Homework 4:
INFO 523
Group 1
Matt Miller
Kai Blumberg

6.3 The Apriori algorithm makes use of prior knowledge of subset support properties.

(a) Prove that all nonempty subsets of a frequent itemset must also be frequent.

s = frequent itemset, *ss* = any non-empty subset of *s*, *mst* = minimum support threshold, *D* = total dataset of transactions, *sc(x)* = support count of itemset *x*.
The minimum support count threshold, *msct*, is:

$$msct = |D| * mst$$

Because *s* is a frequent subset, we know that the support count of *s* is at least *msct:*

$$sc(s) >= msct$$

Any transaction that contains itemset *s* will also contain itemset *ss*, and there could also be transactions in *D* that contain itemset *ss* but not itemset *s*, therefore:

$$sc(ss) >= sc(s) >= msct$$

So itemset *ss* must be a frequent itemset.

(b) Prove that the support of any nonempty subset s′ of itemset s must be at least as great as the support of s.

See part a for some explanation. Every transaction that contains itemset *s* must also contain itemset *s'*, but not every transaction that contains itemset *s*' must contain itemset *s.* For example, itemset *s* = {bread, eggs, milk} and sub-itemset *s'* = {bread, eggs}. Every transaction with *s* contains *s'*, but there could also be transactions from lactose-intolerant people that only buy bread and eggs but not milk. Therefore, the support of *s'* must be greater than or equal to the support of *s.*

(c) Given frequent itemset l and subset s of l, prove that the confidence of the rule "s′ ⇒(l−s′)" cannot be more than the confidence of "s⇒(l−s)," where s′ is a subset of s.

For some s which is a subset of l, the confidence for s⇒(l−s) = support(l) / support(s)
For some s′ which is any non empty subset of s the confidence of s′⇒(l−s′) = support(l) / support(s′).

As was shown in part b, the support(s′) >= support(s). Thus we can infer that the confidence for s⇒(l−s) >= confidence of “s′⇒(l−s′)”. Thus the confidence rule s′ ⇒(l−s′) cannot be more than the confidence of the rule “s⇒(l−s)”.

(d) A partitioning variation of Apriori subdivides the transactions of a database D into n nonoverlapping partitions. Prove that any itemset that is frequent in D must be frequent in at least one partition of D.

To prove that any itemset that is frequent in D must be frequent in at least one partition of D by contradiction we can do the following.

Starting with the assumption that the itemset is not frequent in any of the partitions of D, We will define F to be any frequent itemset. Will will define D to be the task-relevant data, which comprises a set of database transactions. We will define C to be the total number of transactions in D. We will let A be the total number of transactions in D containing the itemset F. Finally we will let min_sup be the minimum support.

If F is a frequent itemset then we can say that A = C × min_sup. If we partition the frequent itemset D into n non overlapping partitions, $d_1,d_2,...,d_n$, we get D = $d_1d_2d_3 ...d_n$.

If we have the elements $c_1c_2c_3 ...c_n$, be the total number of transactions in partitions $d_1d_2d_3 ...d_n$, respectively, then we can say that $C = c_1 + c_2 + c_3 … + c_n$.

If we let $a_1a_2a_3 ...a_n$, be the total number of transactions in partitions $d_1,d_2,...,d_n$ which contain the itemset F, respectively, then we can say that $A = a_1 + a_2 + a_3 … + a_n$.

From that we can rewrite A = C × min_sup as $(a_1 + a_2 + a_3 … + a_n) = (c_1 + c_2 + c_3 … + c_n)$ × min_sup.

At the beginning of the proof, we started with the assumption that F is not frequent in any of the partitions $d_1,d_2,d_3...,d_n$ of D. Which means that $a_1 < c_1$ × min_sup; $a_2 < c_2$ × min_sup ; $a_3 < c_3$ × min_sup; … ; $a_n < c_n$ × min_sup.

