Course: CS634

Subject: Midterm Project Implementation 1

Topic: Apriori Algorithm Implementation

Name: Saket Sandeep Manolkar

Email - sm445@njit.edu

Introduction

Apriori Algorithm Overview

Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Other algorithms are designed for finding association rules in data having no transactions (Winepi and Minepi), or having no timestamps (DNA sequencing). Each transaction is seen as a set of items (an itemset). Given a threshold C, the Apriori algorithm identifies the item sets which are subsets of at least C transactions in the database.

Implementation Overview

The implementation is built from scratch. It uses Java as the main programming language. The implementation uses Database to store the items sets and transactions. The default database is Oracle.

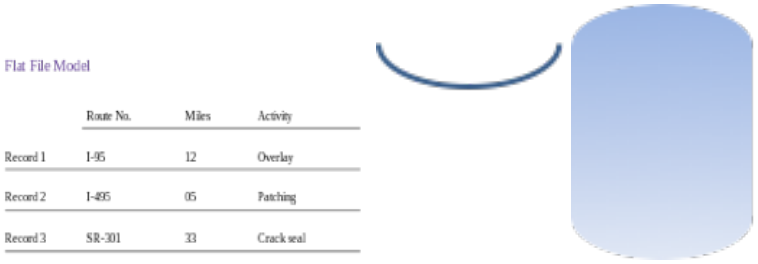
When you run the program, it will prompt you with number of choices to

choose from based on the dataset type, support and confidence.

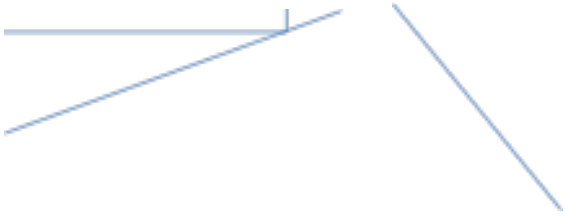
Based on the choice, the program will read the flat file database, parse the items names and transactions, and then load them to the database to parse the transaction for support and confidence to find the association rules using Apriori algorithm.

The implementation also can run in Graphical User Interface. When the graphical interface starts, user can choose from the existing database, specify the support and confidence to run the program.

Implementation Architecture



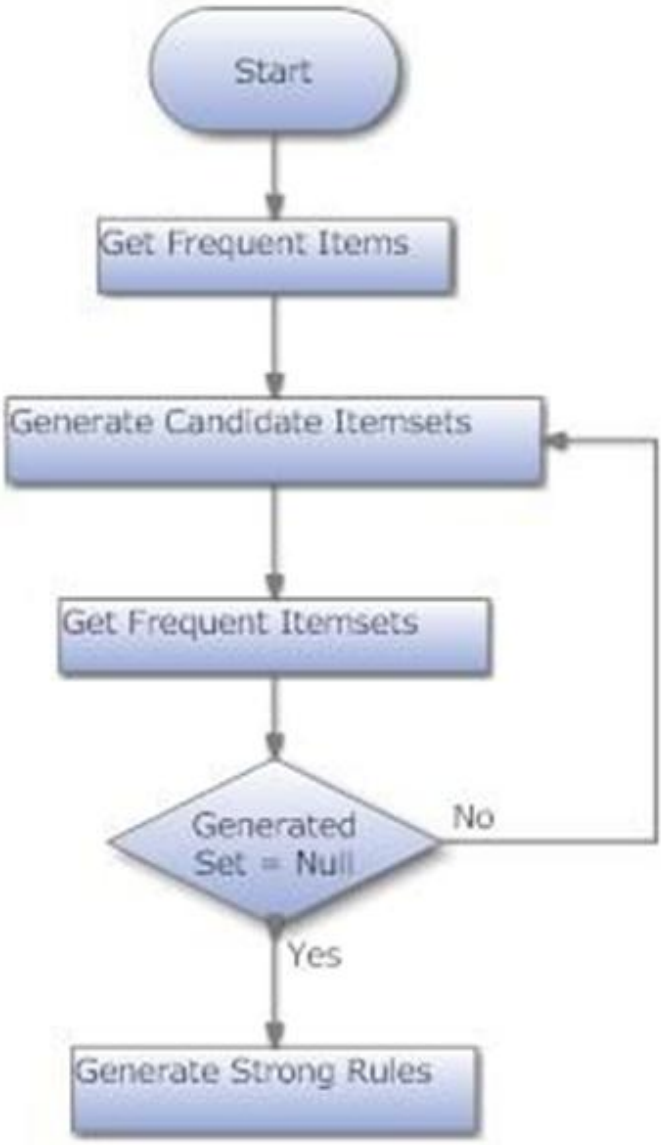
Load Transactions To SQL DB



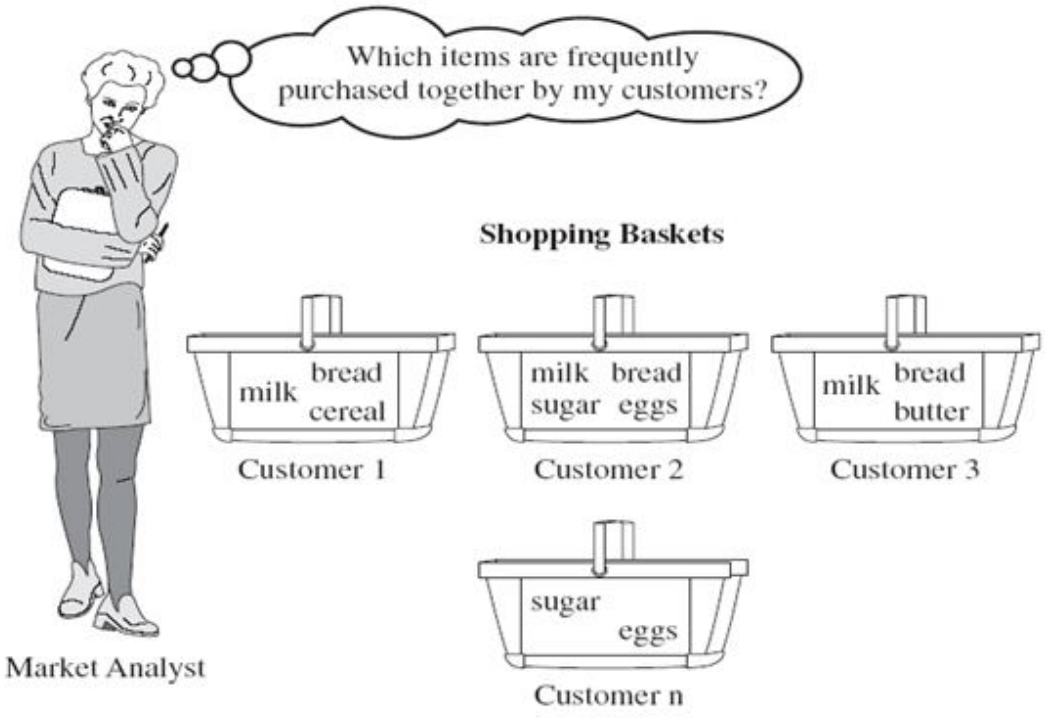
Read File Based Transactions

Run the algorithm implementation

Request Transactions to examine the support/confidence



Strong Rules Result (Association Rules)



Assumptions

The implementation has some assumption to make sure the running of the application flawless and without exception.

The assumptions are:

1. Assumption 1 2. 2
3. ...

Items Names Aliases (optional)

Due to limitation of the number of characters per field in database, the item names are aliased with A, B,C, D, etc.... to avoid any error. This is only used when retrieving data from the database. All calculations are done based the mapped item name.

Requirement

Software

Database Oracle 11g Java JDBC

Java JDK 1.6+
Oracle JDBC 6+

Hardware

Backend Unix, Linux, Windows

Component	Requirement

Component	Requirement
Database	Solaris 11g Windows 7 optional Windows 8 optional

Implementation List of Source Code Files

The implementation is built from scratch. It uses Java as the main programming language. The implementation uses Database to store the items sets and transactions. The default database is Oracle.

The program consists of the following set of Java interfaces and classes.

1) AprioriAlgorithmImpl.java

This is the main class to start the program. It start the programs in console mode. There is no command line arguments required. The program will prompt the user with options menu to select how to execute and run the program.

. 2) AprioriUImpl.java

This class provides graphical interface to the implementation when the option –gui is provided to the command line options for the main class.

. 3) DatabaseConnection.java

This class is creates the Database Connection taking the data from etc/ database.props file. The default database is Oracle. The class forms the JDBC URL and complete Connection to execute the queries.

. 4) DatabaseQuery.java

This class extends the DatabaseConnection to execute the queries to retrieve transaction and support information in the database.

. 5) GenerateItemsTransactions.java

This is a utility class to generate transactions. The transactions are randomly generated

. 6) GetProperty.java

This class loads the information from etc/database.props file to set any customization.

. 7) ItemsScannerIfc.java

This is an interface class to lay down all the abstract methods to scan the items sets to build query as one scan in the database and gets the support and confidence for every items set in the transactions

. 8) ItemsSetsGenerator.java

This class generates the Items Sets names and permutation based on the support and non- support items from previous call and items sets support.

9) ItemsSetsGeneratorTest.java

This is a Unit-Test class to make sure the core is working

10) LogFileWriter.java

Log file wrapper to write messages into log/run.log

11) LogToFile.java

Sends the logs messages to the LogFileWriter to write to the log/run.log

12) SQLItemsScanner.java

Implements the interface ItemsScannerIfc to build the query, scan the transaction for supports in the database and get the confidence.

11 Utils.java

Helper functions to run the program. **NOTE**: List of source code is listed in the Appendix

Compile the Source Code

The source code files are listed in the Appendix 1
To compile the source code, follow the procedure:
Prerequisites: Java JDK 1.6 and higher must be installed in the machine.

. 1) Save all the source code files from the Appendix into a temp directory. Make sure the files are named exactly as they are labeled in the Appendix

. 2) Create parent/children directories as the package name: edu/njit/cs634/projects/impl1
mkdir edu/njit/cs634/projects/impl1

. 3) Run the java compiler command:
javac –d edu/njit/cs634/projects/impl1 *.java
NOTE: if javac is not in the PATH environment variables, use the full path to the executable: JAVA_HOME/jdk/bin/javac
The command will create the necessary classes and place them in the package directory and ready to be used.

How to Run the Application

Prerequisites:

. 1) Java JDK 1.6 and higher must be installed in the machine and "Compile the Source Code" step was performed.

. 2) JDBC Driver jars. Default is oracle. It can be downloaded from http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html

. 3) All the assumptions are in place as explained in the "Assumptions" Section.

. 4) Database is available and ready to connect.

. 5) You selected and configured the database to use (default is oracle)

Console Mode

To Run the program, just run the command:
JAVA_HOME/bin/java -cp .;<name of jdbc jar>
edu.njit.cs634.projects.impl1.AprioriAlgorithmImpl Example:
JAVA_HOME/bin/java -cp .;ojdbc6.jar
edu.njit.cs634.projects.impl1.AprioriAlgorithmImpl

List of Data Sets

The data sets consist of data items names and 20 tuples for each dataset. The items and translations are selected randomly based on data from each site. Shoprite data base is using the labels A,B,C, D, E as items names and transactions.

The Custom data is filled by the user to any other data to use. The custom data must follow the example provided to the user in other data in order to work.

NOTE: Any data set can be changed as long as the file follows the same format

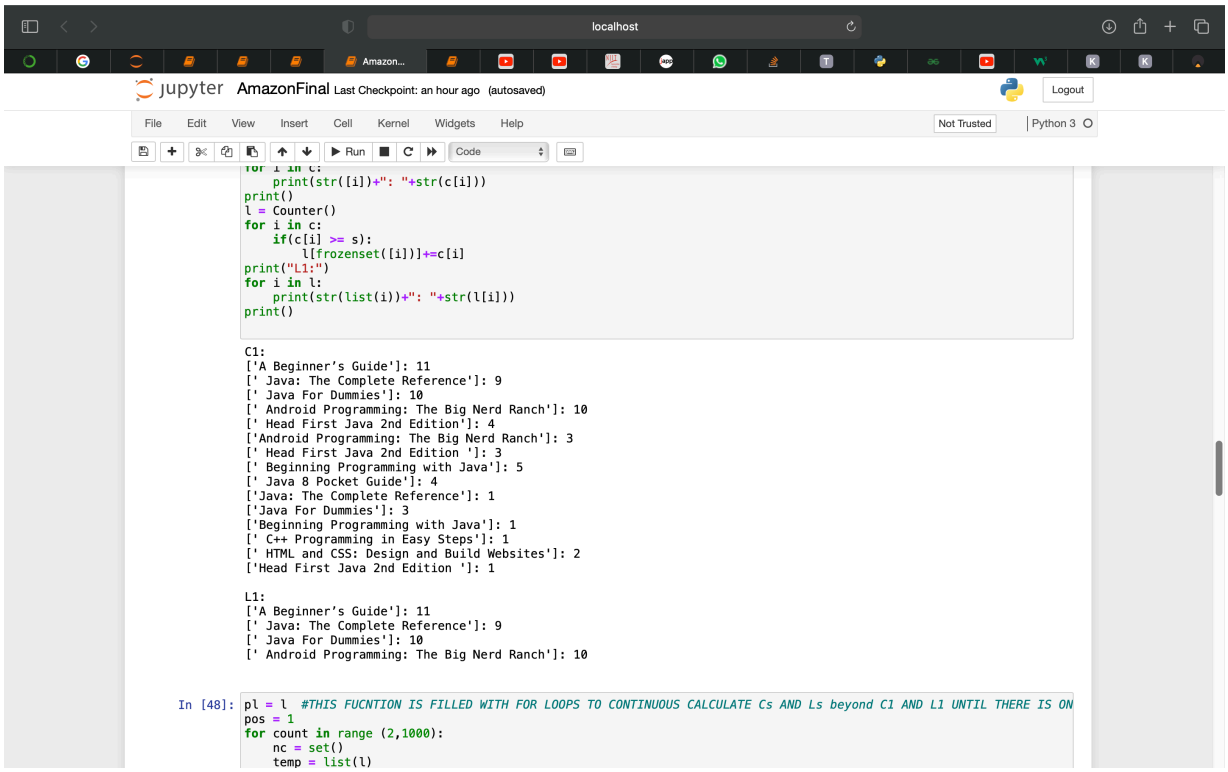
Data sets are included with the project package in the folder data. There are 5 data sets in total: Nike, K- Mart, BestBuy, Amazon, Shoprite. They are in CSV format

