به نام خدا

مهدی فقهی

تمرین سوم داده‌کاوی

Problem 1)

Trace the results of using the Apriori algorithm on the grocery store example with support threshold s=33.34% and confidence threshold c=60%. Show the candidate and frequent itemsets for each database scan. Enumerate all the final frequent itemsets. Also indicate the association rules that are generated and highlight the strong ones, sort them by confidence

Transaction ID	Items
T1	HotDogs, Buns, Ketchup
T2	HotDogs, Buns
T3	HotDogs, Coke, Chips
T4	Chips, Coke
T5	Chips, Ketchup
T6	HotDogs, Coke, Chips

.

سوال اول)

item	count	support percent
Hotdogs	4	4/6 = 0.66
Buns	2	2/6 = 0.33
Ketchup	2	2/6 = 0.33
Coke	3	3/6 = 0.5
Chips	4	4/6 = 0.66

item	count	support percent
Hotdogs,coke	2	2/6 = 0.33
Hot Dogs,chips	2	2/6 = 0.33
coke,chips	2	2/6 = 0.33

با این نوع حساب کتاب به قانونی نمی‌رسیم بیایم 0.34 در نظر بگیرم و به بالا گرد کنیم .

item	count	support percent
Hotdogs	4	4/6 = 0.66
Buns	2	2/6 = 0.34
Ketchup	2	2/6 = 0.34
Coke	3	3/6 = 0.5
Chips	4	4/6 = 0.66

Item	count	support percent
Hotdogs,Buns	2	0.34
Hotdogs,ketchup	1	0.16
Hotdogs,coke	2	0.34
Hotdogs,chips	2	0.34
ketchup,coke	0	0
ketchup,Buns	1	0.16
ketchup,chips	1	0.16

Buns,coke	0	0
Buns,chips	0	0
coke,chips	3	0.50

Item	count	support percent
Hotdogs,coke,chips	2	0.34

قوانین چهارتایی نیز نداریم .

حال confidence قوانین دوتایی را حساب می کنیم .

Hotdogs -> Buns (conf = 2/4 = 0.50)

Buns -> Hotdogs (conf = 2/2 = 1)

Hotdogs -> coke (conf = 2/4 = 0.50)

Coke -> Hotdogs (conf = 2/3 = 0.67)

Hotdogs -> Chips (conf = 2/4 = 0.5)

Chips -> Hotdogs (conf = 2/4 = 0.5)

cock ->chips (conf = 3/3 = 1)

chips -> cock (conf = 3/4 = 0.75)

حال confidence را برحسب قوانین سه‌تای حساب می کنیم .(Hotdogs,coke,chips)

Hotdogs -> coke,chips (conf = 2/4=0.5)_

coke -> Hotdogs,chips (conf = 2/3=0.67)

chips -> Hotdogs,coke (conf = 2/4=0.5)

Hotdogs,coke ->chips (conf = 2/2=1)

Hotdogs,chips -> coke (conf = 2/2=1)

coke,chips -> Hotdogs (conf = 2/3=0.67)

بعد از این کار مرتب می کنیم .

1.Buns -> Hotdogs (conf = 2/2 = 1)

2.cock ->chips (conf = 3/3 = 1)

3.Hotdogs,coke ->chips (conf = 2/2=1)

4.Hotdogs,chips -> coke (conf = 2/2=1)

5.chips -> cock (conf = 3/4 = 0.75)

6.Coke -> Hotdogs (conf = 2/3 = 0.67)

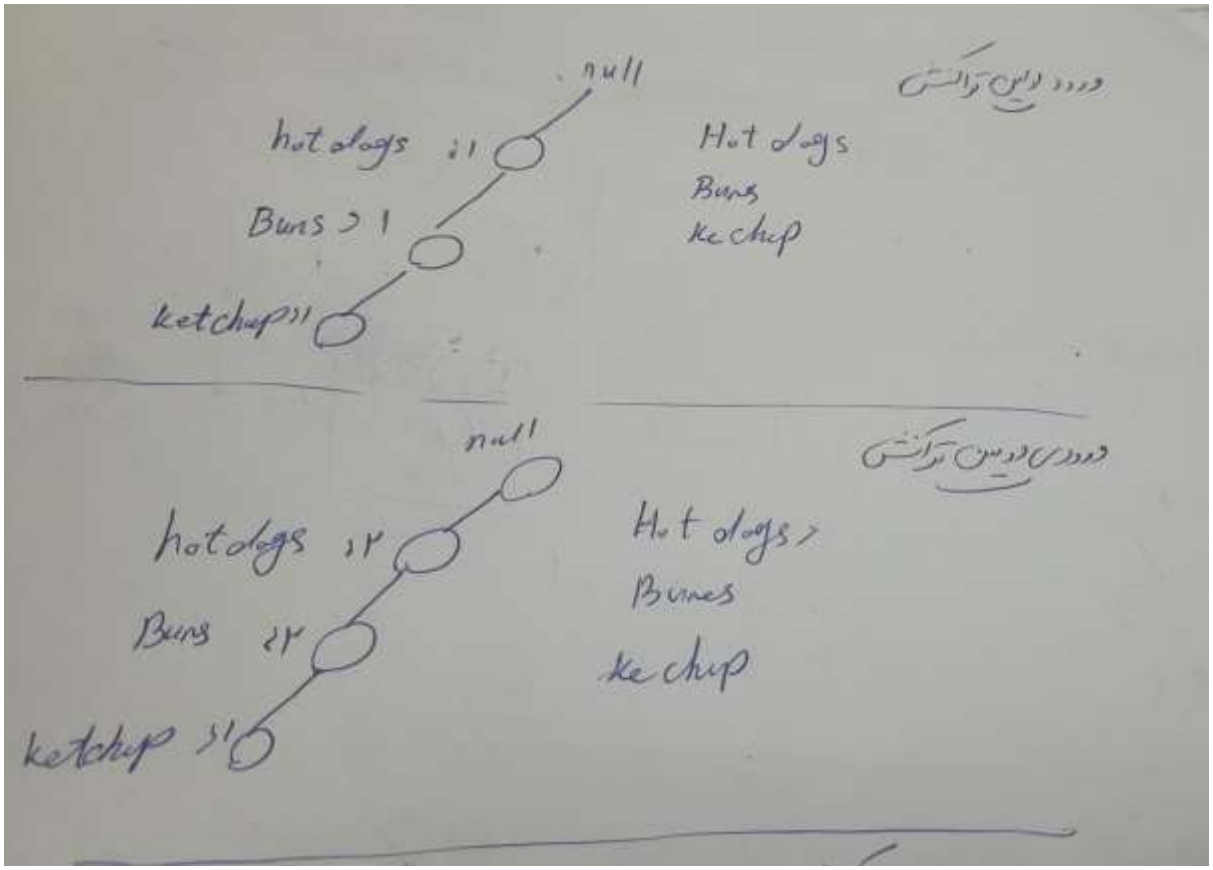
7.coke -> Hotdogs,chips (conf = 2/3=0.67)

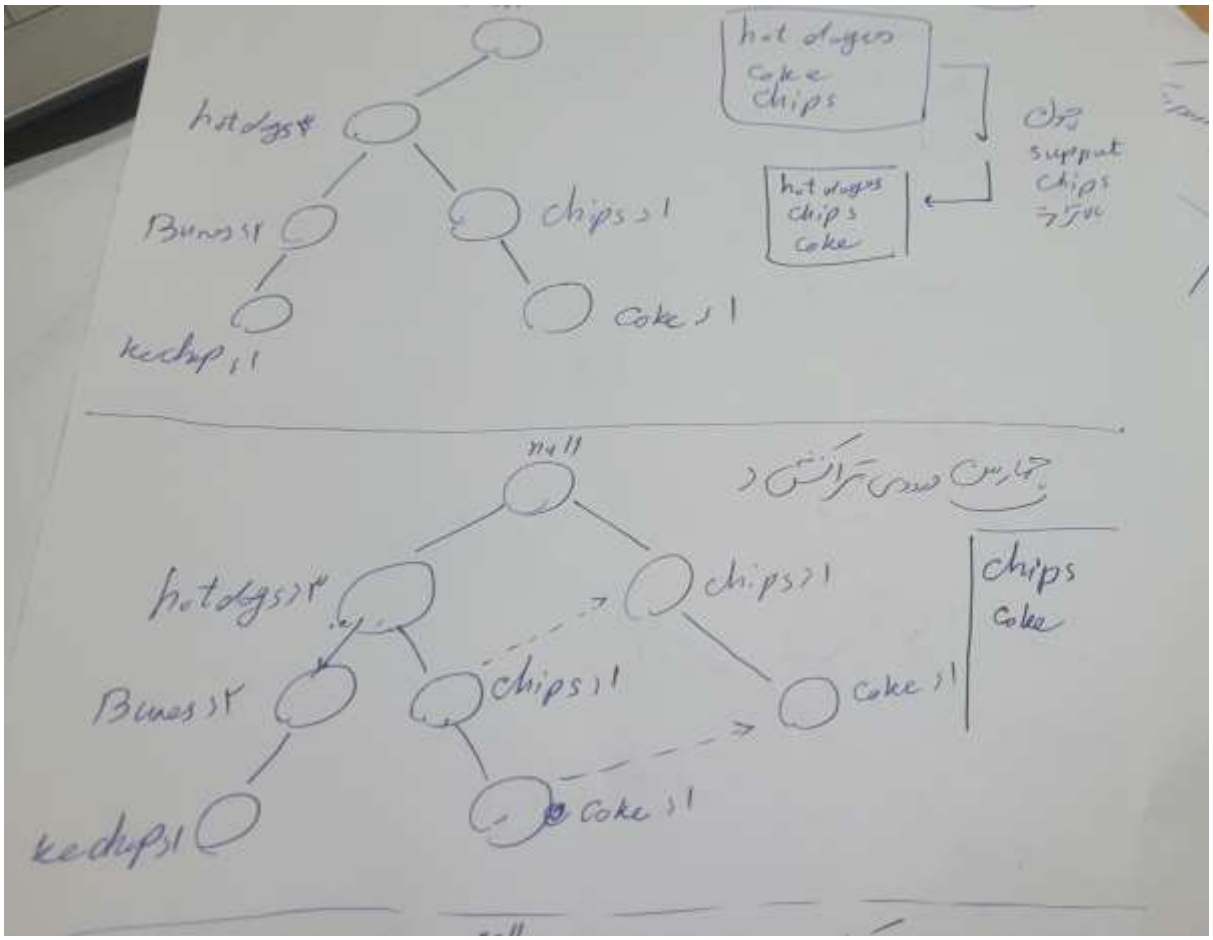
8.coke,chips -> Hotdogs (conf = 2/3=0.67)

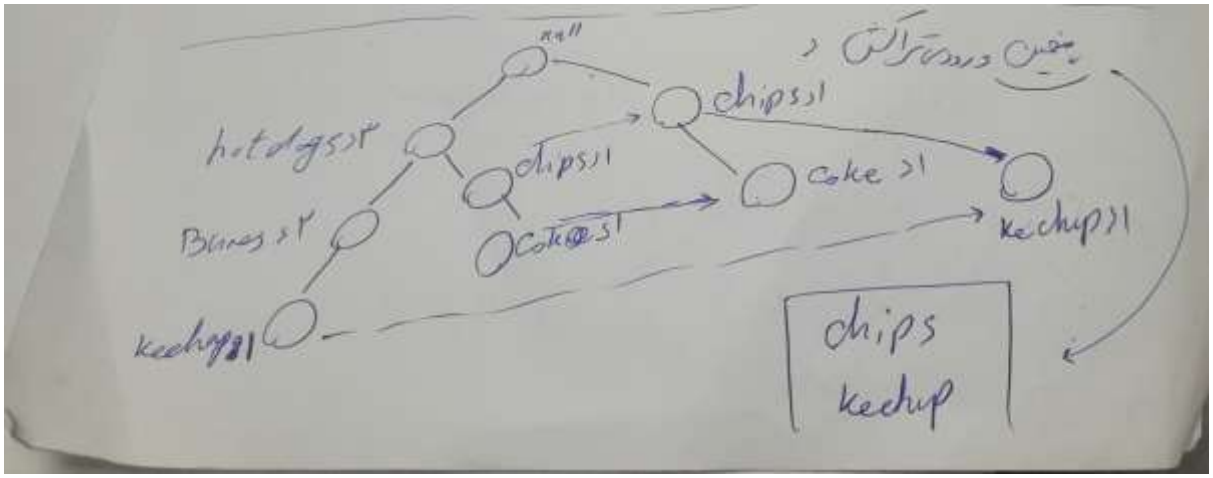
سوال دوم)

