

Apriori Algorithm Documentation

Developed: Dhara Rana



13 October 2018

CS 636: DATA MININING

Table of Contents

[Functions used to create Apriori Algorithm 2](#_Toc527206489)

[findUniqueItems(ItemTransactionDic) 2](#_Toc527206490)

[one\_hot\_encode(data,uniqueItems) 2](#_Toc527206491)

[frequentItemset(support,min\_support,comboList) 2](#_Toc527206492)

[findSupport(combinationList, data) 2](#_Toc527206493)

[combos(One\_Itemset, c) 3](#_Toc527206494)

[getItemsetL1(ItemTransactionDic,min\_support) 3](#_Toc527206495)

[scanNprun (ItemTransactionDic,min\_support) 3](#_Toc527206496)

[findAssociationRules(ItemTransactionDic,min\_support,min\_confid) 3](#_Toc527206497)

[How GUI Works? 3](#_Toc527206498)

[Transaction List: 9](#_Toc527206499)

[Transaction List 1: Grocery Items 9](#_Toc527206500)

[Input file: 9](#_Toc527206501)

[Output File: 10](#_Toc527206502)

[Transaction List 2: Grocery Items 11](#_Toc527206503)

[Input File: 11](#_Toc527206504)

[Output File: 11](#_Toc527206505)

[Transaction List 3: Grocery Items 12](#_Toc527206506)

[Input File: 12](#_Toc527206507)

[Output File: 13](#_Toc527206508)

[Transaction List 4: Grocery Items 14](#_Toc527206509)

[Input File: 14](#_Toc527206510)

[Output File: 14](#_Toc527206511)

[Transaction List 5: Grocery Items 15](#_Toc527206512)

[Input File: 15](#_Toc527206513)

[Output File: 16](#_Toc527206514)

[Code: 17](#_Toc527206515)

[AprioriAlgorithmV5.py : Apriori Algorithm Implementation 17](#_Toc527206516)

[aaGUI.py: Graphical User Interface 23](#_Toc527206517)

# Functions used to create Apriori Algorithm

Function can be found in the script file AprioriAlgorithmV5.py

## findUniqueItems(ItemTransactionDic)

* 1. **Input**: Pass a dictionary with transaction items in it

**Example**: ItemTransactionDic =

{'1001': ['ink', 'pen', 'cheese', 'bag'],

'1002': ['milk', 'pen', 'juice', 'cheese'],

'1003': ['milk', 'juice'],

'1004': ['juice', 'milk', 'cheese']}

* 1. **Output**: All the unique items in the transaction list; returns a list with strings

**Example**:

['ink', 'pen', 'cheese', 'bag', 'milk', 'juice']

## one\_hot\_encode(data,uniqueItems)

* 1. **Input:** (1) Pass a dictionary with transaction items in it (2) uniqueItems list generated from the function 1 above.
  2. **Output:** one hot encoded transaction list

**Example:**

[[1, 1, 1, 1, 0, 0],

[0, 1, 1, 0, 1, 1],

[0, 0, 0, 0, 1, 1],

[0, 0, 1, 0, 1, 1]]

## frequentItemset(support,min\_support,comboList)

* 1. **Input:** (1) support is a list of support value for each single itemset form unique list

e.g ['ink', 'pen', 'cheese', 'bag', 'milk', 'juice']

(2) min\_support: single value set by user; value is between 0-1

e.g: 0.5

(3) comboList: list of uniqueItems generated from function 1 above

* 1. **Output:** (1) One\_Itemset, which is single frequently appearing the item in transaction meeting user define support

Eg. ['pen', 'cheese', 'milk', 'juice']

(2) Item not meeting user defined support: ['ink', 'bag']

(3) Support\_value of the single frequently appearing items that met the user defined support e,g [0.5, 0.75, 0.75, 0.75]

## findSupport(combinationList, data)

Finds support for all the items

* 1. **Input:** (1) CombinationList= type list

(2) data dictionary

* 1. **Output:** Return a list of supports

## combos(One\_Itemset, c)

* 1. **Input:** (1) One\_Itemset: is a list of singe frequent items
  2. # (2) c is the number of items per set; 2= {pen,bag}, 3= {pen,cookie,candy}, etc
  3. #Output: returns all the combo as a list
  4. **Output:**  the list of combinations

## getItemsetL1(ItemTransactionDic,min\_support)

Creating the first frequent Itemset or L1

**Input**: ItemTransactionDic, min\_support,

**Output**: One\_Itemset,ItemRejList,Support\_value

uses the defined functions: findUniqueItems, one\_hot\_encode and frequentItemset

## scanNprun (ItemTransactionDic,min\_support)

Rest of the combinations

=> **Inputs**: One\_Itemset,ItemRejList,Support\_value

=> **Output**: Frequent\_Itemlist, FreqItemsetSup

## findAssociationRules(ItemTransactionDic,min\_support,min\_confid)

Find confidence and assocoation rule

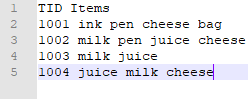
**Input**: Frequent\_Itemlist,FreqItemsetSup

**output**: Cleaned AssociationRules

# How GUI Works?

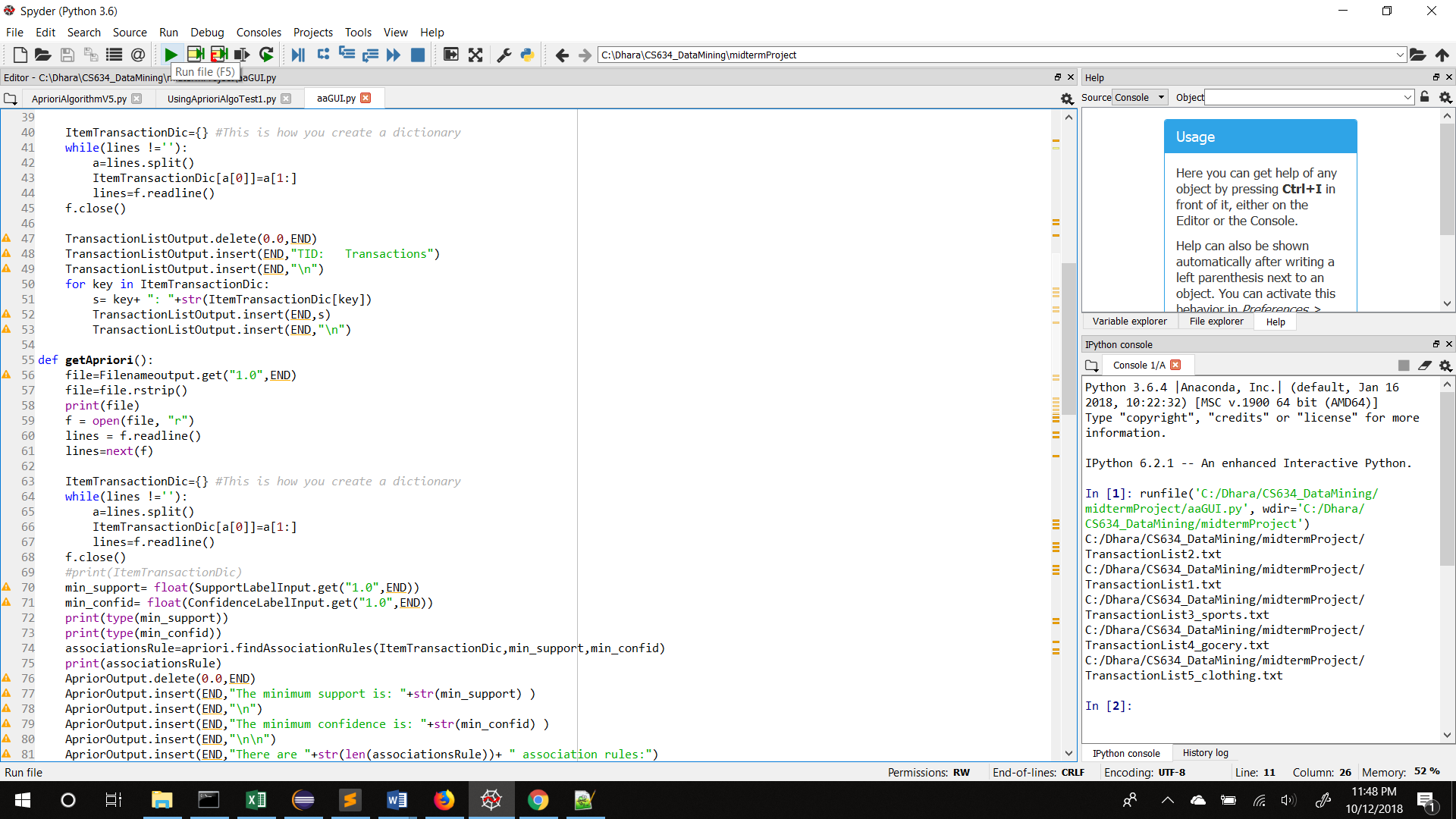
To find the association rule, run the script file aaGUI.py. This will open a graphical user interface allowing you to input a .txt file, the minimum support value, and minimum confidence value.

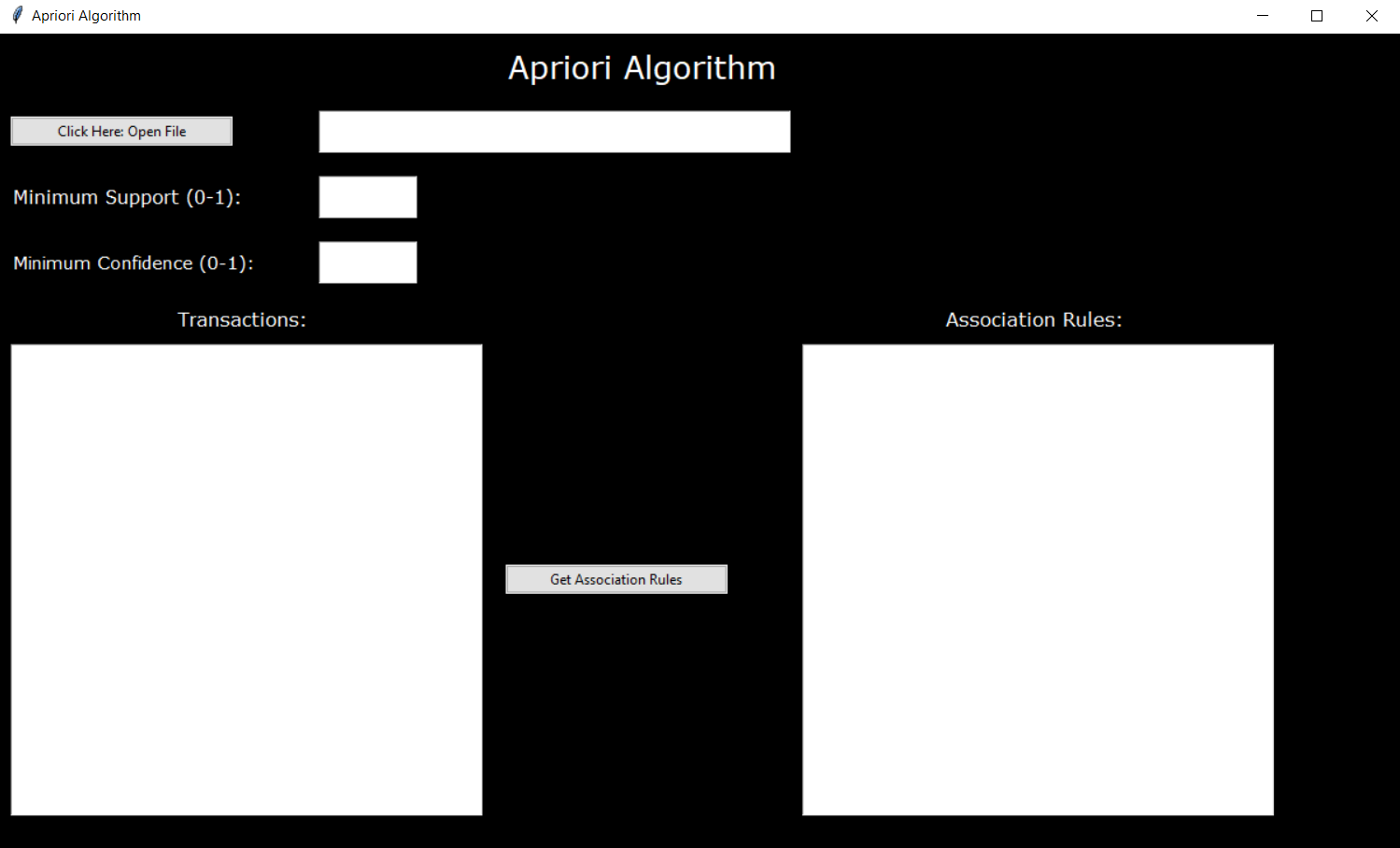
1. The text file must be in the following format:

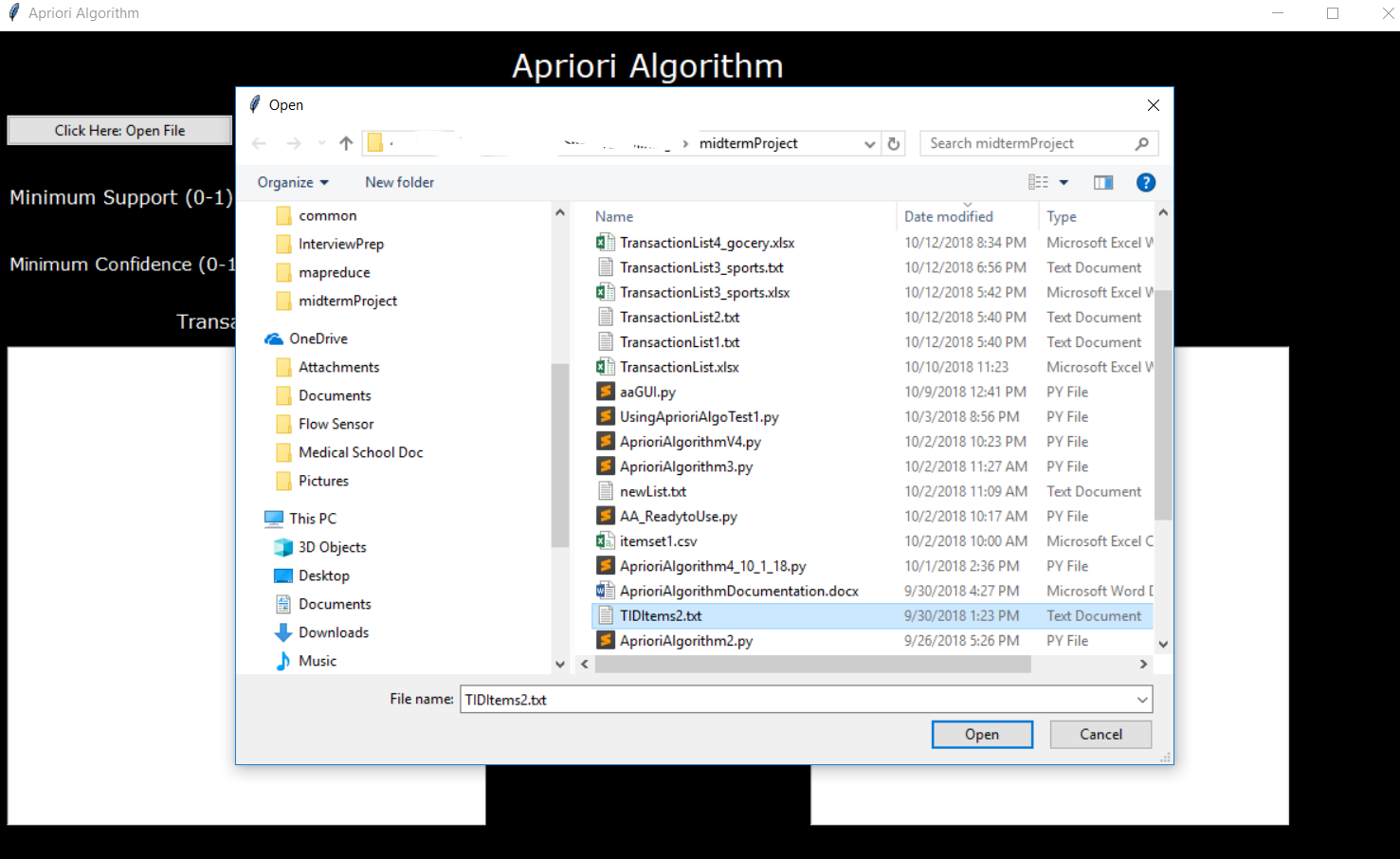
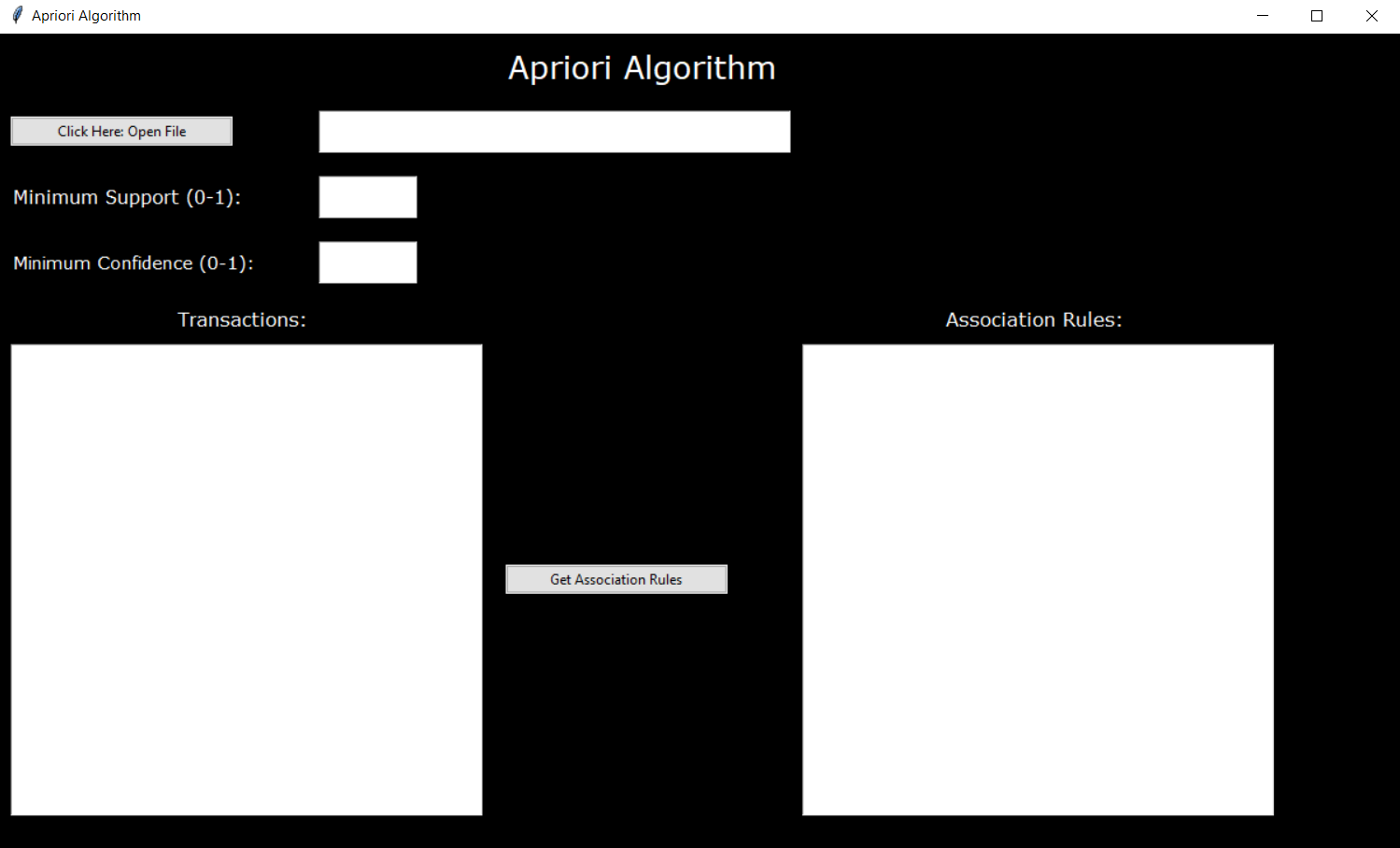


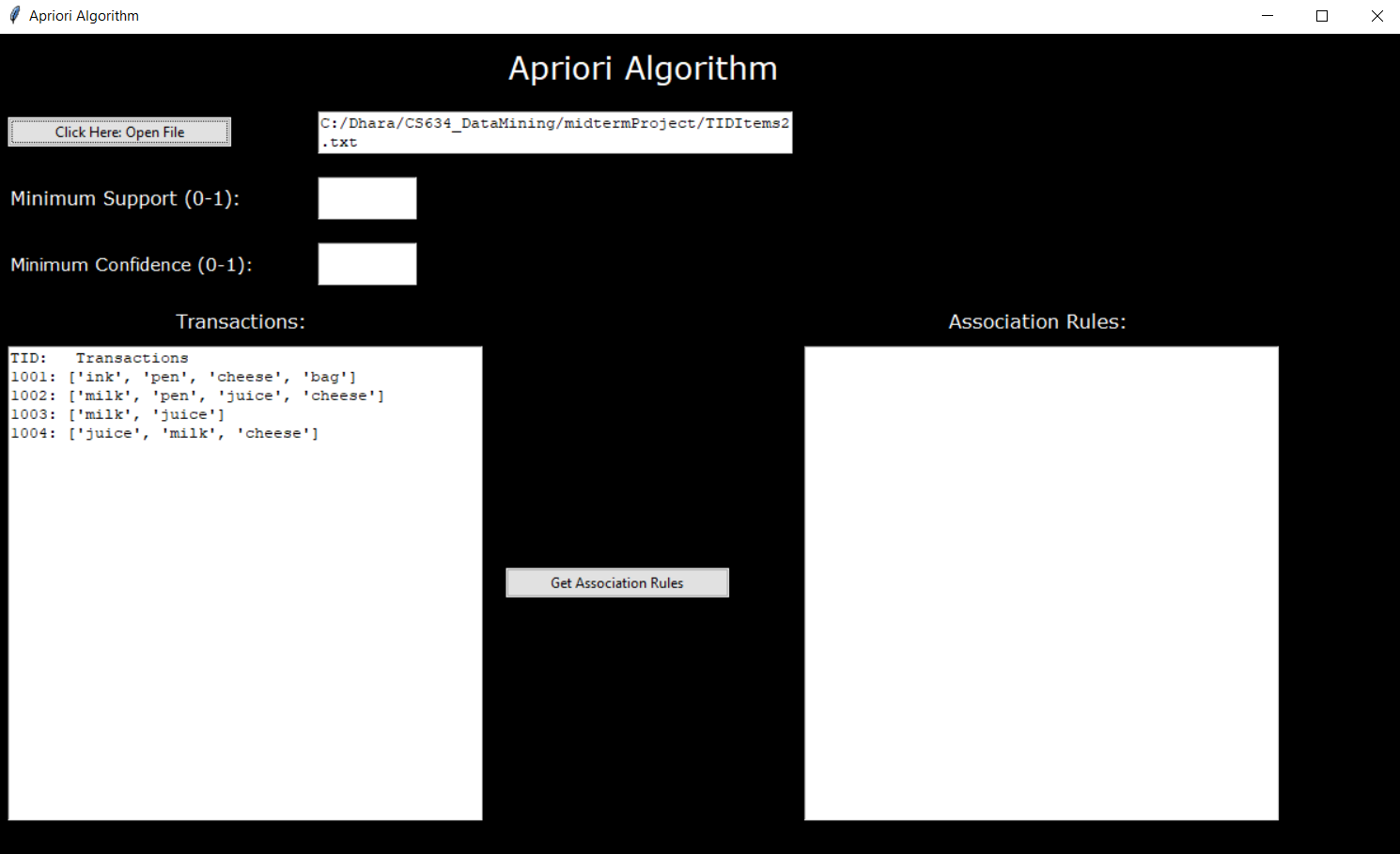
Please note that each items is separated by spaces. So if you have an item called “water bottle”, you must change it to “water\_bottle”; Otherwise, the program will we it as two separate words.

1. Open the file aaGUI.py and press the run button on your preferred Python IDE. Image below is using Spyder IDE for python to run the program. Note the red square is how you run the program.