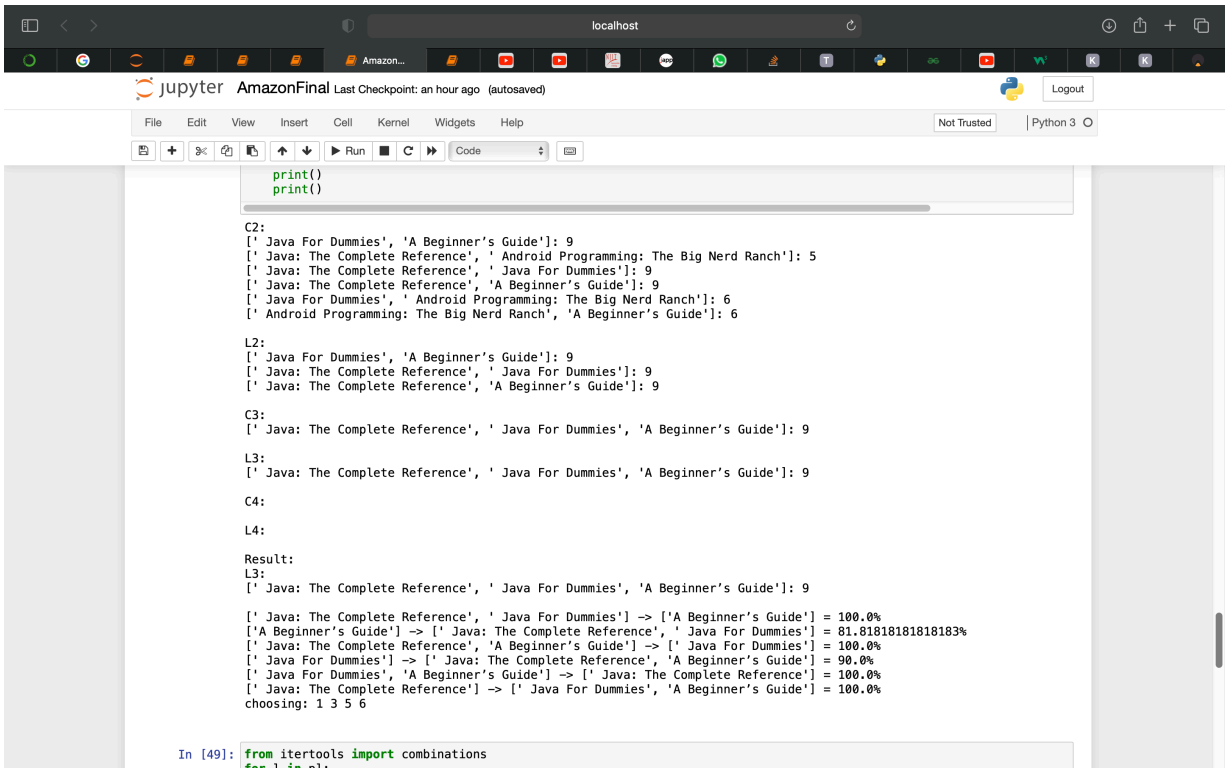
Testing Implementation and Data Set

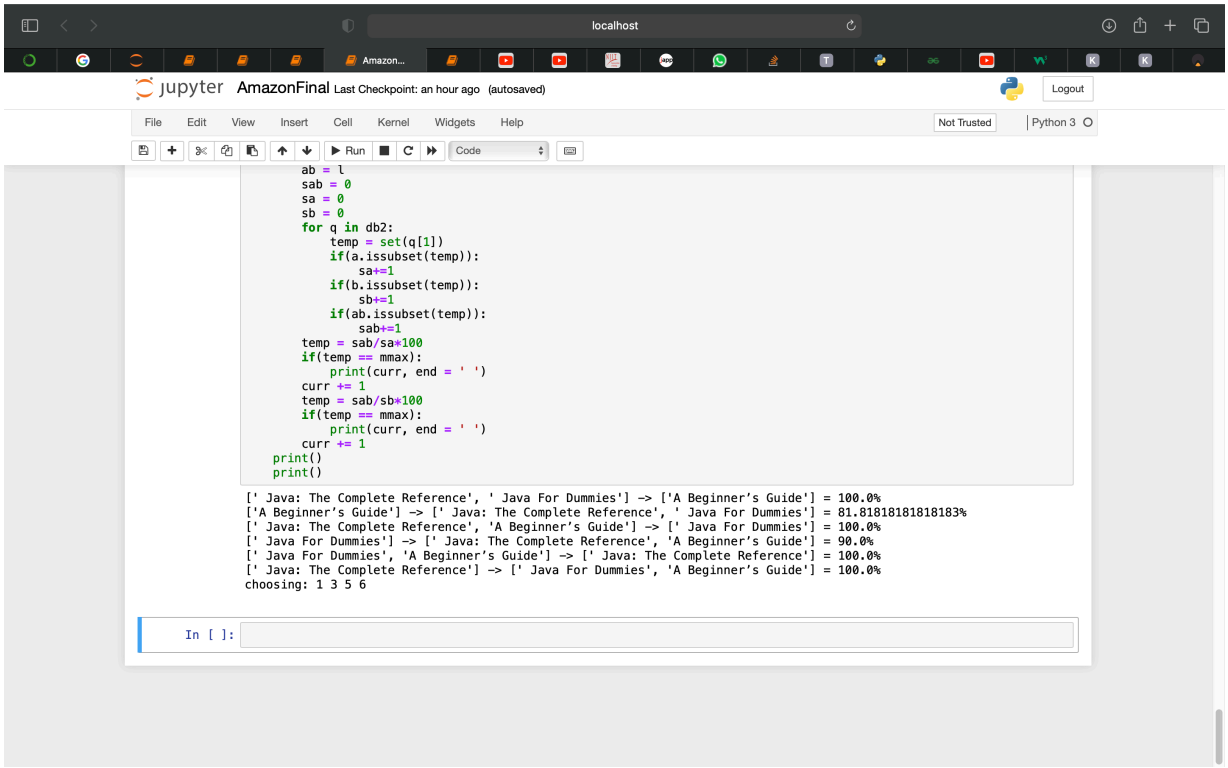
Console Mode

This sections describes several test for each data sets type with different support and confidence values. It tests each data set respectively.

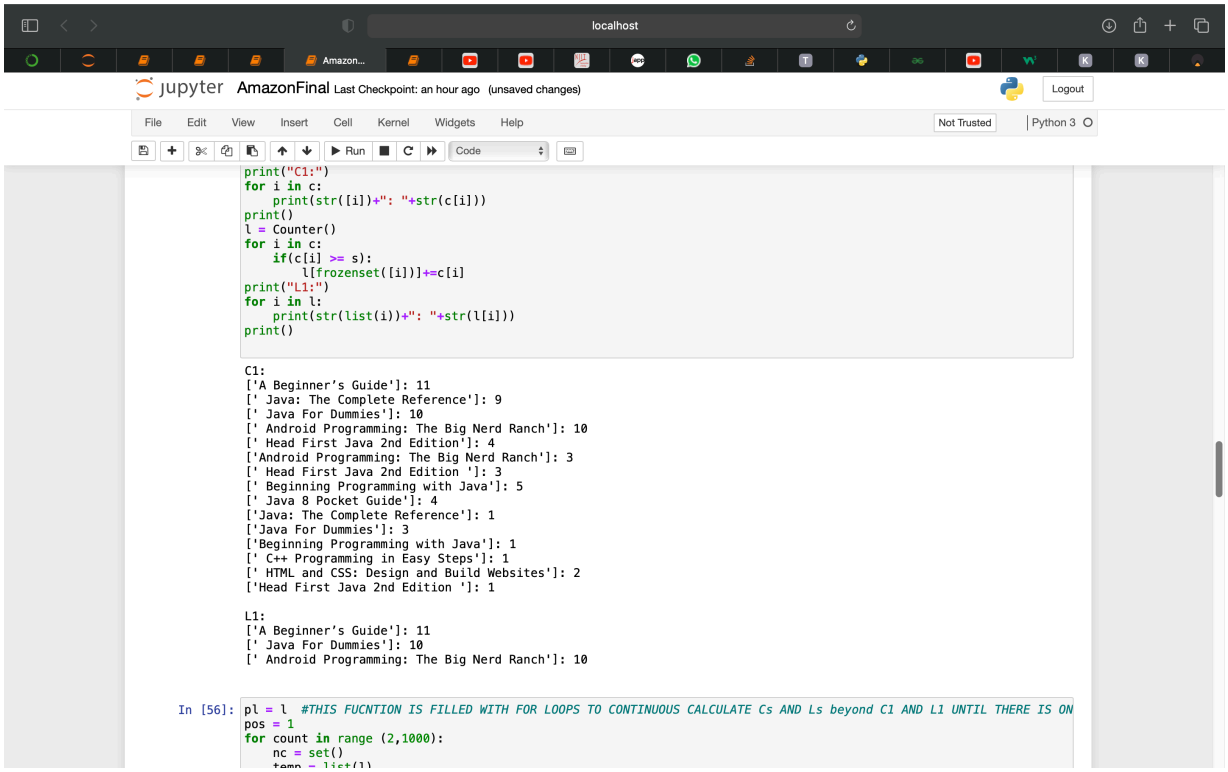
Relations for amazon dataset at support and confidence = 50 %

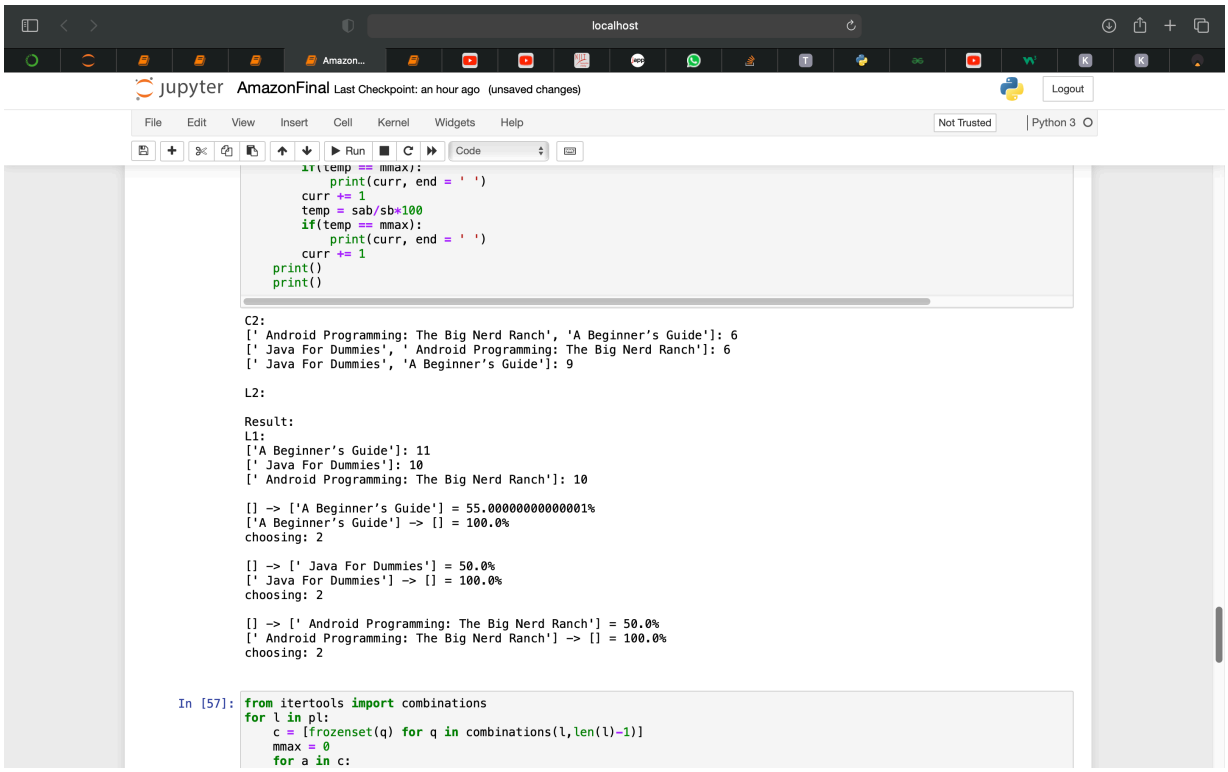


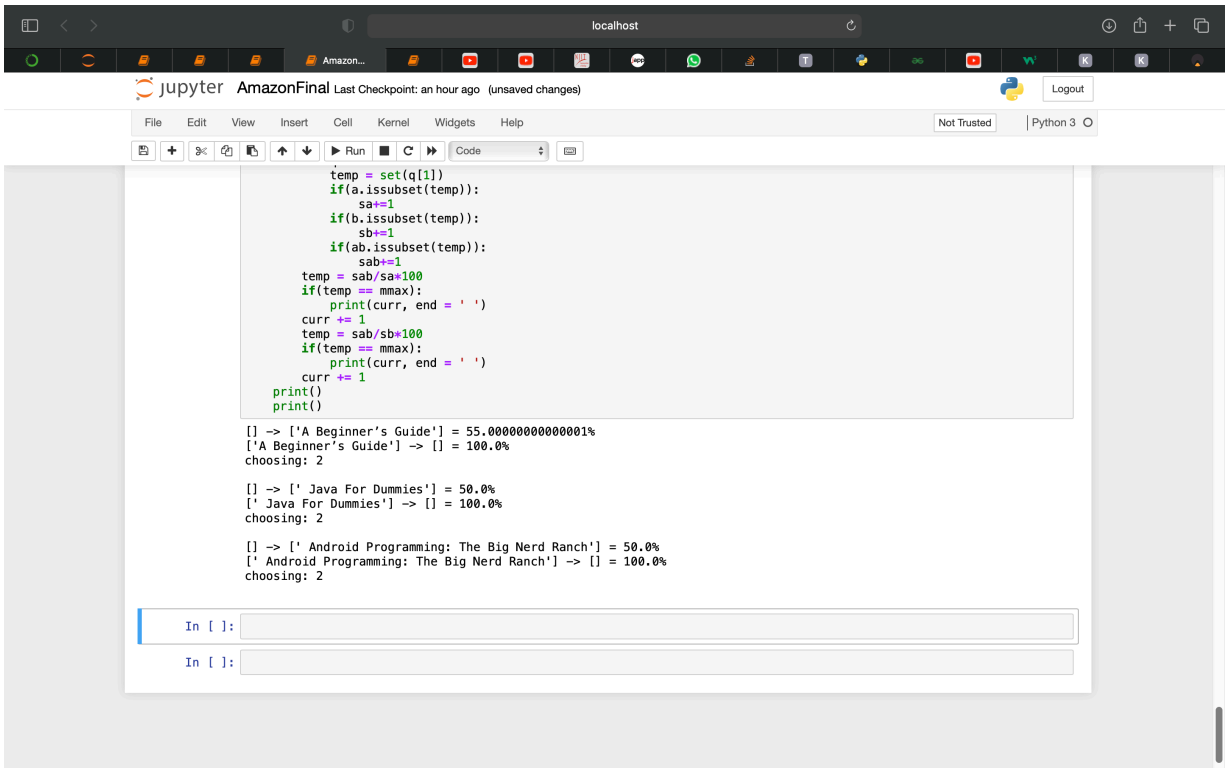

localhost
jupyter AmazonFinal Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
print()
print()
C2:
[' Java For Dummies', 'A Beginner's Guide']: 9
[' Java: The Complete Reference', ' Android Programming: The Big Nerd Ranch']: 5
[' Java: The Complete Reference', ' Java For Dummies']: 9
[' Java: The Complete Reference', 'A Beginner's Guide']: 9
[' Java For Dummies', ' Android Programming: The Big Nerd Ranch']: 6
[' Android Programming: The Big Nerd Ranch', 'A Beginner's Guide']: 6
L2:
[' Java For Dummies', 'A Beginner's Guide']: 9
[' Java: The Complete Reference', ' Java For Dummies']: 9
[' Java: The Complete Reference', 'A Beginner's Guide']: 9
C3:
[' Java: The Complete Reference', ' Java For Dummies', 'A Beginner's Guide']: 9
L3:
[' Java: The Complete Reference', ' Java For Dummies', 'A Beginner's Guide']: 9
C4:
L4:
Result:
L3:
[' Java: The Complete Reference', ' Java For Dummies', 'A Beginner's Guide']: 9
[' Java: The Complete Reference', ' Java For Dummies'] -> ['A Beginner's Guide'] = 100.0%
['A Beginner's Guide'] -> [' Java: The Complete Reference', ' Java For Dummies'] = 81.81818181818183%
[' Java: The Complete Reference', 'A Beginner's Guide'] -> [' Java For Dummies'] = 100.0%
[' Java For Dummies'] -> [' Java: The Complete Reference', 'A Beginner's Guide'] = 90.0%
[' Java For Dummies', 'A Beginner's Guide'] -> [' Java: The Complete Reference'] = 100.0%
[' Java: The Complete Reference'] -> [' Java For Dummies', 'A Beginner's Guide'] = 100.0%
choosing: 1 3 5 6
In [49]:
from itertools import combinations


localhost
jupyter AmazonFinal Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
ab = l
sab = 0
sa = 0
sb = 0
for q in db2:
temp = set(q[1])
if(a.issubset(temp)):
sa+=1
if(b.issubset(temp)):
sb+=1
if(ab.issubset(temp)):
sab+=1
temp = sab/sa*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
temp = sab/sb*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
print()
print()
[' Java: The Complete Reference', ' Java For Dummies'] -> ['A Beginner's Guide'] = 100.0%
['A Beginner's Guide'] -> [' Java: The Complete Reference', ' Java For Dummies'] = 81.81818181818183%
[' Java: The Complete Reference', 'A Beginner's Guide'] -> [' Java For Dummies'] = 100.0%
[' Java For Dummies'] -> [' Java: The Complete Reference', 'A Beginner's Guide'] = 90.0%
[' Java For Dummies', 'A Beginner's Guide'] -> [' Java: The Complete Reference'] = 100.0%
[' Java: The Complete Reference'] -> [' Java For Dummies', 'A Beginner's Guide'] = 100.0%
choosing: 1 3 5 6
In [ ]:

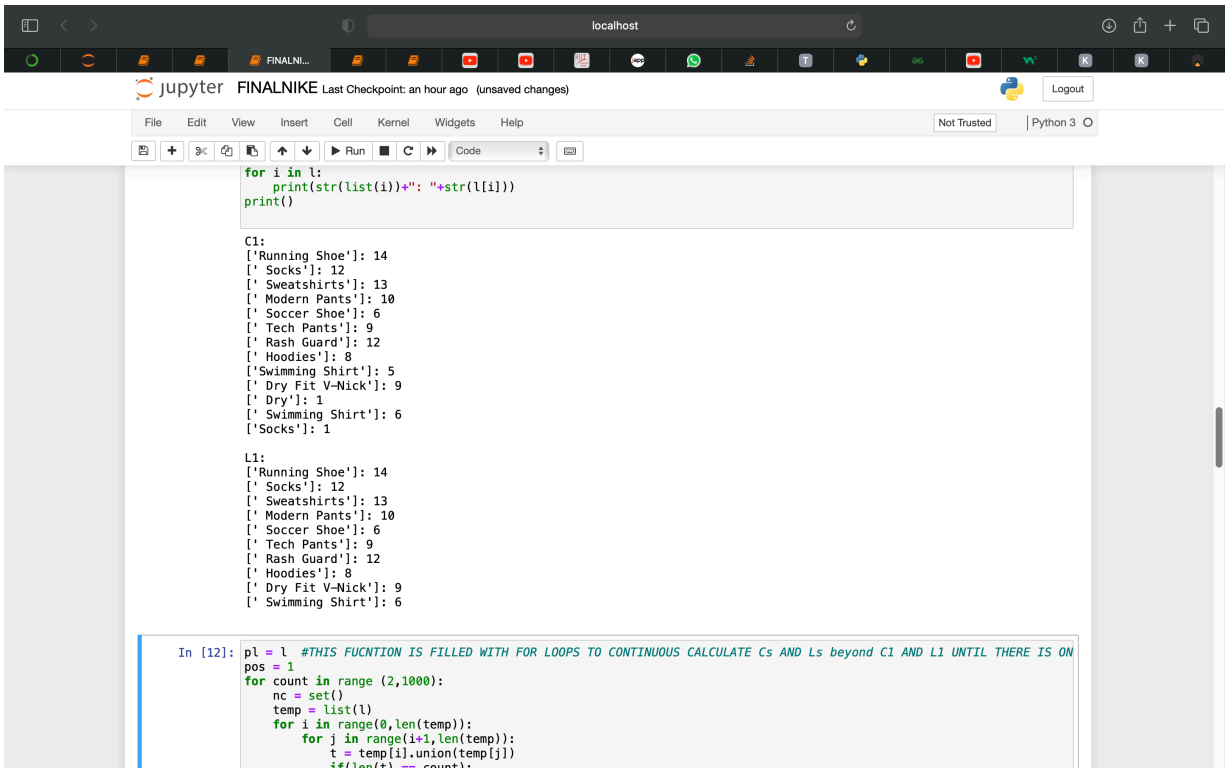
Relations for amazon dataset at support = 70 %

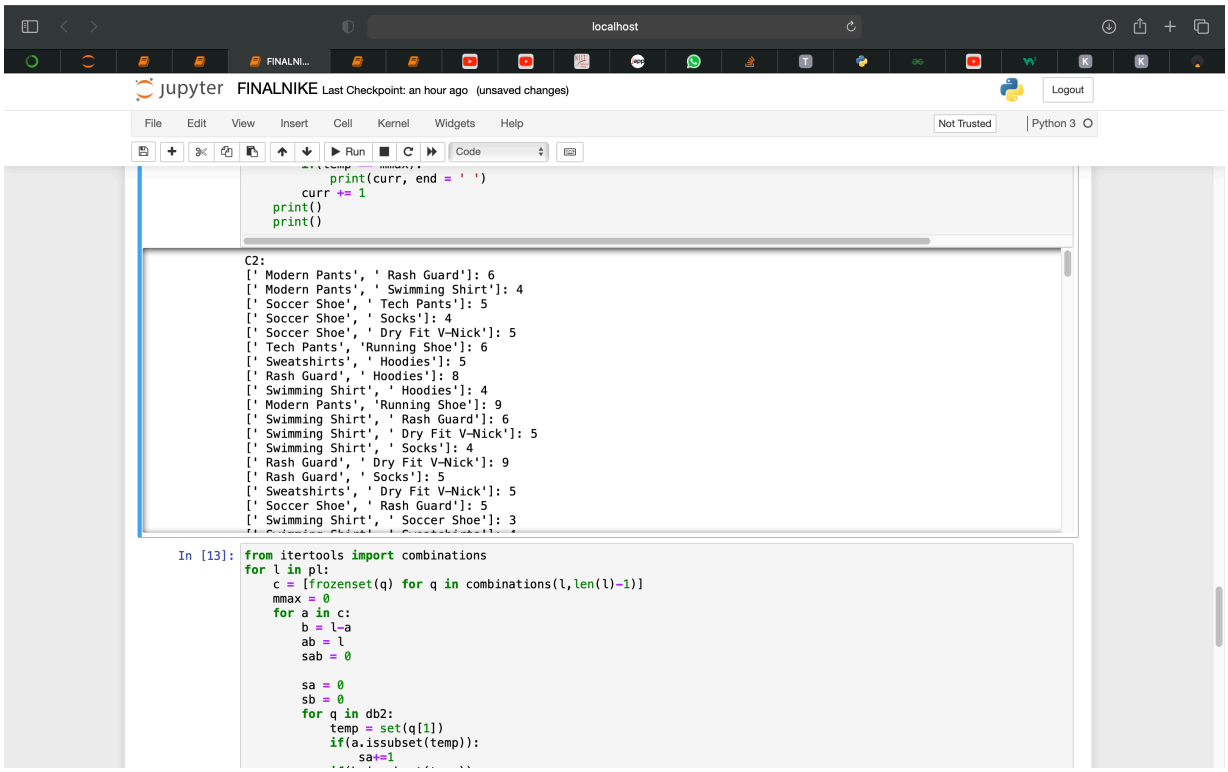


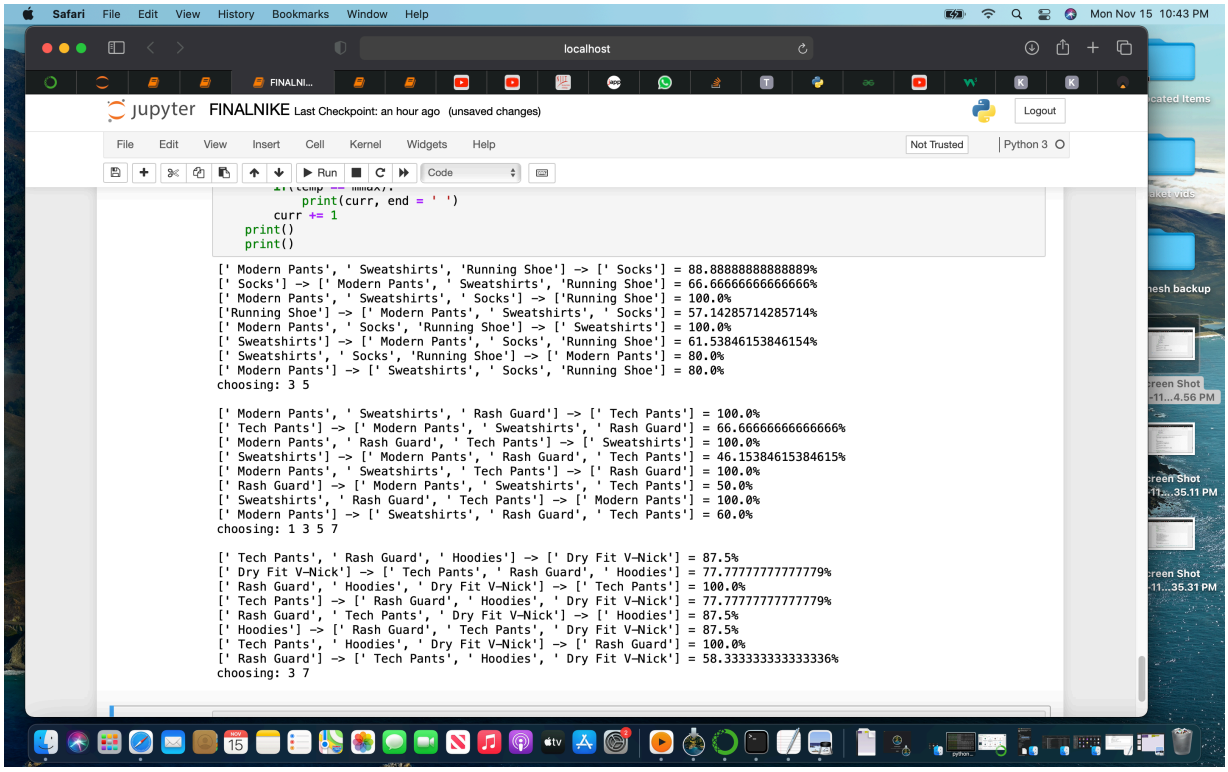

localhost
jupyter AmazonFinal Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
if(temp == mmax):
print(curr, end = ' ')
curr += 1
temp = sab/sb*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
print()
print()
C2:
[' Android Programming: The Big Nerd Ranch', 'A Beginner's Guide']: 6
[' Java For Dummies', ' Android Programming: The Big Nerd Ranch']: 6
[' Java For Dummies', 'A Beginner's Guide']: 9
L2:
Result:
L1:
['A Beginner's Guide']: 11
[' Java For Dummies']: 10
[' Android Programming: The Big Nerd Ranch']: 10
[] -> ['A Beginner's Guide'] = 55.00000000000001%
['A Beginner's Guide'] -> [] = 100.0%
choosing: 2
[] -> [' Java For Dummies'] = 50.0%
[' Java For Dummies'] -> [] = 100.0%
choosing: 2
[] -> [' Android Programming: The Big Nerd Ranch'] = 50.0%
[' Android Programming: The Big Nerd Ranch'] -> [] = 100.0%
choosing: 2
In [57]:
from itertools import combinations
for l in pl:
c = [frozenset(q) for q in combinations(l,len(l)-1)]
mmax = 0
for a in c:



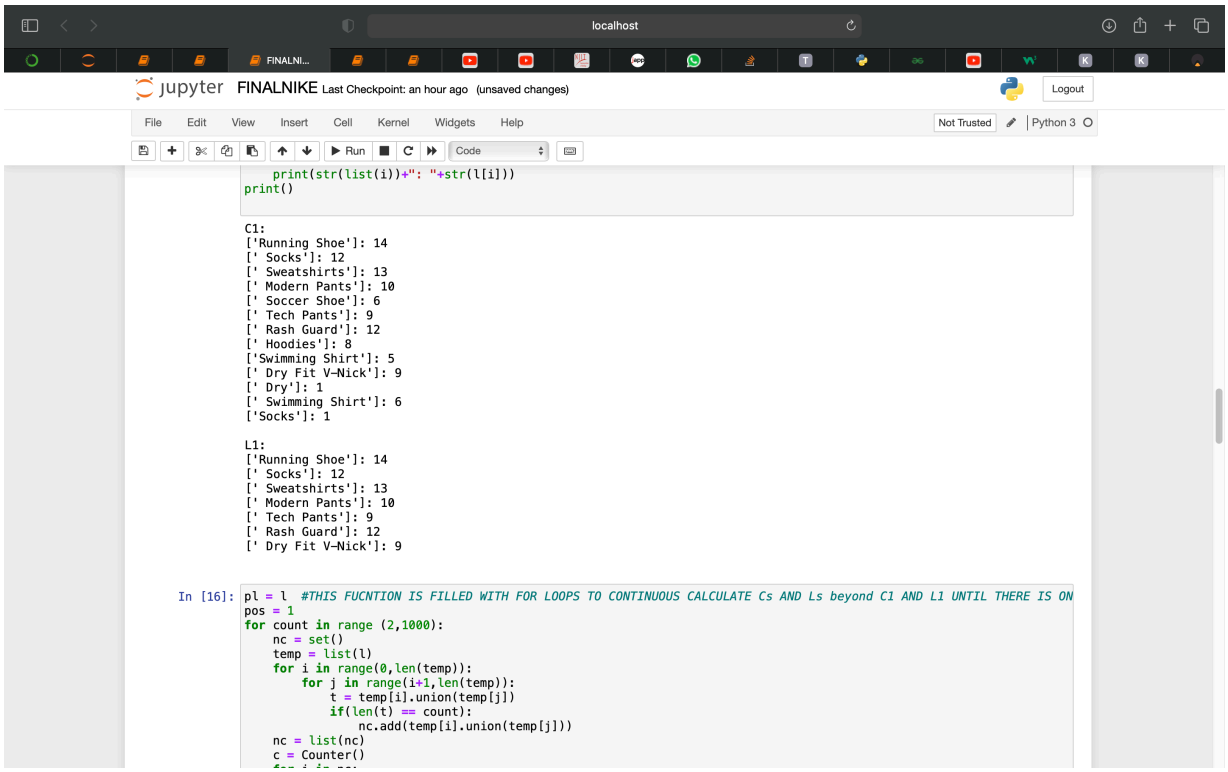
Relations for Nike dataset at support and confidence = 50 %


