Introduction:

- The world is becoming more and more data driven, with endless amounts of data available to work with. Data and information are increasing rapidly, the information's growth rate is so high that the information available to us will be volatile soon. Data is created by hundreds of users, companies and industry. **Data Mining** helps to extract the vital information/knowledge from a large set of data, in data mining we are looking for hidden information but without any idea about what type of information/data we want to find and what we plan to use for it once. In Data Mining according to data we are mining.

Overview:

- In this we discuss about the approaches that are performed on the datasets. Proper problem statements are discussed.
- The solutions for those proposed questions are interpreted by two approaches i) Collaborative filtering Algorithm and ii) Apriori Algorithm results.
- Visualizations and tables are also introduced to enhance the results. The results are carried out by the approaches.

Datasets

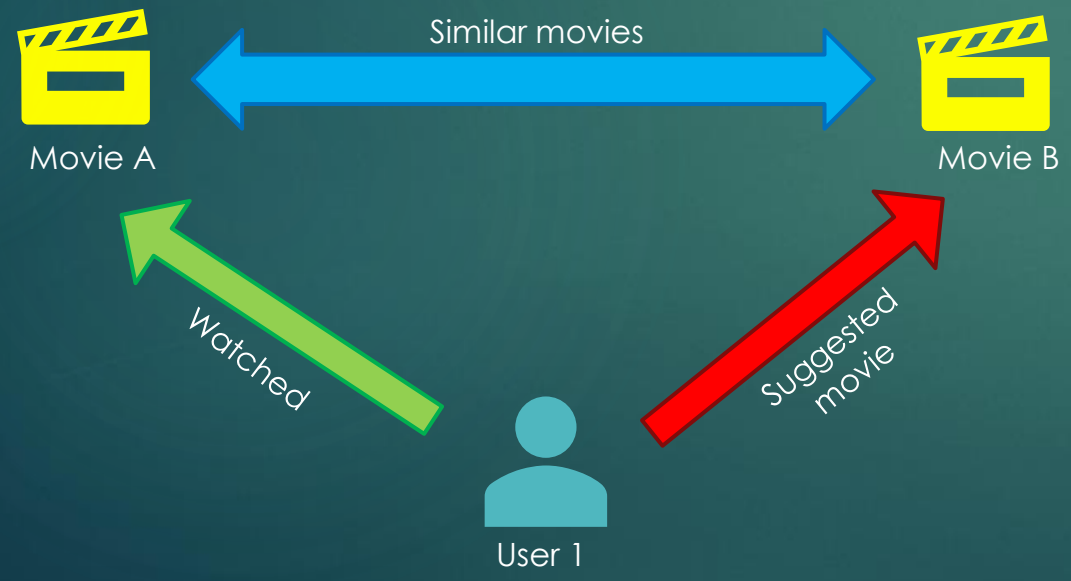
Movies Dataset	Groceries Dataset
In this dataset it contains the rating information of the movies which are given by the users. Several data mining techniques can be performed in this dataset.	In this dataset it contains the transaction history of the items bought. In this Association rules are performed.
Recommended systems by collaborative filtering is used to find the similar movies to a user and suggest new movies to them.	Apriori algorithm approach is used to suggest new items, based on the items that are already purchased. This helps the e-commerce businesses to used this data to help improve their business.

Problem Statements

Movies Dataset	Groceries Dataset
How many similar movies are present based on user ratings to the movies?	What are the Top 10 items bought in the dataset?
How many similar users are present in the database and finding them by their movie ratings?	What are the Top 2 items bought in all transactions?
Collaborative filtering: Suggesting new movies for the particular user.	How many Top 3 items bought in all transactions?
	Apriori Algorithm.

