1/12/2019

**Association Rule Mining on Transaction History of a Debit Card**

ACM373 Scripting Languages Project
2018 Fall Semester / Yeditepe University

Mehmet Ali Özer

## SUMMARY OF DATA

The current account summary of my personal bank account. It has 3700 transaction from 2013 to end of 2018. I exhibit the example set of real data below. I will not share rest of raw data.
Simply, it has **transaction_date**, **channel**, **description**, t_**amount** and **balance**.

I will use **transaction_date** which is necessary to seperate transaction between unique days. description in order to mark places. In this project I actually need this column because I try to understand the association between descriptions which express my preferred expenses.

**t_amount** will present to us whether that is transaction as a spending record or that is transaction as an adding money to the account.

Channel and Balance,transaction will not be used.

| *t_id* | *transaction_date* | *transaction* | *channel* | *description* | *t_amount* | *balance* |
|---|---|---|---|---|---|---|
| *0* | 24.12.2018 | Diğer | Internet - Mobil | INT 450634******0347 2412 1434 | -154,06 TL | xx,67 TL |
| *1* | 24.12.2018 | Diğer | Diğer | POS MANDALIN CAFE RESTOR 3740 2412 | -30,00 TL | xx,73 TL |
| *2* | 24.12.2018 | Diğer | Internet - Mobil | INT 450634******0347 2312 1706 | -316,94 TL | xx,73 TL |
| *3* | 21.12.2018 | Diğer | Diğer | POS KASIMOGLU GIDA UNLU 3740 2112 | -21,50 TL | xx,67 TL |
| *4* | 21.12.2018 | Diğer | Diğer | POS SARRAC TIC.TEKEL GID 3740 2112 | -13,00 TL | xx,17 TL |

## HOW 1 FORGE DATA

Yapıkredi gave me the list of transaction as pdf. I copy whole data and write column name myself, each column was seperated by TAB character. I simply replace all Tab character with
" | " because it is less risky than Tab character while parsing.
I could not use comma " , " because **t_amount** column has already used it.

I changed description info with my keywords as ALPHA, BEITA, E01TA... for personal privacy.

I used notepad++ editor macros and some plugin to handle it.

I also did some of the filtering process with python, deleting ' TL' characters, remove dots and replace with commas bacause 1.234,50 does not mean any numeric value in python.
It waits a dot for decimal side like 1234.50

```
path = "files/account_summary.csv"
D = pd.read_csv(path, sep="|");

#data preprocessing
D['t_amount'] = D['t_amount'].str.replace(' TL','')
D['t_amount'] = D['t_amount'].str.replace('.','')
D['t_amount'] = D['t_amount'].str.replace(',','.')
D['t_amount'] = D['t_amount'].astype('float')

#filter positive t_amounts because they are not expense
D = D[D['t_amount'] < 0]
D.to_csv("files/spend_item.csv")
```

- I only get transactions with negative ammount because I assume if you prefer an item, you get something, so you have to give something else.

After use **t_amount** as an identifier on transaction that imply expenses, I remove the column because both privacy reasons and omission of ***amount-weight***, in this project I just interested in occurence.

## THE AIM AND APPROACH OF THE PROJECT

I will analyze association on expenditures on my 6-year bank transaction.

If we assume one transaction **description** is a ***place***, and a ***place*** can be accepted as an ***item*** is preferred.

- In this paper, I will use ***place*** and ***item*** words interchangeably.

The aim of the project is uncovering relationships between the places, and detecting the association rules of 2-itemsets;
***placeA -> placeB*** when placeA is preferred, what is the degree of preferability of placeB.

and vice versa,

***placeB -> placeA*** when placeB is preferred, what is the degree of preferability of placeA.

Then we need some cluster to understand togetherness.
For example, Is the **placeA** which is preferred in 12.05.2013, together with **placeB** which is preferred in 15.06.2013 ?

Creation of basket with data like that is changeable to our desire and what we want to see.
The group can be semesters, like for Winters 12th, 1st, 2nd months - Spring - Summer and Fall for

each year. We could try to find most preferred 2-itemset in winter, spring...
Even if the product names existed, a clearer analysis could be done.

The another approach would be grouping working days and weekends etc.