localhost
FINALNI...
jupyter FINALNIKE Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
for i in l:
print(str(list(i))+": "+str(l[i]))
print()
C1:
['Running Shoe']: 14
[' Socks']: 12
[' Sweatshirts']: 13
[' Modern Pants']: 10
[' Soccer Shoe']: 6
[' Tech Pants']: 9
[' Rash Guard']: 12
[' Hoodies']: 8
['Swimming Shirt']: 5
[' Dry Fit V-Nick']: 9
[' Dry']: 1
[' Swimming Shirt']: 6
['Socks']: 1
L1:
['Running Shoe']: 14
[' Socks']: 12
[' Sweatshirts']: 13
[' Modern Pants']: 10
[' Soccer Shoe']: 6
[' Tech Pants']: 9
[' Rash Guard']: 12
[' Hoodies']: 8
[' Dry Fit V-Nick']: 9
[' Swimming Shirt']: 6
In [12]:
pl = l #THIS FUCNTION IS FILLED WITH FOR LOOPS TO CONTINUOUS CALCULATE Cs AND Ls beyond C1 AND L1 UNTIL THERE IS ON
pos = 1
for count in range (2,1000):
nc = set()
temp = list(l)
for i in range(0,len(temp)):
for j in range(i+1,len(temp)):
t = temp[i].union(temp[j])


localhost
FINALNI...
jupyter FINALNIKE Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
print(curr, end = ' ')
curr += 1
print()
print()
C2:
[' Modern Pants', ' Rash Guard']: 6
[' Modern Pants', ' Swimming Shirt']: 4
[' Soccer Shoe', ' Tech Pants']: 5
[' Soccer Shoe', ' Socks']: 4
[' Soccer Shoe', ' Dry Fit V-Nick']: 5
[' Tech Pants', 'Running Shoe']: 6
[' Sweatshirts', ' Hoodies']: 5
[' Rash Guard', ' Hoodies']: 8
[' Swimming Shirt', ' Hoodies']: 4
[' Modern Pants', 'Running Shoe']: 9
[' Swimming Shirt', ' Rash Guard']: 6
[' Swimming Shirt', ' Dry Fit V-Nick']: 5
[' Swimming Shirt', ' Socks']: 4
[' Rash Guard', ' Dry Fit V-Nick']: 9
[' Rash Guard', ' Socks']: 5
[' Sweatshirts', ' Dry Fit V-Nick']: 5
[' Soccer Shoe', ' Rash Guard']: 5
[' Swimming Shirt', ' Soccer Shoe']: 3
In [13]:
from itertools import combinations
for l in pl:
c = [frozenset(q) for q in combinations(l,len(l)-1)]
mmax = 0
for a in c:
b = l-a
ab = l
sab = 0
sa = 0
sb = 0
for q in db2:
temp = set(q[1])
if(a.issubset(temp)):
sa+=1



Relations for Nike dataset at support = 70 %



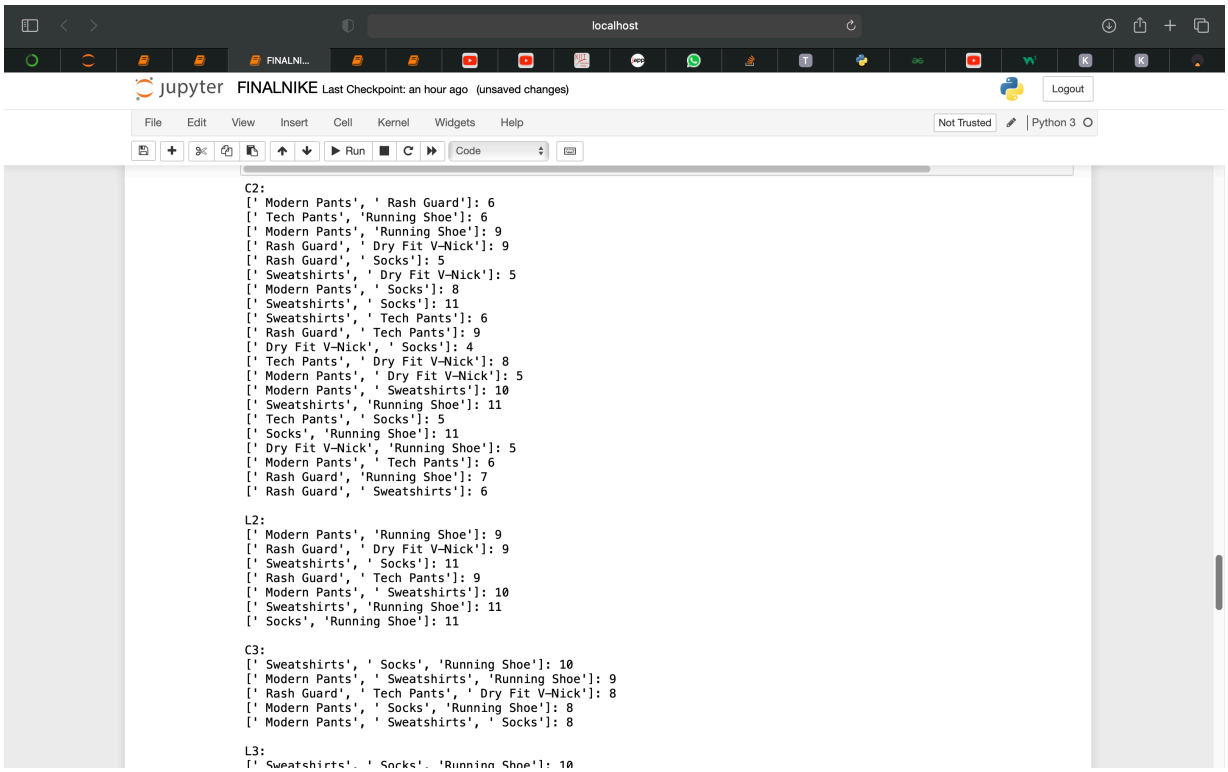

```
    print(str(list(i))+": "+str(l[i]))
print()
```

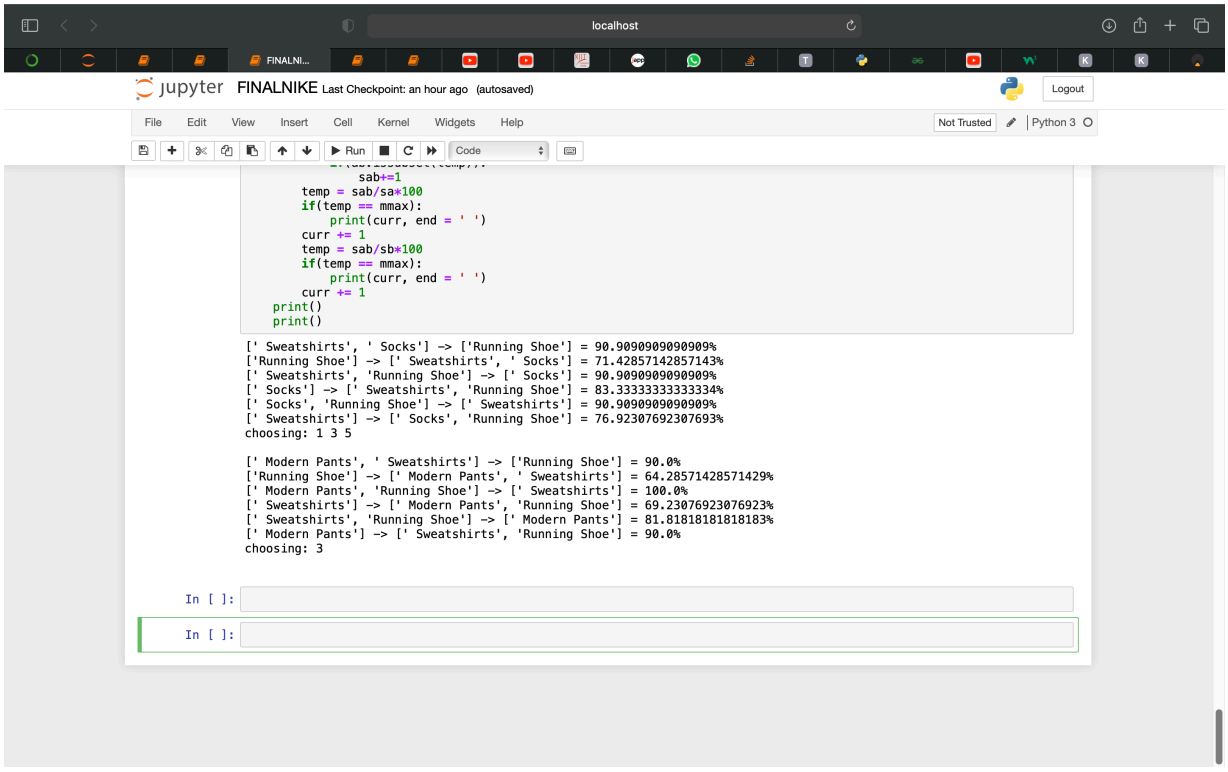
```
C1:
['Running Shoe']: 14
[' Socks']: 12
[' Sweatshirts']: 13
[' Modern Pants']: 10
[' Soccer Shoe']: 6
[' Tech Pants']: 9
[' Rash Guard']: 12
[' Hoodies']: 8
['Swimming Shirt']: 5
[' Dry Fit V-Nick']: 9
[' Dry']: 1
[' Swimming Shirt']: 6
['Socks']: 1

L1:
['Running Shoe']: 14
[' Socks']: 12
[' Sweatshirts']: 13
[' Modern Pants']: 10
[' Tech Pants']: 9
[' Rash Guard']: 12
[' Dry Fit V-Nick']: 9
```

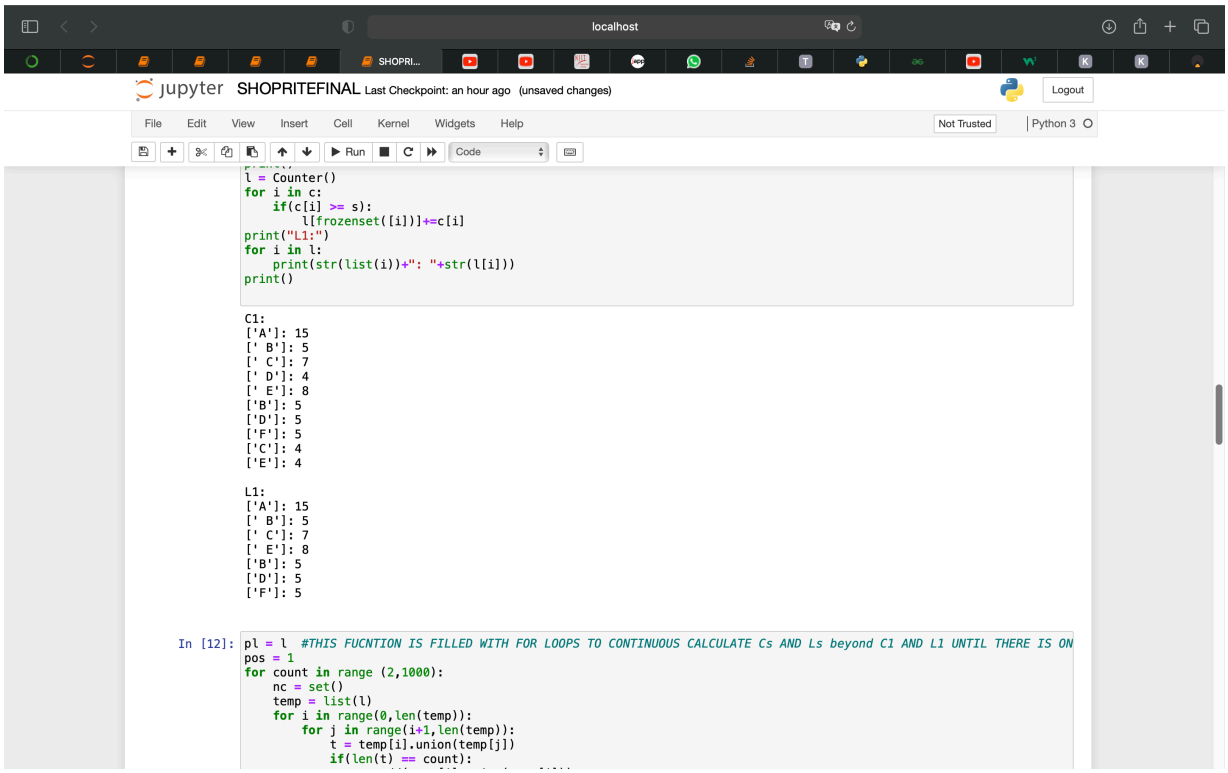
In [16]:

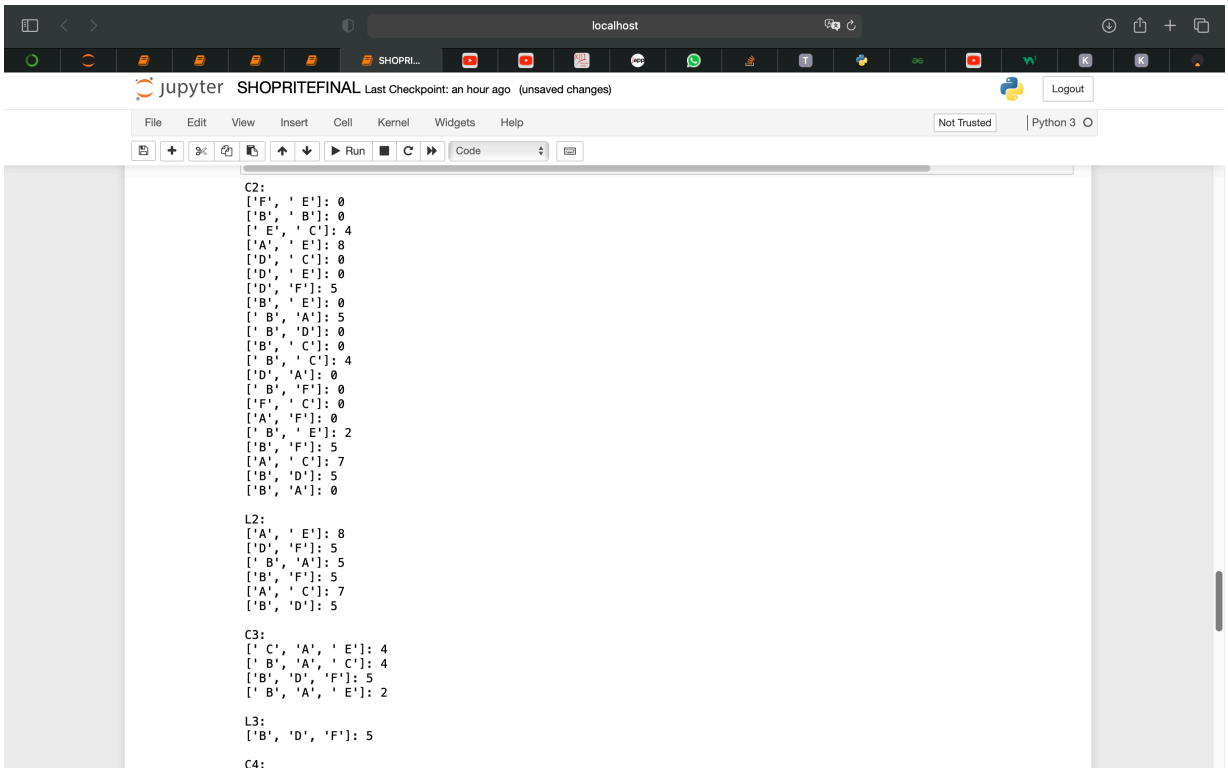
```
pl = l  #THIS FUCNTION IS FILLED WITH FOR LOOPS TO CONTINUOUS CALCULATE Cs AND Ls beyond C1 AND L1 UNTIL THERE IS ON
pos = 1
for count in range (2,1000):
    nc = set()
    temp = list(l)
    for i in range(0,len(temp)):
        for j in range(i+1,len(temp)):
            t = temp[i].union(temp[j])
            if(len(t) == count):
                nc.add(temp[i].union(temp[j]))
    nc = list(nc)
    c = Counter()
```