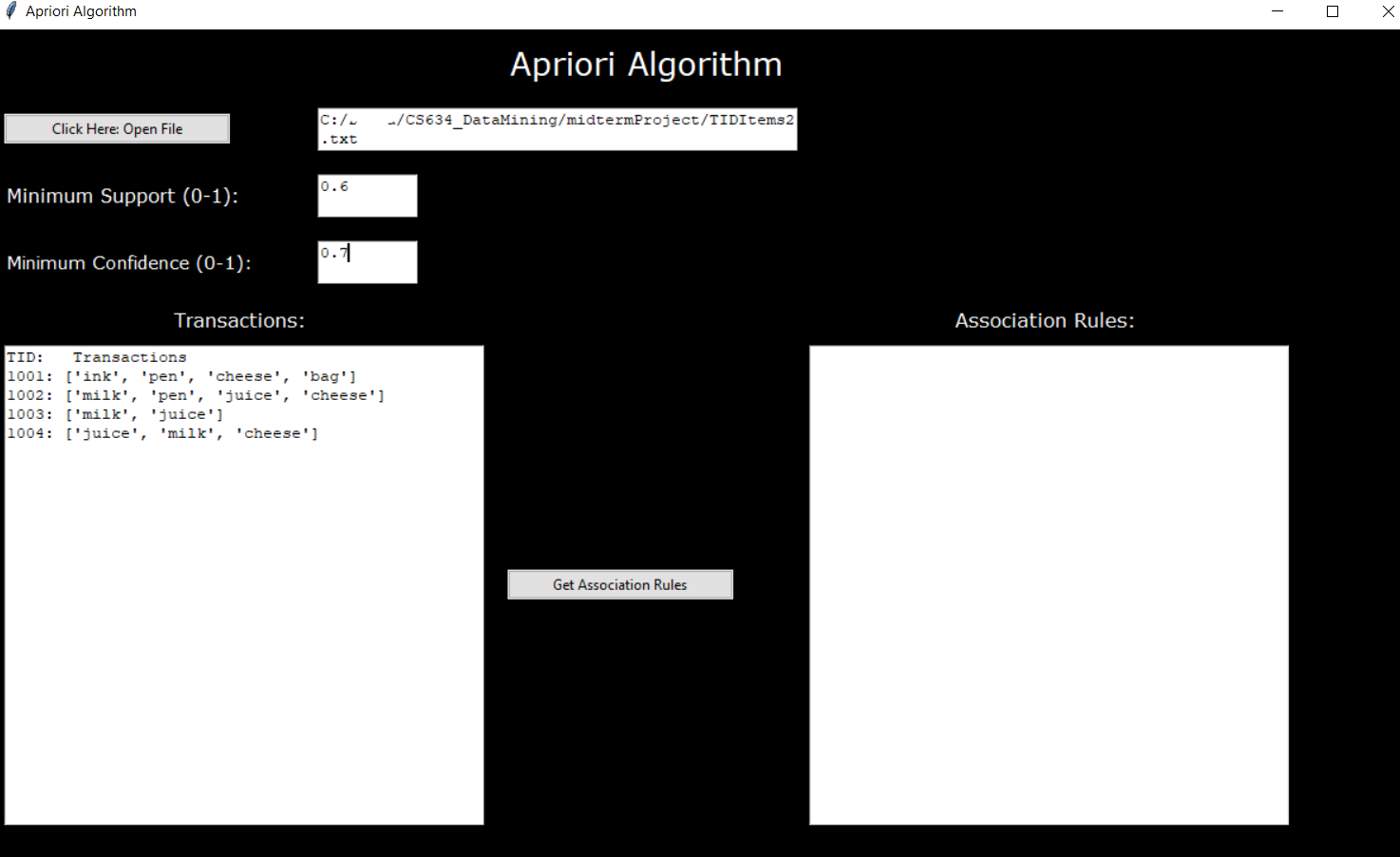


1. After running program, the following window will open.
2. Click on the button titled “Click Here Open File” to load your .txt as followed in step 1. See in red square. After loading the file, the list of transaction will be list in the text under the title “Transaction”.

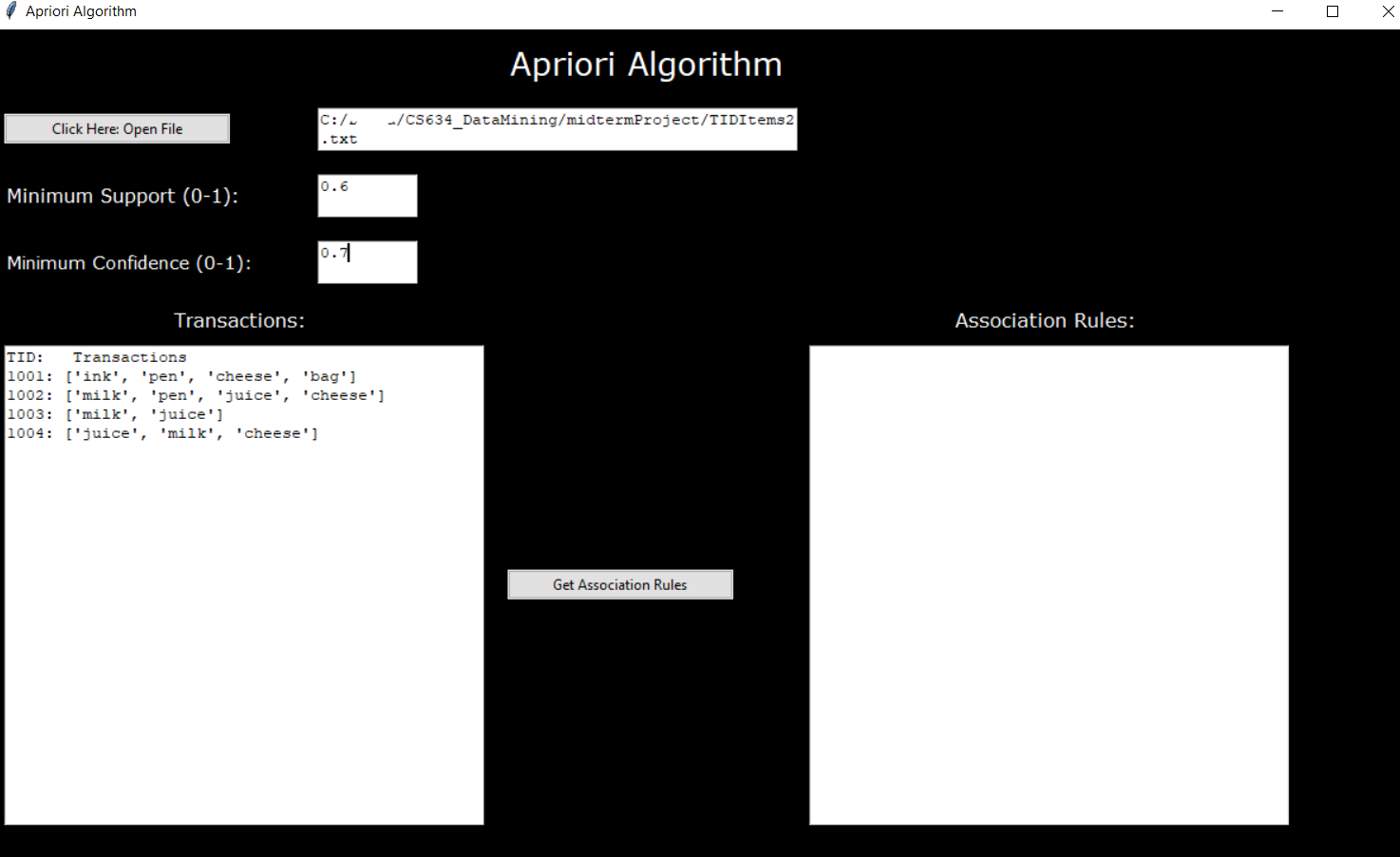


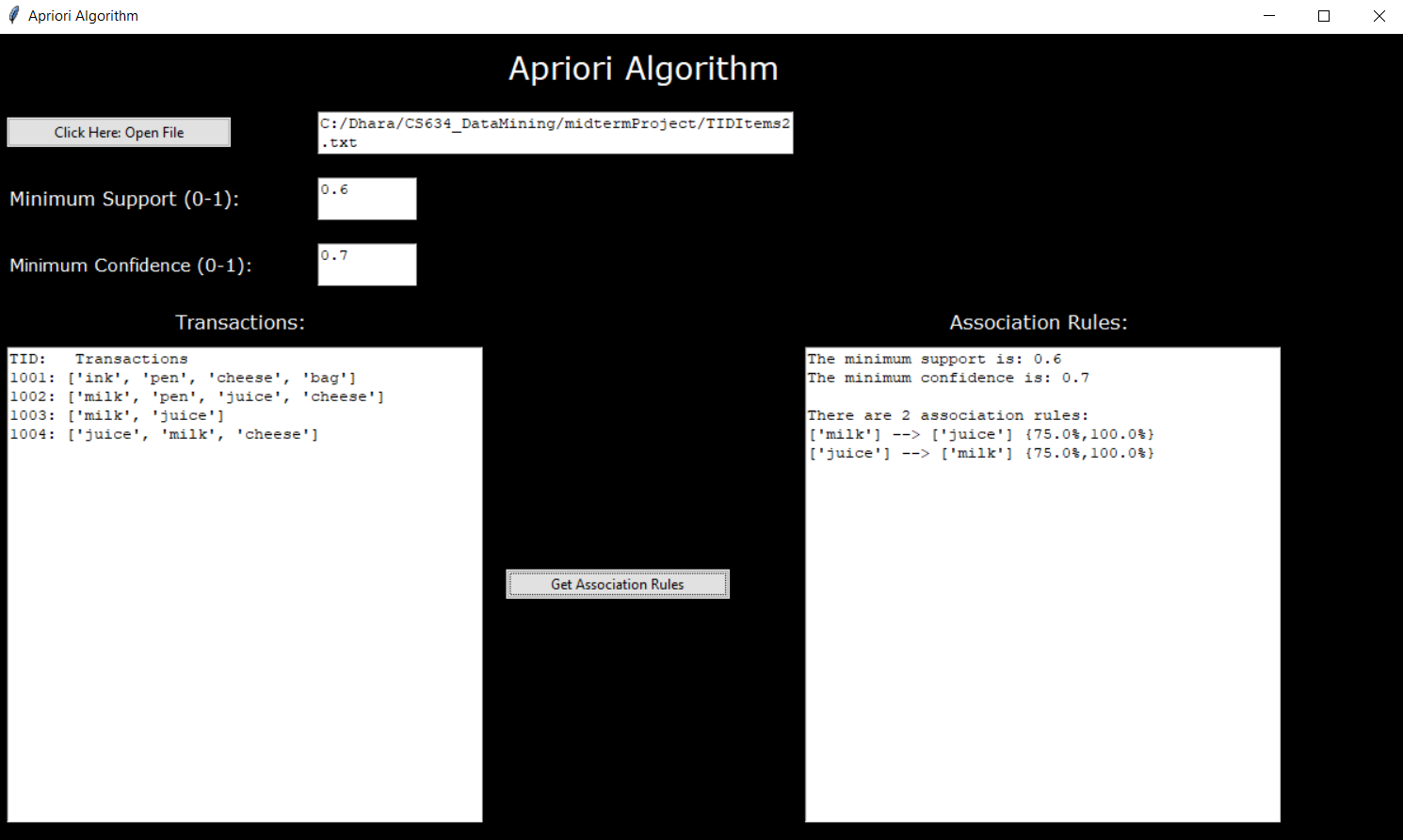


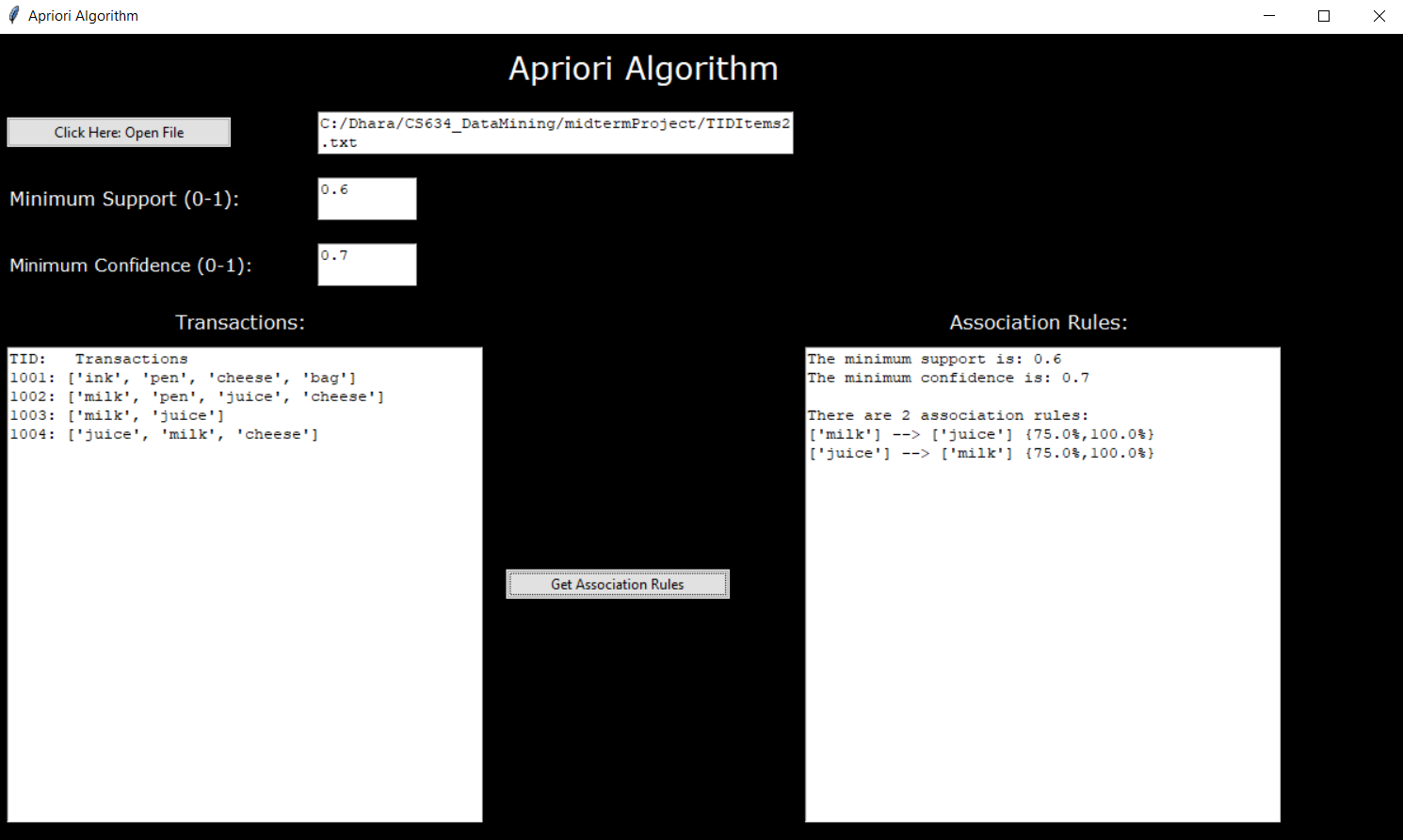
1. Input the minimum support and minimum confidence between 0-1