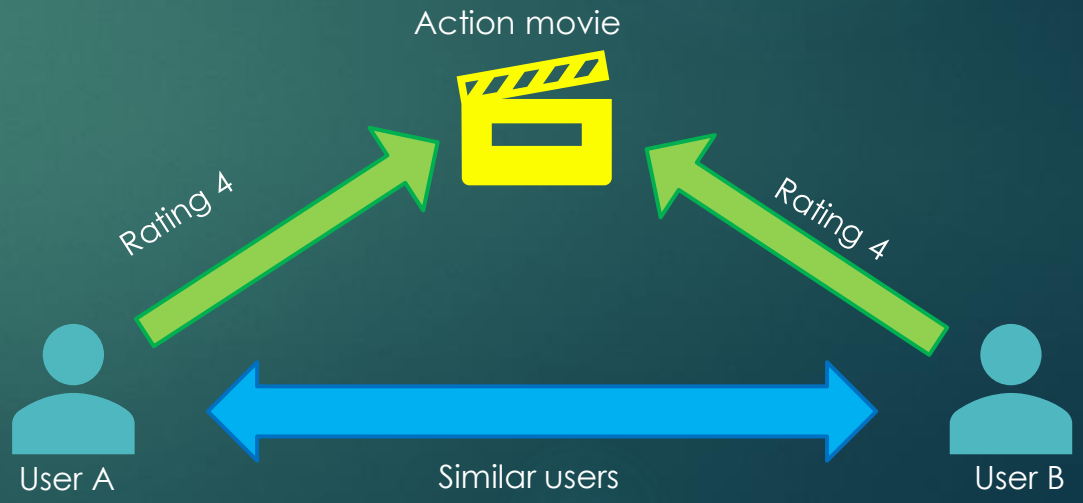
Similar movies:

- By using correlation we can find the relation between two movies A&B. If the relation between them is positive, then they are similar. If the relation between them is negative, then they are not similar.
- These similar movies can be used to suggest to other users who did not watch either A or B.



Similar users:

- By using correlation between users A & B based on their movie ratings we can find the similar users.
- By this we can predict the ratings users A&B to action movies because the both users liked action movies. If a new action movie is released, then there is a high of the users to like that movie.



Similar Movies

Code:

```
import pandas as pd
import matplotlib.pyplot as plt# data visualization library
from sklearn.metrics import pairwise_distances
from scipy.spatial.distance import cosine, correlation


#reading csv file.
ratings = pd.read_csv('ratings.csv')
ratings.drop(['timestamp'],axis=1,inplace =True)


#reading csv file.
movies = pd.read_csv('movies.csv')
movies.drop(['genres'],axis=1, inplace = True)


#Re-arraigning the index, column's and values.
rating_mat = ratings.pivot( index='movieId', columns='userId', values = "rating" ).reset_index(drop=True)


#Replacing NaN values with 0
rating_mat.fillna( 0, inplace = True )


#Finding the relation between the movies.
movie_sim = 1 - pairwise_distances( rating_mat.values, metric="correlation" )
movie_sim_df = pd.DataFrame( movie_sim )


movies['similarity'] = movie_sim_df.iloc[0]
movies.columns = ['movieId', 'title', 'similarity']


def get_similar_movies(movieId, topN= 5):
    movies['similarity']= movie_sim_df.iloc[movieId-1]
    top_n = movies.sort_values(['similarity'], ascending = False)[0:topN]
    print("similar movies to ",)
    return top_n


get_similar_movies( 1,10 )
```

Output:

	movieId	title	similarity
0	1	Toy Story (1995)	1.000000
2353	3112	'night Mother (1986)	0.461761
963	1264	Diva (1981)	0.361540
615	780	Independence Day (a.k.a. ID4) (1996)	0.358473
815	1073	Willy Wonka & the Chocolate Factory (1971)	0.357314
546	648	Mission: Impossible (1996)	0.352847
622	788	Nutty Professor, The (1996)	0.350295
1756	2354	Rugrats Movie, The (1998)	0.345431
322	364	Lion King, The (1994)	0.344248
32	34	Babe (1995)	0.341136

Similar Users

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt# data visualization library
from sklearn.metrics import pairwise_distances
from scipy.spatial.distance import cosine, correlation


#reading csv file.
ratings = pd.read_csv('ratings.csv')
ratings.drop(['timestamp'],axis=1,inplace =True)


#reading csv file.
movies = pd.read_csv('movies.csv')
movies.drop(['genres'],axis=1, inplace = True)


#Re-arrainging the index, columns and values.
user_movies_df = ratings.pivot( index='userId', columns='movieId', values = "rating" ).reset_index(drop=True)


#Replacing NaN values with 0
user_movies_df.fillna(0, inplace = True)


#Finding the relation between the users by pairwise_distances
user_sim= 1 - pairwise_distances( user_movies_df.values, metric="cosine" )
user_sim_df = pd.DataFrame( user_sim )


np.fill_diagonal( user_sim, 0 )


user_sim_df = pd.DataFrame( user_sim )


def u_s_movies(u1,u2):
    c_movies=ratings[ratings.userId==u1].merge(ratings[ratings.userId==u2], on ='movieId', how ='inner')
    return c_movies.merge(movies, on= 'movieId')