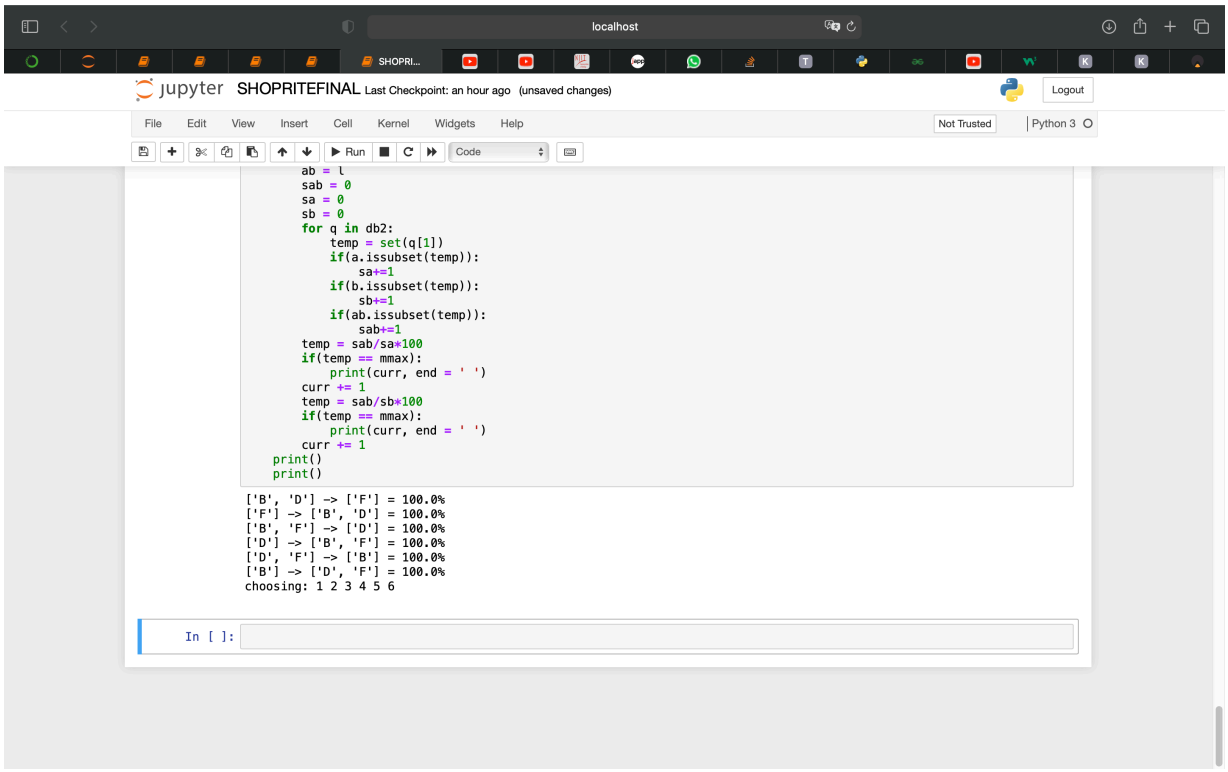

localhost
jupyter FINALNIKE Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3
Run Code
C2:
[' Modern Pants', ' Rash Guard']: 6
[' Tech Pants', 'Running Shoe']: 6
[' Modern Pants', 'Running Shoe']: 9
[' Rash Guard', ' Dry Fit V-Nick']: 9
[' Rash Guard', ' Socks']: 5
[' Sweatshirts', ' Dry Fit V-Nick']: 5
[' Modern Pants', ' Socks']: 8
[' Sweatshirts', ' Socks']: 11
[' Sweatshirts', ' Tech Pants']: 6
[' Rash Guard', ' Tech Pants']: 9
[' Dry Fit V-Nick', ' Socks']: 4
[' Tech Pants', ' Dry Fit V-Nick']: 8
[' Modern Pants', ' Dry Fit V-Nick']: 5
[' Modern Pants', ' Sweatshirts']: 10
[' Sweatshirts', 'Running Shoe']: 11
[' Tech Pants', ' Socks']: 5
[' Socks', 'Running Shoe']: 11
[' Dry Fit V-Nick', 'Running Shoe']: 5
[' Modern Pants', ' Tech Pants']: 6
[' Rash Guard', 'Running Shoe']: 7
[' Rash Guard', ' Sweatshirts']: 6
L2:
[' Modern Pants', 'Running Shoe']: 9
[' Rash Guard', ' Dry Fit V-Nick']: 9
[' Sweatshirts', ' Socks']: 11
[' Rash Guard', ' Tech Pants']: 9
[' Modern Pants', ' Sweatshirts']: 10
[' Sweatshirts', 'Running Shoe']: 11
[' Socks', 'Running Shoe']: 11
C3:
[' Sweatshirts', ' Socks', 'Running Shoe']: 10
[' Modern Pants', ' Sweatshirts', 'Running Shoe']: 9
[' Rash Guard', ' Tech Pants', ' Dry Fit V-Nick']: 8
[' Modern Pants', ' Socks', 'Running Shoe']: 8
[' Modern Pants', ' Sweatshirts', ' Socks']: 8
L3:


localhost
jupyter FINALNIKE Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3
Run Code
sab+=1
temp = sab/sa*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
temp = sab/sb*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
print()
print()
[' Sweatshirts', ' Socks'] -> ['Running Shoe'] = 90.9090909090909%
['Running Shoe'] -> [' Sweatshirts', ' Socks'] = 71.42857142857143%
[' Sweatshirts', 'Running Shoe'] -> [' Socks'] = 90.9090909090909%
[' Socks'] -> [' Sweatshirts', 'Running Shoe'] = 83.33333333333334%
[' Socks', 'Running Shoe'] -> [' Sweatshirts'] = 90.9090909090909%
[' Sweatshirts'] -> [' Socks', 'Running Shoe'] = 76.92307692307693%
choosing: 1 3 5
[' Modern Pants', ' Sweatshirts'] -> ['Running Shoe'] = 90.0%
['Running Shoe'] -> [' Modern Pants', ' Sweatshirts'] = 64.28571428571429%
[' Modern Pants', 'Running Shoe'] -> [' Sweatshirts'] = 100.0%
[' Sweatshirts'] -> [' Modern Pants', 'Running Shoe'] = 69.23076923076923%
[' Sweatshirts', 'Running Shoe'] -> [' Modern Pants'] = 81.81818181818183%
[' Modern Pants'] -> [' Sweatshirts', 'Running Shoe'] = 90.0%
choosing: 3
In [ ]:
In [ ]:

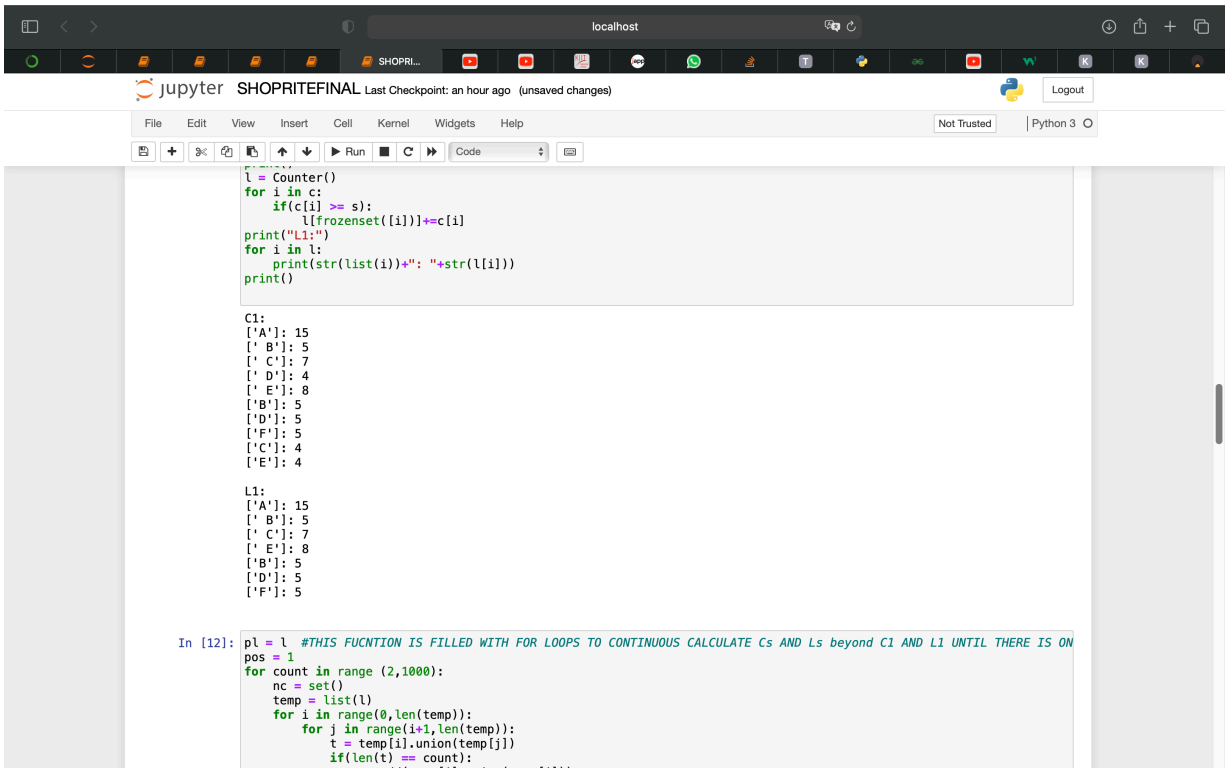
Relations for Shoprite dataset at support and confidence = 50 %

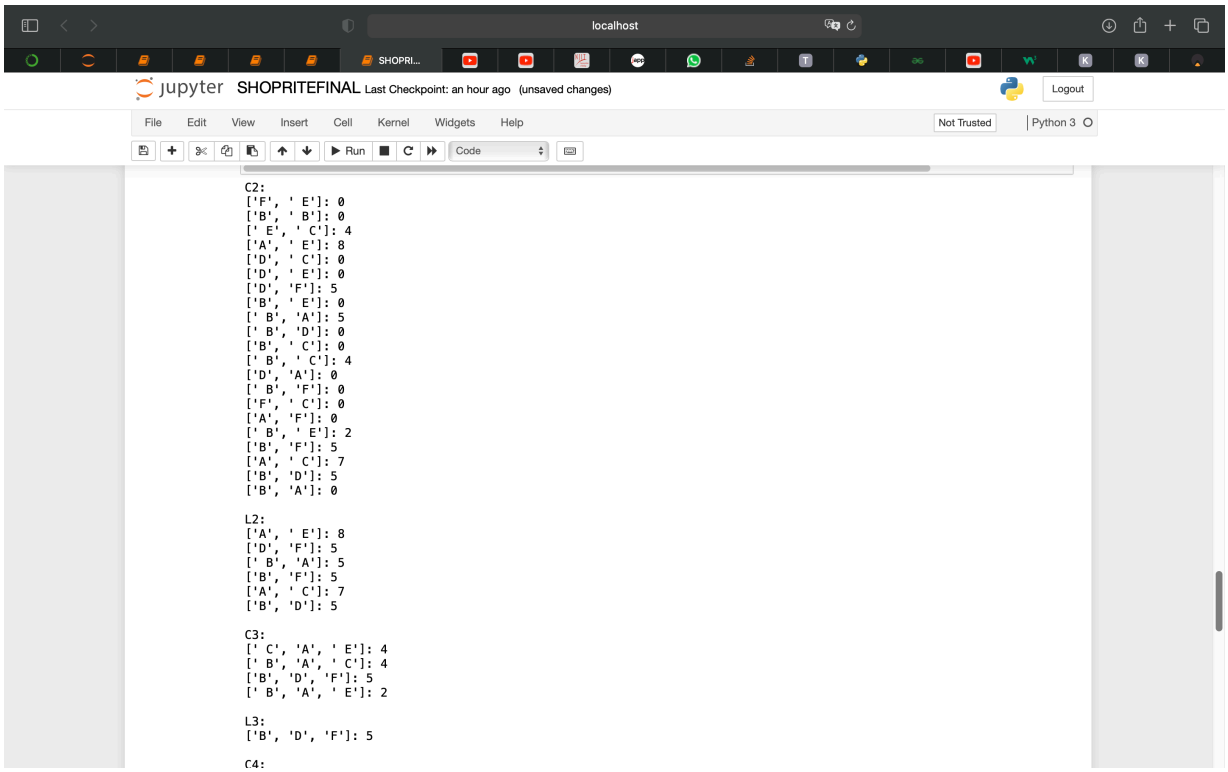


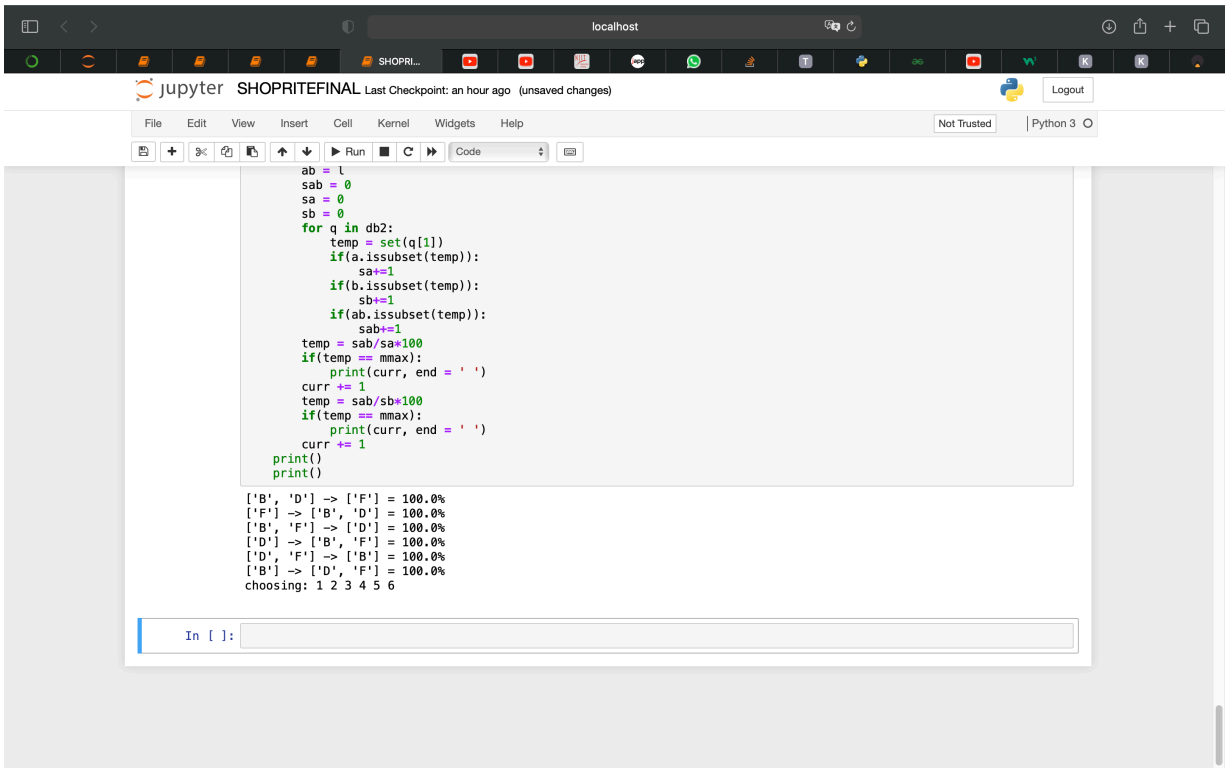

localhost
jupyter SHOPRITEFINAL Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
C2:
['F', ' E']: 0
['B', ' B']: 0
[' E', ' C']: 4
['A', ' E']: 8
['D', ' C']: 0
['D', ' E']: 0
['D', 'F']: 5
['B', ' E']: 0
[' B', 'A']: 5
[' B', 'D']: 0
['B', ' C']: 0
[' B', ' C']: 4
['D', 'A']: 0
[' B', 'F']: 0
['F', ' C']: 0
['A', 'F']: 0
[' B', ' E']: 2
['B', 'F']: 5
['A', ' C']: 7
['B', 'D']: 5
['B', 'A']: 0
L2:
['A', ' E']: 8
['D', 'F']: 5
[' B', 'A']: 5
['B', 'F']: 5
['A', ' C']: 7
['B', 'D']: 5
C3:
[' C', 'A', ' E']: 4
[' B', 'A', ' C']: 4
['B', 'D', 'F']: 5
[' B', 'A', ' E']: 2
L3:
['B', 'D', 'F']: 5
C4:


localhost
jupyter SHOPRITEFINAL Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
ab = l
sab = 0
sa = 0
sb = 0
for q in db2:
temp = set(q[1])
if(a.issubset(temp)):
sa+=1
if(b.issubset(temp)):
sb+=1
if(ab.issubset(temp)):
sab+=1
temp = sab/sa*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
temp = sab/sb*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
print()
print()
['B', 'D'] -> ['F'] = 100.0%
['F'] -> ['B', 'D'] = 100.0%
['B', 'F'] -> ['D'] = 100.0%
['D'] -> ['B', 'F'] = 100.0%
['D', 'F'] -> ['B'] = 100.0%
['B'] -> ['D', 'F'] = 100.0%
choosing: 1 2 3 4 5 6
In [ ]:

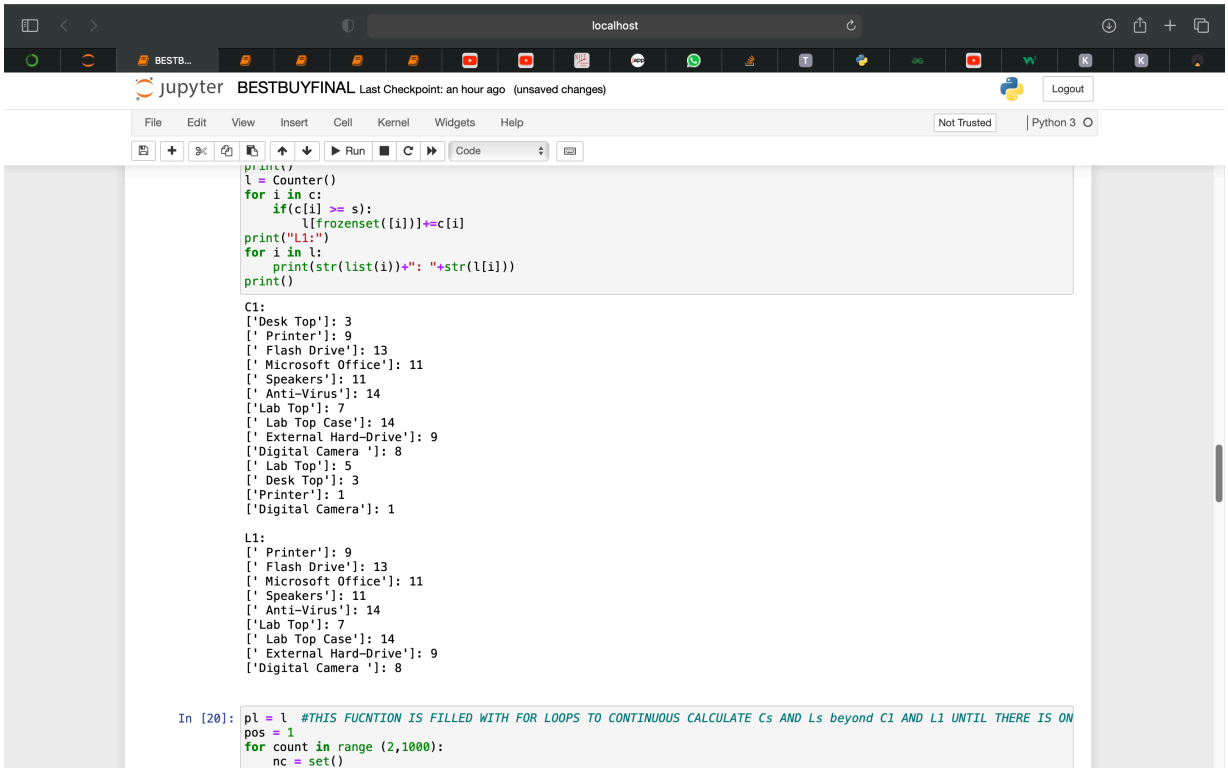
Relations for Shoprite dataset at support = 70 %

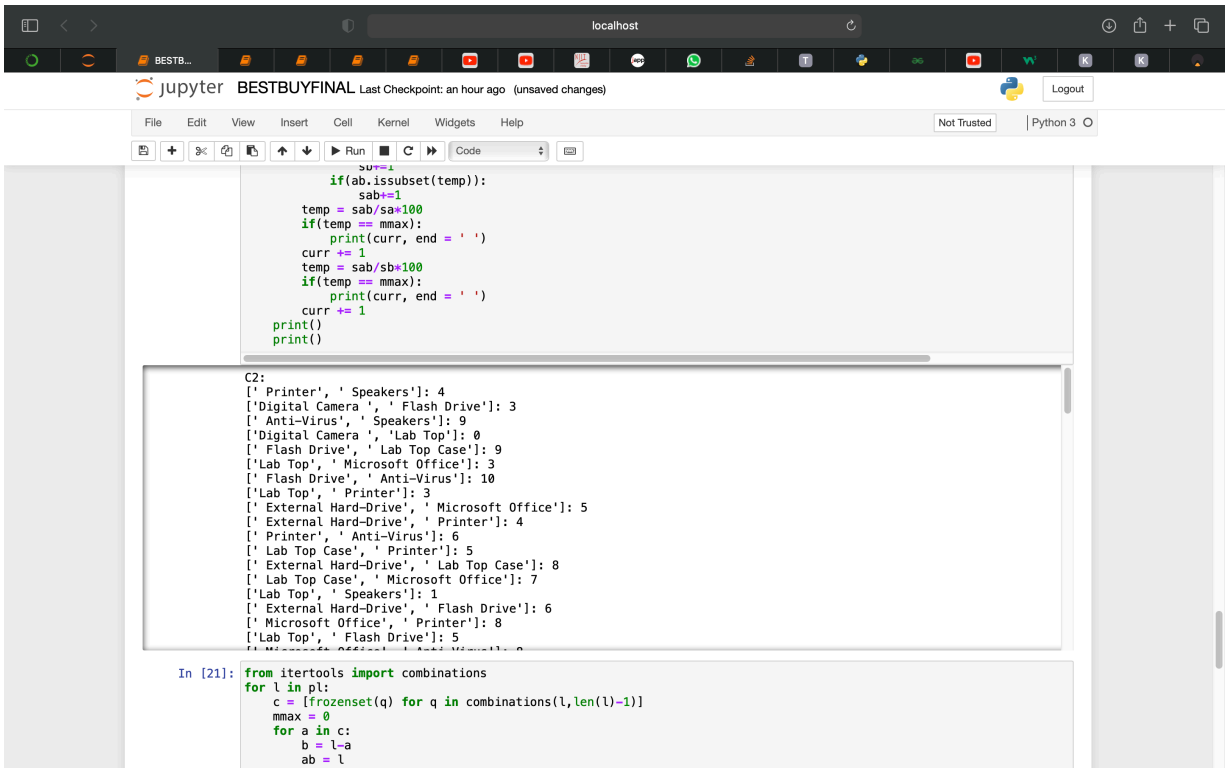


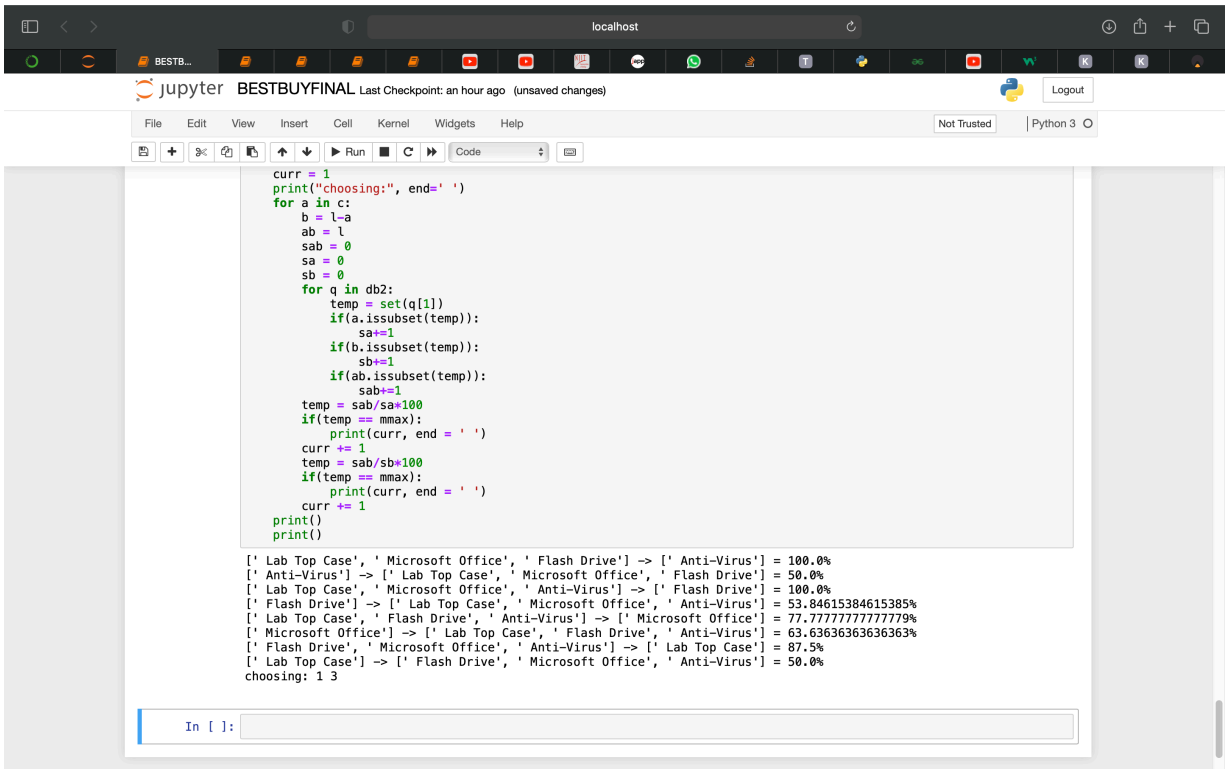

localhost
jupyter SHOPRITEFINAL Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
C2:
['F', ' E']: 0
['B', ' B']: 0
[' E', ' C']: 4
['A', ' E']: 8
['D', ' C']: 0
['D', ' E']: 0
['D', 'F']: 5
['B', ' E']: 0
[' B', 'A']: 5
[' B', 'D']: 0
['B', ' C']: 0
[' B', ' C']: 4
['D', 'A']: 0
[' B', 'F']: 0
['F', ' C']: 0
['A', 'F']: 0
[' B', ' E']: 2
['B', 'F']: 5
['A', ' C']: 7
['B', 'D']: 5
['B', 'A']: 0
L2:
['A', ' E']: 8
['D', 'F']: 5
[' B', 'A']: 5
['B', 'F']: 5
['A', ' C']: 7
['B', 'D']: 5
C3:
[' C', 'A', ' E']: 4
[' B', 'A', ' C']: 4
['B', 'D', 'F']: 5
[' B', 'A', ' E']: 2
L3:
['B', 'D', 'F']: 5
C4:


localhost
jupyter SHOPRITEFINAL Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
ab = l
sab = 0
sa = 0
sb = 0
for q in db2:
temp = set(q[1])
if(a.issubset(temp)):
sa+=1
if(b.issubset(temp)):
sb+=1
if(ab.issubset(temp)):
sab+=1
temp = sab/sa*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
temp = sab/sb*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
print()
print()
['B', 'D'] -> ['F'] = 100.0%
['F'] -> ['B', 'D'] = 100.0%
['B', 'F'] -> ['D'] = 100.0%
['D'] -> ['B', 'F'] = 100.0%
['D', 'F'] -> ['B'] = 100.0%
['B'] -> ['D', 'F'] = 100.0%
choosing: 1 2 3 4 5 6
In [ ]:

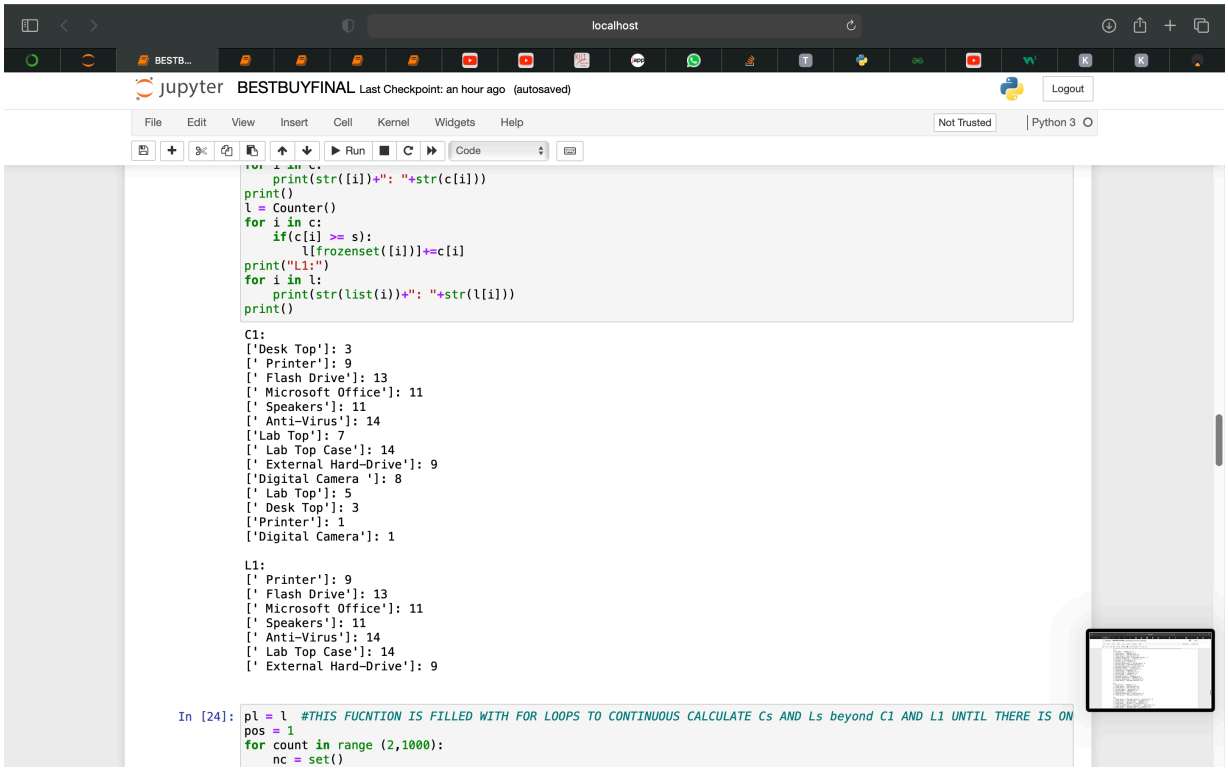
Relations for Best Buy dataset at support and confidence = 50 %