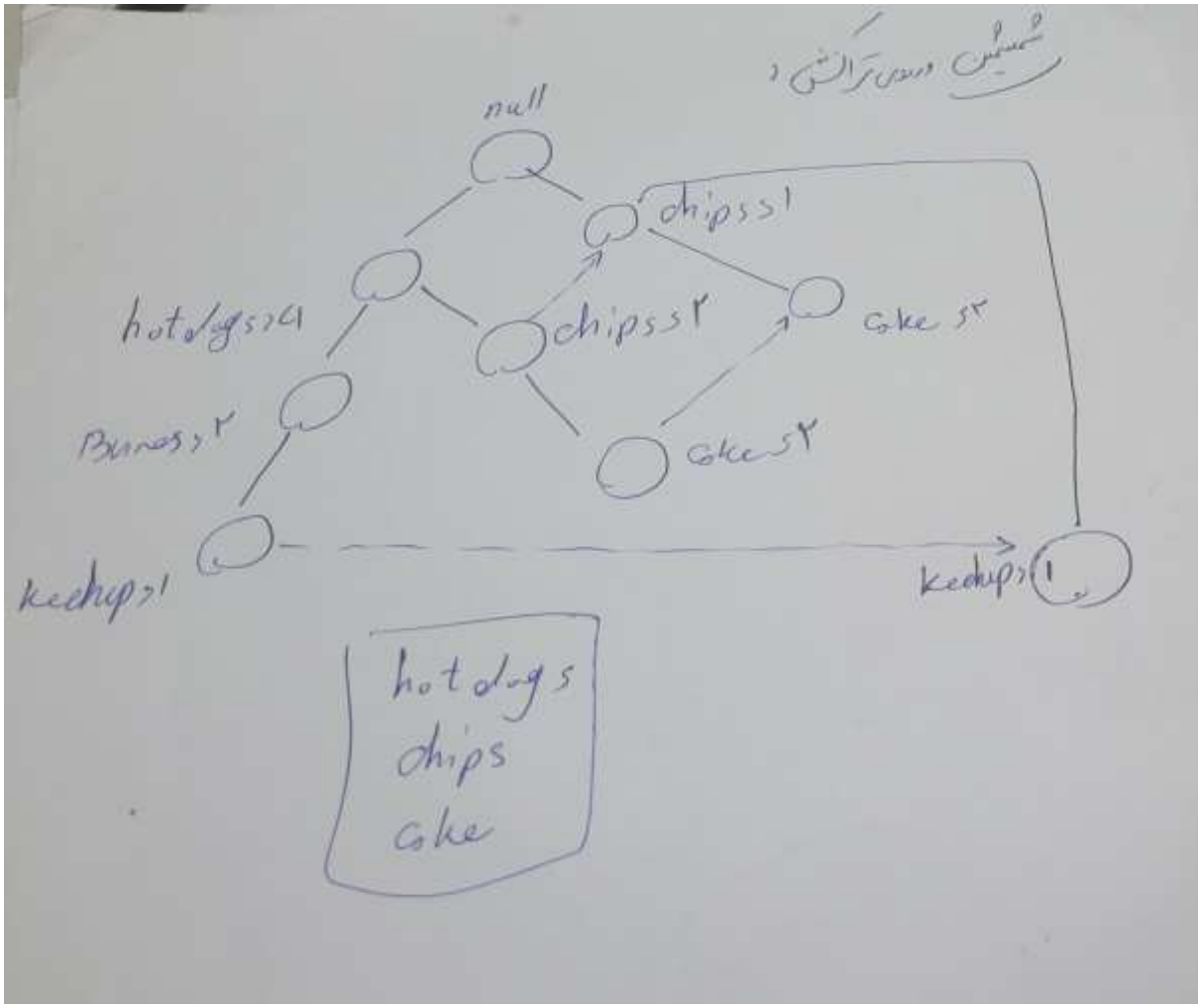
a) Use the transactional database from the previous exercise with same support threshold and build a frequent pattern tree (FP-Tree). Show for each transaction how the tree evolves

Item	support sort
Hotdogs	4
Chips	4
coke	3
Buns	2
Ketchup	2


null
hot dogs :1
Buns :1
ketchup :1
Hot dogs
Buns
ketchup
null
hotdogs :2
Buns :2
ketchup :1
Hot dogs
Buns
ketchup


hot dogs :2
Buns :2
chips :1
ketchup :1
Coke :1
hot dogs
coke
chips
hot dogs
chips
coke
null
hotdogs :2
chips :1
Buns :2
chips :1
coke :1
ketchup :1
Coke :1
chips
Coke





b) Use Fp-Growth to discover the frequent itemsets from this FP-tree.

ما باید یک conditional tree برای هر آیتم frequent بسازیم که از کمترین تعداد شروع می‌شود .

برای سس کچاپ، ما دو شاخه HotDogs-Buns-Ketchup و Chips-Koce داریم، اما از آنجایی که Ketchup دارای پشتیبانی 1 در هر شاخه است،این همه موارد را حذف می کند (زیرا آستانه پشتیبانی 2 است) فقط <2:Ketchup> باقی می ماند.

این منجر به کشف (2: Ketchup) به عنوان آیتم مکرر می شود.
برای (B) Buns، ما فقط یک شاخه HotDogs-Buns داریم. معامله فرعی {HotDogs, Buns} دو بار ظاهر می شود.
بنابراین الگوهای <HotDogs، 2:Buns:2> و <2:Buns> را داریم.
این منجر به کشف (HotDogs, Buns) (2) و (Buns) (2) به عنوان مجموعه موارد مکرر می شود.
برای (Coke)، ما دو شاخه داریم: HotDogs-Chips-Coke و Chips-Coke که منجر به درخت
HotDogs -> Chips(3) -> coke(3) (2 می شود.

بنابراین ما 3 الگو داریم: <Coke:3، Chips>، <HotDogs:2، Chips:2، 3:Coke:2> و <3:Coke>. این منجر به کشف مجموعه‌های اقلام مکرر زیر می‌شود: (Coke, Chips) (2)، (Coke, Chips, HotDogs) (3) و (Coke) (3).
برای تراشه ها (Chips)، دو مسیر HotDogs-Chips و Chips داریم که درخت زیر را نشان می دهد H->(4)Chips(2).
این می دهد الگوهای <HotDogs، 2:Chips:2> و <4:Chips>. بنابراین، مجموعه اقلام (Chips, HotDogs) (2) و (Chips) (4) مکرر هستند.
پس آیتم‌های مکرر یا همان frequent برابر است با :

HotDogs}, {Buns}, {Ketchup}, {Coke}, {Chips}

{HotDogs, Buns}, {HotDogs, Coke}, {HotDogs, Chips}, {Coke, Chips}, {HotDogs, Coke, Chips}

سوال سوم)

In this section we have a paper Model Ensemble for Click Prediction in Bing Search Ads, that is an important part of data mining, explain how the ensemble learning technique is utilized in the paper.

در این مقاله از ensemble learning به دو شکل استفاده شده است .
در حالت اول ما یک GBDT (Gradient-Boosted Decision Tree) داریم که یک مدل ensemble از درخت تصمیم است که به صورت گرادیان کاهشی و با اضافه کردن درخت‌تصمیم‌های مختلف سعی می کند خطا مسئله را کاهش دهد و به عنوان یکی از مدل‌های پایه‌ما استفاده می‌شود .
در حالت دوم ما از ensemble learning برای ترکیب مدل پایه خود استفاده می کنیم .
این کار را به دو شکل مختلف در مقاله انجام داده که حاصل آن ۷ مدل مختلف شده است .
روش اول به صورت stacking ensemble و روش دوم به صورت cascading ensemble و روش سوم که همان boosting ensemble hsj.
در روش اول ما چند مدل را به صورت مجزا روی داده‌های‌آموزش، آموزش می‌دهیم سپس یک meta model به کمک حاصل مدل‌های قبل پیش‌بینی انجام می‌دهد .
همانند مدل GBDT+DNN: که ابتدا یک شبکه عمیق و یک gradient boosted Decision tree به صورت مجزا بر روی داده‌ها آموزش داده شده اند و سپس میانگین نتایج حاصل از آنان به عنوان پیش‌بینی نهایی استفاده شده است.

در روش cascading یک مدل پیش بینی انجام می‌دهد و مدل بعد از پیش بینی مدل قبلی به عنوان یک داده ویژگی به اضافه ویژگی دادهای موجود استفاده می‌کند و دوباره سعی می‌کند پیش بینی کند .
همانند مدل : Cascading GBDT to LR و Cascading LR to GBDT و Cascading GBDT to DNN و Cascading DNN to GBDT .
برای مثال در Cascading DNN to GBDT ابتدا شبکه عمیق ما به صورت کامل بر روی داد آموزش می‌بینید سپس مدل GBDT ما به کمک حاصل بدست آمده از شبکه عمیق و ویژگی‌های موجود و دادگان دوباره بر روی مدل آموزش می‌بینند و پیش‌بینی نهایی را انجام می‌دهد بقیه موارد هم به همین شکل کار را انجام می‌دهند .

در روش Boosting با مدل کم برازش که دارای بایاس بالا و واریانس کم است کار می کند، یعنی مدل نمی تواند رابطه ذاتی در داده ها را به طور کامل توصیف کند. با این بینش که باقیمانده‌های مدل هنوز حاوی اطلاعات مفیدی هستند، تقویت در قلب آن به طور مکرر با مدل جدید بر روی باقیمانده‌ها مطابقت دارد. نتیجه نهایی با جمع کردن همه مدل ها با هم پیش بینی می شود. GBDT پرکاربردترین مدل تقویت کننده است .
همانند boosting DNN with GBDT و Boosting LR with GBDT .

کلا در روش ensemble ما سعی داریم با ترکیب مدل به شکل‌های مختلفی که در بالا مشاهده می‌کنید دقت پایه مدل را افزایش بدیم و همانطور که در قسمت ارزیابی مشاهده می‌کنید سعی کرده است دقت تمامی این حالت‌ها را بدست آورد و آن را گزارش کند تا در نهایت بهترین مدل با دقت را از این حالت‌های مختلف برگزیند .

سوال چهارم :

Using the hash function

$$h(x) = x \bmod 3$$

construct a hash tree with maximum number of itemsets in inner nodes equal to 4 given the following set of candidates

(1, 9, 11)	(2, 5, 10)	(3, 6, 8)	(4, 7, 9)	(6, 12, 13)	(9, 12, 14)
(1, 10, 12)	(2, 5, 12)	(3, 7, 10)	(4, 7, 13)	(6, 12, 14)	(10, 11, 15)
(2, 4, 7)	(2, 9, 10)	(3, 12, 14)	(5, 7, 9)	(8, 11, 11)	(12, 12, 15)
(2, 5, 8)	(3, 3, 5)	(4, 5, 8)	(5, 7, 13)	(8, 11, 15)	(14, 14, 15)

در گره ریشه، مجموعه آیتم ها بر اساس مقدار هش اولین آیتم در مجموعه آیتم ها تقسیم می شوند.

بنابراین، پس از گره ریشه، 3 گره فرزند با محتوا داریم:

البته محتوای هر ستون را مرتب می کنیم .اول برحسب عنصر اول بعد دو

No	N1	N2
3,3,5	1,9,11	2,4,7
3,6,8	1,10,12	2,5,8
3,7,10	4,5,8	2,5,10
3,12,14	4,7,9	2,5,12
6,12,13	4,7,13	2,9,10
6,12,14	10,11,15	5,7,9
9,12,14		5,7,13
12,12,15		8,11,11
		8,11,15
		14,14,15

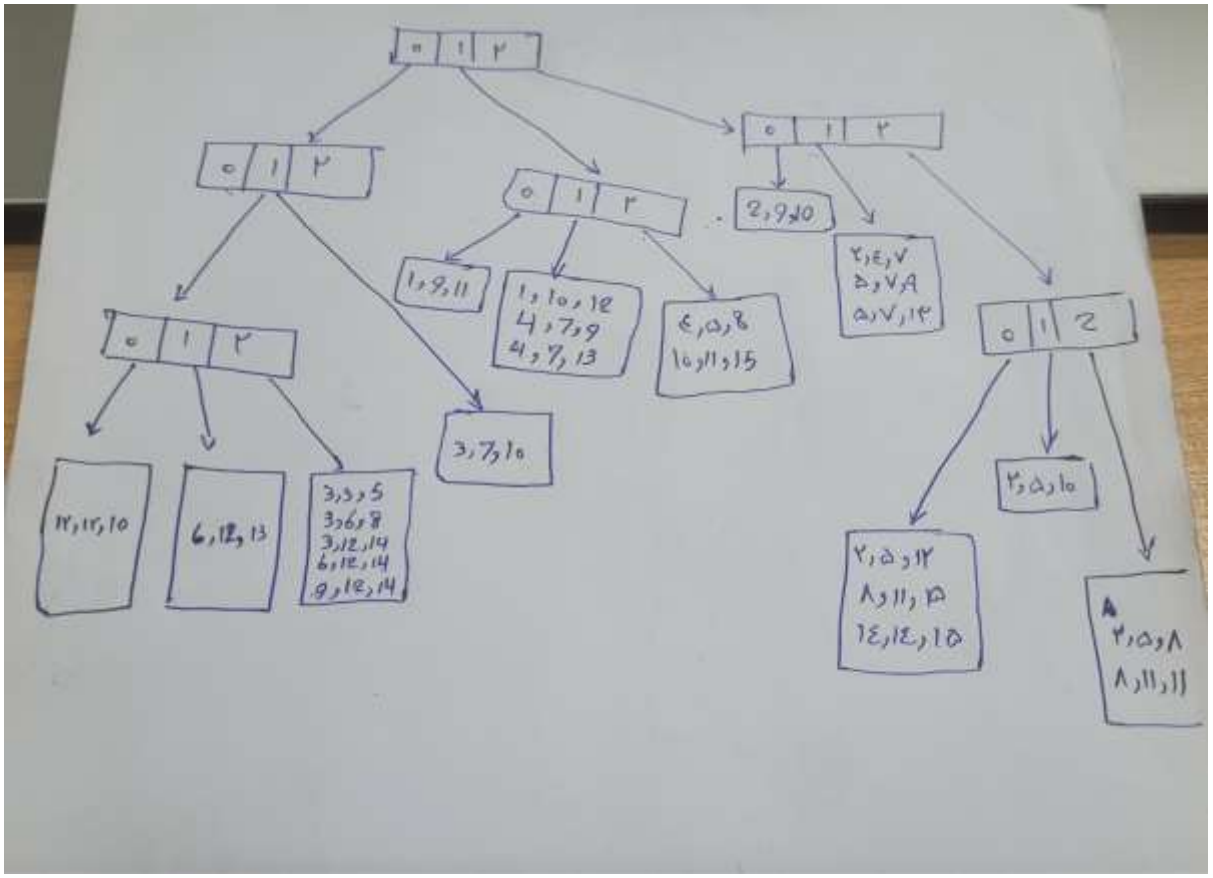
از آنجایی که درجه پر شدن همه گره ها 4 بزرگتر است، اکنون طبق مورد دوم، همه باید تقسیم شوند.

N00	N01	N10	N11	N12	N20	N21	N22
3,3,5	3,7,10	1,9,11	1,10,12	4,5,8	2,9,10	2,4,7	2,5,8
3,6,8			4,7,9	10,11,15		5,7,9	2,5,10
3,12,14			4,7,13			5,7,13	2,5,12
6,12,13							8,11,11
6,12,14							8,11,15
9,12,14							14,14,15
12,12,15							

در اینجا فقط 00N و 22N درجه پر شدن بالاتر از حد مجاز دارند (گره های برگ با * مشخص می شوند). از این رو، آنها دوباره تقسیم می شوند، این بار با استفاده از آیتم سوم. در اینجا، فقط 00N و 2N درجه پر شدن بالاتر از حد مجاز دارند (گره های برگ با * مشخص می شوند). از این رو، آنها دوباره تقسیم می شوند، این بار با استفاده از مورد سوم.