When we add up these inequalities, we get $(a_1 + a_2 + a_3 … + a_n) < (c_1 + c_2 + c_3 … + c_n) \times s$. Stated in a simpler way, A < C × min_sup, which means that F is not a frequent itemset.

Here we find a contradiction as we had defined F as a frequent itemset at the beginning of the proof, hence proving that any itemset that is frequent in D must be frequent in at least one partition of D.

6.6 (no R) A database has five transactions. Let min sup = 60% and min conf = 80%.

TID	items bought
T100	{M,O,N,K,E,Y}
T200	{D,O,N,K,E,Y}
T300	{M,A,K,E}
T400	{M,U,C,K,Y}
T500	{C,O,O,K,I,E}

(a) Find all frequent itemsets using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.

|D| = 5, minimum support count = 5 * .6 = 3

Apriori:

Generate C1

C1

{M}	3
{O}	3
{N}	2
{K}	5
{E}	4
{Y}	3
{D}	1
{A}	1
{U}	1
{C}	2

{I}	1

Filter out itemsets with counts 2 or less, generate frequent itemsets L1

L1

{M}	3
{O}	3
{K}	5
{E}	4
{Y}	3

Generate C2 candidates from L1, get counts

C2

{M,O}	1
{M,K}	3
{M,E}	2
{M,Y}	2
{O,K}	3
{O,E}	3
{O,Y}	2
{K,E}	4
{K,Y}	3
{E,Y}	2

Generate L2

L2

{M,K}	3
{O,K}	3
{O,E}	3
{K,E}	4
{K,Y}	3

Generate C3

C3

{O,K,E}	3
{K,E,Y}	2

Generate L3

L3

{O,K,E}	3

FP-growth

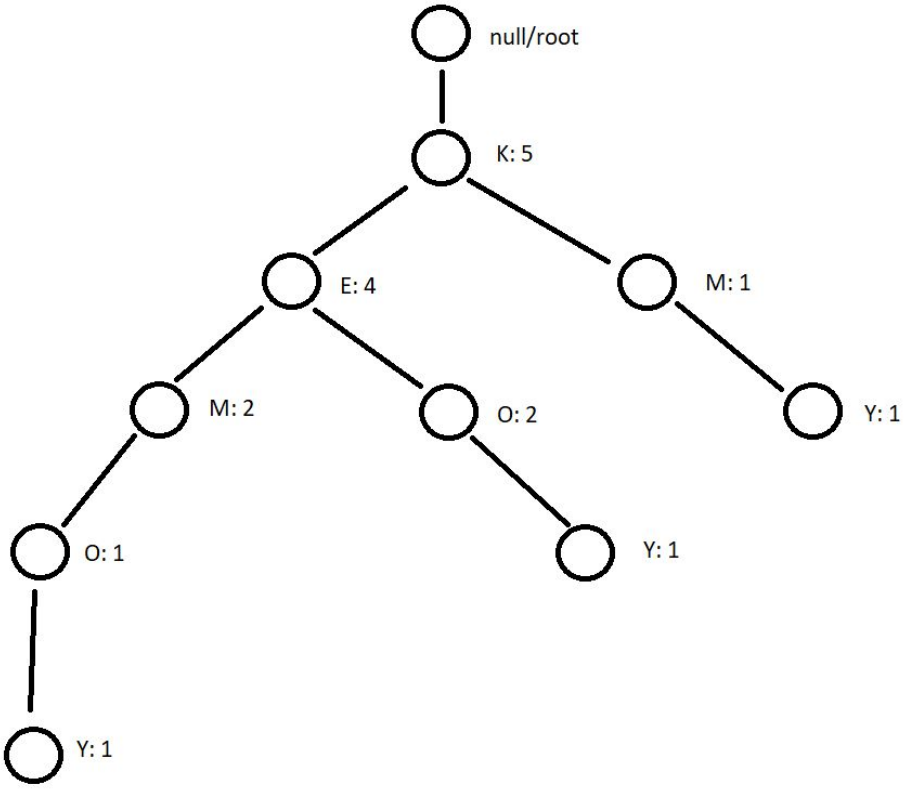
Generate C1

C1

{M}	3
{O}	3
{N}	2
{K}	5
{E}	4
{Y}	3
{D}	1
{A}	1
{U}	1
{C}	2
{I}	1

Generate L1

L1 = {{K: 5}, {E: 4}, {M: 3}, {O: 3}, {Y: 3}}



item	Conditional pattern base	Conditional tree	Frequent patterns
Y	{{K,E,M,O: 1}, {K,E,O: 1}, {K,M: 1}}	K: 3	{K,Y: 3}
O	{{K,E,M: 1}, {K,E: 2}}	K: 3, E: 3	{K,O: 3}, {E,O: 3}, {K,E,O: 3}
M	{{K,E: 2}, {K: 1}}	K: 3	{K,M: 3}
E	{{K: 4}}	K: 4	{K,E: 4}

A-priori needs multiple scans through the dataset, while FP-growth builds its tree with only 2 scans (1-itemsets scan and tree-building scan). Candidate generation is expensive for A-priori, while FP-growth doesn't generate candidates.

(b) List all the strong association rules (with support s and confidence c) matching the following metarule, where X is a variable representing customers, and item$_i$ denotes variables representing items (e.g., "A," "B,"):