#checking the users 608 and 480.
u_s_movies( 608, 480 )
```

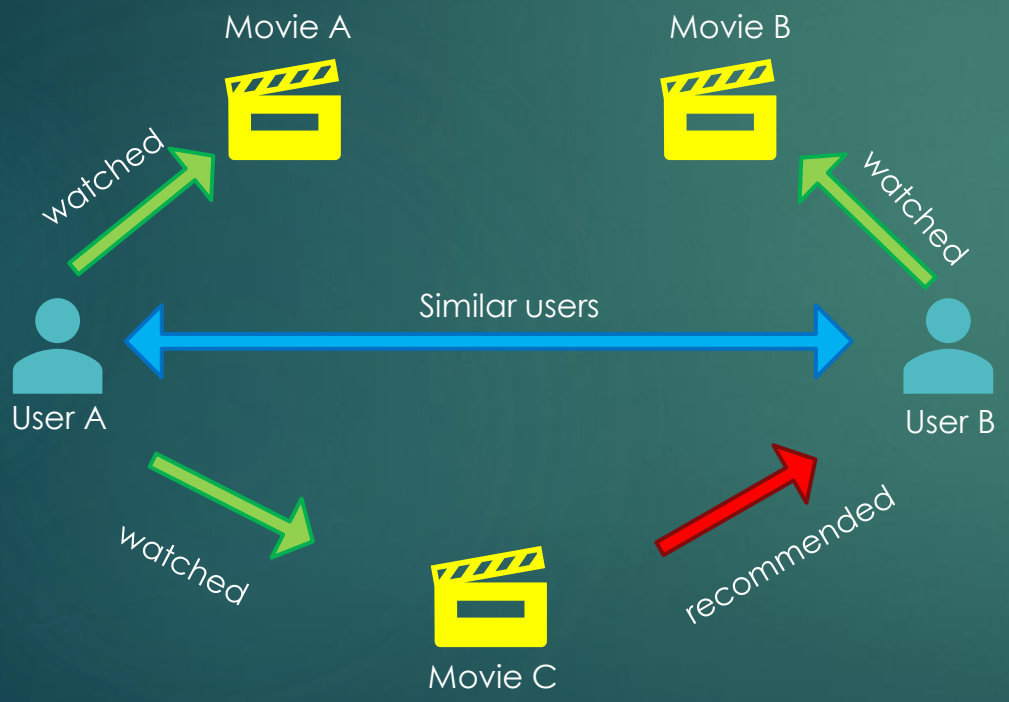
Output:

	userId_x	movieId	rating_x	userId_y	rating_y	title
0	608	1	2.5	480	3.0	Toy Story (1995)
1	608	2	2.0	480	3.0	Jumanji (1995)
2	608	3	2.0	480	2.5	Grumpier Old Men (1995)
3	608	10	4.0	480	4.0	GoldenEye (1995)
4	608	16	4.5	480	4.0	Casino (1995)
...	...	...	...	...	...	...
417	608	44191	4.0	480	4.0	V for Vendetta (2006)
418	608	44665	4.0	480	4.5	Lucky Number Slevin (2006)
419	608	45499	4.0	480	3.0	X-Men: The Last Stand (2006)
420	608	51255	4.5	480	4.0	Hot Fuzz (2007)
421	608	51662	5.0	480	4.0	300 (2007)

422 rows × 6 columns

Collaborative Filtering:

- Collaborative filtering gathers ratings of a product by customer, looks for the similar ratings, and gives suggestions using inter-customer collations.



Code:

```
#importing pandas library, merging the two dataframe ratings and movies
#deleting the columns 'geners' and 'timestamp'
import pandas as pd
ratings = pd.read_csv('ratings.csv')
movies = pd.read_csv('movies.csv')
ratings=pd.merge(movies,ratings).drop(['genres','timestamp'],axis=1)


#reconstructing the data frame with having userId as index , columns as movie titles and the values as ratings
user_ratings= ratings.pivot_table(index=['userId'],columns=['title'],values='rating')


#filling the null values to zero
df1=user_ratings.fillna(0)