N000	N001	N002	N220	N221	N222
12,12,15	6,12,13	3,3,5	2,5,12	2,5,10	2,5,8
		3,6,8	8,11,15		
		3,12,14	14,14,15		
		6,12,14			
		9,12,14			

اگرچه درجه پر شدن 002N بزرگتر از 4 است، هیچ موردی باقی نمانده که برای تقسیم بیشتر استفاده شود.
از این رو، ساخت و ساز درخت هش به پایان می رسد. درخت هش نهایی در زیر نشان داده شده است:


0 1 2
0 1 2
0 1 2
0 1 2
0 1 2
0 1 2
2,9,10
1,9,11
1,10,12
4,7,9
4,7,13
4,5,8
10,11,15
2,4,7
5,7,8
5,7,14
3,7,10
12,13,10
6,12,13
3,3,5
3,6,8
3,12,14
6,12,14
9,12,14
3,5,10
2,5,12
8,11,5
13,14,15
2,5,8
8,11,11

سوال پنجم)

Given the lattice structure shown in Figure and the transactions given in Table, label each node with the following letter(s

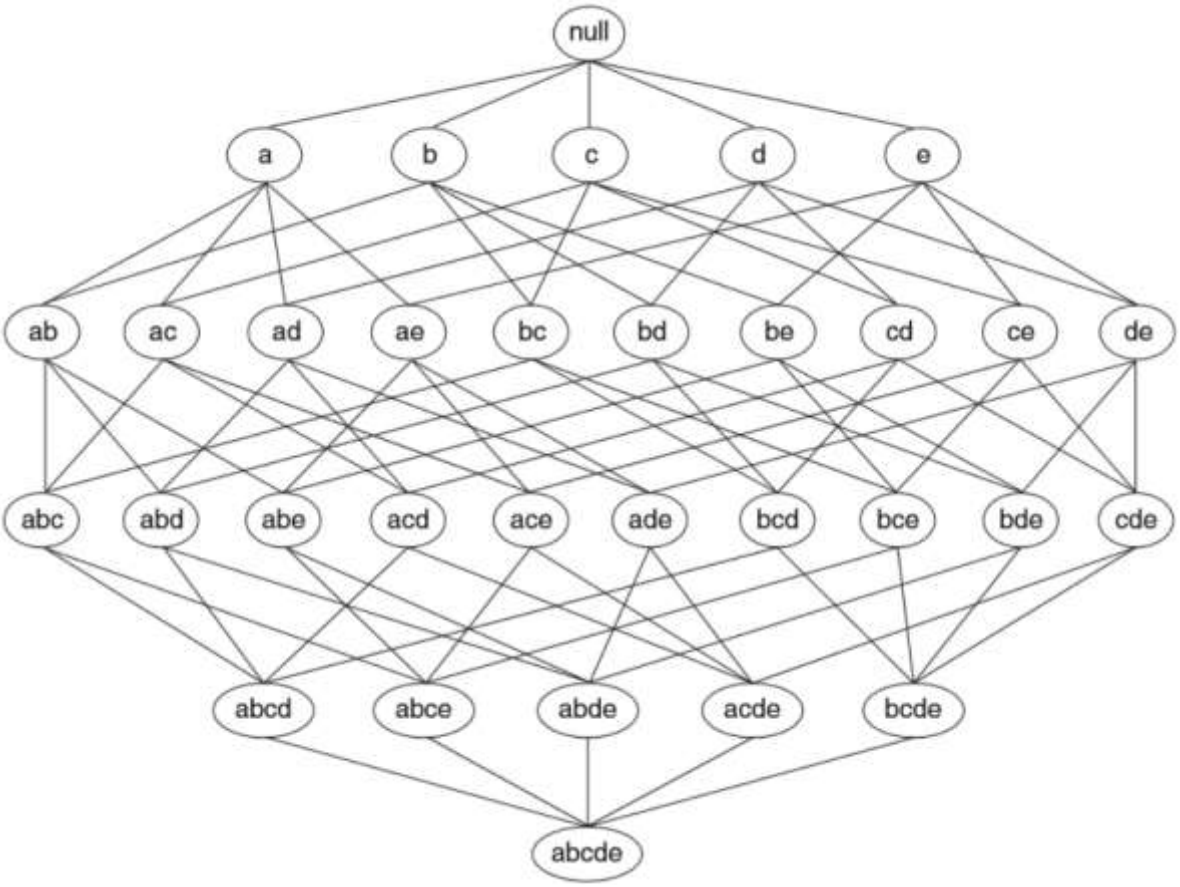
M if the node is a maximal frequent itemset,

C if it is a closed frequent itemset,

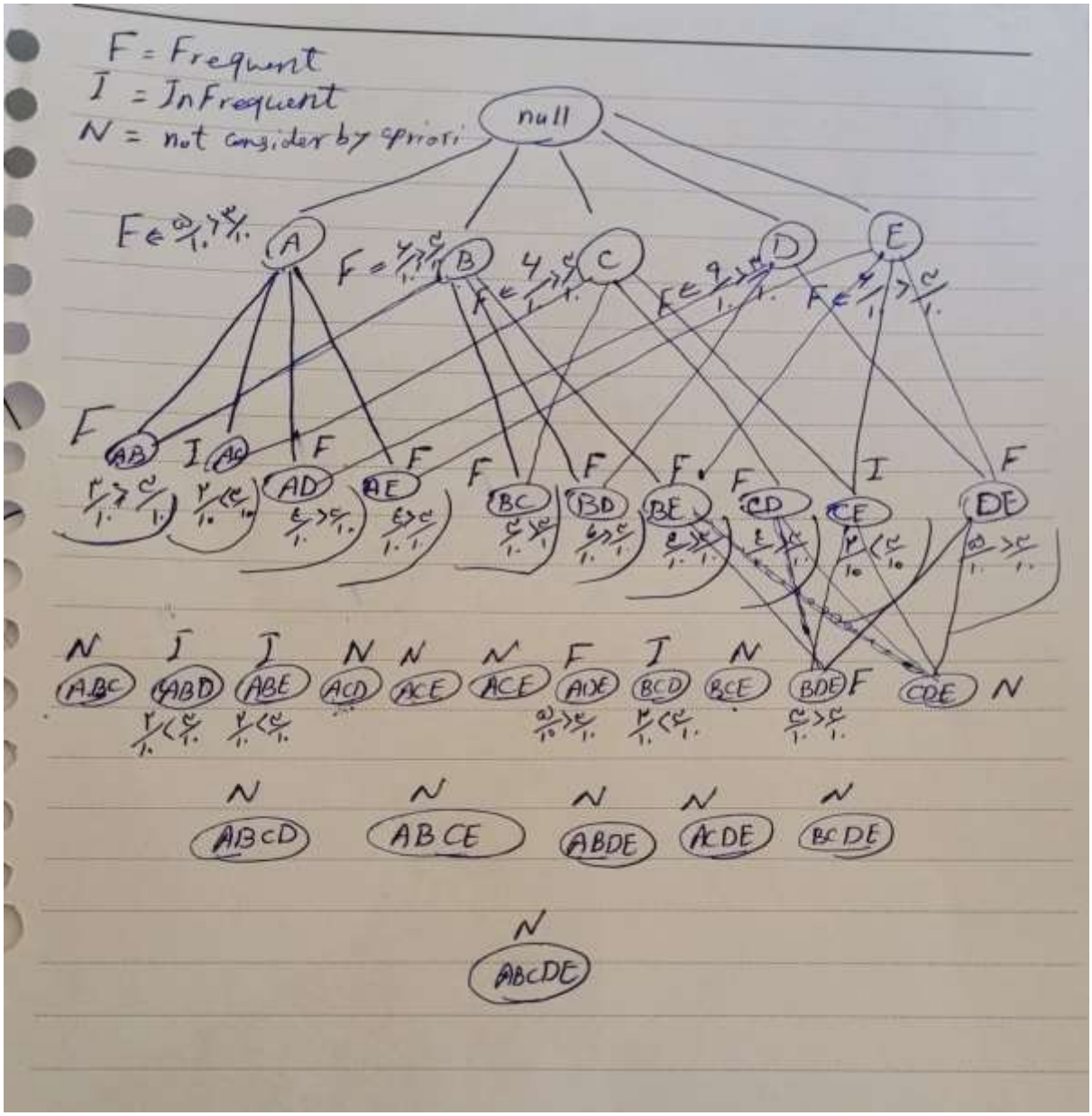
N if it is frequent but neither maximal nor closed, and

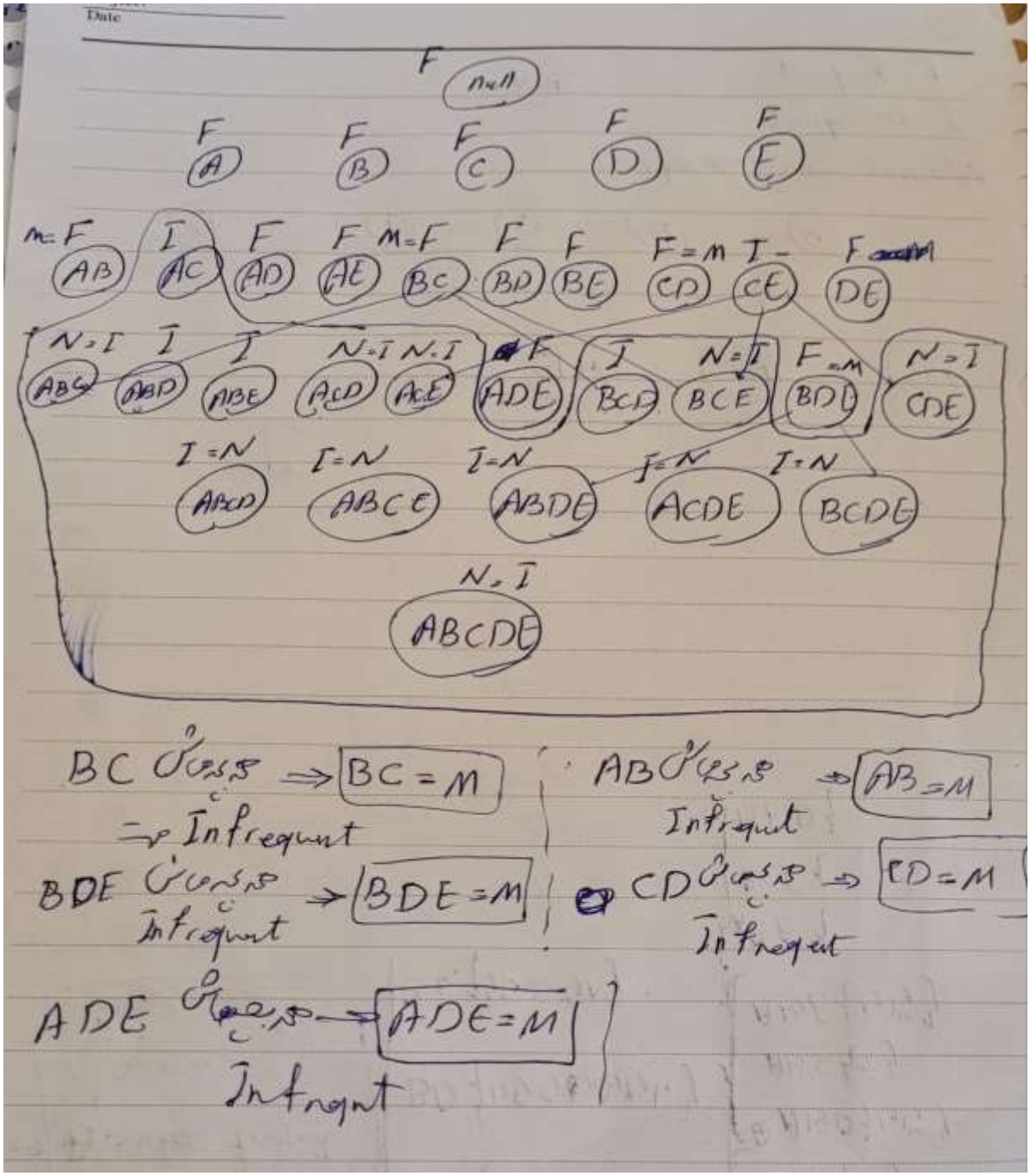
I if it is infrequent.

Assume that the support threshold is equal to 30%.



