B565 HW4

Chia-Hsuan Chou (chou5)

1. For drawing ROC curve and calculate the area, we can first generate the posterior probability for each instance and sort it based on the decreasing order. Then, give it threshold and calculate the TP and FP by count the number of TP, FP, TN, FN at each threshold. Therefore, the maximum number of thresholds will be n+(-1) and the true class will be like +,-,+,-,+,-…. And the minimum number of thresholds will be 1 and the case will look like …..+,+,+,+,-,-,-,-……

2. Let's say we have n=5 data points and we have already known the distance between any two points which we calculate it by the Euclidean distances. Now assume that we have three cluster which represents m=3 and we know that p1 is belong to c1 (cluster 1); p2, p3, p4 are belong to c2 (cluster2) and p5 is belong to c3 (cluster 3). Then, we can easily get that {d11} is belong to c1; {d22, d23, d24, d33, d44, d34} is belong to c2, {d55} is belong to c3. Third, we use D to stand for the set of all the indexes in each cluster which will be D1={1}, D2={2,3,4}, D3={5} so that we can calculate the sum of the distance if we combine D1 with D2 and D1 with D3. For example, for c1, we can get d12, d13, d14 according to D2 and d15 according to D3. For c2, we can get d25, d35, d45, according to D3. Finally we can merge two cluster by comparing the sum of their distance between different clusters and get the smallest sum in order to merge. For example, if d12, d13, d14 is the smallest sum, then we can merge c1 and c2.

3.
   (a) Following result is by running "q3_a.py" (used F1*Fk-1 method) and "q3_a-2.py" (used Fk-1*Fk-1 method) which take "item_1000.txt" for input and reference the attribute file called "goods.txt" and it is from taking minimum support=0.02 for threshold.

Method	F1*Fk-1	Fk-1*Fk-1
Numbers of total candidates	3930	122
Run Time (sec)	1.5971	0.1675

   By seeing the table, we can easily find out that using Fk-1*Fk-1 method is better than using F1*Fk-1 method since it generates few candidates. Also, Fk-1*Fk-1 method decrease the run time. (The generated candidate item sets and the total number of frequent item sets is in the output file after you run the program.)

   (b) Following result is by running "q3_b.py". The input will be three different data sets from UCI Machine learning repository and they are "item_1000.txt", "car.txt" and "chess.txt". The level of the minimum support threshold are shown in the table below. Their output file will contain generated frequent item sets and candidates in detail. The table below is just for comparing the difference.

Data sets		item_1000.txt (1000 examples)	
Method		F1*Fk-1	Fk-1*Fk-1
Min_sup=0.02	Number of candidate	3930	122
	Number of Frequent Item sets	110	110
	Run time (sec)	1.687	0.1938
Min_sup=0.05	Number of candidate	820	42
	Number of Frequent Item sets	42	42
	Run time (sec)	0.3396	0.0862
Min_sup=0.08	Number of candidate	18	18
	Number of Frequent Item sets	18	18
	Run time (sec)	0.0985	0.0952

Data sets		car.txt (1728 examples)	
Method		F1*Fk-1	Fk-1*Fk-1
Min_sup=0.05	Number of candidate	2032	1634
	Number of Frequent Item sets	349	349
	Run time (sec)	2.2224	0.9465
Min_sup=0.1	Number of candidate	993	172
	Number of Frequent Item sets	86	86
	Run time (sec)	0.7092	0.1715
Min_sup=0.2	Number of candidate	276	31
	Number of Frequent Item sets	31	31
	Run time (sec)	0.2988	0.1092

Data sets		chess.txt (28056 examples)	
Method		F1*Fk-1	Fk-1*Fk-1
Min_sup=0.05	Number of candidate	1225	123
	Number of Frequent Item sets	123	123
	Run time (sec)	33.7369	12.3161
Min_sup=0.08	Number of candidate	946	50
	Number of Frequent Item sets	50	50
	Run time (sec)	10.1027	2.1376
Min_sup=0.1	Number of candidate	780	46

	Number of Frequent Item sets	46	46
	Run time (sec)	8.4067	1.9287

By seeing the table, we can easily find out that using Fk-1*Fk-1 method is better than using F1*Fk-1 method since it generates few candidates. Also, Fk-1*Fk-1 method decrease the run time. Also, minimum support threshold influence the number of the generation of candidates a lot. Also, we can find out that when the minimum support increase, the number of candidate and the number of frequent item set will approach the same; however, it performs much obvious when using fk-1*fk-1 method.

(c) For this question, the number of frequent item sets is based on question 2 and the detail is also shown in running “q2_b.py”. Closed Item sets and Maximal Item Sets can be gotten by running “q3_c.py”. The minimum support and the data set used here is as same as in question 2.