#applying the pearson correlation function
#removing the movies similarities which are rated both movies by less than 100 people by using min_periods function
#if the movie X is rated by 50 people then the movie X is deleted because it may effect the result.
df = df1.corr(method='pearson',min_periods = 100)


#taking the movie ratings of the user1 and dropping null values i.e. movies which are not rated
user1= user_ratings.loc[1].dropna()


#finding the similar kind of movies for the selected user


simCandidates = pd.Series()
for i in range (0,len(user1.index)): #going through all the movies that user 1 have rated
    print ("similar for" + user1.index[i] + ".....")
    #creating a correlation with user1 and dropping null values
    sims= df[user1.index[i]].dropna()
    #scaling the similiter by how user1 had rated the movies
    #gives more similar movie for high ratings vice-versa
    sims= sims.map(lambda x:x*user1[i])
    #building the similar movies
    simCandidates = simCandidates.append(sims)
```

```
#sorting the results and printing them.
print("sorting....")
simCandidates.sort_values(inplace=True, ascending=False)


#printing the suggested movies to the user1 along with the movies that are rated by user1
#with this there is a chance of multiple duplicate movies in the suggested movies
#it is caused by( consider there are A,B,C,D,E. movies if user1 as rated movie A=5 and B=5)
#if the movie C is correlated related to both A and B movies then the movie C is suggested 2 times
#inorder to solve this we should add the duplicate movies to get the valid result
simCandidates.sort_values(inplace = True, ascending= False)


#printing the suggested movies for user1 by removing the movies which are rated by the user1
newdf= simCandidates.drop(user1.index)


#adding multiple movies similarity in a new file
simCandidates_new = newdf.groupby(newdf.index).sum()


#sorting the suggested movie list by decending order
#displaying the top 10 suggested movies for the user1 based on his previous movies ratings.
simCandidates_new.sort_values(inplace = True, ascending=False)


#convertind the simCandidates_new pd series file to dataframe
#and renaming the column to Similarity
#the required top 10 suggested movies for user1 based on his previous movies ratings.
finaldf=.to_frame()
newfinal.resimCandidates_new
newfinal=finaldfname(columns={0:"Similarity"}).head(10)
```

Output:

	Similarity
Ferris Bueller's Day Off (1986)	268.633323
Gremlins (1984)	252.530342
Mars Attacks! (1996)	247.042148
Escape from New York (1981)	246.939713
Breakfast Club, The (1985)	238.098542
Animal House (1978)	236.546920
Fifth Element, The (1997)	236.071226
Untouchables, The (1987)	235.496581
Jaws (1975)	230.952061
Splash (1984)	230.438345

Most frequently bought items:

- The items which are purchased more no of times in all the transactions is called frequently bought items.
- With this the e-commerce businesses will have an idea about the products which are bought more no of times so that they can improve their selling's.

Example:

Customer	Item 1	Item 2	Item 3
1	A	C	D
2	B	E	A
3	A	D	C

- Support = Count of an item.
- Frequently bought item is A i.e. support(A) = 3.

Code:

```
import pandas as pd
df=pd.read_csv('groceries.csv',header=None)

#no of items that are repeated in each column
dfu=df.apply(pd.value_counts).fillna(0)


#adding all columns to new column Total
dfu['total']=dfu[0]+dfu[1]+dfu[2]+dfu[3]+dfu[4]+dfu[5]+dfu[6]+dfu[7]+dfu[8]+dfu[9]+dfu[10]+dfu[11]+dfu[12]+dfu[13]+dfu[1
4]+dfu[15]+dfu[16]+dfu[17]+dfu[18]+dfu[19]+dfu[20]+dfu[21]+dfu[22]+dfu[23]+dfu[24]+dfu[25]+dfu[26]+dfu[27]+dfu[28]+df
u[29]+dfu[30]+dfu[31]

#removing all columns except total column
dfu.drop([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31], axis=1, inplace=True)