$\forall x \in$transaction,buys(X,item1)$\land$buys(X,item2)$\Rightarrow$buys(X,item3) [s,c]

Support({O,K,E}) = 0.6
Confidence(O,K -> E) = 3/3 = 1
Confidence(O,E -> K) = 3/3 = 1
Confidence(K,E -> O) = ¾ =0.75
O,K -> E [0.6, 1]
E,O -> K [0.6, 1]

This video the instructor made should explain https://www.screencast.com/t/TaV8rW2y should explain how this is done.

6.8 (R) A database has four transactions. Let min sup = 60% and min conf = 80%.

cust ID	TID	items bought (in the form of brand-item category)
01	T100	{King's-Crab, Sunset-Milk, Dairyland-Cheese, Best-Bread}
02	T200	{Best-Cheese, Dairyland-Milk, Goldenfarm-Apple, Tasty-Pie, Wonder-Bread}
01	T300	{Westcoast-Apple, Dairyland-Milk, Wonder-Bread, Tasty-Pie}
03	T400	{Wonder-Bread, Sunset-Milk, Dairyland-Cheese}

(a) At the granularity of item category (e.g., $item_i$ could be "Milk"), for the rule template,
$\forall X \in transaction, buys(X,item1) \wedge buys(X,item2) \Rightarrow buys(X,item3)\ [s,c]$,

list the frequent k-itemset for the largest k, and all the strong association rules (with their support s and confidence c) containing the frequent k-itemset for the largest k.

```
> inspect(subset(apri_c, subset=is.maximal(apri_c)))
    lhs               rhs     support confidence lift count
[1] {Bread,Cheese} => {Milk}  0.75    1          1    3
[2] {Cheese,Milk}  => {Bread} 0.75    1          1    3
```

(b) At the granularity of brand-item category (e.g., itemi could be “Sunset-Milk”), for the rule template,

∀X ∈customer, buys(X,item1)∧buys(X,item2)⇒buys(X,item3), list the frequent k-itemset for the largest k (but do not print any rules).

```
> inspect(subset(freqsets_b, subset=is.maximal(freqsets_b)))
    items                                       support   count
[1] {Dairyland-Cheese,Sunset-Milk,Wonder-Bread} 0.6666667 2
[2] {Dairyland-Milk,Tasty-Pie,Wonder-Bread}     0.6666667 2
```

6.13 Give a short example to show that items in a strong association rule actually may be negatively correlated.

Items X and Y are in the table.