Because of my lifestyle, I simply assume each day as a basket, each day I had 24 hours to purchase something together. When I prefer to eat something in a restaurant then I may choose to drink something in another coffee shop or I can order a pizza then purchase a beer.
Next day, however, is too far to choose new place (item) is dependent to the place choosen one day before.

Based on this proposition "If items that are frequently exist together, they may be associated with each other", I used the Apriori algorithm which help me to count the itemsets progressively.

## APRıORı ALGORıTHM AND USAGE

The approach of Apriori algorithm is that get 1-itemset L1 and count frequency for each item on the transaction records (baskets). The candidate itemset Candidate-1 whose size is higher by 1 (support_count) and also determine the minimum support, if items exceed min support threshold, they are called Candidate-1. Then derive 2-k itemset with binary combination of remainig itemlist, called join step, from Candidate-1.

e.g. Assume min_sup = 0.40 and the transactions like that;

L1 = {A,B,C,D,E}

| | |
|---|---|
| basket01 | **{A,B}** |
| basket02 | **{A,C}** |
| basket03 | **{A,D}** |
| basket04 | **{A,B,C }** |
| basket05 | **{A,B}** |
| basket06 | **{C,D,E }** |

| item_name | support_count<br>freq (item) | Support - freq(item) / total transactions |
|---|---|---|
| **placeA** | 5 (times preferred) | 5/6 (83.3%) |
| **place B** | 3 | 3/6 (50%) |
| **place C** | 3 | 3/6 (50%) |
| **place D** | 2 | 2/6 (33.3%) |
| **placeE** | 1 | 1/6 (16.6%) |

Candidate-1 = {A,B,C}

**placeE** is pruned because it does not supply frequency. It only occurs 1 times, we cannot use it to make for 2k-itemset because it is not frequent. There are no other basket has pair with placeE.

Although **placeD** pass the frequency threshold, it is caught when we look at support, when we determine min_support threshold as **40%**.

L2 = { {A,B}, {A,C}, {B,C} }

Then if any 2-k itemset has exceeded min frequency and support threshold, the item-pairs are Candidate-2 then add 3$^{rd}$ item in join step, get frequency and repeat as soon as there is no itemset provide minimum frequency and support value.

As a note, I do not use Candidate-2 itemset to make 3-itemset, because I only want to see what is the relationship of preferability between two places.

Three association rules for understanding relationship;

**1) Support**: the ratio of how many basket has the itempair. freq(k-itemset) / total basket number

**2) Confidence**: It is the association rule;
place1 -> place2 is the ratio of the place2 was preferred when place1 was already preferred. [3]
They develop this association rule with use bayes theorem.

The probability of two events A and B happening, $P(A \cap B)$, is the probability of A, $P(A)$, times the probability of B given that A has occurred, $P(B|A)$. [4]

$$\text{if, } P(A \cap B) = P(A)P(B|A)$$

$$\text{then } P(B|A) = P(A \cap B) / P(A)$$

Therefore the rules of P(place2 | place1) given place1 conditions
the probability of preferability of place2 is **support( {place1, place2} ) / support ({place1})**

$$\text{place1} \rightarrow \text{place2} = \text{support}(\{\text{place1}, \text{place2}\}) / \text{support}(\text{place1})$$

**3) Lift**: the ratio of the observed support to that expected if X and Y were independent.

For first, $\mathbf{Lift_{place1>place2}} = $ **confidence / support(place2)**

If confidence = support({place1, place2}) / support(place1)

Then $Lift_{place1>place2}$ = [support({place1, place2}) / support(place1)] / support(place2)

**$Lift_{place1>place2}$ = support({place1, place2}) / [support(place1) * support(place2)]**

For second, $Lift_{place2>place1}$ = confidence / support(place1)

If confidence = support({place1, place2}) / support(place2)

Then $Lift_{place2>place1}$ = [support({place1, place2}) / support(place2)] / support(place1)

$Lift_{place2>place1}$ = support({place1, place2}) / [support(place2) * support(place1)]

The equality $Lift_{place1>place2}$ = $Lift_{place2>place1}$ shows that there is no effect on lift the direction of places. Confidence, however, lack of this feature.