Data sets		item_1000.txt (1000 examples)	
Method		F1*Fk-1	Fk-1*Fk-1
Min_sup=0.02	Number of Frequent Item Sets	110	110
	Number of Closed Item Sets	99	99
	Number of Maximal Item Sets	31	31
Min_sup=0.05	Number of Frequent Item Sets	42	42
	Number of Closed Item Sets	42	42
	Number of Maximal Item Sets	38	38
Min_sup=0.08	Number of Frequent Item sets	18	18
	Number of Closed Item Sets	18	18
	Number of Maximal Item Sets	18	18

Data sets		car.txt (1728 examples)	
Method		F1*Fk-1	Fk-1*Fk-1
Min_sup=0.05	Number of Frequent Item Sets	349	349
	Number of Closed Item Sets	309	309
	Number of Maximal Item Sets	195	195
Min_sup=0.1	Number of Frequent Item Sets	86	86
	Number of Closed Item Sets	73	73
	Number of Maximal Item Sets	43	43
Min_sup=0.2	Number of Frequent Item sets	31	31

	Number of Closed Item Sets	29	29
	Number of Maximal Item Sets	22	22

Data sets		chess.txt (28056 examples)	
Method		F1*Fk-1	Fk-1*Fk-1
Min_sup=0.05	Number of Frequent Item Sets	123	123
	Number of Closed Item Sets	121	121
	Number of Maximal Item Sets	80	80
Min_sup=0.08	Number of Frequent Item Sets	50	50
	Number of Closed Item Sets	50	50
	Number of Maximal Item Sets	43	43
Min_sup=0.1	Number of Frequent Item sets	46	46
	Number of Closed Item Sets	46	46
	Number of Maximal Item Sets	39	39

It is obviously to see that either use f1*fk-1 method or fk-1*fk-1 method does not change the result of the frequent item sets, closed item sets and maximal item sets. Also we can find out that the number of frequent item sets ≧ the number of closed item sets ≧ the number of maximal item sets no matter in which minimum support threshold we provides.

(d) Following outcomes can be gotten by running "q3_d.py". Here I use the same data sets as previous question but use different minimum support for the threshold, since if I use the same minimum support as previous question, then I'll get very less rules and will not be able for the comparison. Also, I set the minimum confidence threshold for each data set respectively which is shown in the table below. The number of candidate by using brute-force can be calculated in the program as well and will be printed on the screen after executing the program, but the output will not show any candidate information about using brute-force method. The calculation of brute-force method is all based on the equation which is 2^k-2. Below is the table for comparison the requirement given from the question 3(d).

Data sets		item_1000.txt (1000 examples)	
Method		Confidence-based pruning	Brute-force
Min_sup=0.02	Min_conf=0.3	115	430
	Min_conf=0.5	60	320
	Min_conf=0.7	38	276
Min_sup=0.03	Min_conf=0.3	76	264

	Min_conf=0.5	41	194
	Min_conf=0.7	16	120
Min_sup=0.04	Min_conf=0.3	27	66
	Min_conf=0.5	20	52
	Min_conf=0.7	3	18

Data sets		car.txt (1728 examples)	
Method		Confidence-based pruning	Brute-force
Min_sup=0.03	Min_conf=0.3	163	978
	Min_conf=0.5	73	446
	Min_conf=0.7	23	136
Min_sup=0.04	Min_conf=0.3	134	708
	Min_conf=0.5	61	318
	Min_conf=0.7	21	114
Min_sup=0.05	Min_conf=0.3	113	574
	Min_conf=0.5	50	248
	Min_conf=0.7	11	60

Data sets		chess.txt (28056 examples)	
Method		Confidence-based pruning	Brute-force
Min_sup=0.02	Min_conf=0.3	181	474
	Min_conf=0.5	28	108
	Min_conf=0.7	11	50
Min_sup=0.03	Min_conf=0.3	156	344
	Min_conf=0.5	19	54
	Min_conf=0.7	6	20
Min_sup=0.04	Min_conf=0.3	102	216
	Min_conf=0.5	13	30
	Min_conf=0.7	5	14

By observing the table above, we can find out that using confidence-based pruning can decrease a lot of redundant association rules. Also, by giving each data set different minimum support and minimum confidence threshold, we can find out that high minimum support and high minimum confidence threshold decrease the generation of association rule quickly which means the data set does not show high cluster in specific data points.

(e) Following result can get by running "q3_f.py". The output file shows the top ten association rules for the confidence threshold which you entered when you typed the command line for running the program. Here I choose several confidence and support threshold and list the result below. Also, I consider the situation that several association rules have same confidence and I decide to use their support to generate the ranking. Therefore, it means that the top ten association rules is based on considering confidence first, then their support second. (The output file also shows the top ten rules for the lift threshold which is the result for question f. Therefore, I'll discuss it in the next question.)