Transaction ID	Items Bought
1	$\{a, b, d, e\}$
2	$\{b, c, d\}$
3	$\{a, b, d, e\}$
4	$\{a, c, d, e\}$
5	$\{b, c, d, e\}$
6	$\{b, d, e\}$
7	$\{c, d\}$
8	$\{a, b, c\}$
9	$\{a, d, e\}$
10	$\{b, d\}$


F = Frequent
I = InFrequent
N = not consider by apriori
null
A
B
C
D
E
AB
AC
AD
AE
BC
BD
BE
CD
CE
DE
ABC
ABD
ABE
ACD
ACE
ACE
ADE
BCD
BCE
BDE
CDE
ABCD
ABCE
ABDE
ACDE
BCDE
ABCDE



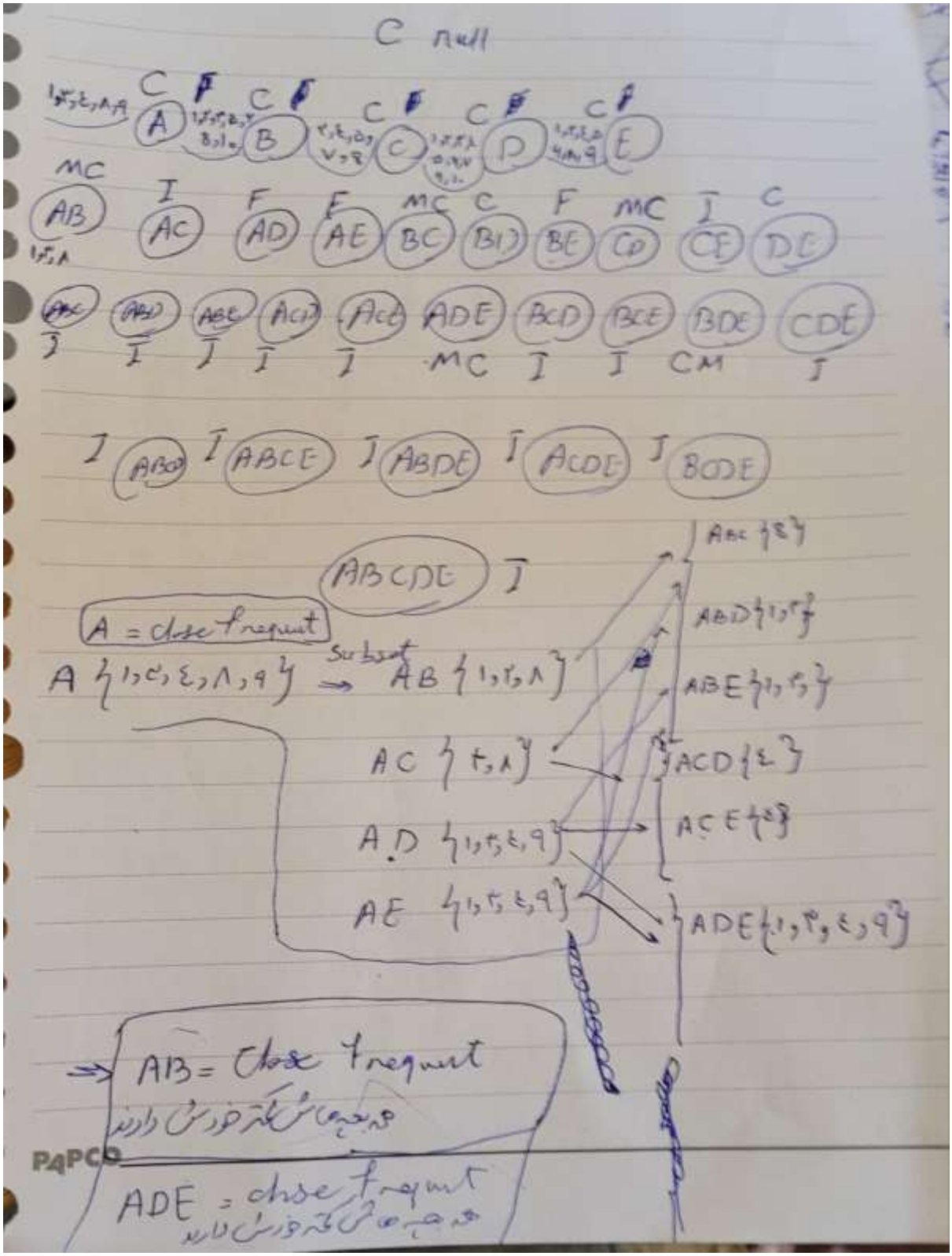
BC ... ⇒ BC = M
⇒ Infrequent

BDE ... ⇒ BDE = M
Infrequent

ADE ... ⇒ ADE = M
Infrequent

AB ... ⇒ AB = M
Infrequent

CD ... ⇒ CD = M
Infrequent


C null
A
B
C
D
E
AB
AC
AD
AE
BC
BD
BE
CD
CE
DE
ABC
ABD
ABE
ACD
ACE
ADE
BCD
BCE
BDE
CDE
ABCD
ABCE
ABDE
ACDE
BCDE
ABCDE
A = close frequnt
Subset
A {1,3,4,8,9}
AB {1,3,8}
AC {4,8}
AD {1,3,4,9}
AE {1,3,4,9}
ABC {4}
ABD {1,3}
ABE {1,3}
ACD {4}
ACE {4}
ADE {1,3,4,9}
AB = Close frequnt
ADE = close frequnt

Subject:
Date

B = {1, 3, 4, 5, 6, 8, 10}

B = close frequent
است که مجموعه‌ی کمتر خودش نیست

AB
- ABC {8}
- BCD {2, 5}
- BCE {5}

BC {2, 5, 4, 8}
- ABD {1, 4}
- BCD {2, 5}
- BDE {1, 3, 5, 6}

BD {1, 3, 4, 5, 6, 10}

BE {1, 3, 5, 6}
- ABE {1, 3}
- BCE {5}
- BED {}

BC = close frequent
که مجموعه‌ی کمتر خودش دارد

BDE close frequent
است

AC {4, 8}
- ABC {8}
- ACD {4}
- ACE {4}

BC {2, 5, 8}
- ABC
- BCE
- BCD

C = {2, 4, 5, 7, 8}
close frequent
که مجموعه‌ی کمتر خودش دارد

CD {2, 4, 5, 7}
- ACD {4}
- BCD {2, 5}
- CDE {}

CE = {4, 5}
- ACE
- BCE
- CDE

CD
close frequent
که مجموعه‌های کمتر خودش

PAPCO

D ⇒ تنها یک فرزند frequent دارد

CD → تعداد تراکنش‌هایش با D close frequent

{۱،۲،۳،۴، ۵،۶،۷،۸،۹،۱۰} ⇒ CD {۲، ۴، ۵، ۷}

→ maximal

خودش ایضا ← close Frequent هم هست.

E → {۱، ۲، ۴، ۵، ۶، ۹}

- AE {۱، ۳، ۴، ۹}
- BE {۱، ۳، ۵، ۶، [illegible]}
- CE {۴، ۵}
- DE {۱، ۲، ۳، ۴، ۵، ۶، ۹}

DE = E چون

فقط E بیش

Frequent = ۲

DE:

- ADE mc {۱، ۳، ۴، ۹}
- BDE mc {۱، ۳، ۵، ۶}
- CDE I

↓

DE

close هست چون تعداد تراکنش‌هایش

کمتر فرزندش هست

Problem 6)* (Project)

We want to implement the apriori algorithm with python.

In this section, you are going to apply frequent pattern mining on a market transaction dataset via the apriori algorithm. The dataset is available in the assignment folder. First, generate all frequent itemsets in a table (e.g., Figure 1), then print all possible association rules as shown in Figure 2. Write your code in an ipynb notebook format, then answer the following questions:

1. How did you handle Nan values in the dataset?

با تبدیل کردن دیتاست به آرایه‌ای از خریدها و در نظر نگرفتن داده‌های nan به هدفم رسیدم . البته جزئیات را در پایین توضیح داده‌ام .

2. In what aspect is this similar to the Naïve Bayes algorithm? Why?

به طور کلی، هم الگوریتم‌های بیز و هم الگوریتم‌های قانون انجمن در یادگیری ماشین برای طبقه‌بندی و مدل‌سازی روابط بین متغیرها یا ویژگی‌های مختلف در یک مجموعه داده استفاده می‌شوند. آنها هر دو شامل محاسبات احتمال و گروه بندی داده ها بر اساس معیارهای خاص هستند. بنابراین، شباهت بین این الگوریتم‌ها در کاربرد آن‌ها برای تحلیل و مدل‌سازی داده‌ها و استفاده از روش‌های آماری برای نتیجه‌گیری در مورد روابط و الگوهای موجود در داده‌ها است.

همچنین معیار confidence در اینجا همان معیار مورد استفاده احتمالاتی در بیز است .به طوری که در هر دو ما از معیار

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})} \qquad \text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

در جهت هدف خود استفاده می‌کنیم در یکی برای پیدا کردن بالاترین مقدار آن برای محاسبه و تقسیم بندی یک داده بین کلاس‌های موجود . و در یکی برای اینکه ببینیم قانون بدست آمده چقدر ارزشمند است .

3. How do min_sup and min_conf parameters affect your results?

به کمک min sup ما بسیاری از تراکنش‌هایی که می‌دانیم برای استخراج قوانین به کمک ما نمی‌آیند را حذف می کنیم و به کمک min confidence می‌توانیم بر روی قوانینی که از لحاظ احتمالاتی نظر ما جلب نمی کنند را راحت کنار بگذاریم تا بتوانیم نظر و توجه خود را بر روی قوانین با ارزش بالاتر بگذاریم .

In the final part, you are used to support and confidence all the time and maybe the gini index. There are lots of measures proposed in the literature of the class. Some measures are good for certain applications, but not for others.

In this part What about Apriori-style support based pruning? How does it affect these measures?

استفاده از هرس مبتنی بر حمایت به سبک Apriori می تواند تأثیر قابل توجهی بر میزان confidence و support داشته باشد. با حذف مجموعه‌های اقلام نادر از بررسی، support به معیار معناداری برای قدرت ارتباط بین اقلام تبدیل می‌شود. confidence نیز تحت تأثیر قرار می گیرد، زیرا هرس مجموعه‌های اقلام نادر، تعداد ارتباط‌های مثبت کاذب را کاهش می‌دهد و در نتیجه مقادیر confidence بالاتری برای پیوندهای باقی‌مانده ایجاد می کند. تأثیر هرس مبتنی بر حمایت به سبک Apriori بر شاخص جینی کمتر واضح است، زیرا این معیار معمولاً در استخراج قوانین انجمن استفاده نمی شود. با این حال، ممکن است هرس مجموعه‌های اقلام نادر می‌تواند توزیع فرکانس اقلام را تغییر دهد و در نتیجه بر شاخص جینی تأثیر بگذارد. به طور کلی، استفاده از هرس مبتنی بر حمایت به سبک Apriori یک تکنیک مهم برای استخراج قوانین تداعی کارآمد است و می‌تواند تأثیر قابل‌توجهی بر تفسیر میزان support و confidence داشته باشد.

So you should use this three important measurements and after results, compare together,

which one was better and if you can say why, you can have bonus scores

Interest

Kappa

Gini Index

از نظر من و با توجه به بررسی که بر روی قوانین بدست آمده، معیار interest به نسبت دو تا معیار دیگر قوانین جذاب‌تر ومعنادار تری را حتی نسبت به confidence پیدا می‌کند .

همانطور که در انتها و در نتایج آخر برحسب confidence آمده است با بالا بردن میزان confidence و رساندن آن به 0.5 تنها و تقریبا قوانینی باقی می‌مانند که طرف سمت راست آن آب معدنی هست اما خوب این قوانین قوانین منطقی و خوبی نیستند زیرا به طور معمول در تمامی سبدهای خرید آب معدنی بود و وجود این آیتم‌ها به معنی اینکه آب معدنی به دلیل برداشتن آیتم‌های قبلی برداشته شده است بی معنی است .

اما اگر به جدول زیر که بر حسب معیار interest ساخته شده است می‌توان به قوانین بهتری دست یافت .علت نیز تقسیم شدن حاصل confidence بر احتمال رخداد سمت راست است که عملا به کمک این تقسیم برای مثال اگر آب معدنی را در نظر بگیریم به دلیل اینکه احتمال رخداد آن در تمامی ترکنش‌ها بالا است پس با این تقسیم این اثر کاهش پیدا می‌کند .

	rule	left	conf	support	Kappa	interest	gini_index
1232	pasta -> escalope	pasta	0.372881	0.005866	0.099787	4.700812	0.993460
1231	pasta -> shrimp	pasta	0.322034	0.005066	0.092816	4.506672	0.994646
2310	herb & pepper,spaghetti -> ground beef	herb & pepper,spaghetti	0.393443	0.006399	0.086256	4.004360	0.990082
2346	herb & pepper,mineral water -> ground beef	herb & pepper,mineral water	0.390625	0.006666	0.089120	3.975683	0.990055
1664	tomato sauce -> ground beef	tomato sauce	0.377358	0.005333	0.071968	3.840659	0.990147
1688	mushroom cream sauce -> escalope	mushroom cream sauce	0.300699	0.005733	0.088512	3.790833	0.993344
2298	tomatoes,spaghetti -> frozen vegetables	tomatoes,spaghetti	0.318471	0.006666	0.083211	3.341054	0.990476
1397	herb & pepper -> ground beef	herb & pepper	0.323450	0.015998	0.161430	3.291994	0.987900
2304	grated cheese,spaghetti -> ground beef	grated cheese,spaghetti	0.322581	0.005333	0.066496	3.283144	0.990073
2354	ground beef,cooking oil -> spaghetti	ground beef,cooking oil	0.571429	0.004799	0.037164	3.281995	0.969615
2142	olive oil,frozen vegetables -> milk	olive oil,frozen vegetables	0.423529	0.004799	0.048282	3.268410	0.983080
1702	mineral water,shrimp -> frozen vegetables	mineral water,shrimp	0.305085	0.007199	0.086520	3.200616	0.990357
1897	spaghetti,frozen vegetables -> ground beef	spaghetti,frozen vegetables	0.311005	0.008666	0.098273	3.165328	0.989570
2292	cooking oil,spaghetti -> ground beef	cooking oil,spaghetti	0.302521	0.004799	0.058389	3.078982	0.990095
1862	ground beef,shrimp -> spaghetti	ground beef,shrimp	0.523256	0.005999	0.044090	3.005315	0.969554

توضیحات مربوط به کد :

برای حذف داده‌های null من دیتافریم را تبدیل به یک آرایه از آرایه‌ها کردم، که هر کدام از این آرایه‌ها یک transaction است که توی این آرایه‌ها هیچ داده null وجود ندارد .

```
# assume you have a 2D numpy array named "numpay_market_data" containing NaN values
transcations_market = []
for transaction in numpay_market_data :
  # create a boolean mask marking the NaN values as True and non-NaN values as False
  # print(transaction)
  mask = transaction!= 'nan'
  # filter the array by selecting only the non-NaN values
  clear_transation = transaction[mask]
  transcations_market.append(clear_transation)
# print the resulting filtered array
transcations_market = np.array(transcations_market)
print(transcations_market)
```

```
[array(['shrimp', 'almonds', 'avocado', 'vegetables mix', 'green grapes',
        'whole weat flour', 'yams', 'cottage cheese', 'energy drink',
        'tomato juice', 'low fat yogurt', 'green tea', 'honey', 'salad',
        'mineral water', 'salmon', 'antioxydant juice', 'frozen smoothie',
        'spinach', 'olive oil'], dtype=object)
 array(['burgers', 'meatballs', 'eggs'], dtype=object)
 array(['chutney'], dtype=object) ... array(['chicken'], dtype=object)
 array(['escalope', 'green tea'], dtype=object)
 array(['eggs', 'frozen smoothie', 'yogurt cake', 'low fat yogurt'],
       dtype=object)                                                ]
```