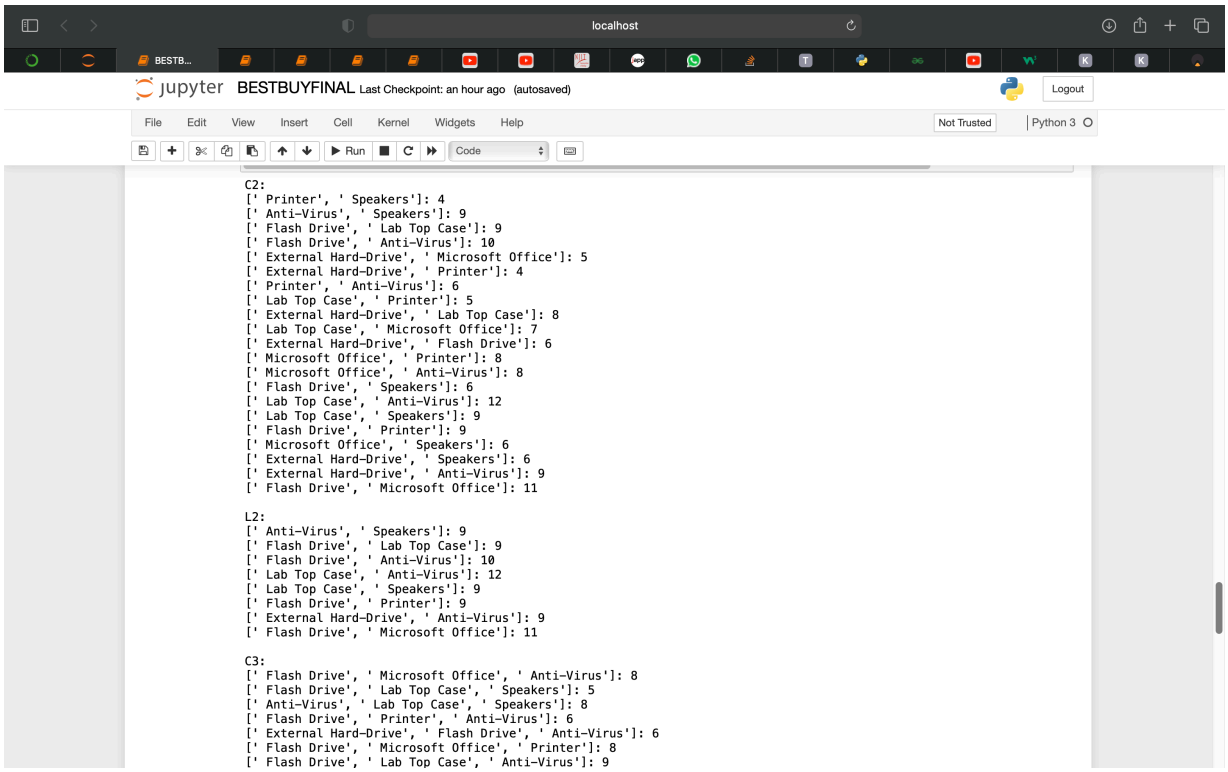



localhost
jupyter BESTBUYFINAL Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
if(ab.issubset(temp)):
sab+=1
temp = sab/sa*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
temp = sab/sb*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
print()
print()
C2:
[' Printer', ' Speakers']: 4
['Digital Camera ', ' Flash Drive']: 3
[' Anti-Virus', ' Speakers']: 9
['Digital Camera ', 'Lab Top']: 0
[' Flash Drive', ' Lab Top Case']: 9
['Lab Top', ' Microsoft Office']: 3
[' Flash Drive', ' Anti-Virus']: 10
['Lab Top', ' Printer']: 3
[' External Hard-Drive', ' Microsoft Office']: 5
[' External Hard-Drive', ' Printer']: 4
[' Printer', ' Anti-Virus']: 6
[' Lab Top Case', ' Printer']: 5
[' External Hard-Drive', ' Lab Top Case']: 8
[' Lab Top Case', ' Microsoft Office']: 7
['Lab Top', ' Speakers']: 1
[' External Hard-Drive', ' Flash Drive']: 6
[' Microsoft Office', ' Printer']: 8
['Lab Top', ' Flash Drive']: 5
In [21]:
from itertools import combinations
for l in pl:
c = [frozenset(q) for q in combinations(l,len(l)-1)]
mmax = 0
for a in c:
b = l-a
ab = l


localhost
jupyter BESTBUYFINAL Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
curr = 1
print("choosing:", end=' ')
for a in c:
b = l-a
ab = l
sab = 0
sa = 0
sb = 0
for q in db2:
temp = set(q[1])
if(a.issubset(temp)):
sa+=1
if(b.issubset(temp)):
sb+=1
if(ab.issubset(temp)):
sab+=1
temp = sab/sa*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
temp = sab/sb*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
print()
print()
[' Lab Top Case', ' Microsoft Office', ' Flash Drive'] -> [' Anti-Virus'] = 100.0%
[' Anti-Virus'] -> [' Lab Top Case', ' Microsoft Office', ' Flash Drive'] = 50.0%
[' Lab Top Case', ' Microsoft Office', ' Anti-Virus'] -> [' Flash Drive'] = 100.0%
[' Flash Drive'] -> [' Lab Top Case', ' Microsoft Office', ' Anti-Virus'] = 53.84615384615385%
[' Lab Top Case', ' Flash Drive', ' Anti-Virus'] -> [' Microsoft Office'] = 77.77777777777779%
[' Microsoft Office'] -> [' Lab Top Case', ' Flash Drive', ' Anti-Virus'] = 63.63636363636363%
[' Flash Drive', ' Microsoft Office', ' Anti-Virus'] -> [' Lab Top Case'] = 87.5%
[' Lab Top Case'] -> [' Flash Drive', ' Microsoft Office', ' Anti-Virus'] = 50.0%
choosing: 1 3
In [ ]:

Relations for Best Buy dataset at support = 70 %

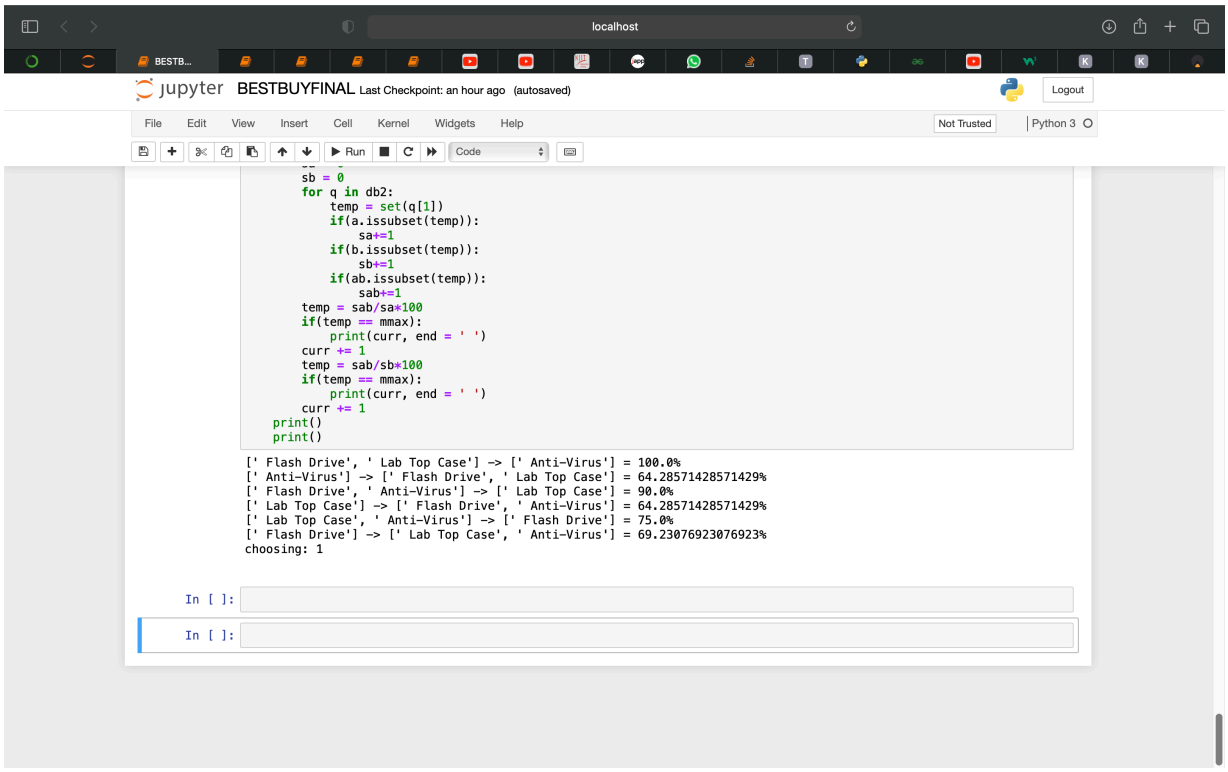





```
C2:
[' Printer', ' Speakers']: 4
[' Anti-Virus', ' Speakers']: 9
[' Flash Drive', ' Lab Top Case']: 9
[' Flash Drive', ' Anti-Virus']: 10
[' External Hard-Drive', ' Microsoft Office']: 5
[' External Hard-Drive', ' Printer']: 4
[' Printer', ' Anti-Virus']: 6
[' Lab Top Case', ' Printer']: 5
[' External Hard-Drive', ' Lab Top Case']: 8
[' Lab Top Case', ' Microsoft Office']: 7
[' External Hard-Drive', ' Flash Drive']: 6
[' Microsoft Office', ' Printer']: 8
[' Microsoft Office', ' Anti-Virus']: 8
[' Flash Drive', ' Speakers']: 6
[' Lab Top Case', ' Anti-Virus']: 12
[' Lab Top Case', ' Speakers']: 9
[' Flash Drive', ' Printer']: 9
[' Microsoft Office', ' Speakers']: 6
[' External Hard-Drive', ' Speakers']: 6
[' External Hard-Drive', ' Anti-Virus']: 9
[' Flash Drive', ' Microsoft Office']: 11

L2:
[' Anti-Virus', ' Speakers']: 9
[' Flash Drive', ' Lab Top Case']: 9
[' Flash Drive', ' Anti-Virus']: 10
[' Lab Top Case', ' Anti-Virus']: 12
[' Lab Top Case', ' Speakers']: 9
[' Flash Drive', ' Printer']: 9
[' External Hard-Drive', ' Anti-Virus']: 9
[' Flash Drive', ' Microsoft Office']: 11

C3:
[' Flash Drive', ' Microsoft Office', ' Anti-Virus']: 8
[' Flash Drive', ' Lab Top Case', ' Speakers']: 5
[' Anti-Virus', ' Lab Top Case', ' Speakers']: 8
[' Flash Drive', ' Printer', ' Anti-Virus']: 6
[' External Hard-Drive', ' Flash Drive', ' Anti-Virus']: 6
[' Flash Drive', ' Microsoft Office', ' Printer']: 8
[' Flash Drive', ' Lab Top Case', ' Anti-Virus']: 9
```



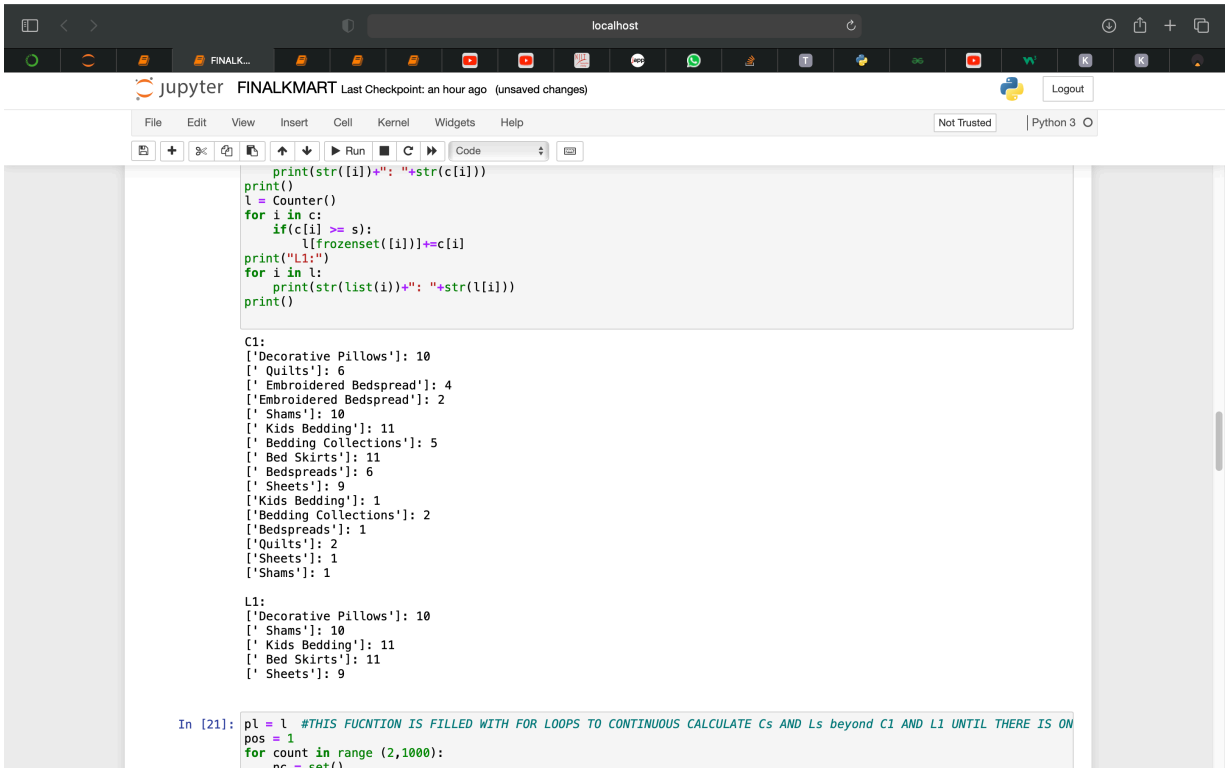

```
        sb = 0
        for q in db2:
            temp = set(q[1])
            if(a.issubset(temp)):
                sa+=1
            if(b.issubset(temp)):
                sb+=1
            if(ab.issubset(temp)):
                sab+=1
        temp = sab/sa*100
        if(temp == mmax):
            print(curr, end = ' ')
        curr += 1
        temp = sab/sb*100
        if(temp == mmax):
            print(curr, end = ' ')
        curr += 1
    print()
    print()
```