#sorting the values in the total column in desceding order.
dfu.sort_values(['total'],ascending=0,inplace=True)
dfu.head(10)
```

Output:

	0	total
whole milk	717.0	2513.0
other vegetables	460.0	1903.0
rolls/buns	336.0	1809.0
soda	294.0	1715.0
yogurt	225.0	1372.0
bottled water	170.0	1087.0
root vegetables	288.0	1072.0
tropical fruit	482.0	1032.0
shopping bags	48.0	969.0
sausage	825.0	924.0

Frequent two Itemset:

- The two items which are bought in the transaction is called two itemset. The total no of occurrences of the same two itemset in dataset is called frequency of the itemset.

Customer	Item 1	Item 2	Item 3	Item 4
1	A	C	E	B
2	F	A	C	H
3	A	G	D	C

- Frequently bought two item set is item A and Item C, having support (A,C) = 3.

Frequent three Itemset:

- The Three items which are bought in the transaction is called three itemset. The total no of occurrences of the same three itemset in the dataset is called frequency of the itemset.

Customer	Item 1	Item 2	Item 3	Item 4	Item 5
1	A	D	B	C	E
2	D	A	F	B	C
3	F	E	A	B	C
4	A	B	D	C	E

- Frequently bought two item set is item A and Item C, having support (A,B,C) = 4.

Frequent two Itemset

Code:

```
import pandas as pd
df=pd.read_csv('groceries.csv',header=None)


df1=df.fillna(0)
df1


#removing duplicates and getting unique transactions
dfv=df.drop_duplicates()


#merging columns to a single column Transactions with coma
dfv['transactions'] =
dfv[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]].apply(lambda x:
','.join(x[x.notnull()]), axis=1)


#deleted all the coloums and merged into a singe column transaction
dfv.drop([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31], axis=1, inplace=True)



#most common top 10 2 items brought together
from itertools import combinations
from collections import Counter


count=Counter()


for row in dfv['transactions']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list,2)))

count.most_common(10)
```

Output:

```
[(('other vegetables', 'whole milk'), 726),
 (('whole milk', 'yogurt'), 539),
 (('whole milk', 'rolls/buns'), 532),
 (('root vegetables', 'whole milk'), 478),
 (('root vegetables', 'other vegetables'), 459),
 (('other vegetables', 'yogurt'), 419),
 (('other vegetables', 'rolls/buns'), 410),
 (('tropical fruit', 'whole milk'), 407),
 (('whole milk', 'soda'), 387),
 (('tropical fruit', 'other vegetables'), 347)]
```

Frequent three itemset

Code:

```
import pandas as pd
df=pd.read_csv('groceries.csv',header=None)


df1=df.fillna(0)
df1


#removing duplicates and getting unique transactions
dfv=df.drop_duplicates()


#merging columns to a single column Transactions with coma
dfv['transactions'] =
dfv[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]].apply(lambda x:
','.join(x[x.notnull()]), axis=1)


#deleted all the coloums and merged into a singe column transaction
dfv.drop([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31], axis=1, inplace=True)



#most common top 10 3 items brought together
from itertools import combinations
from collections import Counter


count=Counter()


for row in dfv['transactions']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list,3)))

count.most_common(10)
```

Output:

```
('root vegetables', 'other vegetables', 'whole milk') 228
('other vegetables', 'whole milk', 'yogurt') 218
('other vegetables', 'whole milk', 'rolls/buns') 176
('tropical fruit', 'other vegetables', 'whole milk') 168
('whole milk', 'yogurt', 'rolls/buns') 153
('tropical fruit', 'whole milk', 'yogurt') 148
('other vegetables', 'whole milk', 'whipped/sour cream') 143
('root vegetables', 'whole milk', 'yogurt') 143
('other vegetables', 'whole milk', 'soda') 137
('pip fruit', 'other vegetables', 'whole milk') 133
```

Apriori Algorithm:

- This approach is used to suggest the items based on the items that are already purchased.

The major components of Apriori algorithm:

- Support S:
  It is the popularity of an item in the dataset.

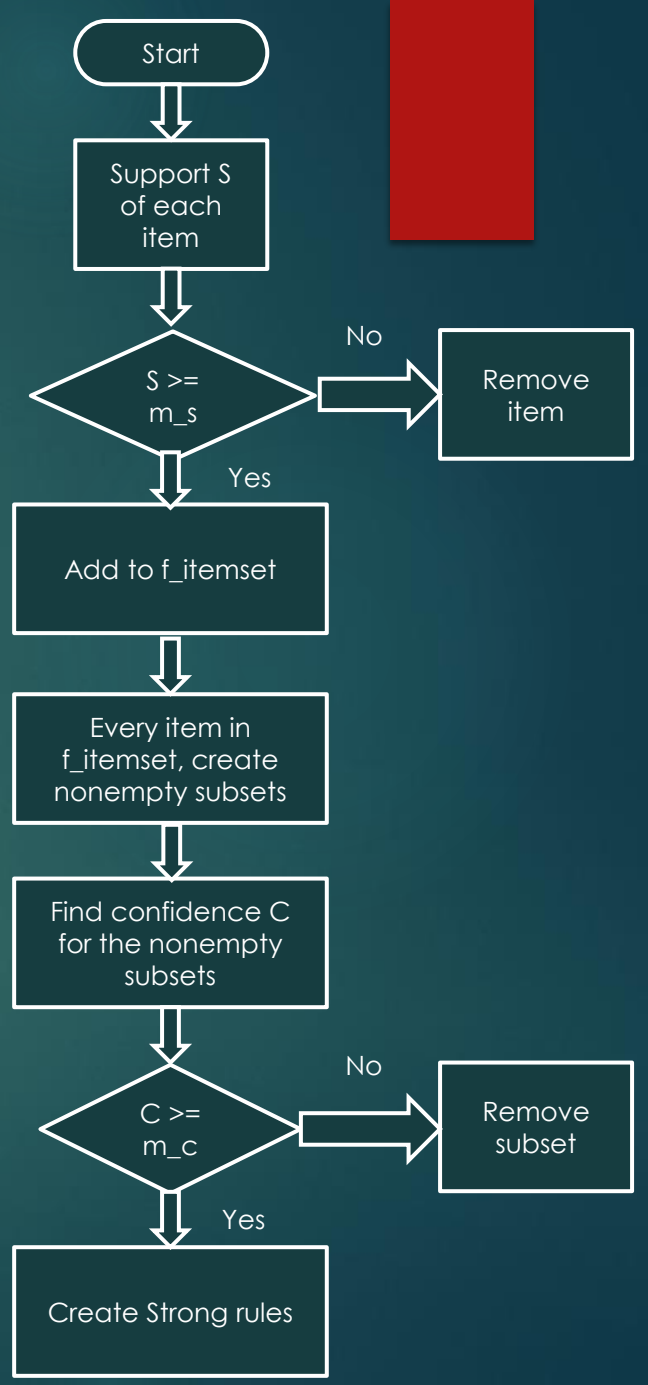
$$S = \frac{\text{Frequency}(X, Y)}{\text{Total transactions}}$$

- Confidence C:
  It signifies the likelihood than an item to be bought with another item.

$$C = \frac{\text{Frequency}(X, Y)}{\text{Frequency}(X)}$$

In flow chart,

- m_s = minimum Support
- f_itemset = frequent itemset
- m_c = minimum Confidence



Apriori Algorithm

Code:

```
import pandas as pd
df=pd.read_csv('groceries.csv',header=None)

#Removing duplicate transactions.
df=df.drop_duplicates()

leng = len(df)

#Creating a list of all transactions.
records = []
for i in range(0, leng):
    records.append([str(df.values[i,j]) for j in range(0, 20)])

#Implementing apriori algorithm.
from apyori import apriori

association_rules = apriori(records, min_support=0.0053, min_confidence=0.20, min_lift=3, min_length=2)
association_results = list(association_rules)

results = []
for item in association_results:

    pair = item[0]
    items = [x for x in pair]

    value0 = str(items[0])
    value1 = str(items[1])
    value2 = str(item[1])[:7]
    value3 = str(item[2][0][2])[:7]
    value4 = str(item[2][0][3])[:7]

    rows = (value0, value1,value2,value3,value4)
    results.append(rows)

labels = ['item 1','item 2','Support','Confidence','Lift']
df1 = pd.DataFrame.from_records(results, columns = labels)

df1.sort_values(['Confidence'],ascending=0,inplace=True)
df1.head(10)
```

Output:

	item 1	item 2	Support	Confidence	Lift
4	soda	beef	0.00542	0.48101	3.28052
3	citrus fruit	beef	0.00542	0.45783	3.12242
15	whole milk	butter	0.00556	0.39	3.76623
18	yogurt	tropical fruit	0.00556	0.39	5.07289
5	whole milk	berries	0.00584	0.35652	3.78723
13	nan	whole milk	0.00570	0.34782	3.79845
24	tropical fruit	root vegetables	0.00984	0.33333	3.21900
17	citrus fruit	root vegetables	0.00813	0.32758	3.16350
26	tropical fruit	whole milk	0.00627	0.32352	3.12433
19	whole milk	domestic eggs	0.00627	0.31654	3.05689