سپس شروع می‌کنیم به پیدا کردن frequent item ها که برای این کار تعداد هر آیتم‌را بدست آوریم و سپس حاصل را بر تعداد ترنزکشن‌ها تقسیم کنیم .

```
dic_of_frequent_item = {}
number_of_transaction = transcations_market.shape[0]
for transaction in transcations_market :

    for item in transaction :

        if item in dic_of_frequent_item.keys():

            dic_of_frequent_item[item] += 1
        else :
            dic_of_frequent_item[item] = 1


print(number_of_transaction,dic_of_frequent_item)
```

```
7501 {'shrimp': 536, 'almonds': 153, 'avocado': 250, 'vegetables mix': 193, 'green grapes': 68, 'whole
```

```
[ ] data_show_list = []
    list_of_name = dic_of_frequent_item.keys()
    for key in list_of_name:

        data_show_list.append({'itemsets':key,'support':dic_of_frequent_item[key]/number_of_transaction})

    df = pd.DataFrame.from_dict(data_show_list)
    df
```

	itemsets	support
0	shrimp	0.071457
1	almonds	0.020397
2	avocado	0.033329
3	vegetables mix	0.025730
4	green grapes	0.009065
...	...	...
115	burger sauce	0.005866
116	oatmeal	0.004399
117	asparagus	0.000133
118	cream	0.000933
119	napkins	0.000667

سپس تابع زیر را می‌نویسیم که frequent آیتم‌ها را در هر نوع را پیدا می‌کند یعنی اینکه ابتدا frequent آیتم‌های قوانین یک آیتمی، سپس بر اساس آن قوانین دو آیتمی و ... تا در انتها هیچ قانونی وجود نداشته باشد. حال ما به ازای هر دسته از قوانین دوتایی، سه تایی و چهارتایی تمامی frequent ها را داریم .

```python
def generate_frequent_itemsets(transactions, min_support):
    # Initialize the frequent itemsets of length 1
    frequent_itemsets = []
    infrequent_itemsets_k = []
    candidate_itemsets = []
    for transaction in transactions:
        for item in transaction:
            if not {item} in candidate_itemsets:
                candidate_itemsets.append({item})

    candidate_itemsets = list(map(frozenset, candidate_itemsets))
    candidate_itemsets.sort()
    frequent_itemsets_k,infrequent_itemsets_k = count_frequent_itemsets(candidate_itemsets,transactions,min_support)
    print(frequent_itemsets_k)
    print(infrequent_itemsets_k)
    print("_______")
    frequent_itemsets.append(frequent_itemsets_k)
    # Generate frequent itemsets of length k+1 until no new frequent itemsets are identified
    k = 1
    while True:
        candidate_itemsets = generate_candidate_itemsets(frequent_itemsets, k,infrequent_itemsets_k)
        if len(candidate_itemsets) == 0:
            break
        frequent_itemsets_k,infrequent_itemsets_k = count_frequent_itemsets(candidate_itemsets, transactions, min_support)
        if len(frequent_itemsets_k) == 0:
            break
        frequent_itemsets.append(frequent_itemsets_k)
        k += 1

    # Return the frequent itemsets
    return frequent_itemsets
```

سپس براساس frequent آیتم‌ها سعی می‌کنیم قوانین را بسازیم .

```python
import itertools

def generate_rules(items):
    rules = {}
    for i in range(1, len(items)):
        for antecedent_items in itertools.combinations(items, i):
            # print(antecedent_items)
            # print(antecedent)
            consequent = items.difference(antecedent_items)
            antecedent = ','.join(antecedent_items)

            consequent = ','.join(consequent)
            # print(consequent)
            rules[antecedent] = consequent
    return rules
```

تابع بالا یک لیست از آیتم‌ها را می‌گیرد و سپس تمامی قوانینی که می‌شود با آن لیست ساخت را برمی‌گرداند .
سپس براساس این تابع تابع زیر را تعریف می‌کنیم .

```python
def find_all_possible_rule(frequent_item_can_make_rule) :
    return generate_rules(frequent_item_can_make_rule)


def find_problit_on_transaction(left,right,transactions):

  left_set = set(left.split(','))
  right_set = set(right.split(','))
  set_main = left_set.union(right_set)
  left_sum = 0
  right_sum = 0
  main_sum = 0
  not_sum =0
  for transaction in transactions:
      set_transaction = set(transaction)

      if set_main.issubset(set_transaction):
          main_sum += 1

      if left_set.issubset(set_transaction):
          left_sum += 1

      if right_set.issubset(set_transaction):
          right_sum += 1
      if not right_set.issubset(set_transaction) and not left_set.issubset(set_transaction):
          not_sum += 1

  return (left_sum/len(transactions),right_sum/len(transactions),main_sum/len(transactions),not_sum/len(transactions))
```

سپس بر اساس سمت چپ و سمت راست قوانین به ترتیب احتمال رخداد سمت چپ، احتمال سمت راست، اشتراک رخداد هم سمت چپ و هم سمت ، احتمال اینکه کلا نه سمت چپ اتفاق بیفتد و نه سمت راست را حساب می کنیم .

```python
def make_rule_from_frequent_item(frequent_item_can_make_rule,transcations):
    dic_of_rule = {}
    for frequent_list in frequent_item_can_make_rule:

        for set_frequent in frequent_list :

          all_possible_rule = find_all_possible_rule(set_frequent)
          for rule in all_possible_rule.keys() :
            left =  rule
            right = all_possible_rule[rule]
            if rule in dic_of_rule.keys():

                (pa,pb,pab,pnot) = find_problit_on_transaction(left,right,transcations)
                dic_of_rule[rule].append((all_possible_rule[rule],pa,pb,pab,pnot))

            else :
              (pa,pb,pab,pnot) = find_problit_on_transaction(left,right,transcations)
              dic_of_rule[rule] = [(all_possible_rule[rule],pa,pb,pab,pnot)]

    return dic_of_rule
```

سپس به کمک تابع بالا تمامی قوانین را به کمک تمامی احتمالاتی که دارند در یک دیکشنری ذخیره می کنیم .

```
dic_for_make_dataframe = {'rule':[],'left':[],'conf':[],'support':[],'Kappa':[],'interest':[],'gini_index':[]}
for key in dic_of_rule.keys():
    for item in dic_of_rule[key]:
      dic_for_make_dataframe['rule'].append(f'{key} -> {item[0]}')
      dic_for_make_dataframe['left'].append(key)
      dic_for_make_dataframe['conf'].append(item[3]/item[1])
      dic_for_make_dataframe['support'].append(item[3])
      dic_for_make_dataframe['interest'].append(item[3]/(item[1]*item[2]))
      kappa = (item[3]+item[4] - item[1]*item[2] - ((1-item[1])*(1-item[2])))/(1- item[1]*item[2] - ((1-item[1])*(1-item[2])))
      dic_for_make_dataframe['Kappa'].append(kappa)

      gini_index = 1 - ((item[1])**2+(item[2])**2)
      dic_for_make_dataframe['gini_index'].append(gini_index)

df = pd.DataFrame(dic_for_make_dataframe)
```

سپس یک دیتا فریم بر اساس اطلاعات جدید برحسب آن دیکشنری که در بالا ساختیم می‌سازیم و برحسب احتمالاتی که داریم kappa ، interest ، support ،confidence و gini index را بدست می‌آوریم .
که حاصل یک جدول مانند زیر خواهد شد که براساس support مرتب شده است .

	rule	left	conf	support	Kappa	interest	gini_index
161	mineral water -> spaghetti	mineral water	0.250559	0.059725	0.110619	1.439085	0.912866
894	spaghetti -> mineral water	spaghetti	0.343032	0.059725	0.110619	1.439085	0.912866
1066	chocolate -> mineral water	chocolate	0.321400	0.052660	0.083950	1.348332	0.916335
165	mineral water -> chocolate	mineral water	0.220917	0.052660	0.083950	1.348332	0.916335
467	eggs -> mineral water	eggs	0.283383	0.050927	0.048673	1.188845	0.910885
...	...	...	...	...	...	...	...
2099	salmon,mineral water -> chocolate	salmon,mineral water	0.265625	0.004533	0.019813	1.621199	0.972864
55	shrimp -> olive oil,mineral water	shrimp	0.063433	0.004533	0.053849	2.298598	0.994132
2100	salmon,chocolate -> mineral water	salmon,chocolate	0.425000	0.004533	0.016319	1.782956	0.943067
552	eggs -> cooking oil,chocolate	eggs	0.025223	0.004533	0.022174	1.854847	0.967520
925	spaghetti -> white wine	spaghetti	0.026034	0.004533	0.017898	1.574828	0.969412

2368 rows × 7 columns

سپس برحسب confidence بالای 0.3 و 0.5، قوانین را بدست می‌آوریم.

```
thershold_conf = 0.3

mask = df['conf'] <= thershold_conf  # Create a boolean mask for the rows that need to be removed
new_df = df[~mask]            # Select only the rows that don't match the mask

new_df.sort_values(by=['conf'],ascending = False)
```

	rule	left	conf	support	Kappa	interest	gini_index
2261	frozen vegetables,soup -> mineral water	frozen vegetables,soup	0.633333	0.005066	0.026050	2.656954	0.943117
2297	pancakes,cooking oil -> mineral water	pancakes,cooking oil	0.593220	0.004666	0.023021	2.488672	0.943119
2118	soup,olive oil -> mineral water	soup,olive oil	0.582090	0.005199	0.025264	2.441976	0.943101
2126	frozen vegetables,olive oil -> mineral water	frozen vegetables,olive oil	0.576471	0.006532	0.031366	2.418404	0.943052
2353	cooking oil,ground beef -> spaghetti	cooking oil,ground beef	0.571429	0.004799	0.037164	3.281995	0.969615
...	...	...	...	...	...	...	...
2004	green tea,pancakes -> spaghetti	green tea,pancakes	0.300813	0.004933	0.022486	1.727717	0.969417
799	soup -> milk	soup	0.300792	0.015198	0.103591	2.321232	0.980655
1688	mushroom cream sauce -> escalope	mushroom cream sauce	0.300699	0.005733	0.088512	3.790833	0.993344
2253	frozen vegetables,french fries -> milk	frozen vegetables,french fries	0.300699	0.005733	0.045401	2.320520	0.982845
1983	mineral water,french fries -> spaghetti	mineral water,french fries	0.300395	0.010132	0.043443	1.725318	0.968548

311 rows × 7 columns

```
thershold_conf = 0.5
mask = df['conf'] <= thershold_conf  # Create a boolean mask for the rows that need to be removed
new_df_50 = df[~mask]                # Select only the rows that don't match the mask
new_df_50.sort_values(by=['conf'],ascending = False)
```

	rule	left	conf	support	Kappa	interest	gini_index
2261	frozen vegetables,soup -> mineral water	frozen vegetables,soup	0.633333	0.005066	0.026050	2.656954	0.943117
2297	pancakes,cooking oil -> mineral water	pancakes,cooking oil	0.593220	0.004666	0.023021	2.488672	0.943119
2118	soup,olive oil -> mineral water	soup,olive oil	0.582090	0.005199	0.025264	2.441976	0.943101
2126	frozen vegetables,olive oil -> mineral water	frozen vegetables,olive oil	0.576471	0.006532	0.031366	2.418404	0.943052
2353	cooking oil,ground beef -> spaghetti	cooking oil,ground beef	0.571429	0.004799	0.037164	3.281995	0.969615
2233	soup,milk -> mineral water	soup,milk	0.561404	0.008532	0.039863	2.355194	0.942950
1699	olive oil,shrimp -> mineral water	olive oil,shrimp	0.557377	0.004533	0.021385	2.338303	0.943114
2264	soup,chocolate -> mineral water	soup,chocolate	0.552632	0.005599	0.026135	2.318395	0.943078
2361	frozen vegetables,spaghetti,milk -> mineral water	frozen vegetables,spaghetti,milk	0.548387	0.004533	0.021117	2.300588	0.943112
2183	cooking oil,eggs -> mineral water	cooking oil,eggs	0.545455	0.006399	0.029469	2.288286	0.943043
2178	soup,eggs -> mineral water	soup,eggs	0.544118	0.004933	0.022802	2.282677	0.943098
2269	frozen vegetables,ground beef -> mineral water	frozen vegetables,ground beef	0.543307	0.009199	0.041767	2.279277	0.942894
2219	turkey,milk -> mineral water	turkey,milk	0.541176	0.006133	0.028092	2.270338	0.943052
2316	pancakes,chicken -> mineral water	pancakes,chicken	0.529412	0.004799	0.021706	2.220983	0.943098
2262	soup,spaghetti -> mineral water	soup,spaghetti	0.523364	0.007466	0.033075	2.195614	0.942977
1864	ground beef,shrimp -> spaghetti	ground beef,shrimp	0.523256	0.005999	0.044090	3.005315	0.969554
2265	soup,ground beef -> mineral water	soup,ground beef	0.520548	0.005066	0.022560	2.183798	0.943086
2298	chicken,chocolate -> mineral water	chicken,chocolate	0.518182	0.007599	0.033355	2.173871	0.942966
2320	pancakes,ground beef -> mineral water	pancakes,ground beef	0.513761	0.007466	0.032539	2.155327	0.942969
2271	frozen vegetables,ground beef -> spaghetti	frozen vegetables,ground beef	0.511811	0.008666	0.061764	2.939582	0.969399
2355	chicken,ground beef -> spaghetti	chicken,ground beef	0.507042	0.004799	0.034961	2.912193	0.969596
2207	eggs,ground beef -> mineral water	eggs,ground beef	0.506667	0.010132	0.043123	2.125563	0.942781
2128	frozen vegetables,olive oil -> spaghetti	frozen vegetables,olive oil	0.505882	0.005733	0.041429	2.905531	0.969557
2097	salmon,spaghetti -> mineral water	salmon,spaghetti	0.504950	0.006799	0.029253	2.118363	0.942999
2130	olive oil,chocolate -> mineral water	olive oil,chocolate	0.504065	0.008266	0.035285	2.114649	0.942912

Problem 7

In this part, you have to know that a data analyzer has to learn how to use data mining toolkits. One of the most important toolkits is RapidMiner. According to the previous part, you have to deal with dataset of previous problem. There is an introduction to RapidMiner in link below that you can use to do this problem (https://docs.rapidminer.com/latest/studio/getting-started/), and you can download the software in this link (https://my.rapidminer.com).