1. To get Association rules, click the button titled “Get Association Rules”. This will generate the rules in the textbox under “Association Rules”. See red Square





1. Click the window close button to exit out of the GUI. See Red square.

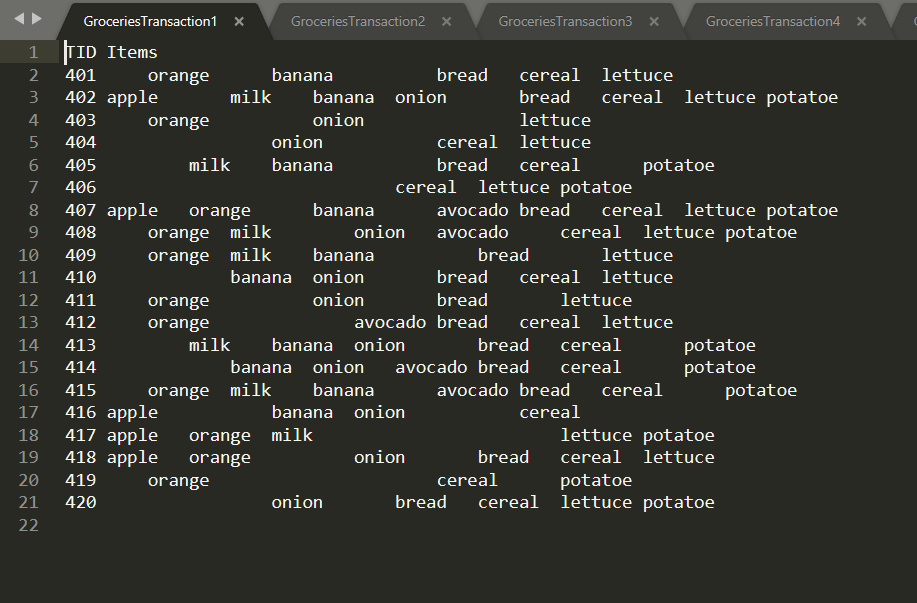
# Transaction List:

Each transaction list contains the following 10 items:

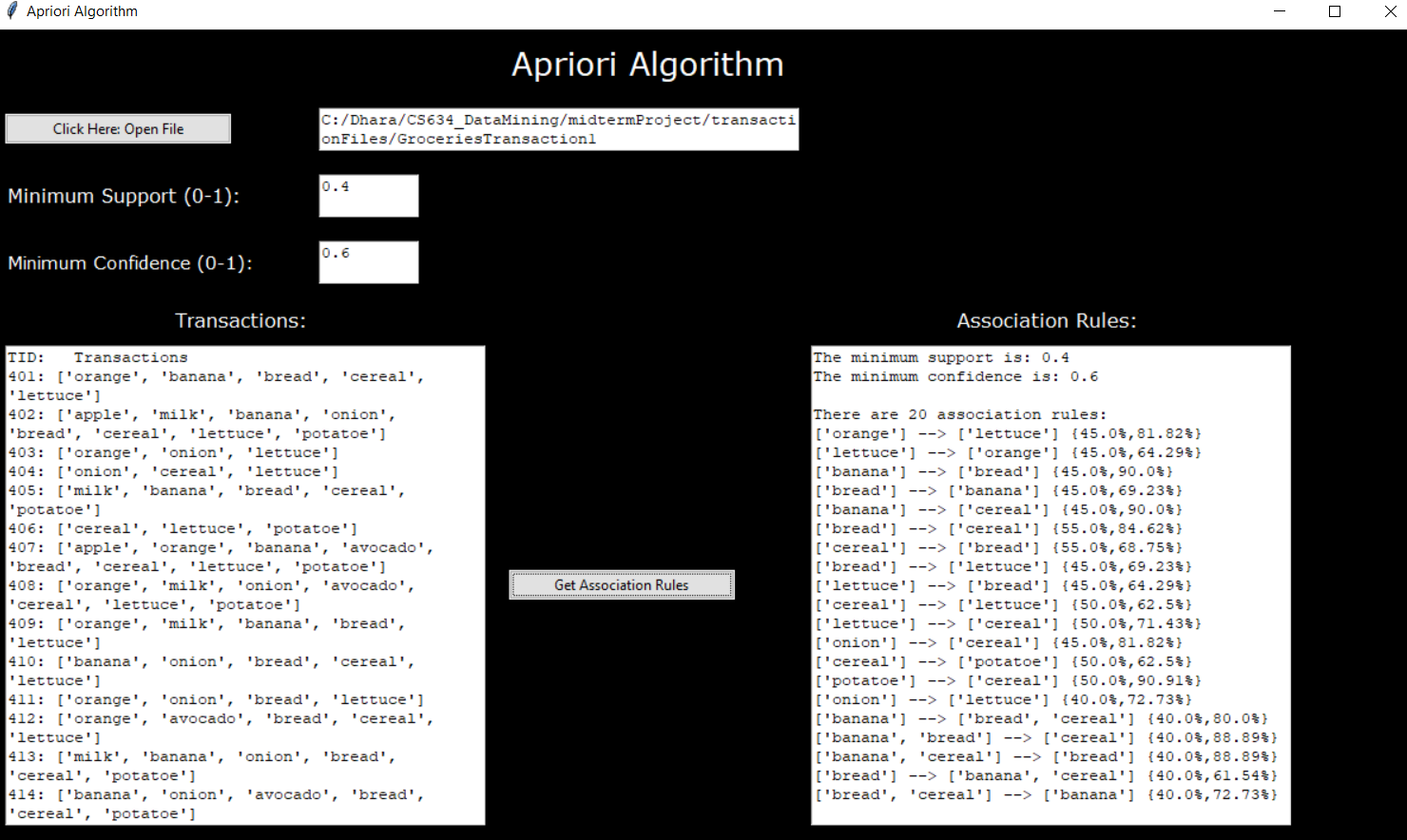
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| apple | orange | milk | banana | onion | avocado | bread | cereal | lettuce | potatoe |

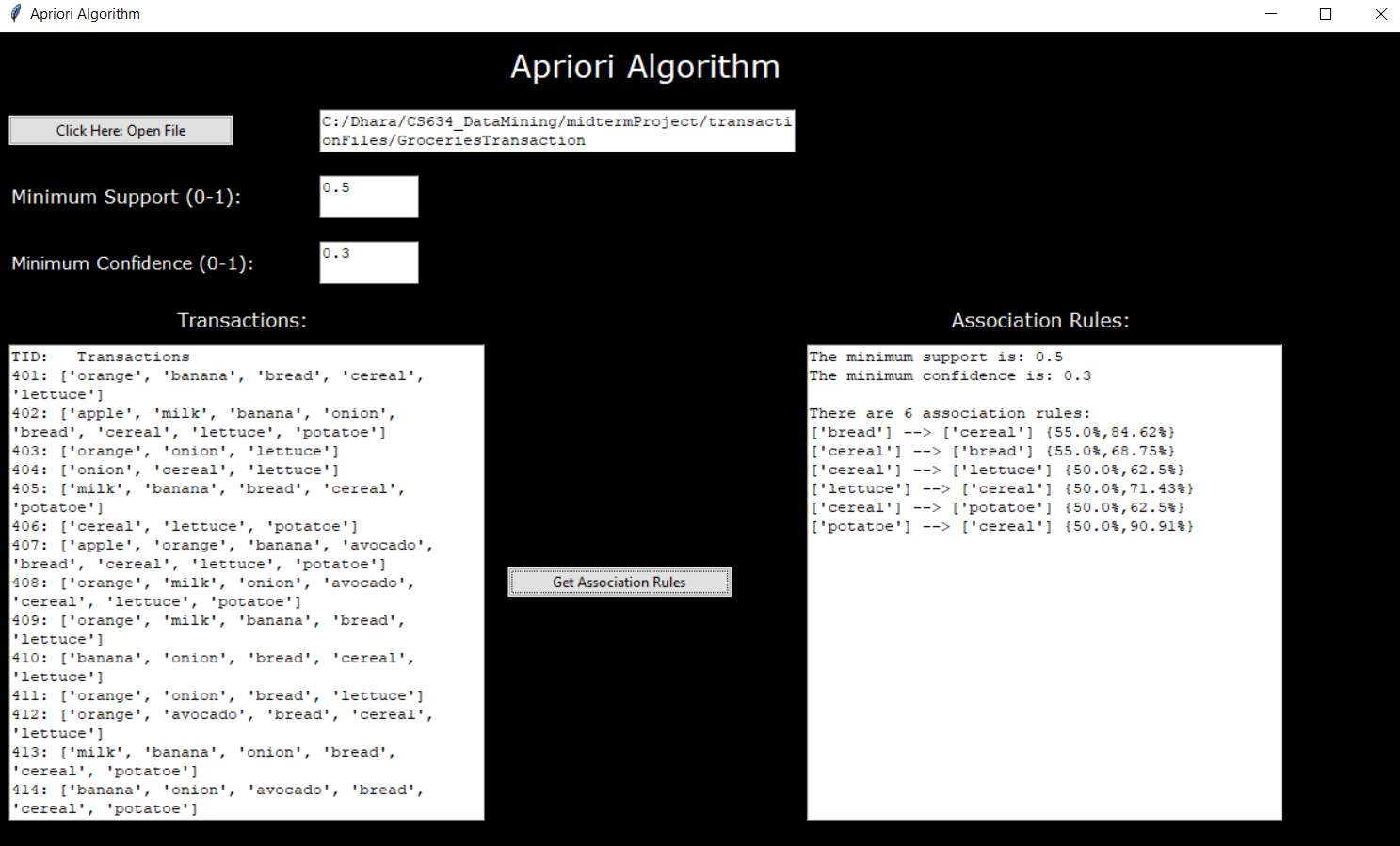
# Transaction List 1: Grocery Items

## Input file:

These are the list of transaction being passed

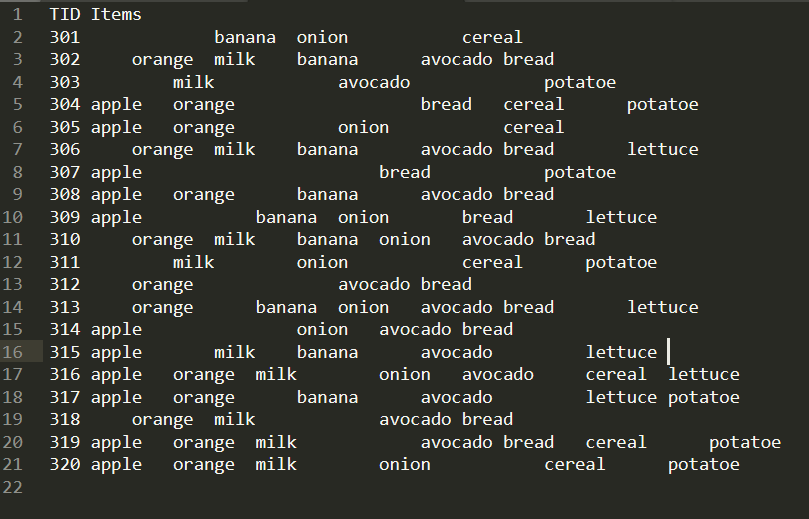
## Output File:

**Minimum support = 0.4 and minimum confidence = 0.6**

**Minimum support = 0.50 and minimum confidence = 0.3**

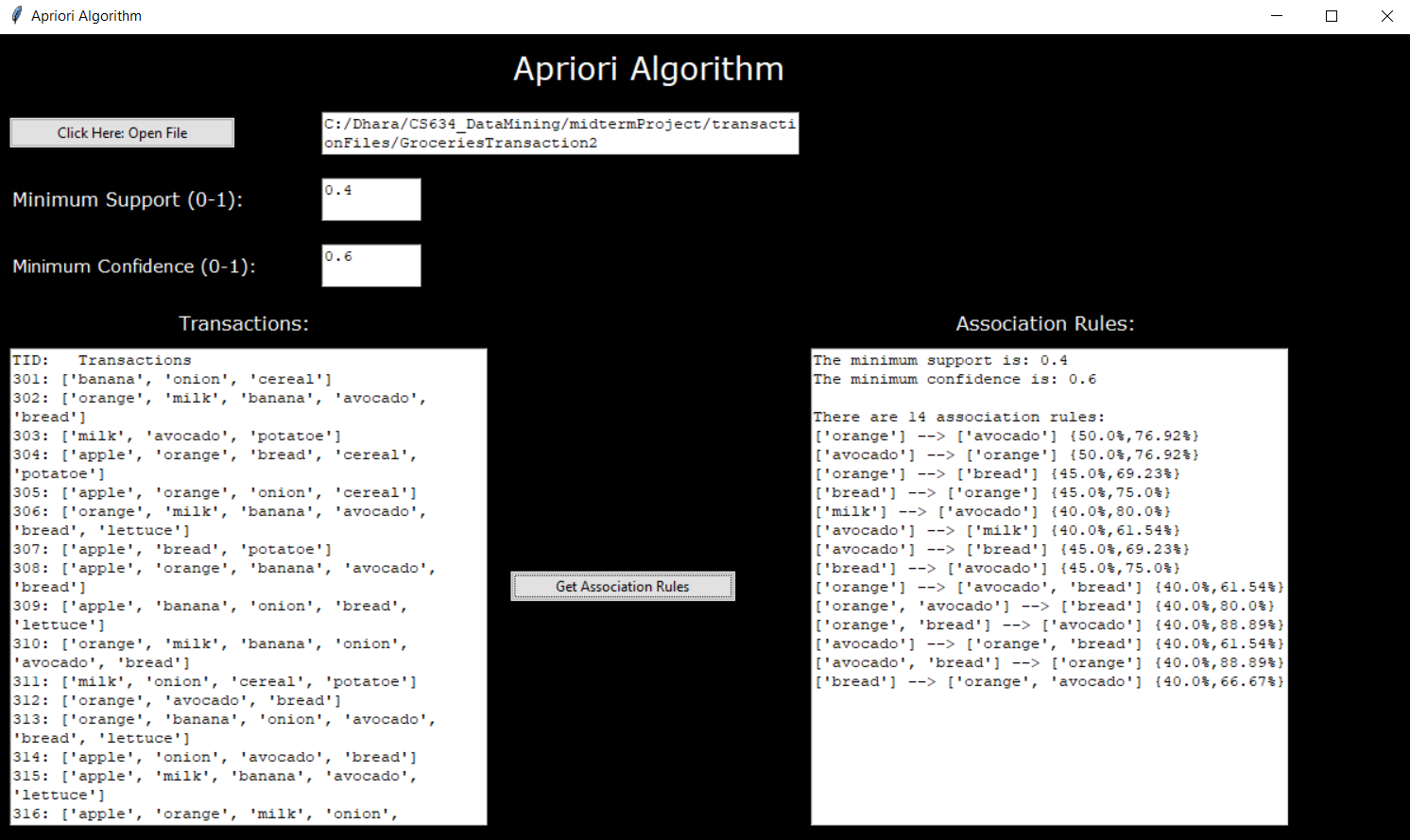
# Transaction List 2: Grocery Items

## Input File:

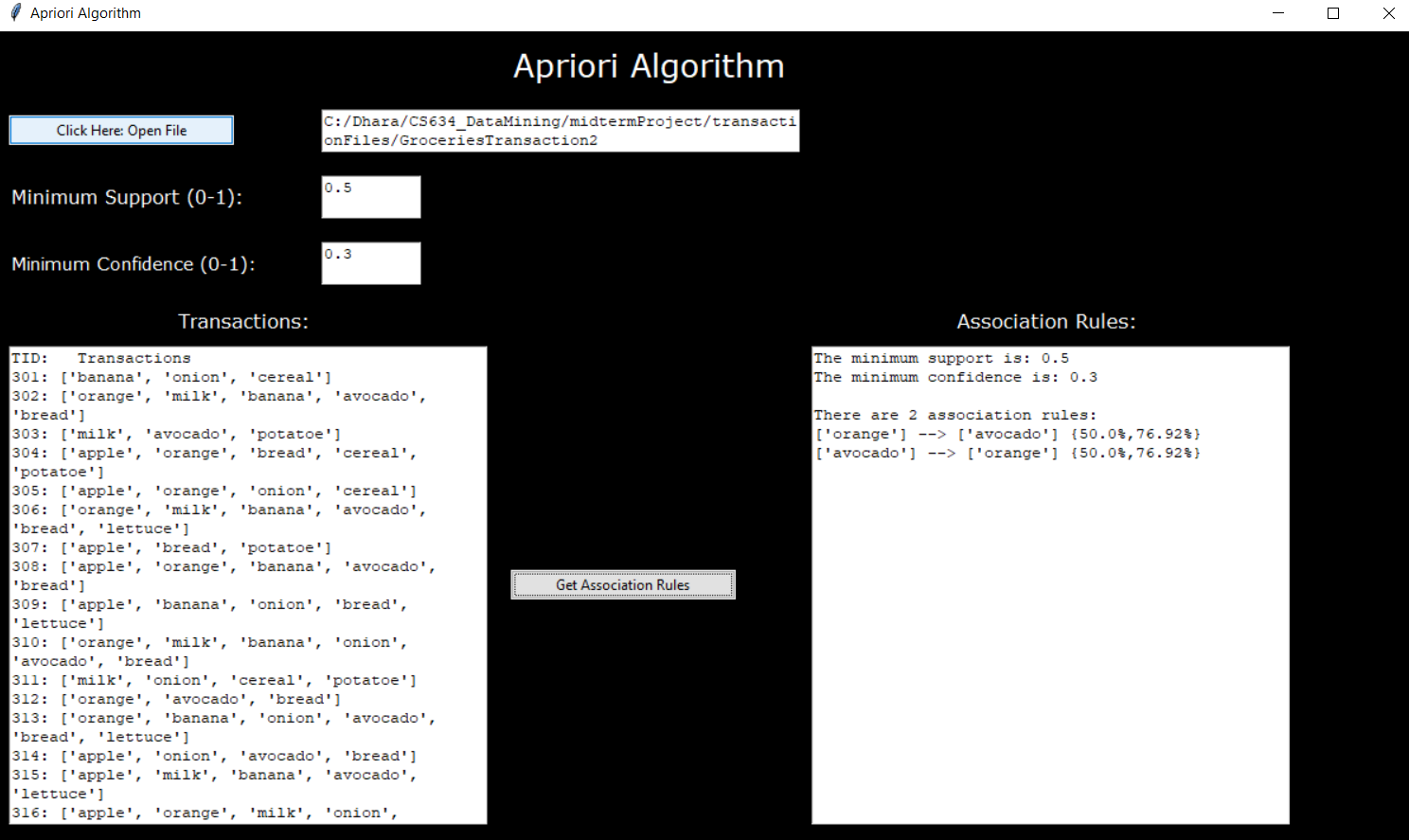


## Output File:

**Minimum support = 0.4 and minimum confidence = 0.6**

****

**Minimum support = 0.50 and minimum confidence = 0.3**

****

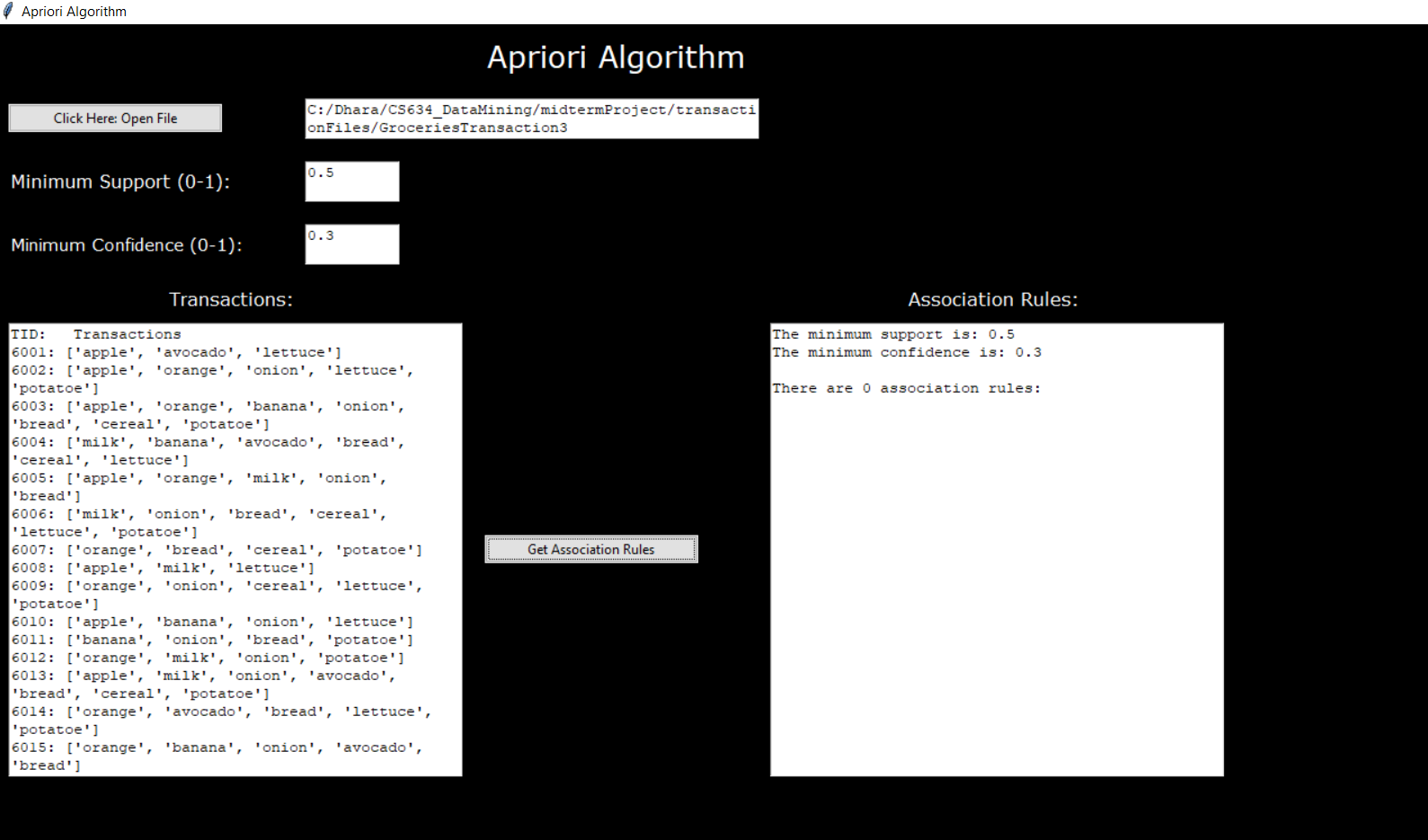
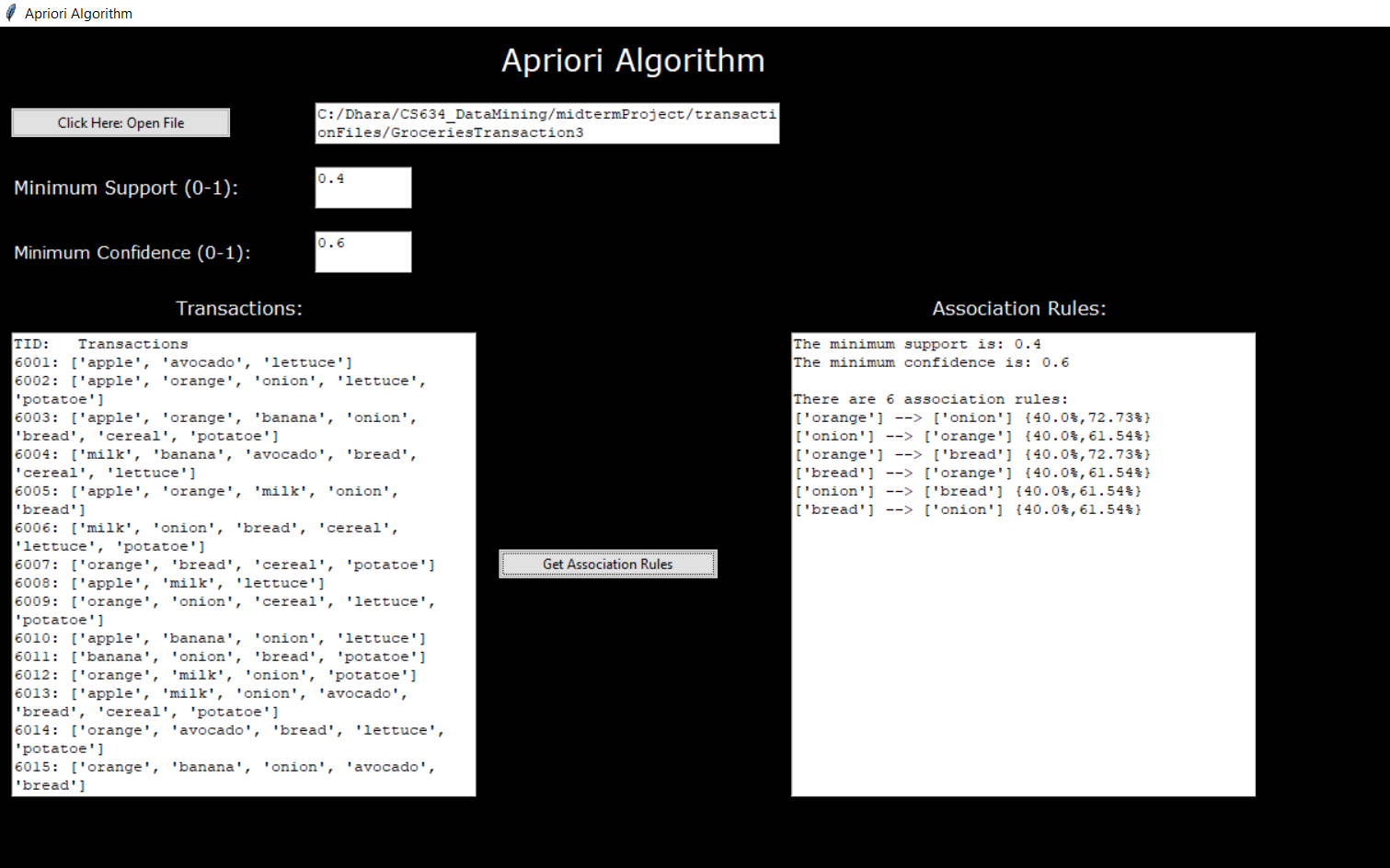
# Transaction List 3: Grocery Items