	X	!X	Σrow
Y	35	15	50
!Y	25	5	30
Σrow	60	20	80

We set our minimum support threshold = 40%, minimum confidence threshold = 60%. For this dataset, X->Y is a strong rule, because support for X->Y = 35/80 = 43.75%, and confidence = 35/50 = 70%. The lift for X->Y = support(X->Y)/P(X)*P(Y) = .4375/(0.75*0.625) = 0.933. Since the lift is less than 1, X and Y must be negatively correlated.

6.14 The following contingency table summarizes supermarket transaction data, where hot dogs refers to the transactions containing hot dogs, ~~hot dogs~~ refers to the transactions that do not contain hot dogs, hamburgers refers to the transactions containing hamburgers, and ~~hamburgers~~ refers to the transactions that do not contain hamburgers.

	hot dogs	~~hot dogs~~	Σ_{row}
hamburgers	2000	500	2500
~~hamburgers~~	1000	1500	2500
Σ_{col}	3000	2000	5000

(a) Suppose that the association rule "hot dogs ⇒ hamburgers" is mined. Given a minimum support threshold of 25% and a minimum confidence threshold of 50%, is this association rule strong?

Mining the association rule: "hot dogs ⇒ hamburgers", we get a support which = 2000/5000 = 40%. The confidence is = 2000/3000 = 66.7%. Therefore, we observe the association rule is strong.

(b) Based on the given data, is the purchase of hot dogs independent of the purchase of hamburgers? If not, what kind of correlation relationship exists between the two?

The correlation between A and B, corr{A,B} = P({A, B})/(P({A}) * P({B})), substituting in hotdog and hamburger for A and B: corr{hotdog,hamburger} = P({hot dog, hamburger})/(P({hot dog}) * P({hamburger})). Such calculation is also referred to as the lift. As Lift: which is given by the formula = P({A,B})/(P({A})×P({B}))

Thus P({hamburger}))=0.4/(0.5 × 0.6) = 1.33. 1.33 is > 1, thus the purchase of hotdogs is not independent of the purchase of hamburgers, and there is a positive correlation between them.

(c) Compare the use of the all_confidence, max_confidence, Kulczynski, and cosine measures with lift and correlation on the given data.

all_confidence:
all_conf(A,B) = sup(A U B) / max { sup(A), sup(B)} = min{P(A|B), P(B|A)}
all_conf(hot dog , hamburger) = sup(hot dog U hamburger) / max { sup(hot dog), sup(hamburger)} = min{P(hot dog|hamburger), P(hamburger|hot dog)}
all_conf(hot dog , hamburger) = min { (0.8), (0.67) } = 0.67

max_confidence:
Max_conf(A| B) = max{P(A|B), P(B|A)}
Max_conf(hot dog| hamburger) = max{P(hot dog|hamburger), P(hamburger|hot dog)}
Max_conf(hot dog| hamburger) = max{ (0.8), (0.67) } = 0.8

Kulczynski:
Kulc(A, B) = ½(P(A | B) + P(A | B))
Kulc(hot dog, hamburger) = ½(P(hot dog | hamburger) + P(hamburger | hot dog))
Kulc(hot dog, hamburger) = ½ (0.8 + 0.67) = 0.735

Cosine:
cosine(A, B) = sqrt(P(A | Br) * P(A | B))
cosine(hot dog, hamburger) = sqrt(P(hot dog | hamburger) * P(hamburger | hot dog))
cosine(hot dog, hamburger) = sqrt(0.8 * 0.67) = sqrt(0.536) = 0.732

Thus in summary, Cosine and Kulczynski measures are very similar to one another (0.732, 0.735 respectively) but are much lower than the lift of 1.33, whereas all_confidence and max_confidence differ more widely from one another (0.67 and 0.8 respectively) with the max_confidence being closest to the lift of 1.33.