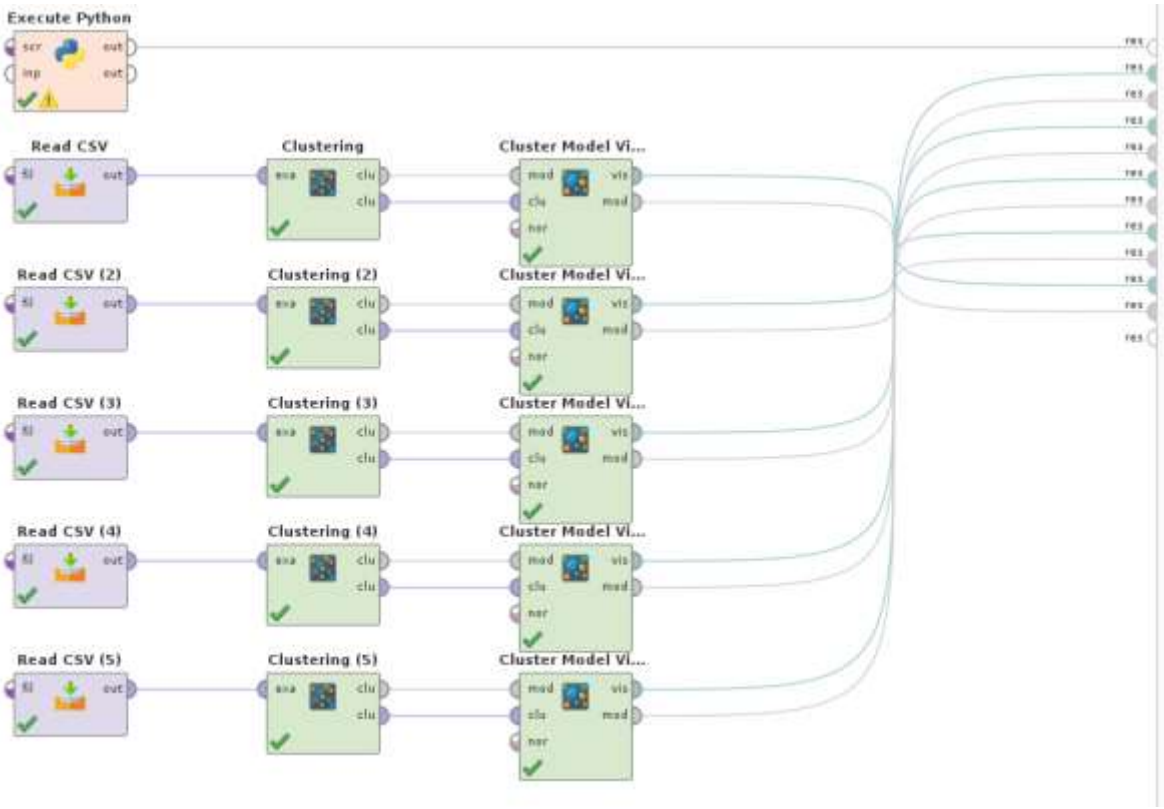
In this project, you will use RapidMiner to perform customer segmentation on a dataset. The goal of the project is to segment customers based on their purchasing behavior, demographics, and other relevant variables. You will need to perform the following tasks:

1. Data preparation: You will need to clean and preprocess the dataset, handle missing values, and convert categorical variables to numerical ones.
2. Exploratory Data Analysis: You will explore the dataset to gain insights into the variables, perform statistical analysis, and visualize the data using different charts and graphs.
3. Feature Engineering: You will create new features based on the existing variables or domain knowledge to improve the model performance.
4. Model Selection and Evaluation: You will choose appropriate clustering algorithms.
5. Interpretation: Finally, students will interpret the results of the clustering algorithm and create customer profiles based on the segments identified.

After learning these parts and a brief document about that, you have to create association rules and compare it with your previous results.

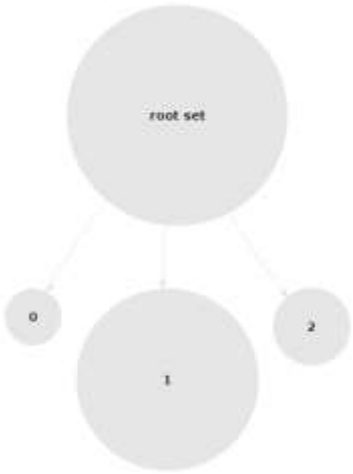
خوب من در ابتدا یک کد پایتون را اجرا کردم که توسط آن preprocese اولیه انجام می‌شود و به یک csv تبدیل می‌شود که فقط در آن transaction ها وجود دارند .

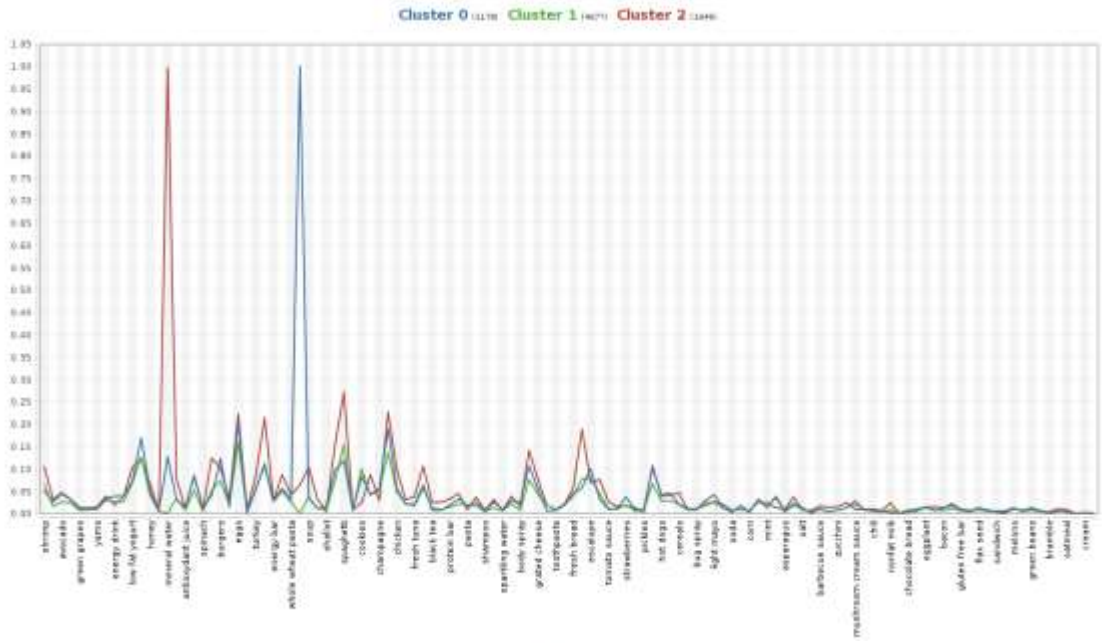
سپس در مرحله بعد از این csv جدید می‌خوانم و به بخش clustering حاصل را می‌دهم و در ادامه کمک cluster model visualisation حاصل مدل را نشان می‌دهیم که به توصیف پذیری مدل کمک می کند که در ادامه خواهید دید .



در ادامه حاصل را به ازای تعداد تقسیم بندی‌های مختلف مشاهده می‌کنیم که باید براساس این نگاه بدست آمده بهترین k را انتخاب کنیم .

k means (k = 3):





Cluster Model

```
Cluster 0: 1178 items
Cluster 1: 4677 items
Cluster 2: 1646 items
Total number of items: 7501
```

Number of Clusters: 3

Cluster 0 1,178

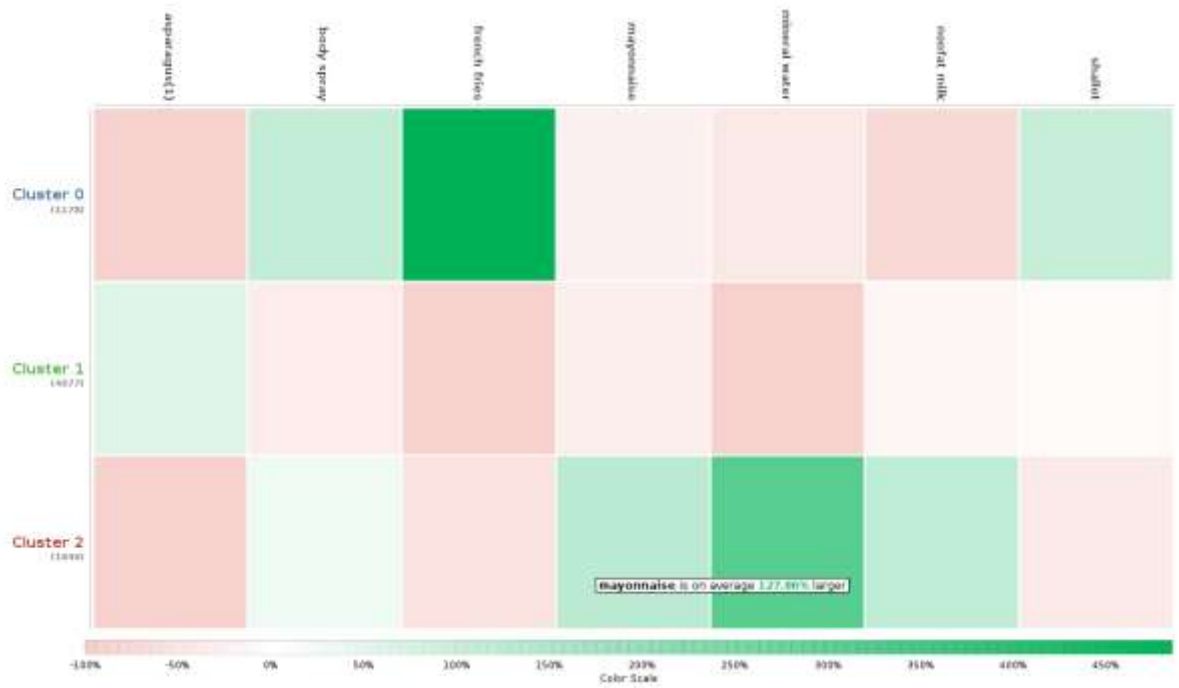
french fries is on average **485.10%** larger, **body spray** is on average **114.72%** larger, **shallot** is on average **108.59%** larger

Cluster 1 4,677

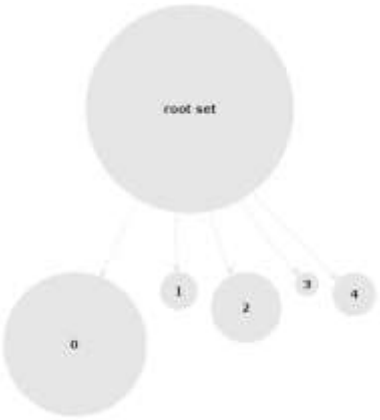
mineral water is on average **100.00%** smaller, **french fries** is on average **100.00%** smaller, **asparagus(1)** is on average **60.38%** larger

Cluster 2 1,646

mineral water is on average **317.73%** larger, **mayonnaise** is on average **127.86%** larger, **nonfat milk** is on average **122.01%** larger


asparagus(1)
body spray
french fries
mayonnaise
mineral water
nonfat milk
shallot
Cluster 0
Cluster 1
Cluster 2
mayonnaise is on average 127.8% larger
-100%
-50%
0%
50%
100%
150%
200%
250%
300%
350%
400%
450%
Color Scale

k means (k = 5):



Cluster Model

```
Cluster 0: 3639 items
Cluster 1: 818 items
Cluster 2: 1585 items
Cluster 3: 515 items
Cluster 4: 944 items
Total number of items: 7501
```

Number of Clusters: 5

Cluster 0 3,639

asparagus(1) is on average **106.13%** larger, **water spray** is on average **106.13%** larger, **mineral water** is on average **100.00%** smaller

Cluster 1 818

spaghetti is on average **474.35%** larger, **tomato sauce** is on average **142.22%** larger, **pepper** is on average **130.40%** larger