## Input File:

****

## Output File:

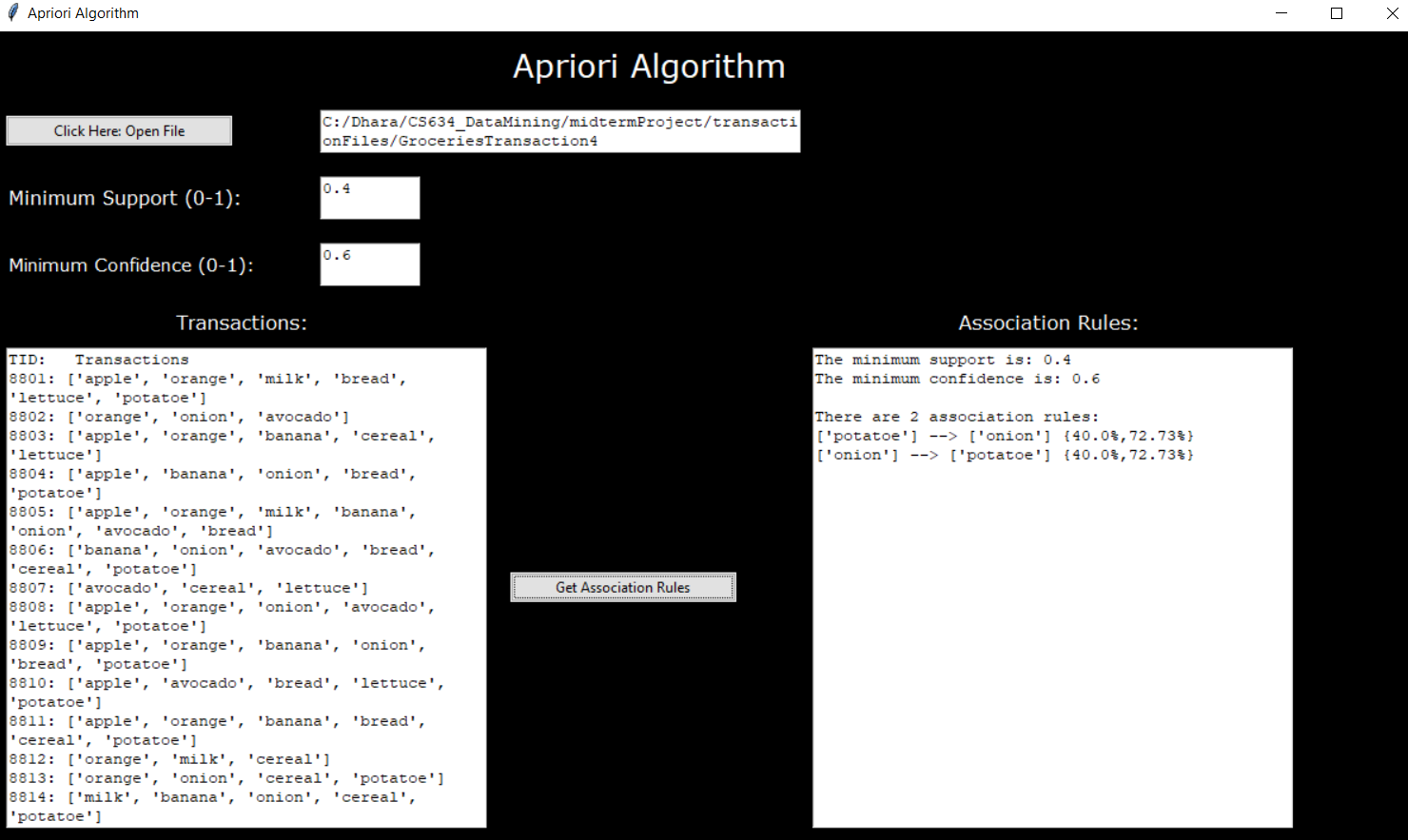
**Minimum support = 0.4 and minimum confidence = 0.6**

**Minimum support = 0.50 and minimum confidence = 0.3**

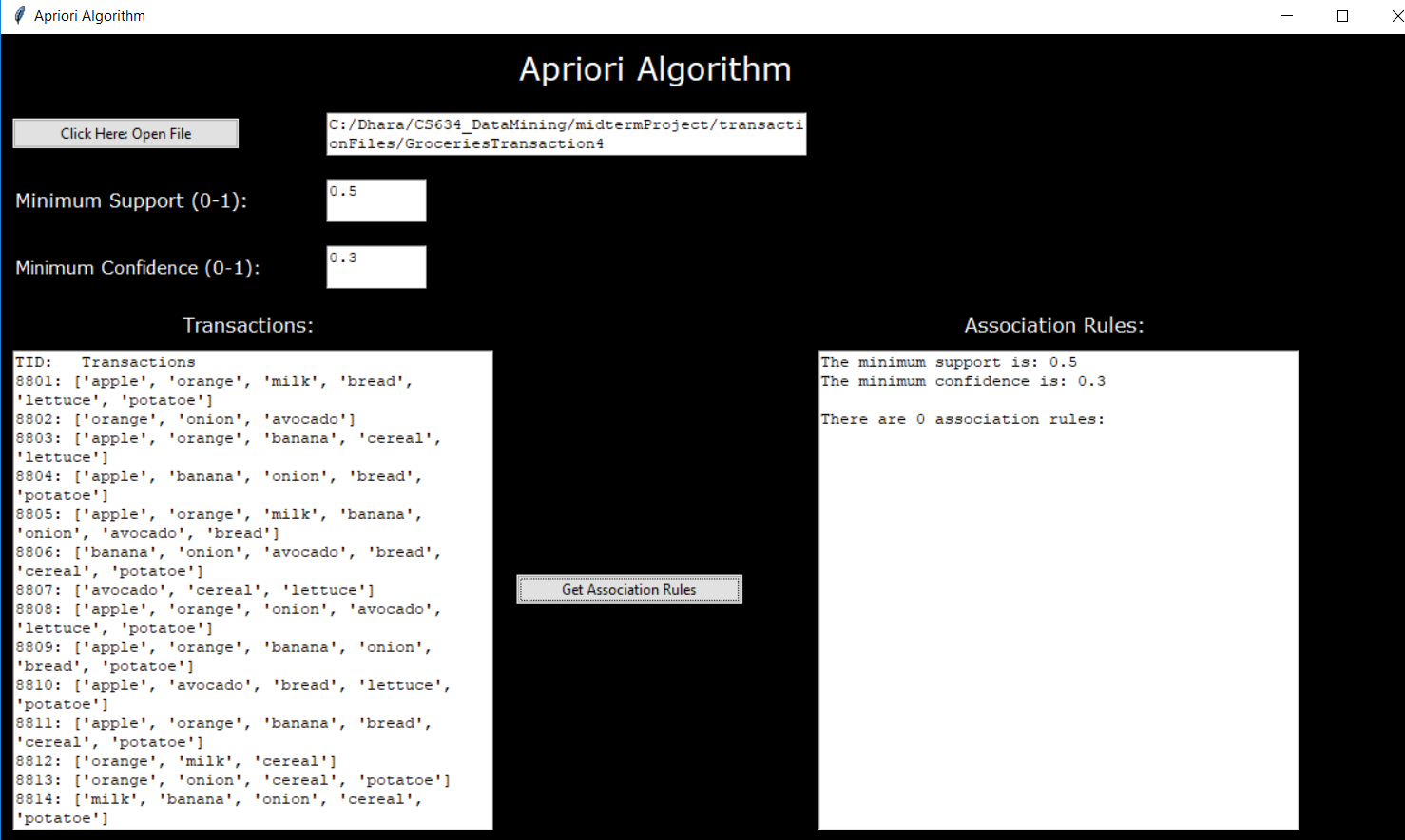
# Transaction List 4: Grocery Items

## Input File:

## Output File:

**Minimum support = 0.4 and minimum confidence = 0.6**

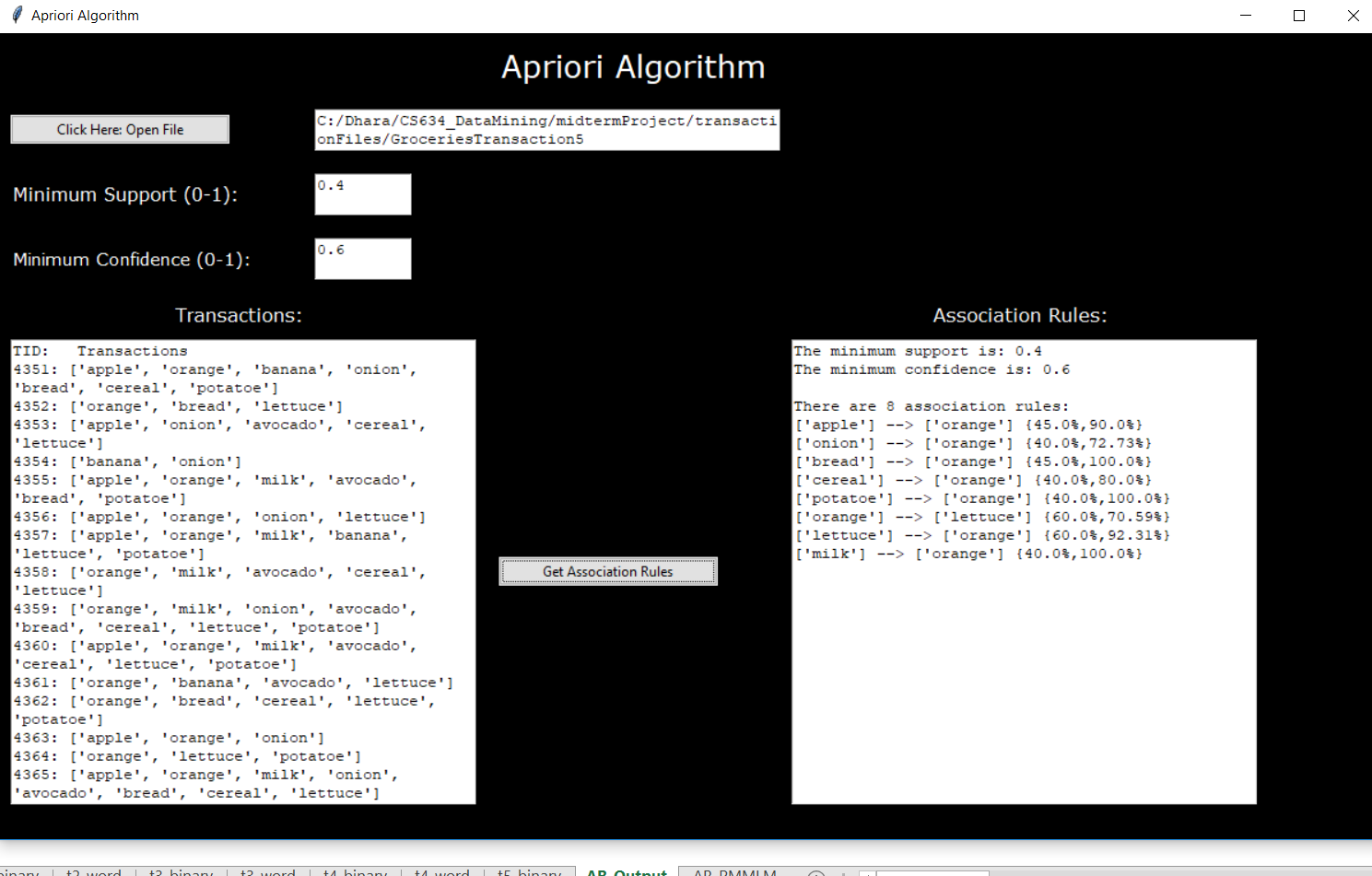
**Minimum support = 0.50 and minimum confidence = 0.3**

