 0:
break
frequent.append(current)
return frequent, support


class Rule:

def __init__(self, left, right, all):
self.left = list(left)
self.left.sort()
self.right = list(right)
self.right.sort()
self.all = all

def _str_(self):
return ",".join(self.left)+" => "+",".join(self.right)

def _hash_(self):
"""
Store support value to dict
:return: hash value in the object
"""
return hash(str(self))


def generate_rules(frequent, support):
all_rule = set()
all_result = []
for k_freq in frequent:
if len(k_freq) == 0:
continue
```

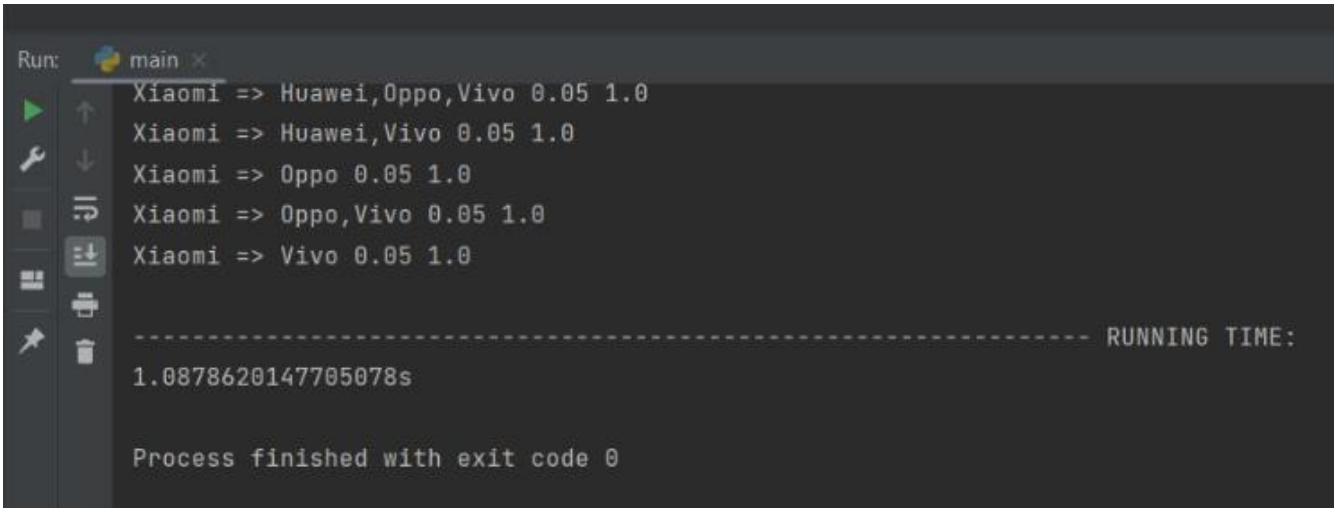
```
if len(k_freq[0]) < 2:
continue
for freq in k_freq:
for i in range(1, len(freq)):
for left in generate_k(freq, i):
tmp = freq.copy()
right = [x for x in tmp if x not in left]
all_rule.add(Rule(left, right, freq))
for rule in all_rule:
confidence = support[frozenset(rule.all)] / support[frozenset(rule.left)]
if confidence >= min_confidence:
all_result.append([rule, support[frozenset(rule.all)], confidence])

all_result.sort(key=lambda x: str(x[0]))

return all_result


if __name__ == '__main__':
start_time = time.time()
f, s = generate_frequent_and_support()
all_result = generate_rules(f, s)
end_time = time.time()
print("\n---------------------------------------------------- RULES SUPPORT CONFIDENCE:")
for r in all_result:
print(r[0], r[1], r[2])
print("\n---------------------------------------------------------------- RUNNING TIME:")
print(str(end_time - start_time) + "s")
```

Output



Comparing Apriori and Bruteforce:

When we try to generate association rules for transaction all data bases with both Apriori and Bruteforce algorithms, we found that Apriori algorithm is way faster than Brute force.

Here are the time results:

ForApriori

0.007231712341308594

For Brute Force

1.0878620147705078

Conclusion:

From the above observations in the data sets,we can conclude that the Apriori algorithm is faster than the Brute force.