Cluster 2 1,585

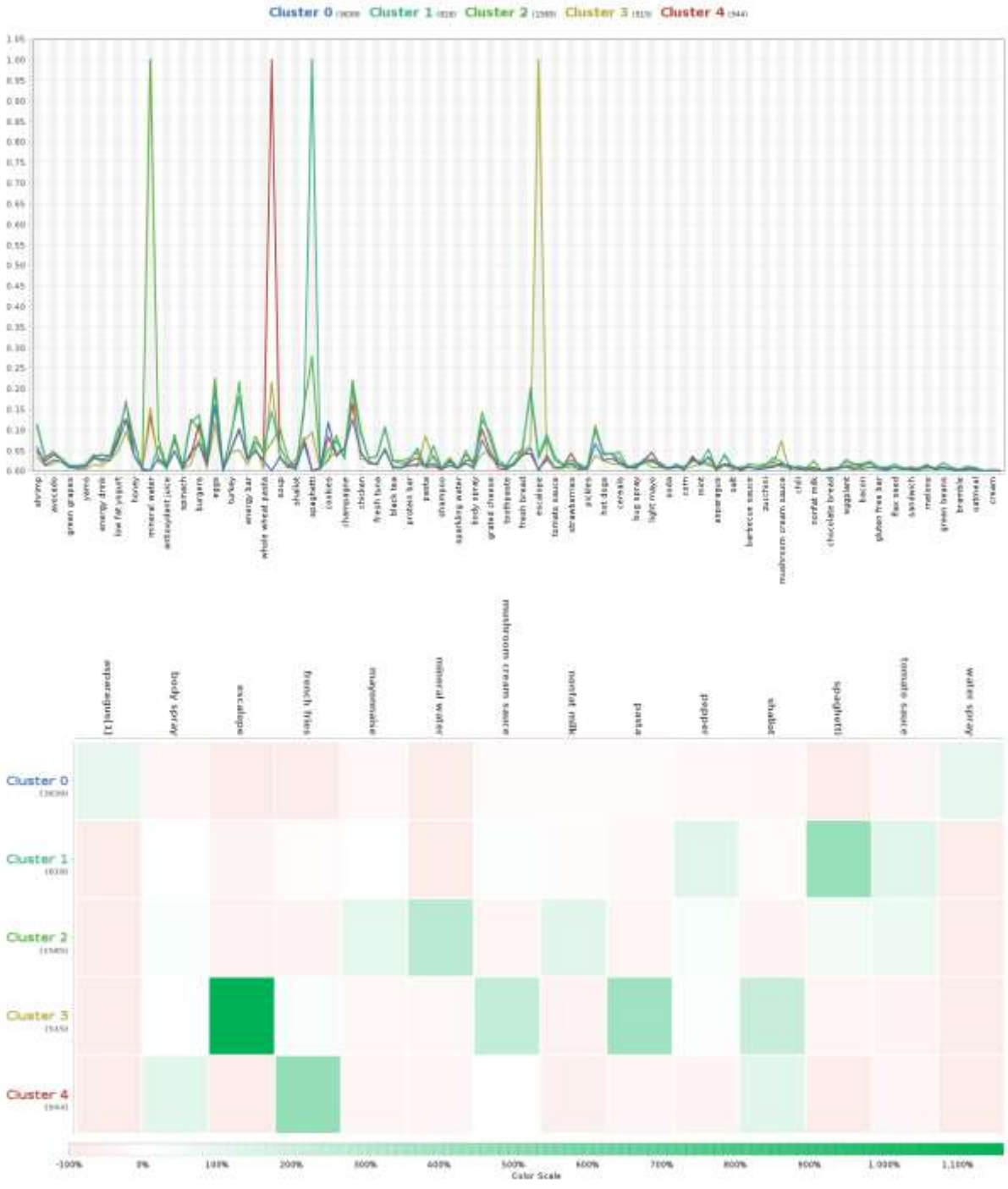
mineral water is on average **319.25%** larger, **nonfat milk** is on average **130.56%** larger, **mayonnaise** is on average **116.05%** larger

Cluster 3 515

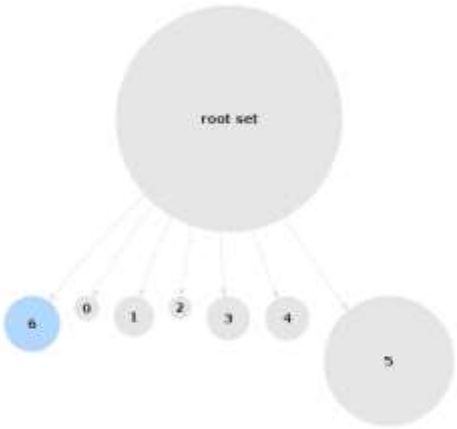
escalope is on average **1,160.67%** larger, **pasta** is on average **430.76%** larger, **mushroom cream sauce** is on average **276.86%** larger

Cluster 4 944

french fries is on average **485.10%** larger, **shallot** is on average **132.90%** larger, **body spray** is on average **121.75%** larger


Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
shrimp
avocado
green grapes
yerba
energy drink
low fat yogurt
honey
mineral water
antioxidant juice
spinach
bulgaria
eggs
turkey
energy bar
whole wheat pasta
soup
shallot
spaghetti
cookies
chemicalpie
chicken
fresh tuna
black tea
protein bar
pasta
shampoo
sparkling water
body spray
grated cheese
toothpaste
fresh bread
excelsior
tomato sauce
strawberries
pickles
hot dogs
cereals
bug spray
light mayo
soda
corn
rice
asparagus
salt
barbecue sauce
zucchini
mushroom cream sauce
chili
nonfat milk
chocolate bread
eggplant
bacon
gluten free bar
flax seed
sandwich
melons
green beans
bramble
oatmeal
cream
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
-100%
0%
100%
200%
300%
400%
500%
600%
700%
800%
900%
1,000%
1,100%
Color Scale

k means (k = 7):



Cluster Model

```
Cluster 0: 495 items
Cluster 1: 798 items
Cluster 2: 447 items
Cluster 3: 867 items
Cluster 4: 879 items
Cluster 5: 2886 items
Cluster 6: 1129 items
Total number of items: 7501
```

Number of Clusters: 7

Cluster 0 495

escalope is on average 1,160.67% larger, **pasta** is on average 439.36% larger, **mushroom cream sauce** is on average 302.68% larger

Cluster 1 798

eggs is on average 456.45% larger, **shallot** is on average 100.00% smaller, **asparagus** is on average 100.00% smaller

Cluster 2 447

asparagus(1) is on average 1,578.08% larger, **ground beef** is on average 917.77% larger, **napkins** is on average 571.23% larger

Cluster 3 867

french fries is on average 485.10% larger, **body spray** is on average 111.26% larger, **shallot** is on average 108.83% larger

Cluster 4 879

mineral water is on average 319.52% larger, **chocolate bread** is on average 113.34% larger, **spaghetti** is on average 100.00% smaller

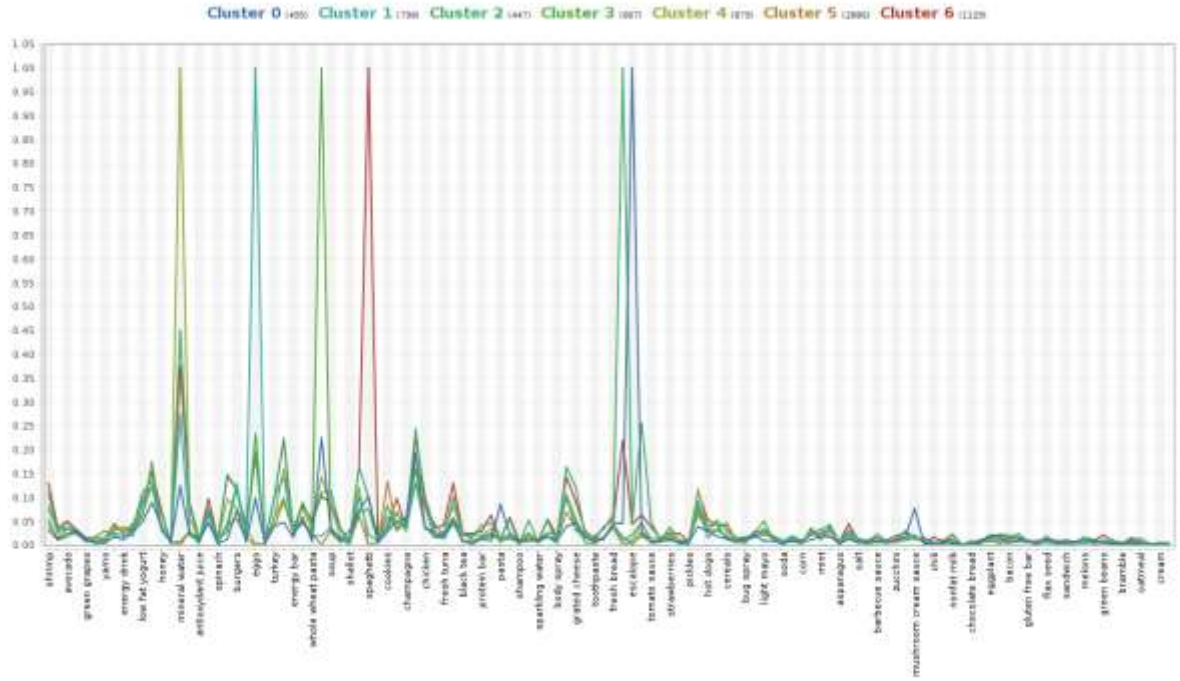
Cluster 5 2,886

water spray is on average 159.91% larger, **mineral water** is on average 100.00% smaller, **eggs** is on average 100.00% smaller

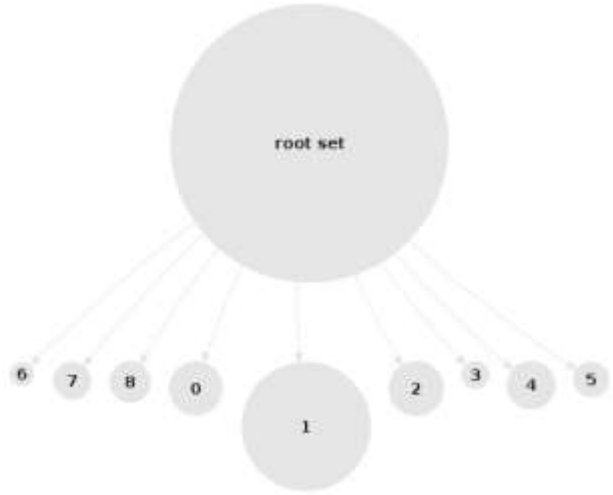
Cluster 6 1,129

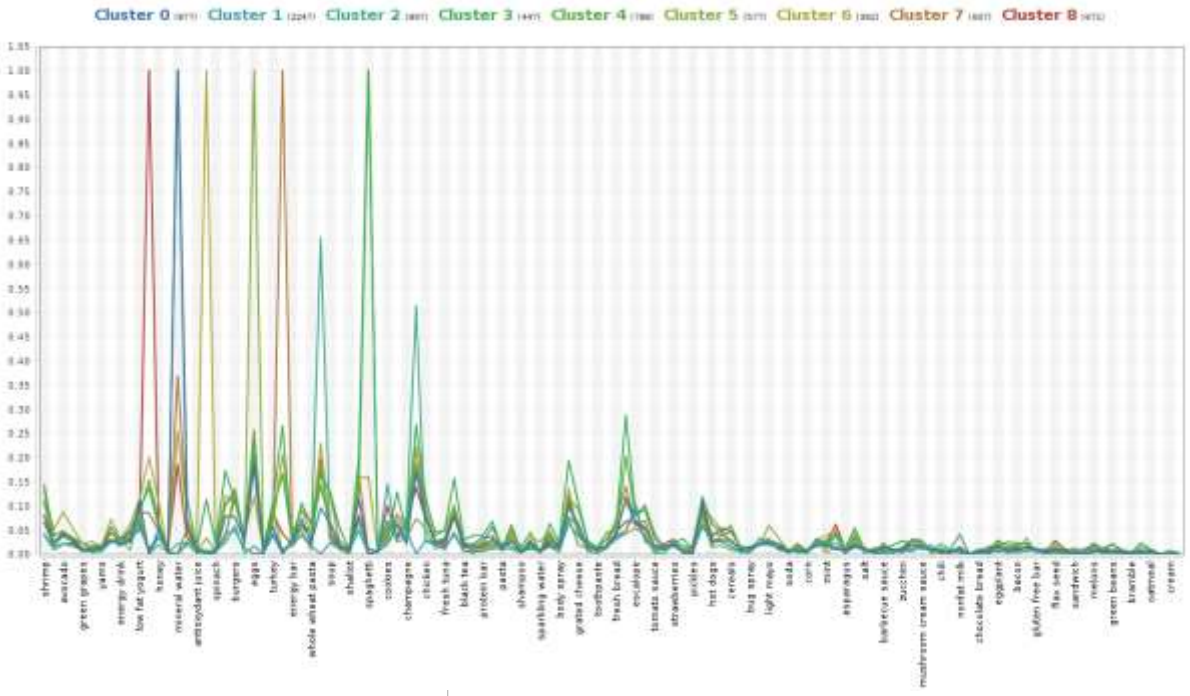
spaghetti is on average 474.35% larger, **strong cheese** is on average 209.29% larger, **tomato sauce** is on average 163.25% larger


antipasto
asparagus(1)
body spray
chocolate bread
eggs
escalopin
french fries
ground beef
mineral water
mushroom cream sauce
napkins
pasta
shallot
spaghetti
strong cheese
tomato sauce
water spray
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
-100%
0%
100%
200%
300%
400%
500%
600%
700%
800%
900%
1,000%
1,100%
1,200%
1,300%
1,400%
1,500%
Color Scale


Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6

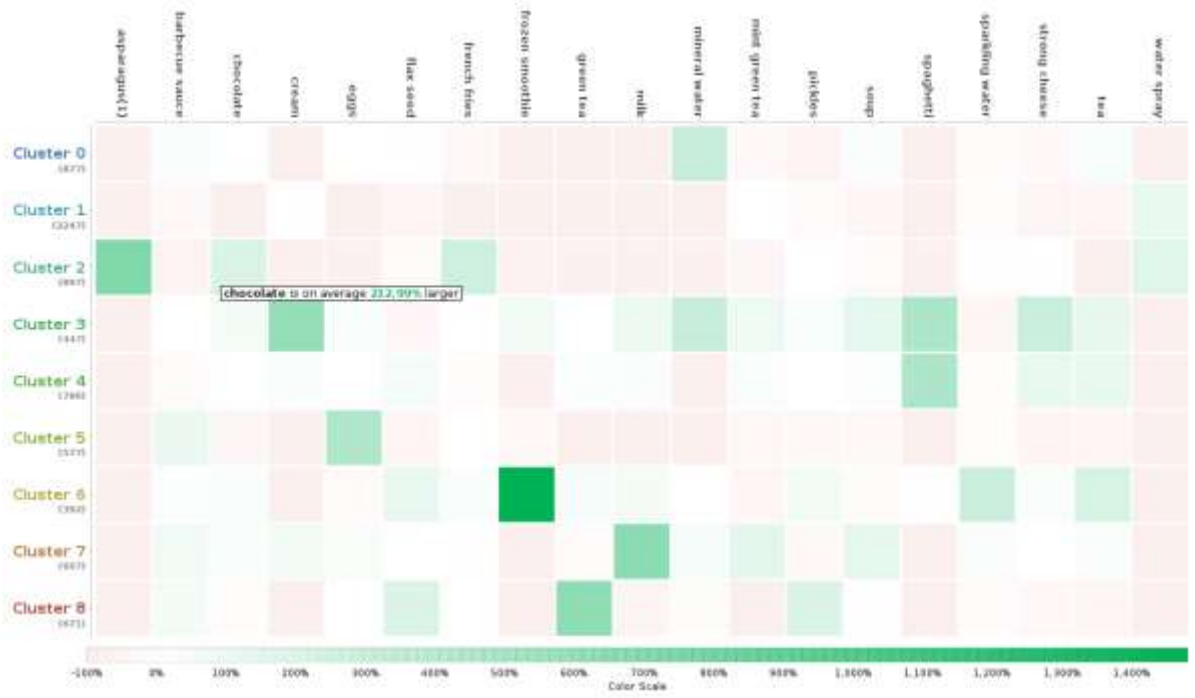
k means (k = 9):