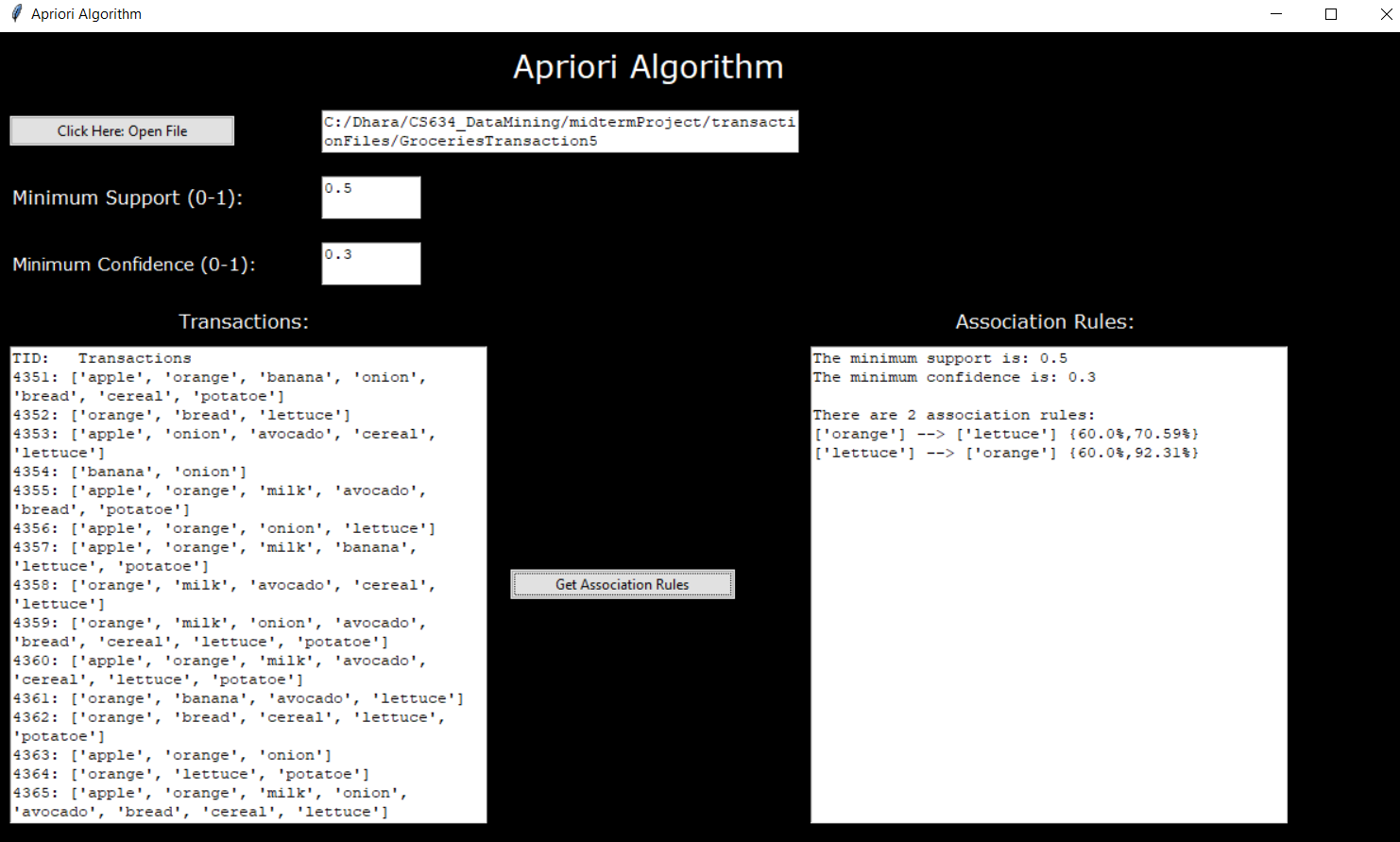


# Transaction List 5: Grocery Items

## Input File:

## Output File:

**Minimum support = 0.4 and minimum confidence = 0.6**

**Minimum support = 0.50 and minimum confidence = 0.3**

# Code:

## AprioriAlgorithmV5.py : Apriori Algorithm Implementation

|  |
| --- |
| """  Version 5: Going to convert everything into functions.;  So we can call only one function to get all association rules  """  **from** itertools **import** combinations**,** permutations  **def** findUniqueItems**(**ItemTransactionDic**):**  #Input: Pass a dictionary with items in it  #Output: All the unique items in the transaction list; returns a list with strings  uniqueItems**=[]**  **for** key **in** ItemTransactionDic**:**  **for** val **in** ItemTransactionDic**[**key**]:**  **if** val **in** uniqueItems**:**  **continue**  **else:**  uniqueItems**.**append**(**val**)**  **return(**uniqueItems**)**  **def** one\_hot\_encode**(**data**,**uniqueItems**):**  # creates one hot encoding for the tranaction list with 0s and 1s    one\_hot\_encodeArr**=[]**  temp**=[**0**]\***len**(**uniqueItems**)**  **for** key **in** data**:**  **for** item **in** uniqueItems**:**  **if** item **in** data**[**key**]:**  idx**=**uniqueItems**.**index**(**item**)**  temp**[**idx**]=**1  one\_hot\_encodeArr**.**append**(**temp**)**  temp**=[**0**]\***len**(**uniqueItems**)**  **return** one\_hot\_encodeArr  **def** frequentItemset**(**support**,**min\_support**,**comboList**):**  # Input: (1) support is a list of suport value for each itemset  # (2) min\_support: single value set by user  # (3) comboList: list of uniqueITems in  # outputs: index value  passItem**=[]**  rejectItem**=[]**  support\_value**=[]**  **for** i **in** range**(**0**,**len**(**support**)):**  **if** support**[**i**]** **>=** min\_support**:**  passItem**.**append**(**comboList**[**i**])**  support\_value**.**append**(**support**[**i**])**  **else:**  rejectItem**.**append**(**comboList**[**i**])**  **return** passItem**,**rejectItem**,**support\_value  **def** findSupport**(**combinationList**,** data**):**  #Finds support for all the items  #Input: (1) CombinationList= type list  # (2) data dictionary!  #Return a list of supports  total\_entries**=**len**(**data**)**  support**=[]**  count**=**0  **for** j **in** combinationList**:**  #print(j)  **for** key **in** data**:**  **if** set**(**j**).**issubset**(**set**(**data**[**key**])):**  count**=**count **+**1  #print(count)  support**.**append**(**count**)**  count**=**0  support**=[**j**/**total\_entries **for** j **in** support**]**  **return** support  **def** combos**(**One\_Itemset**,** c**):**  #Input: (1) One\_Itemset: is a list of singe frequent items  # (2) c is the number of items per set; 2= {pen,bag}, 3= {pen,cookie,candy}, etc  #Output: returns all the combo as a list  TotalCombo**=**combinations**(**One\_Itemset**,** c**)**  combo**=[]**  **for** i **in** list**(**TotalCombo**):**  combo**.**append**(**i**)**  #print(i)    **return** combo  """  Creating the first frequent Itemset or L1  Input: ItemTransactionDic, min\_support,  Output: One\_Itemset,ItemRejList,Support\_value  uses the defined functions: findUniqueItems, one\_hot\_encode and frequentItemset  """  **def** getItemsetL1**(**ItemTransactionDic**,**min\_support**):**  total\_entries**=**len**(**ItemTransactionDic**)**    """  Find all unique items in datasets  """  uniqueItems**=** findUniqueItems**(**ItemTransactionDic**)**    """  Convert each tuple into a one hot encoder  """  ohe**=**one\_hot\_encode**(**ItemTransactionDic**,**uniqueItems**)**      """  Find First Frequenct Itemsets  """  #Going to sum each column and find Support of each item  support**=[]**  **for** i **in** range**(**0**,**len**(**uniqueItems**)):**  s**=**sum**([**pair**[**i**]** **for** pair **in** ohe**])**  support**.**append**(**s**)**  support**=[**j**/**total\_entries **for** j **in** support **]**  # Filter by min support and return passing items and rejected items for just single items  One\_Itemset**,**ItemRejList**,**Support\_value**=**frequentItemset**(**support**,**min\_support**,**uniqueItems**)**  #print("\nfor c =", 1)  #print("Items passed by support: ",One\_Itemset)  #print("Items rejected by support: ",ItemRejList)  **return(**One\_Itemset**,**ItemRejList**,**Support\_value**)**  """  Rest of the combinations  => Inputs: One\_Itemset,ItemRejList,Support\_value  => Output: Frequent\_Itemlist, FreqItemsetSup  """  **def** scanNprun**(**ItemTransactionDic**,**min\_support**):**  One\_Itemset**,**ItemRejList**,**Support\_value**=**getItemsetL1**(**ItemTransactionDic**,**min\_support**)**  maxCombo**=**len**(**One\_Itemset**)**    Frequent\_Itemlist**=[]**  supports\_Itemlist**=[]**  Frequent\_Itemlist**.**append**(**One\_Itemset**)**  supports\_Itemlist**.**append**(**Support\_value**)**    **for** c **in** range**(**2**,**maxCombo**):**  #find combinations  #print("\n\*\*\*\*for c =", c," \*\*\*\*")  combo**=**combos**(**One\_Itemset**,** c**)**  #print("Before combinations are filtered w/ rej: \n",combo)  #Check if rejected is in combination  **for** j **in** ItemRejList**:**  **for** i **in** combo**:**  **if** set**(**j**).**issubset**(**set**(**i**)):**  combo**.**remove**(**i**)**  #print(combo)  #print("After combinations are filtered with Rej: \n",combo)  # Scan dataset and calculate support  sup**=**findSupport**(**combo**,** ItemTransactionDic**)**  #print("Support value: ",s)  #Make a list of passing items and reject items  Pass\_Itemset**,**ItemRejList**,**Support\_value**=**frequentItemset**(**sup**,**min\_support**,**combo**)**  #print("Items passed by support: ",Pass\_Itemset)  #print("Items rejected by support: ",ItemRejList)  Frequent\_Itemlist**.**append**(**Pass\_Itemset**)**  supports\_Itemlist**.**append**(**Support\_value**)**    #modifies the only items from list of string to list of lists!!!  **for** i **in** range**(**0**,** len**(**Frequent\_Itemlist**[**0**])):**  Frequent\_Itemlist**[**0**][**i**]=[**Frequent\_Itemlist**[**0**][**i**]]**    **for** j **in** range**(**1**,**len**(**Frequent\_Itemlist**)):**  **for** idx **in** range**(**0**,**len**(**Frequent\_Itemlist**[**j**])):**  Frequent\_Itemlist**[**j**][**idx**]=**list**(**Frequent\_Itemlist**[**j**][**idx**])**      """  Create a FreqItemsetSup dictionary with frequent Item as key and support as value  eg. ('cheese', 'juice'): 0.5 etc for items group 2 or greater  """    mapped**=[]**  **for** i **in** range**(**0**,**len**(**Frequent\_Itemlist**)):** # note that len of Frequent\_Itemlist will equal len(supports\_Itemlist) ALWAYSS IF NOT there is a bug    m**=**list**(**zip**(**Frequent\_Itemlist**[**i**],**supports\_Itemlist**[**i**]))**  mapped**.**append**(**m**)**    FreqItemsetSup**={}**  **for** i **in** range**(**0**,**len**(**mapped**)):**  **for** j **in** range**(**0**,**len**(**mapped**[**i**])):**    strItem **=** mapped**[**i**][**j**][**0**]** # e.g['pen'] or ['cheese', 'milk', 'juice']  supItem **=** mapped**[**i**][**j**][**1**]** # 0.5, or 0.75    **if** len**(**strItem**)** **==** 1**:**# have to do this bc single item like tuple(['pen'])= ('cheese',) annoying  FreqItemsetSup**[**strItem**[**0**]]=[**strItem**,**supItem**]**  **else:**  FreqItemsetSup**[**tuple**(**strItem**)]=[**strItem**,**supItem**]**    **return(**Frequent\_Itemlist**,** FreqItemsetSup**)**  """  Find confidence and assocoation rule  Input: Frequent\_Itemlist,FreqItemsetSup  output: Cleaned AssociationRules  """  **def** findAssociationRules**(**ItemTransactionDic**,**min\_support**,**min\_confid**):**    Frequent\_Itemlist**,**FreqItemsetSup**=**scanNprun**(**ItemTransactionDic**,**min\_support**)**    AssociationRules**=[]** # save the rule: Antecedent, consequent, support,confidence  **for** i **in** range**(**1**,**len**(**Frequent\_Itemlist**)):**  **for** item **in** Frequent\_Itemlist**[**i**]:** # just gets the item eg 'cheese', 'milk', 'juice')  perm**=**permutations**(**item**,** len**(**item**))**  #print("THis is item: ",item)    #find support; remain same despite change in order; Finding the numaerator of equation  sup**=**FreqItemsetSup**[**tuple**(**item**)][**1**]\***100    #Step2: find the permuation and find the denominator of equation  **for** p **in** list**(**perm**):**#Iterates trhough the permutations  #print("-----",p,"-----")  **for** k **in** FreqItemsetSup**:**# iterates to the dictionary of frequent items (1 item, 2 items, 3 item etc and corresponding supports)    p**=**list**(**p**)**  **for** v **in** range**(**1**,**len**(**p**)):**  l**=**len**(**p**[**0**:**len**(**p**)-**v**])**  compareVal**=**FreqItemsetSup**[**k**][**0**]**  #print("\nComparing to K: ", compareVal,len(list(compareVal)))  #print("Looking at AR: ",p[0:len(p)-v],"-->",p[-v:], l) # Note the logically error is in the length of key-pen give 3 instead of 1 smh    **if** set**(**p**[**0**:**len**(**p**)-**v**]).**issubset**(**compareVal**)** **and** len**(**compareVal**)==**l**:**  denominator**=**FreqItemsetSup**[**k**][**1**]**  confidence**=**round**(**sup**/**denominator**,**2**)**  **if** confidence **>=** **(**min\_confid**\***100**):**  antecedent**=**p**[**0**:**len**(**p**)-**v**]**  consequent**=**p**[-**v**:]**  supportVal**=**sup  AssociationRules**.**append**([**antecedent**,**consequent**,**round**(**supportVal**,**2**),**confidence**])**  #print(p[0:len(p)-v],"-->",p[-v:],"{",sup,"%,",confidence,"%}\n")    """  Clean up: Removing Duplicate Association Rules  Input: AssociationRules  output: Cleaned AssociationRules and printing results  """  removeIDK**=[]**  **for** a **in** range**(**0**,**len**(**AssociationRules**)):**  **for** IDK\_nextAR **in** range**(**a**+**1**,**len**(**AssociationRules**)):**  antcdentA**=**AssociationRules**[**a**][**0**]**  antcdentNext**=**AssociationRules**[**IDK\_nextAR**][**0**]**  precdentA**=**AssociationRules**[**a**][**1**]**  precdentNext**=**AssociationRules**[**IDK\_nextAR**][**1**]**    # Must check if len of antcendentand precedent of first rule is te same as the next one  **if** len**(**antcdentA**)==**len**(**antcdentNext**)** **and** len**(**precdentA**)==**len**(**precdentNext**):**  **if** set**(**antcdentA**).**issubset**(**antcdentNext**)** **and** set**(**precdentA**).**issubset**(**precdentNext**):**  removeIDK**.**append**(**IDK\_nextAR**)**  AssociationRules **=** **[**i **for** j**,** i **in** enumerate**(**AssociationRules**)** **if** j **not** **in** removeIDK**]**  #for a in AssociationRules:  #print(a[0],"-->",a[1],"{",a[2],"%, ",a[3],"%}")    **return(**AssociationRules**)** |