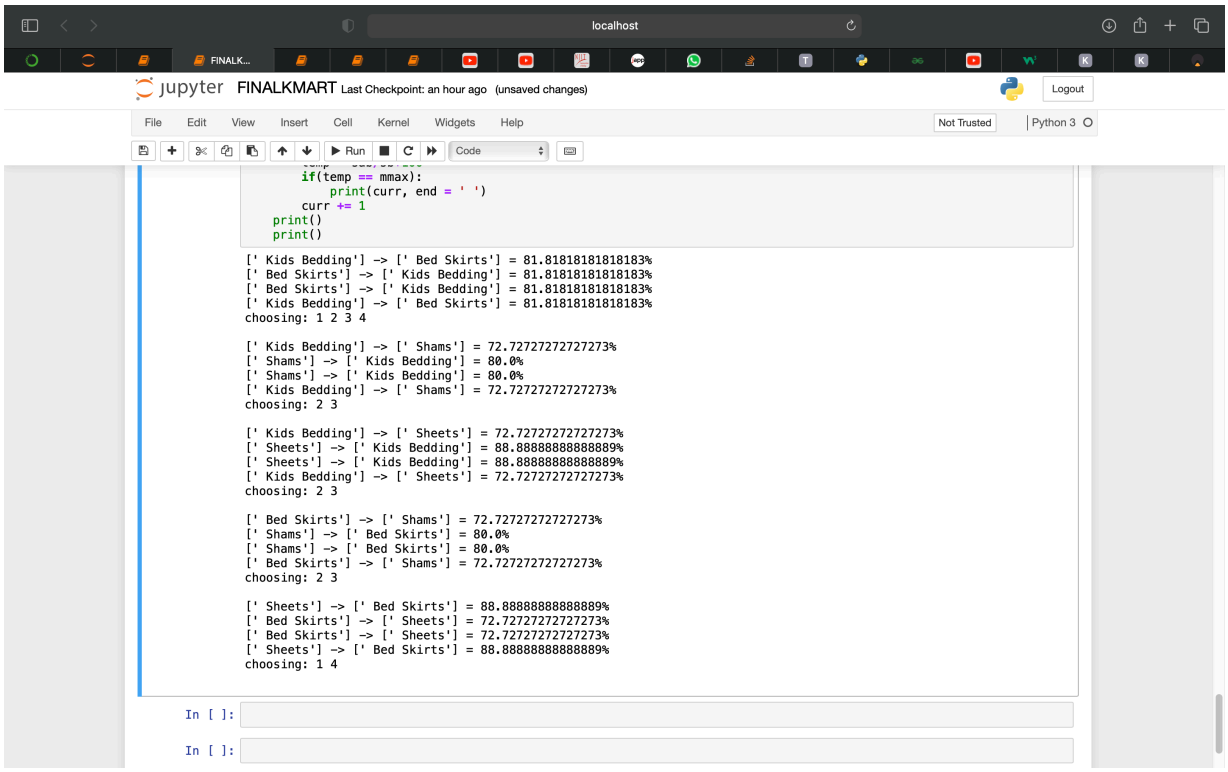
```
[' Flash Drive', ' Lab Top Case'] -> [' Anti-Virus'] = 100.0%
[' Anti-Virus'] -> [' Flash Drive', ' Lab Top Case'] = 64.28571428571429%
[' Flash Drive', ' Anti-Virus'] -> [' Lab Top Case'] = 90.0%
[' Lab Top Case'] -> [' Flash Drive', ' Anti-Virus'] = 64.28571428571429%
[' Lab Top Case', ' Anti-Virus'] -> [' Flash Drive'] = 75.0%
[' Flash Drive'] -> [' Lab Top Case', ' Anti-Virus'] = 69.23076923076923%
choosing: 1
```

In [ ]:

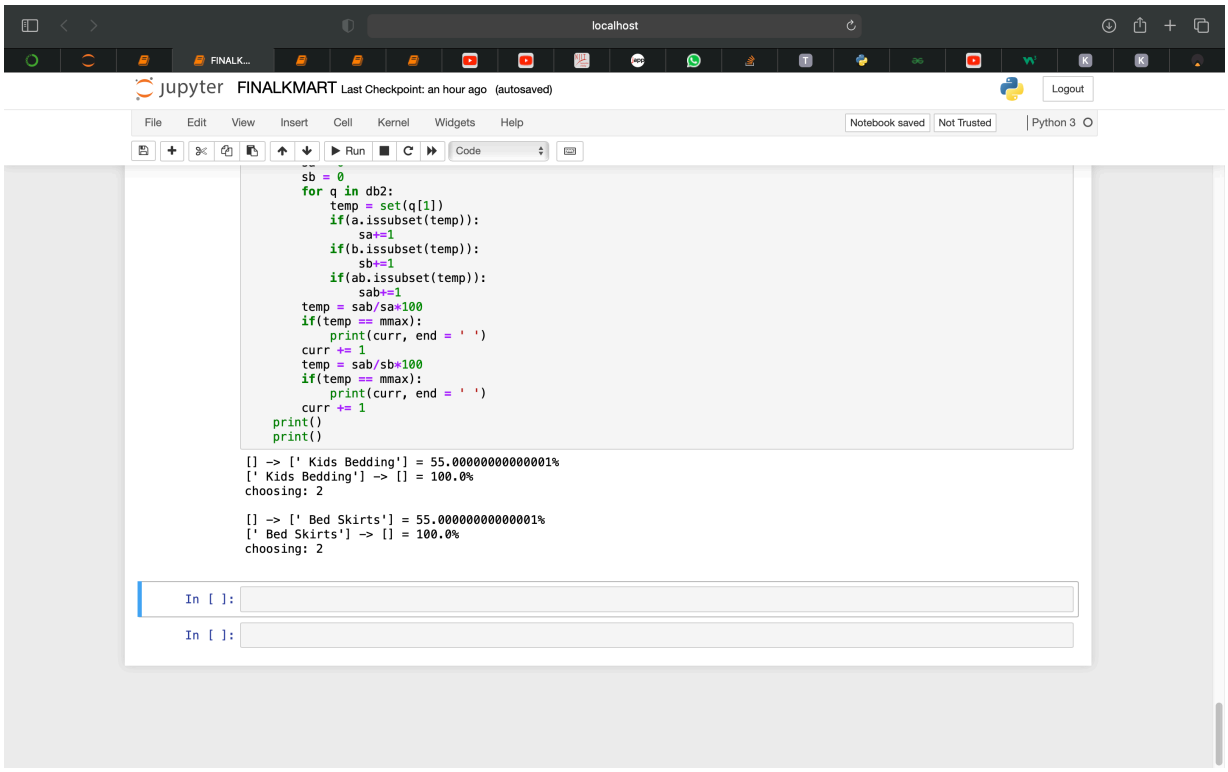
In [ ]:

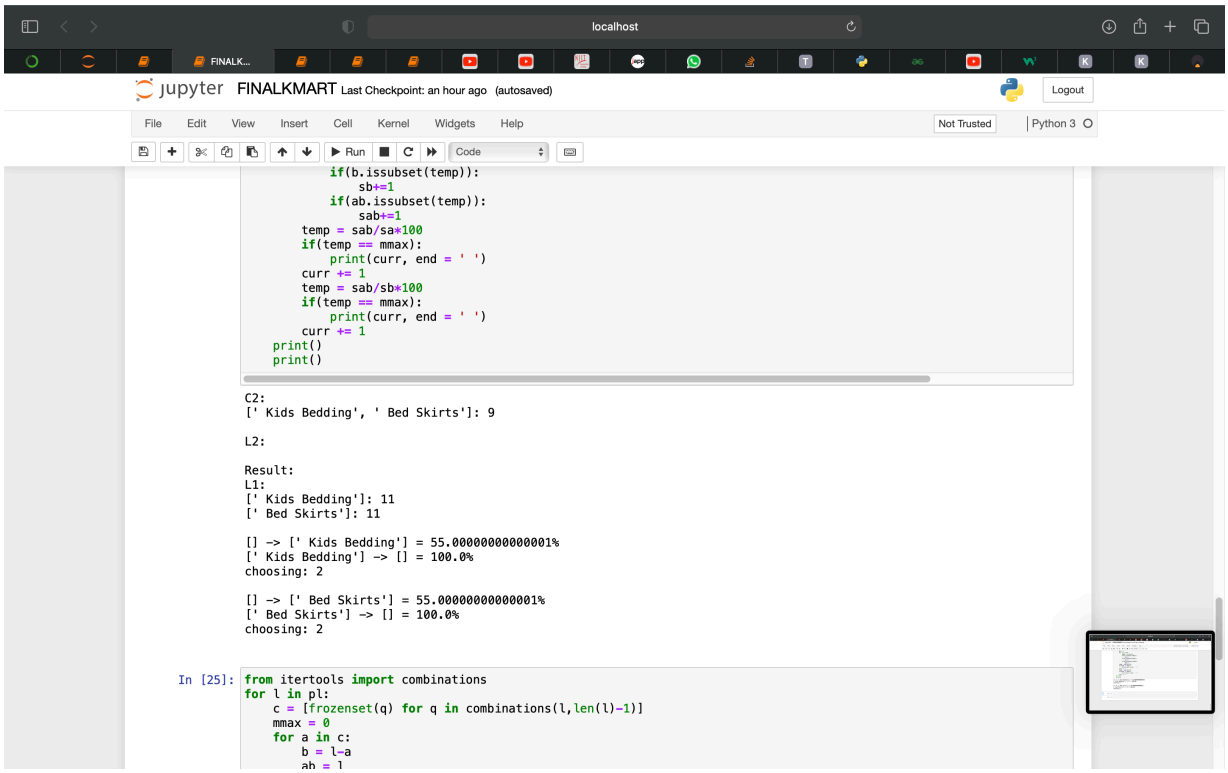
Relations for K Mart dataset at support and confidence = 50 %

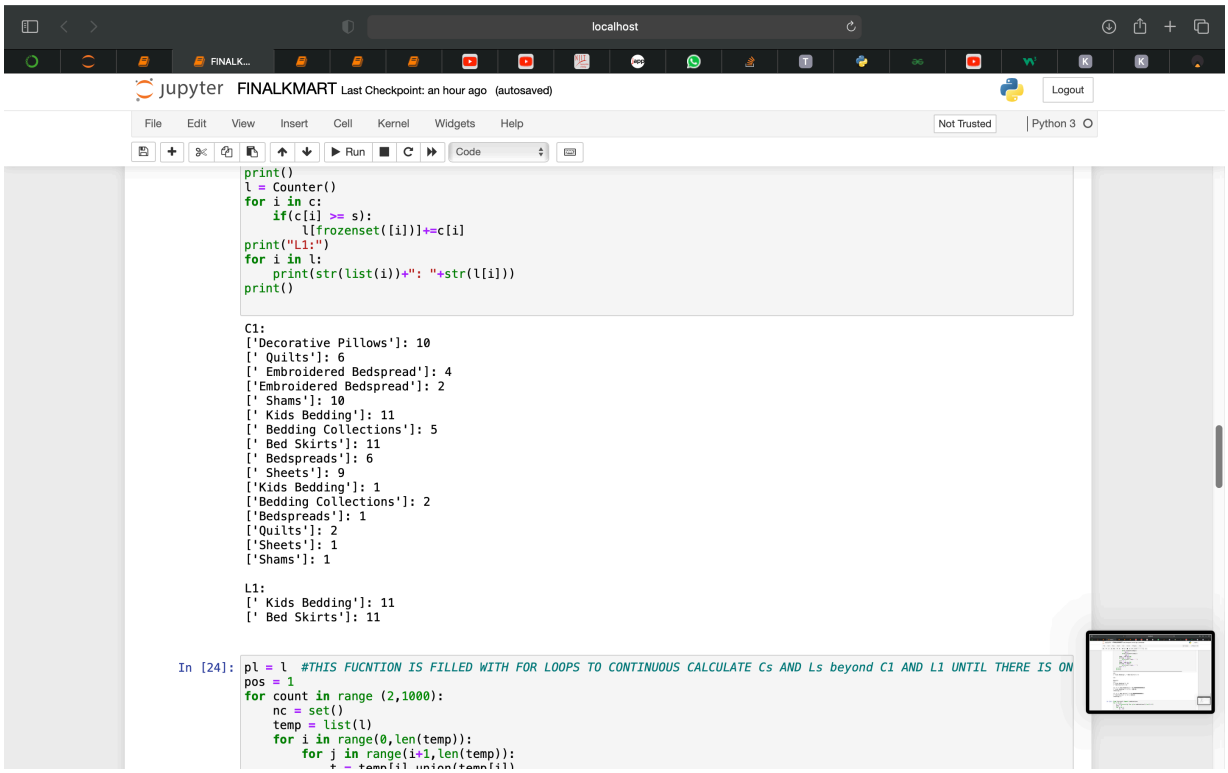




Relations for K Mart dataset at support = 70 %


localhost
FINALK...
jupyter FINALKMART Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Notebook saved
Not Trusted
Python 3
Run
Code
sb = 0
for q in db2:
temp = set(q[1])
if(a.issubset(temp)):
sa+=1
if(b.issubset(temp)):
sb+=1
if(ab.issubset(temp)):
sab+=1
temp = sab/sa*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
temp = sab/sb*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
print()
print()
[] -> [' Kids Bedding'] = 55.00000000000001%
[' Kids Bedding'] -> [] = 100.0%
choosing: 2
[] -> [' Bed Skirts'] = 55.00000000000001%
[' Bed Skirts'] -> [] = 100.0%
choosing: 2
In [ ]:
In [ ]:


localhost
FINALK...
jupyter FINALKMART Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
if(b.issubset(temp)):
sb+=1
if(ab.issubset(temp)):
sab+=1
temp = sab/sa*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
temp = sab/sb*100
if(temp == mmax):
print(curr, end = ' ')
curr += 1
print()
print()
C2:
[' Kids Bedding', ' Bed Skirts']: 9
L2:
Result:
L1:
[' Kids Bedding']: 11
[' Bed Skirts']: 11
[] -> [' Kids Bedding'] = 55.00000000000001%
[' Kids Bedding'] -> [] = 100.0%
choosing: 2
[] -> [' Bed Skirts'] = 55.00000000000001%
[' Bed Skirts'] -> [] = 100.0%
choosing: 2
In [25]:
from itertools import combinations
for l in pl:
c = [frozenset(q) for q in combinations(l,len(l)-1)]
mmax = 0
for a in c:
b = l-a
ab = l


localhost
FINALK...
jupyter FINALKMART Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
print()
l = Counter()
for i in c:
if(c[i] >= s):
l[frozenset([i])]+=c[i]
print("L1:")
for i in l:
print(str(list(i))+": "+str(l[i]))
print()
C1:
['Decorative Pillows']: 10
[' Quilts']: 6
[' Embroidered Bedspread']: 4
['Embroidered Bedspread']: 2
[' Shams']: 10
[' Kids Bedding']: 11
[' Bedding Collections']: 5
[' Bed Skirts']: 11
[' Bedspreads']: 6
[' Sheets']: 9
['Kids Bedding']: 1
['Bedding Collections']: 2
['Bedspreads']: 1
['Quilts']: 2
['Sheets']: 1
['Shams']: 1
L1:
[' Kids Bedding']: 11
[' Bed Skirts']: 11
In [24]:
pl = l #THIS FUCNTION IS FILLED WITH FOR LOOPS TO CONTINUOUS CALCULATE Cs AND Ls beyond C1 AND L1 UNTIL THERE IS ON
pos = 1
for count in range (2,1000):
nc = set()
temp = list(l)
for i in range(0,len(temp)):
for j in range(i+1,len(temp)):