## PYTHON APPLICATION

```
#data_read and frame
import pandas as pd

#calculation
from itertools import combinations, groupby
from collections import Counter

#visualization
import seaborn as sns
from IPython.display import display
```

```
#frequency function
def freq(objSeries):
    #pd series already have value_counts() method
    if type(objSeries) == pd.core.series.Series:
        return objSeries.value_counts()
    #pair_items is a list, freq'll be counted by counter and returned a series
    else:
        return pd.Series(Counter(objSeries))
```

```
#combinator function
def get_pairs(ts):
    ts = ts.reset_index().values
    #get 2-k itemsets
    twokitemset = list()
    for t_date, basket in groupby(ts, lambda x: x[0]):
        item_list = set()
        for item in basket:
            item_list.add(item[1])

      #because of combinator returns a tuple for each combinations,
      mirror effect is serious problem, (A,B) and (B,A) generate
      different frequency.
      #e.g. ('EPSILON','DELTA') occurs 56 times, ('DELTA','EPSILON')
      occurs 17 times but actual count is that set of {'EPSILON', 'DELTA'}
occurs 73 times. Seperation in frequency cannot be accepted.
      #Although they are not different itemset, different values
      generated for each sight.
      #so we need alphabetical sorting command, it will work because
      combinator just follows the place of the item, not item itself.
        for pairs in combinations(sorted(item_list), 2):
            twokitemset.append(pairs)
    return twokitemset
```

```
path = "files/spend_item.csv"
df = pd.read_csv(path, sep=",");

#conversion dates column string to date
df['t_date'] = pd.to_datetime(df['t_date'])

#the adverse conditions on my bank-account_transaction dataset may contain
same item(place) more than one in same day.
#it means same basket can contain twice or more times
the same preferred item
#if we think that a basket is like a set, it should contain one
element only once
#because of its effect on frequency, in this scenario,
preferability-weight of a place in a same timestamp is omitted.

#I interested on that it is just preferred on that day or not,
so duplicates are dropped.

#if t_date, item_code pairs are duplicated, drop the duplicated items
except first item, to get first item is inevitable.
#e.g. a day may contain 3 times DELTA, it gets first then drops other two.
df = df.drop_duplicates(keep="first")

#dataframe to series conversion
#now we can use date as index
tseries = df.set_index('t_date')['item_code']

day_size = freq(tseries.index).rename("preferred_loc")
print("Total basket number is : {0} ".format(len(day_size)))

#get timestamp of the days which have minimum 2 preferred place.
```

```
the_days_we_care = day_size[day_size > 1].index
#pandas select column by condition
tseries = tseries[tseries.index.isin(the_days_we_care)]

print("The eliminated 1-item basket number is : {0}
".format(len(day_size)-len(set(tseries.index))))

print("Remaining basket Number is : {0}".format(len(set(tseries.index))))
item_freq = freq(tseries).rename("occurence")

#get item support bigger than 5

#5 item support threshold means the item preferred at least 5% in
remaining day size

items_we_care = item_freq[(item_freq/len(set(tseries.index))) * 100 > 5].index

print("Inspected item size : {0}".format(len(items_we_care)))
#playing with index to filter tseries with items_we_care
tseries = tseries.reset_index()
tseries = tseries.set_index("item_code")

tseries = tseries[tseries.index.isin(items_we_care)]

#make index as date again
tseries = tseries.reset_index()
tseries = tseries.set_index("t_date")["item_code"]

item_info = freq(tseries).to_frame("occurence")
item_info['support'] = (item_info["occurence"] / len(set(tseries.index)))*100

print("Remaining basket number after elimination of items which could not pass
threshold : {0} ".format(len(day_size)-len(set(tseries.index))))

#produce 2-itemsets
pair_items = get_pairs(tseries)

pairs_info = freq(pair_items).to_frame("occurence_pairs")
pairs_info['support_pairs'] = pairs_info['occurence_pairs'] /
len(set(tseries))

# 1 means the 2-itemsets preferred at least 20 times
pairs_info = pairs_info[pairs_info['support_pairs'] >= 1]

#seperate 2-itemsets as column for each item
pairs_info = pairs_info.reset_index().rename(columns={'level_0': 'place1',
'level_1': 'place2'})

#merge data frames like sql join statement
pairs_info =
pairs_info.merge(item_info.rename(columns={'occurence':'sup_countLeft','sup-
port':'supportLeft'}), left_on="place1", right_index=True)

pairs_info =
pairs_info.merge(item_info.rename(columns={'occurence':'sup_countRight','sup-
port':'supportRight'}), left_on="place2", right_index=True)
```

```
#confidence(A->B) and confidence(B->A) calculations
#confidence(A->B) supportAB / supportA, if we know the preferability of place
A and we want to see what is the possibility to prefer B.
#for vice versa, confidence(B->A)supportAB / supportB
pairs_info['confP1->P2'] = pairs_info['support_pairs'] /
pairs_info['supportLeft']
pairs_info['confP2->P1'] = pairs_info['support_pairs'] /
pairs_info['supportRight']

#lift calcuation
pairs_info['lift'] = pairs_info['support_pairs'] / (pairs_info['supportLeft']
* pairs_info['supportRight'])

#visualisation of report | desc order by lift
cm = sns.light_palette("orange", as_cmap=True)
display(pairs_info.sort_values(by=['lift'],
ascending=False).style.background_gradient(cmap=cm))

#export as excel | desc order by lift
writer = pd.ExcelWriter('apriori_report.xlsx')

pairs_info.sort_values(by=['lift'],
ascending=False).style.background_gradient(cmap=cm).to_excel(writer, 'Sheet1')

writer.save()
```

