lOMoARcPSD|17241975

**Experiment 9**

**Aim:** Implementation of Association Rule Mining algorithm (Apriori)

# **Theory:**

The **Apriori algorithm** is used for mining frequent itemsets and devising association rules from a transactional database. The parameters “support” and “confidence” are used. **Support** refers to items’ frequency of occurrence; **confidence** is a conditional probability.

Items in a transaction form an item set. The algorithm begins by identifying frequent, individual items (items with a frequency greater than or equal to the given support) in the database and continues to extend them to larger, frequent itemsets.

# **Algorithm:**

The following are the main steps of the algorithm:

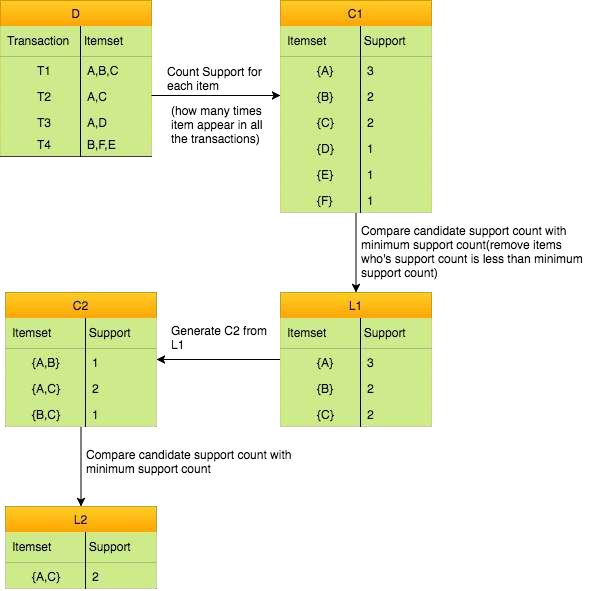
1. Calculate the support of item sets (of size k = 1) in the transactional database (note that support is the frequency of occurrence of an itemset). This is called *generating the candidate set*.
2. Prune the candidate set by eliminating items with a support less than the given threshold.
3. Join the frequent itemsets to form sets of size k + 1, and repeat the above sets until no more itemsets can be formed. This will happen when the set(s) formed have a

support *less than* the given support.

# **Example:**

Consider the following database(D) with 4 transactions (T1, T2, T3 and T4). Let minimum support = 2%

Let minimum confidence = 50%



Now generate Association rule:



# **Program:**

*import* itertools list = []

support = float(input("Enter the minimum support: "))

items = ['i1', 'i2', 'i3', 'i4', 'i5']

num = int(input("Enter number of transactions: "))

*for* i *in* range(num):

print("Enter the items bought in transaction ", i + 1, " separated by a comma: ")

val = input() list.append(val)

print("Transactions are as follows: ")

*for* i *in* list: print(i)

print("The candidate set C1 is : ", items)

*# calculate support for all items in candidate set C1*

supportc1 = []

*for* item *in* items: val = 0

*for* i *in* range(len(list)):

*if* item *in* list[i]: val += 1

supportc1.append(float(val / 5))

*for* i *in* range(5):

print("Support for ", items[i], " is : ", supportc1[i])

print("Genrating L1 which is frequent 1-item set from C1")

l1 = []

*for* i *in* range(len(items)):

*if* supportc1[i] >= support: l1.append(items[i])

print("L1 is : ", l1)

*#*

*# Generating Candidate set C2*

c2 = []

*for* val *in* itertools.combinations(items, 2): c2.append(val)

*# calculating support for all items in c2* print("Candidate set c2 is : ", c2) supportc2 = []

*for* i *in* range(len(c2)): val = 0

*for* item *in* list:

*if* c2[i][0] *in* item *and* c2[i][1] *in* item: val += 1

supportc2.append(float(val / 5))

*for* i *in* range(len(c2)):

print("Support for ", c2[i], " is : ", supportc2[i])

*# generating L2 from C2*

l2 = []

*for* i *in* range(len(c2)):

*if* supportc2[i] >= support: l2.append(c2[i])

print(l2)

*#*

c3 = []

*for* val *in* itertools.combinations(items, 3): c3.append(val)

supportc3 = []

*for* i *in* range(len(c3)): val = 0

*for* item *in* list:

*if* c3[i][0] *in* item *and* c3[i][1] *in* item *and* c3[i][2] *in* item: val += 1

supportc3.append(float(val / 5))

*for* i *in* range(len(c3)):

print("Support for : ", c3[i], " is: ", supportc3[i])

*# generating L3 from C3*

l3 = []

*for* i *in* range(len(c3)):

*if* supportc3[i] >= support:

l3.append(c3[i])

print("L3 is : ", l3)

*#*

confidence = []

*for* i *in* range(len(l3)): val = 0

div = 0

*for* item *in* list:

*if* l3[i][0] *in* item:

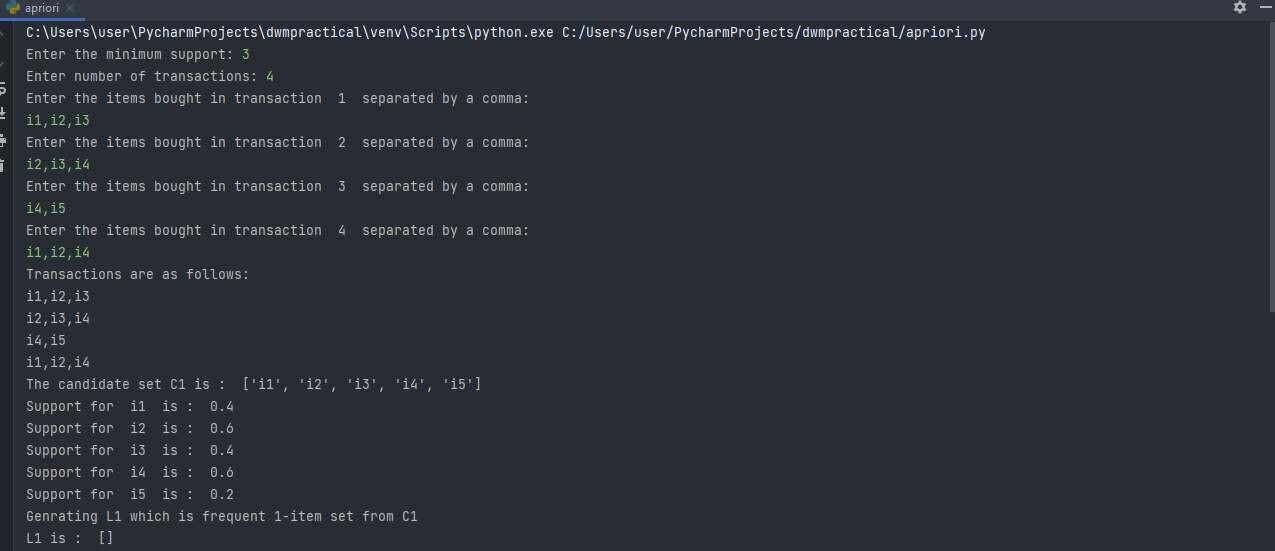
*if* l3[i][0] *in* item *and* l3[i][1] *in* item *and* l3[i][2] *in* item: val += 1

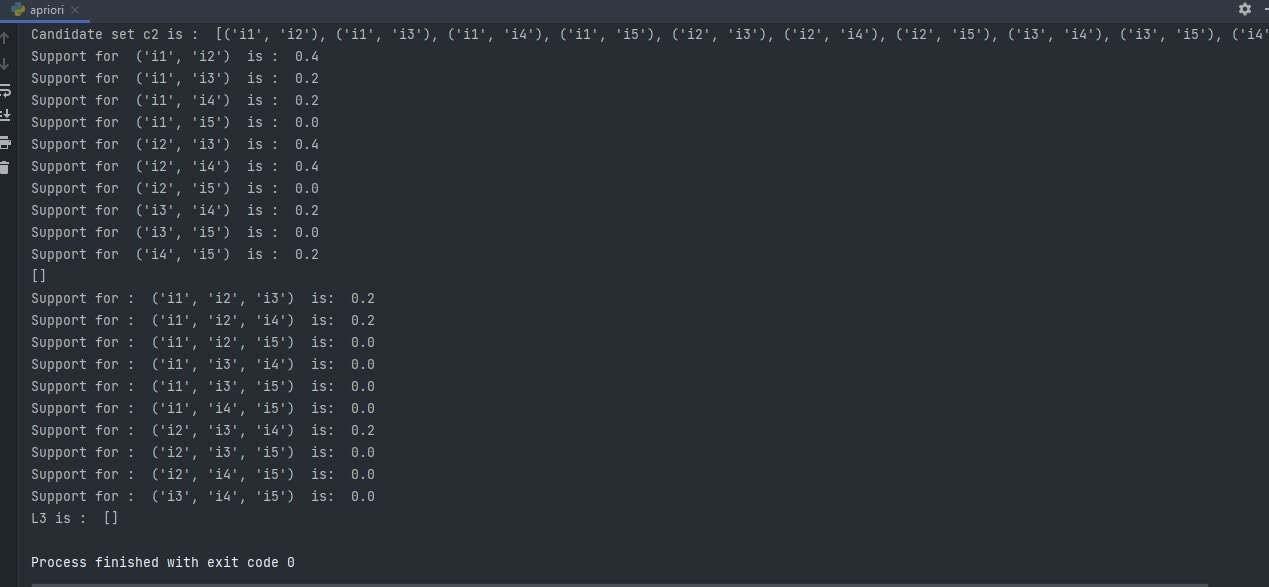
div += 1 confidence.append(float(val / div))

*for* i *in* range(len(l3)):

print("Confidence for ", l3[i], " is: ", confidence[i])

# **Output:**





**Conclusion:** Thus, we successfully implemented Association Rule Mining algorithm (Apriori)