Cluster Model

```
Cluster 0: 877 items
Cluster 1: 2247 items
Cluster 2: 897 items
Cluster 3: 447 items
Cluster 4: 786 items
Cluster 5: 577 items
Cluster 6: 392 items
Cluster 7: 607 items
Cluster 8: 671 items
Total number of items: 7501
```


asparagus(1)
barbecue sauce
chocolate
cream
eggs
flax seed
french fries
frozen smoothie
green tea
milk
mineral water
mint green tea
pickles
soup
spaghetti
sparkling water
strong cheese
tea
water spray
Cluster 0
(877)
Cluster 1
(3247)
Cluster 2
(897)
Cluster 3
(447)
Cluster 4
(2988)
Cluster 5
(519)
Cluster 6
(350)
Cluster 7
(667)
Cluster 8
(671)
chocolate is on average 232.99% larger
-100%
0%
100%
200%
300%
400%
500%
600%
700%
800%
900%
1,000%
1,100%
1,200%
1,300%
1,400%
Color Scale

Number of Clusters: 9

Cluster 0

877

mineral water is on average **319.52%** larger, **green tea** is on average **100.00%** smaller, **frozen smoothie** is on average **100.00%** sma

Cluster 1

2,247

water spray is on average **122.55%** larger, **green tea** is on average **100.00%** smaller, **mineral water** is on average **100.00%** smaller

Cluster 2

897

asparagus(1) is on average **736.23%** larger, **french fries** is on average **282.24%** larger, **chocolate** is on average **212.99%** larger

Cluster 3

447

cream is on average **619.18%** larger, **spaghetti** is on average **474.35%** larger, **mineral water** is on average **319.52%** larger

Cluster 4

786

spaghetti is on average **474.35%** larger, **strong cheese** is on average **130.35%** larger, **tea** is on average **130.35%** larger

Cluster 5

577

eggs is on average **456.45%** larger, **barbecue sauce** is on average **108.64%** larger, **green tea** is on average **100.00%** smaller

Cluster 6

392

frozen smoothie is on average **1,479.16%** larger, **sparkling water** is on average **307.13%** larger, **tea** is on average **229.92%** larger

Cluster 7

607

milk is on average **671.71%** larger, **mint green tea** is on average **164.80%** larger, **soup** is on average **151.06%** larger

Cluster 8

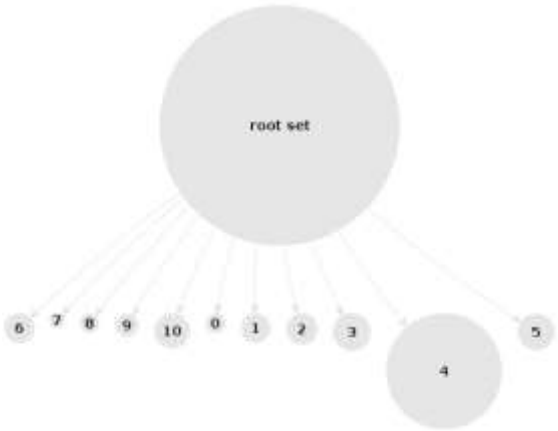
671

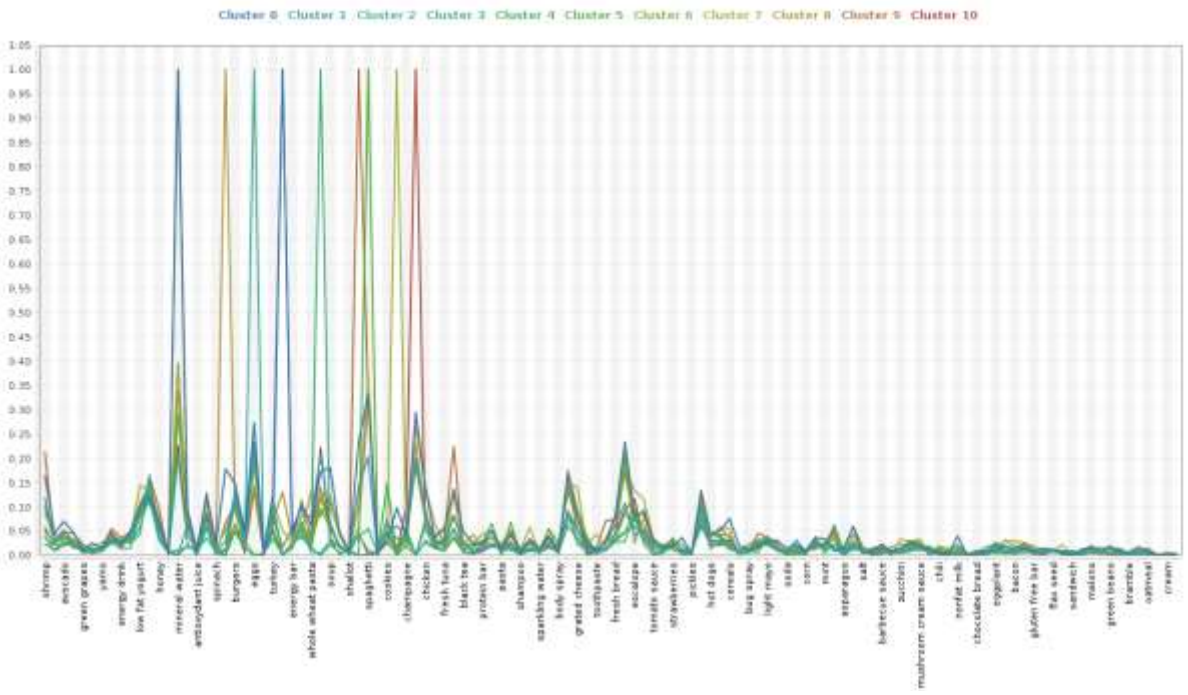
green tea is on average **656.91%** larger, **pickles** is on average **198.10%** larger, **flax seed** is on average **195.91%** larger

k means (k = 11):

Cluster Model

```
Cluster 0: 361 items
Cluster 1: 543 items
Cluster 2: 587 items
Cluster 3: 706 items
Cluster 4: 2350 items
Cluster 5: 684 items
Cluster 6: 560 items
Cluster 7: 279 items
Cluster 8: 353 items
Cluster 9: 425 items
Cluster 10: 653 items
Total number of items: 7501
```





Number of Clusters: 11

Cluster 0

361

milk is on average **671.71%** larger, **strong cheese** is on average **365.72%** larger, **mineral water** is on average **318.36%** larger

Cluster 1

543

milk is on average **671.71%** larger, **cider** is on average **197.26%** larger, **mint green tea** is on average **196.01%** larger

Cluster 2

587

eggs is on average **456.45%** larger, **barbecue sauce** is on average **105.09%** larger, **antioxydant juice** is on average **100.00%** smaller

Cluster 3

706

french fries is on average **485.10%** larger, **body spray** is on average **110.02%** larger, **chutney** is on average **100.00%** smaller

Cluster 4

2,350

mineral water is on average **100.00%** smaller, **olive oil** is on average **100.00%** smaller, **eggs** is on average **100.00%** smaller

Cluster 5

684

spaghetti is on average **474.35%** larger, **cream** is on average **369.99%** larger, **napkins** is on average **338.65%** larger

Cluster 6

560

mineral water is on average **319.52%** larger, **eggplant** is on average **102.95%** larger, **olive oil** is on average **100.00%** smaller

Cluster 7

279

cooking oil is on average **1,858.49%** larger, **toothpaste** is on average **428.89%** larger, **hand protein bar** is on average **313.62%** larger

Cluster 8

353

olive oil is on average **1,418.42%** larger, **whole wheat pasta** is on average **342.29%** larger, **extra dark chocolate** is on average **254.15%** larger

Cluster 9 425

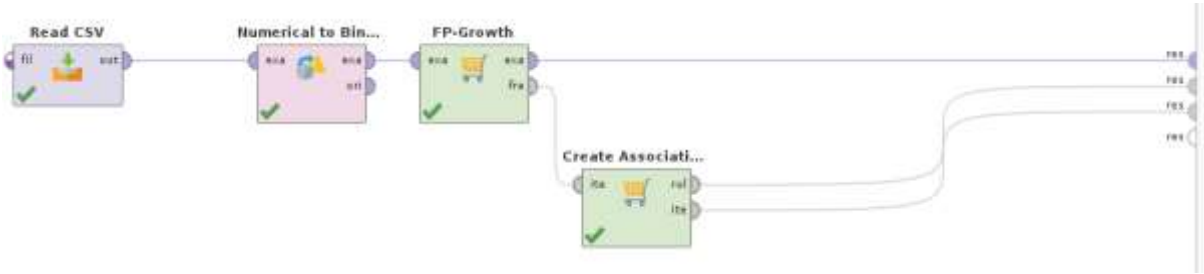
frozen vegetables is on average **949.09%** larger, **water spray** is on average **488.31%** larger, **tea** is on average **265.16%** larger

Cluster 10 653

asparagus(1) is on average **1,048.70%** larger, **chocolate** is on average **510.33%** larger, **water spray** is on average **282.90%** larger

Association Rule :

After learning these parts and a brief document about that, you have to create association rules and compare it with your previous results.



حاصل csv بدست آمده از قسمت قبل را ابتدا به کمک Numerical to binary به صورت لیستی از boolean ها می‌کنیم که هر true یا false نشان دهنده وجود یک آیتم در سبد خرید در آن تراکنش خاص هست سپس به کمک سلول FP growth حاصل را به ازای تمامی حالت‌های ممکن پیدا می‌کنیم و سپس به کمک create Association rule قوانین مربوطه را پیدا می‌کنیم که در پایین مشاهده می‌نمایید همانطور که مشاهده می‌کنید حاصل شبیه به حاصل بدست آمده از قسمت قبل است با در نظر گرفتن confidence بزرگتر از 0.5 که البته به صورت از کوچک به بزرگ مرتب شده است حاصل در اینجا.

و سپس گراف ارتباط را به شکل زیرین نشان خواهیم داد .

Size	Support	Item 1	Item 2	Item 3	Item 4
3	0.006	french fries	chocolate	frozen vegetables	
3	0.005	french fries	chocolate	pancakes	
3	0.005	french fries	chocolate	burgers	
3	0.005	french fries	chocolate	escalope	
3	0.005	french fries	green tea	pancakes	
3	0.005	french fries	green tea	burgers	
3	0.005	french fries	green tea	cookies	
3	0.006	french fries	milk	frozen vegetables	
3	0.006	french fries	milk	burgers	
3	0.005	chocolate	green tea	milk	
3	0.006	chocolate	milk	ground beef	
3	0.006	chocolate	milk	frozen vegetables	
3	0.005	chocolate	milk	pancakes	
3	0.005	chocolate	milk	burgers	
3	0.005	chocolate	milk	shrimp	
3	0.005	chocolate	milk	olive oil	
3	0.005	chocolate	milk	chicken	
3	0.006	chocolate	ground beef	frozen vegetables	
3	0.005	chocolate	frozen vegetables	shrimp	
3	0.006	milk	ground beef	frozen vegetables	
3	0.005	milk	ground beef	olive oil	
3	0.005	milk	frozen vegetables	shrimp	
3	0.005	milk	frozen vegetables	olive oil	
4	0.005	mineral water	eggs	spaghetti	chocolate
4	0.005	mineral water	spaghetti	chocolate	milk
4	0.005	mineral water	spaghetti	milk	frozen vegetables

No.	Premises	Conclusion	Support	Confidence	LaPlace	Gain	p-s	Lift	Convi...
10	ground beef, pancakes	mineral water	0.007	0.514	0.993	-0.022	0.004	2.155	1.566
11	chocolate, chicken	mineral water	0.008	0.518	0.993	-0.022	0.004	2.174	1.581
12	ground beef, soup	mineral water	0.005	0.521	0.995	-0.014	0.003	2.184	1.589
13	ground beef, shrimp	spaghetti	0.006	0.523	0.995	-0.017	0.004	3.005	1.732
14	spaghetti, soup	mineral water	0.007	0.523	0.993	-0.021	0.004	2.196	1.598
15	pancakes, chicken	mineral water	0.005	0.529	0.996	-0.013	0.003	2.221	1.618
16	milk, turkey	mineral water	0.006	0.541	0.995	-0.017	0.003	2.270	1.660
17	ground beef, frozen vegetables	mineral water	0.009	0.543	0.992	-0.025	0.005	2.279	1.668
18	eggs, soup	mineral water	0.005	0.544	0.996	-0.013	0.003	2.283	1.671
19	eggs, cooking oil	mineral water	0.006	0.545	0.995	-0.017	0.004	2.288	1.676
20	spaghetti, milk, frozen vegetables	mineral water	0.005	0.548	0.996	-0.012	0.003	2.301	1.686
21	chocolate, soup	mineral water	0.006	0.553	0.996	-0.015	0.003	2.318	1.702
22	shrimp, olive oil	mineral water	0.005	0.557	0.996	-0.012	0.003	2.338	1.721
23	milk, soup	mineral water	0.009	0.561	0.993	-0.022	0.005	2.355	1.737
24	ground beef, cooking oil	spaghetti	0.005	0.571	0.996	-0.012	0.003	3.282	1.927
25	frozen vegetables, olive oil	mineral water	0.007	0.576	0.995	-0.016	0.004	2.418	1.798
26	olive oil, soup	mineral water	0.005	0.582	0.996	-0.013	0.003	2.442	1.822
27	pancakes, cooking oil	mineral water	0.005	0.593	0.997	-0.011	0.003	2.489	1.872
28	frozen vegetables, soup	mineral water	0.005	0.633	0.997	-0.011	0.003	2.657	2.077