OUTPUT:

```
Total basket number is : 955
The eliminated 1-item basket number is : 229
Remaining basket Number is : 726
Inspected item size : 19
Remaining basket number after elimination of items which could not pass
threshold : 234
```

| | place1 | place2 | occurence_pairs | support_pairs | sup_countLeft | supportLeft | sup_countRight | supportRight | conf P1->P2 | conf P2->P1 | lift |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **60** | LAMBDA | PIO1 | 23 | 1.21053 | 77 | 10.6796 | 108 | 14.9792 | 0.113349 | 0.0808138 | 0.00756711 |
| **64** | DELTA | MEMORY | 32 | 1.68421 | 347 | 48.1276 | 38 | 5.27046 | 0.0349947 | 0.319557 | 0.00663978 |

| | place1 | place2 | occurence_pairs | support_pairs | sup_countLeft | supportLeft | sup_countRight | supportRight | conf P1->P2 | conf P2->P1 | lift |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **8** | ALPHA | RHO | 19 | 1 | 132 | 18.3079 | 64 | 8.87656 | 0.0546212 | 0.112656 | 0.00615342 |
| **26** | BLACK AND WHITE | EITA01 | 27 | 1.42105 | 170 | 23.5784 | 74 | 10.2635 | 0.0602693 | 0.138457 | 0.00587219 |
| **54** | PIO1 | TELTA | 52 | 2.73684 | 108 | 14.9792 | 225 | 31.2067 | 0.18271 | 0.0877006 | 0.00585483 |
| **62** | GAMMA | PIO1 | 20 | 1.05263 | 88 | 12.2053 | 108 | 14.9792 | 0.086244 | 0.0702729 | 0.00575759 |
| **61** | LAMBDA | TELTA | 36 | 1.89474 | 77 | 10.6796 | 225 | 31.2067 | 0.177416 | 0.0607158 | 0.00568521 |
| **28** | IOTA | OMICRON | 52 | 2.73684 | 144 | 19.9723 | 176 | 24.4105 | 0.137032 | 0.112117 | 0.00561365 |
| **51** | OMICRON | PIO1 | 38 | 2 | 176 | 24.4105 | 108 | 14.9792 | 0.0819318 | 0.133519 | 0.00546971 |
| **57** | BLACK AND WHITE | LAMBDA | 26 | 1.36842 | 170 | 23.5784 | 77 | 10.6796 | 0.0580372 | 0.128134 | 0.00543439 |