## aaGUI.py: Graphical User Interface

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """  Created on Wed Oct 3 13:46:38 2018  @author: dhara  """  **from** tkinter **import** **\***  **from** tkinter **import** ttk  **from** tkinter **import** filedialog **as** fd  **import** AprioriAlgorithmV5 **as** apriori  **def** close\_window**():**  window**.**destory**()**  exit**()**  **def** mfileopen**():**  # Load Data  file **=** fd**.**askopenfile**()**  Filenameoutput**.**delete**(**0.0**,**END**)**  Filenameoutput**.**insert**(**END**,**file**.**name**)**    #Give Preview of data  file**=**Filenameoutput**.**get**(**"1.0"**,**END**)**  file**=**file**.**rstrip**()**  **print(**file**)**  f **=** open**(**file**,** "r"**)**  lines **=** f**.**readline**()**  lines**=**next**(**f**)**    ItemTransactionDic**={}** #This is how you create a dictionary  **while(**lines **!=**''**):**  a**=**lines**.**split**()**  ItemTransactionDic**[**a**[**0**]]=**a**[**1**:]**  lines**=**f**.**readline**()**  f**.**close**()**    TransactionListOutput**.**delete**(**0.0**,**END**)**  TransactionListOutput**.**insert**(**END**,**"TID: Transactions"**)**  TransactionListOutput**.**insert**(**END**,**"\n"**)**  **for** key **in** ItemTransactionDic**:**  s**=** key**+** ": "**+**str**(**ItemTransactionDic**[**key**])**  TransactionListOutput**.**insert**(**END**,**s**)**  TransactionListOutput**.**insert**(**END**,**"\n"**)**    **def** getApriori**():**  file**=**Filenameoutput**.**get**(**"1.0"**,**END**)**  file**=**file**.**rstrip**()**  **print(**file**)**  f **=** open**(**file**,** "r"**)**  lines **=** f**.**readline**()**  lines**=**next**(**f**)**    ItemTransactionDic**={}** #This is how you create a dictionary  **while(**lines **!=**''**):**  a**=**lines**.**split**()**  ItemTransactionDic**[**a**[**0**]]=**a**[**1**:]**  lines**=**f**.**readline**()**  f**.**close**()**  #print(ItemTransactionDic)  min\_support**=** float**(**SupportLabelInput**.**get**(**"1.0"**,**END**))**  min\_confid**=** float**(**ConfidenceLabelInput**.**get**(**"1.0"**,**END**))**  **print(**type**(**min\_support**))**  **print(**type**(**min\_confid**))**  associationsRule**=**apriori**.**findAssociationRules**(**ItemTransactionDic**,**min\_support**,**min\_confid**)**  **print(**associationsRule**)**  ApriorOutput**.**delete**(**0.0**,**END**)**  ApriorOutput**.**insert**(**END**,**"The minimum support is: "**+**str**(**min\_support**)** **)**  ApriorOutput**.**insert**(**END**,**"\n"**)**  ApriorOutput**.**insert**(**END**,**"The minimum confidence is: "**+**str**(**min\_confid**)** **)**  ApriorOutput**.**insert**(**END**,**"\n\n"**)**  ApriorOutput**.**insert**(**END**,**"There are "**+**str**(**len**(**associationsRule**))+** " association rules:"**)**  ApriorOutput**.**insert**(**END**,**"\n"**)**  **for** a **in** associationsRule**:**  s**=**str**(**a**[**0**])** **+**" --> " **+**str**(**a**[**1**])+** " {"**+** str**(**a**[**2**])+**"%,"**+** str**(**a**[**3**])+**"%}"  ApriorOutput**.**insert**(**END**,**s**)**  ApriorOutput**.**insert**(**END**,**"\n"**)**    #print(a[0],"-->",a[1],"{",a[2],"%, ",a[3],"%}")    ##### main####:  window**=**Tk**()**      window**.**title**(**"Apriori Algorithm"**)**  window**.**configure**(**background**=**"black"**)**  window**.**geometry**(**"1200x700"**)**  #window.resizable(0, 0)  # Get file from computer and show path being used  titleLabel**=**ttk**.**Label**(**window**,**text**=**"Apriori Algorithm"**,**foreground**=**"white"**,**background**=**"black"**)**  titleLabel**.**config**(**font**=(**"Verdana"**,** 20**))**  titleLabel**.**grid**(**row**=**0**,**column**=**2**,** padx**=**10**,** pady**=**10**)**  filename**=**ttk**.**Button**(**window**,**text**=**"Click Here: Open File"**,** width**=**30**,**command**=**mfileopen**)**  filename**.**grid**(**row**=**1**,**column**=**0**,**sticky**=**W**,** padx**=**10**,** pady**=**10**)**  Filenameoutput**=**Text**(**window**,** width**=**50**,**height**=**2**,**wrap**=**WORD**,**background**=**"white"**)**  Filenameoutput**.**grid**(**row**=**1**,**column**=**1**,**columnspan**=**2**,**sticky**=**E**,**pady**=**10**)**  SupportLabel**=**ttk**.**Label**(**window**,**text**=**"Minimum Support (0-1): "**,**foreground**=**"white"**,**background**=**"black"**)**  SupportLabel**.**config**(**font**=(**"Verdana"**,** 12**))**  SupportLabel**.**grid**(**row**=**2**,**column**=**0**,**columnspan**=**2**,**sticky**=**W**,**padx**=**10**,** pady**=**10**)**  SupportLabelInput**=** Text**(**window**,** width**=**10**,**height**=**2**,**wrap**=**WORD**,**background**=**"white"**)**  SupportLabelInput**.**grid**(**row**=**2**,**column**=**1**,**columnspan**=**2**,**sticky**=**W**,** pady**=**10**)**  ConfidenceLabel**=**ttk**.**Label**(**window**,**text**=**"Minimum Confidence (0-1): "**,**foreground**=**"white"**,**background**=**"black"**)**  ConfidenceLabel**.**config**(**font**=(**"Verdana"**,** 11**))**  ConfidenceLabel**.**grid**(**row**=**3**,**column**=**0**,**columnspan**=**2**,**sticky**=**W**,**padx**=**10**,** pady**=**10**)**  ConfidenceLabelInput**=** Text**(**window**,** width**=**10**,**height**=**2**,**wrap**=**WORD**,**background**=**"white"**)**  ConfidenceLabelInput**.**grid**(**row**=**3**,**column**=**1**,**columnspan**=**2**,**sticky**=**W**,**pady**=**10**)**  TransactionLabel**=**ttk**.**Label**(**window**,**text**=**"Transactions: "**,**foreground**=**"white"**,**background**=**"black"**)**  TransactionLabel**.**config**(**font**=(**"Verdana"**,** 12**))**  TransactionLabel**.**grid**(**row**=**4**,**column**=**0**,**columnspan**=**2**,**padx**=**10**,** pady**=**10**)**  AssociationRuleLabel**=**ttk**.**Label**(**window**,**text**=**"Association Rules: "**,**foreground**=**"white"**,**background**=**"black"**)**  AssociationRuleLabel**.**config**(**font**=(**"Verdana"**,** 12**))**  AssociationRuleLabel**.**grid**(**row**=**4**,**column**=**10**,**columnspan**=**2**,**padx**=**10**,** pady**=**10**)**  TransactionListOutput**=**Text**(**window**,** width**=**50**,**height**=**25**,**wrap**=**WORD**,**background**=**"white"**)**  TransactionListOutput**.**grid**(**row**=**10**,**column**=**0**,**columnspan**=**2**,**sticky**=**W**,**padx**=**10**)**  # Run the Aprior Algorithm and show results  ApriorButton**=**ttk**.**Button**(**window**,**text**=**"Get Association Rules"**,** width**=**30**,**command**=**getApriori**)**  ApriorButton**.**grid**(**row**=**10**,**column**=**2**,**sticky**=**W**,**padx**=**10**)**  ApriorOutput**=**Text**(**window**,** width**=**50**,**height**=**25**,**wrap**=**WORD**,**background**=**"white"**)**  ApriorOutput**.**grid**(**row**=**10**,**column**=**10**,**columnspan**=**2**,**sticky**=**E**,**padx**=**10**)**  ######tun the main loop  window**.**mainloop**()** |