EXCEL OUTPUT:

| | A | place1 | place2 | occurence_pairs | support_pairs | sup_countLeft | supportLeft | sup_countRight | supportRight | confP1->P2 | confP2->P1 | lift |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 60 | LAMBDA | PIO1 | 23 | 1.210526316 | 77 | 10.67961165 | 108 | 14.97919556 | 0.113349282 | 0.08081384 | 0.007567 |
| 3 | 64 | DELTA | MEMORY | 32 | 1.684210526 | 347 | 48.12760055 | 38 | 5.270457698 | 0.034994691 | 0.319556787 | 0.00664 |
| 4 | 8 | ALPHA | RHO | 19 | 1 | 132 | 18.30790569 | 64 | 8.876560333 | 0.054621212 | 0.11265625 | 0.006153 |
| 5 | 26 | BLACK AND WHITE | EITA01 | 27 | 1.421052632 | 170 | 23.57836338 | 74 | 10.26352288 | 0.06026935 | 0.138456615 | 0.005872 |
| 6 | 54 | PIO1 | TELTA | 52 | 2.736842105 | 108 | 14.97919556 | 225 | 31.20665742 | 0.182709552 | 0.087700585 | 0.005855 |
| 7 | 62 | GAMMA | PIO1 | 20 | 1.052631579 | 88 | 12.20527046 | 108 | 14.97919556 | 0.086244019 | 0.070272904 | 0.005758 |
| 8 | 61 | LAMBDA | TELTA | 36 | 1.894736842 | 77 | 10.67961165 | 225 | 31.20665742 | 0.177416268 | 0.060715789 | 0.005685 |
| 9 | 28 | IOTA | OMICRON | 52 | 2.736842105 | 144 | 19.97226075 | 176 | 24.41054092 | 0.137032164 | 0.112117225 | 0.005614 |
| 10 | 51 | OMICRON | PIO1 | 38 | 2 | 176 | 24.41054092 | 108 | 14.97919556 | 0.081931818 | 0.133518519 | 0.00547 |
| 11 | 57 | BLACK AND WHITE | LAMBDA | 26 | 1.368421053 | 170 | 23.57836338 | 77 | 10.67961165 | 0.058037152 | 0.128133971 | 0.005434 |
| 12 | 58 | LAMBDA | ZEILTA | 34 | 1.789473684 | 77 | 10.67961165 | 226 | 31.34535368 | 0.167559809 | 0.057088961 | 0.005346 |
| 13 | 6 | ALPHA | EITA01 | 19 | 1 | 132 | 18.30790569 | 74 | 10.26352288 | 0.054621212 | 0.097432432 | 0.005322 |
| 14 | 50 | PIO1 | ZEILTA | 47 | 2.473684211 | 108 | 14.97919556 | 226 | 31.34535368 | 0.165141326 | 0.078917094 | 0.005268 |
| 15 | 18 | EPSILON | GAMMA | 21 | 1.105263158 | 127 | 17.61442441 | 88 | 12.20527046 | 0.062747617 | 0.09055622 | 0.005141 |
| 16 | 63 | ALPHA | LAMBDA | 19 | 1 | 132 | 18.30790569 | 77 | 10.67961165 | 0.054621212 | 0.093636364 | 0.005115 |
| 17 | 48 | TELTA | ZEILTA | 95 | 5 | 225 | 31.20665742 | 226 | 31.34535368 | 0.160222222 | 0.159513274 | 0.005112 |
| 18 | 27 | EITA01 | OMICRON | 24 | 1.263157895 | 74 | 10.26352288 | 176 | 24.41054092 | 0.123072546 | 0.051746411 | 0.005042 |
| 19 | 23 | ALPHA | BLACK AND WHITE | 41 | 2.157894737 | 132 | 18.30790569 | 170 | 23.57836338 | 0.117866826 | 0.091520124 | 0.004999 |
| 20 | 32 | ALPHA | OMICRON | 42 | 2.210526316 | 132 | 18.30790569 | 176 | 24.41054092 | 0.120741627 | 0.09055622 | 0.004946 |
| 21 | 47 | BLACK AND WHITE | GAMMA | 27 | 1.421052632 | 170 | 23.57836338 | 88 | 12.20527046 | 0.06026935 | 0.116429426 | 0.004938 |
| 22 | 33 | BLACK AND WHITE | CAROUSEL | 22 | 1.157894737 | 170 | 23.57836338 | 72 | 9.986130374 | 0.049108359 | 0.115950292 | 0.004918 |
| 23 | 59 | LAMBDA | OMICRON | 24 | 1.263157895 | 77 | 10.67961165 | 176 | 24.41054092 | 0.118277512 | 0.051746411 | 0.004845 |
| 24 | 19 | EPSILON | IOTA | 31 | 1.631578947 | 127 | 17.61442441 | 144 | 19.97226075 | 0.092627435 | 0.081692251 | 0.004638 |
| 25 | 0 | EITA01 | TELTA | 28 | 1.473684211 | 74 | 10.26352288 | 225 | 31.20665742 | 0.143584637 | 0.047223392 | 0.004601 |
| 26 | 38 | EITA01 | ZEILTA | 28 | 1.473684211 | 74 | 10.26352288 | 226 | 31.34535368 | 0.143584637 | 0.047014439 | 0.004581 |
| 27 | 11 | DELTA | EPSILON | 73 | 3.842105263 | 347 | 48.12760055 | 127 | 17.61442441 | 0.07983164 | 0.218122669 | 0.004532 |
| 28 | 35 | ALPHA | ZEILTA | 49 | 2.578947368 | 132 | 18.30790569 | 226 | 31.34535368 | 0.140865231 | 0.082275268 | 0.004494 |
| 29 | 36 | BLACK AND WHITE | ZEILTA | 63 | 3.315789474 | 170 | 23.57836338 | 226 | 31.34535368 | 0.140628483 | 0.105782487 | 0.004486 |
| 30 | 39 | OMICRON | ZEILTA | 65 | 3.421052632 | 176 | 24.41054092 | 226 | 31.34535368 | 0.140146531 | 0.109140661 | 0.004471 |
| 31 | 45 | BLACK AND WHITE | PIO1 | 30 | 1.578947368 | 170 | 23.57836338 | 108 | 14.97919556 | 0.066965944 | 0.105409357 | 0.004471 |
| 32 | 9 | ALPHA | TELTA | 47 | 2.473684211 | 132 | 18.30790569 | 225 | 31.20665742 | 0.13511563 | 0.079267836 | 0.00433 |
| 33 | 12 | DELTA | GAMMA | 48 | 2.526315789 | 347 | 48.12760055 | 88 | 12.20527046 | 0.052492037 | 0.206985646 | 0.004301 |
| 34 | 44 | EPSILON | OMICRON | 35 | 1.842105263 | 127 | 17.61442441 | 176 | 24.41054092 | 0.104579362 | 0.075463517 | 0.004284 |
| 35 | 3 | DELTA | EITA01 | 40 | 2.105263158 | 347 | 48.12760055 | 74 | 10.26352288 | 0.043743364 | 0.20512091 | 0.004262 |
| 36 | 46 | DELTA | PIO1 | 58 | 3.052631579 | 347 | 48.12760055 | 108 | 14.97919556 | 0.063427878 | 0.203791423 | 0.004234 |
| 37 | 37 | DELTA | ZEILTA | 120 | 6.315789474 | 347 | 48.12760055 | 226 | 31.34535368 | 0.131230093 | 0.201490452 | 0.004187 |
| 38 | 34 | BLACK AND WHITE | EPSILON | 33 | 1.736842105 | 170 | 23.57836338 | 127 | 17.61442441 | 0.073662539 | 0.098603398 | 0.004182 |
| 39 | 1 | RHO | TELTA | 22 | 1.157894737 | 64 | 8.876560333 | 225 | 31.20665742 | 0.130444079 | 0.037104094 | 0.00418 |
| 40 | 43 | RHO | ZEILTA | 22 | 1.157894737 | 64 | 8.876560333 | 226 | 31.34535368 | 0.130444079 | 0.036939916 | 0.004162 |
| 41 | 5 | ALPHA | DELTA | 69 | 3.631578947 | 132 | 18.30790569 | 347 | 48.12760055 | 0.198361244 | 0.075457303 | 0.004122 |
| 42 | 49 | OMICRON | TELTA | 59 | 3.105263158 | 176 | 24.41054092 | 225 | 31.20665742 | 0.127209928 | 0.099506433 | 0.004076 |
| 43 | 16 | DELTA | SOLAR | 28 | 1.473684211 | 347 | 48.12760055 | 55 | 7.628294036 | 0.030620355 | 0.193186603 | 0.004014 |

# REFERENCES


1) http://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf
2) https://www.saedsayad.com/association_rules.htm
3) http://www.hep.upenn.edu/~johnda/Papers/Bayes.pdf
4) https://www.researchgate.net/publication/262325976_A_Bayesian_Association_Rule_Mining_Algorithm