Data sets		item_1000.txt (1000 examples)
Method		Top 10 association rules (ex: rules: confidence/support)
Min_sup=0.01	Min_conf=0.3	{Chocolate Croissant}->{Chocolate Tart}: 0.6133/0.0460 {Vanilla Meringue}->{Marzipan Cookie}: 0.5889/0.0530 {Strawberry Cake}->{Ganache Cookie}: 0.5631/0.0580 {Single Espresso}->{Hot Coffee}: 0.5595/0.0470 {Apple Pie}->{Almond Croissant}: 0.5570/0.0440 {Lemon Cake}->{Single Espresso}: 0.5556/0.0400 {Hot Coffee}->{Single Espresso}: 0.5529/0.0470 {Chocolate Tart}->{Chocolate Croissant}: 0.5476/0.0460 {Vanilla Eclair}->{Opera Cake}: 0.5444/0.0490 {Lemon Cake}->{Hot Coffee}: 0.5417/0.0390
	Min_conf=0.4	{Chocolate Croissant}->{Chocolate Tart}: 0.6133/0.0460 {Vanilla Meringue}->{Marzipan Cookie}: 0.5889/0.0530 {Strawberry Cake}->{Ganache Cookie}: 0.5631/0.0580 {Single Espresso}->{Hot Coffee}: 0.5595/0.0470 {Apple Pie}->{Almond Croissant}: 0.5570/0.0440 {Lemon Cake}->{Single Espresso}: 0.5556/0.0400 {Hot Coffee}->{Single Espresso}: 0.5529/0.0470 {Chocolate Tart}->{Chocolate Croissant}: 0.5476/0.0460 {Vanilla Eclair}->{Opera Cake}: 0.5444/0.0490 {Lemon Cake}->{Hot Coffee}: 0.5417/0.0390
	Min_conf=0.5	{Chocolate Croissant}->{Chocolate Tart}: 0.6133/0.0460 {Vanilla Meringue}->{Marzipan Cookie}: 0.5889/0.0530 {Strawberry Cake}->{Ganache Cookie}: 0.5631/0.0580 {Single Espresso}->{Hot Coffee}: 0.5595/0.0470 {Apple Pie}->{Almond Croissant}: 0.5570/0.0440 {Lemon Cake}->{Single Espresso}: 0.5556/0.0400 {Hot Coffee}->{Single Espresso}: 0.5529/0.0470 {Chocolate Tart}->{Chocolate Croissant}: 0.5476/0.0460 {Vanilla Eclair}->{Opera Cake}: 0.5444/0.0490 {Lemon Cake}->{Hot Coffee}: 0.5417/0.0390

Min_sup=0.02	Min_conf=0.3	{Chocolate Croissant}->{Chocolate Tart}: 0.6133/0.0460 {Vanilla Meringue}->{Marzipan Cookie}: 0.5889/0.0530 {Strawberry Cake}->{Ganache Cookie}: 0.5631/0.0580 {Single Espresso}->{Hot Coffee}: 0.5595/0.0470 {Apple Pie}->{Almond Croissant}: 0.5570/0.0440 {Lemon Cake}->{Single Espresso}: 0.5556/0.0400 {Hot Coffee}->{Single Espresso}: 0.5529 /0.0470 {Chocolate Tart}->{Chocolate Croissant}: 0.5476/0.0460 {Vanilla Eclair}->{Opera Cake}:0.5444/0.0490 {Lemon Cake}->{Hot Coffee}:0.5417/0.0390
	Min_conf=0.4	{Chocolate Croissant}->{Chocolate Tart}: 0.6133/0.0460 {Vanilla Meringue}->{Marzipan Cookie}: 0.5889/0.0530 {Strawberry Cake}->{Ganache Cookie}: 0.5631/0.0580 {Single Espresso}->{Hot Coffee}: 0.5595/0.0470 {Apple Pie}->{Almond Croissant}: 0.5570/0.0440 {Lemon Cake}->{Single Espresso}: 0.5556/0.0400 {Hot Coffee}->{Single Espresso}: 0.5529/0.0470 {Chocolate Tart}->{Chocolate Croissant}: 0.5476/0.0460 {Vanilla Eclair}->{Opera Cake}: 0.5444/0.0490 {Lemon Cake}->{Hot Coffee}: 0.5417/0.0390
	Min_conf=0.5	{Chocolate Croissant}->{Chocolate Tart}: 0.6133/0.0460 {Vanilla Meringue}->{Marzipan Cookie}: 0.5889/0.0530 {Strawberry Cake}->{Ganache Cookie}: 0.5631/0.0580 {Single Espresso}->{Hot Coffee}: 0.5595/0.0470 {Apple Pie}->{Almond Croissant}: 0.5570/0.0440 {Lemon Cake}->{Single Espresso}: 0.5556/0.0400 {Hot Coffee}->{Single Espresso}: 0.5529/0.0470 {Chocolate Tart}->{Chocolate Croissant}: 0.5476/0.0460 {Vanilla Eclair}->{Opera Cake}: 0.5444/0.0490 {Lemon Cake}->{Hot Coffee}: 0.5417/0.0390
Min_sup=0.03	Min_conf=0.3	{Chocolate Croissant}->{Chocolate Tart}: 0.6133/0.0460 {Vanilla Meringue}->{Marzipan Cookie}: 0.5889/0.0530 {Strawberry Cake}->{Ganache Cookie}: 0.5631/0.0580 {Single Espresso}->{Hot Coffee}: 0.5595/0.0470 {Apple Pie}->{Almond Croissant}: 0.5570/0.0440 {Lemon Cake}->{Single Espresso}: 0.5556/0.0400 {Hot Coffee}->{Single Espresso}: 0.5529/0.0470 {Chocolate Tart}->{Chocolate Croissant}: 0.5476/0.0460 {Vanilla Eclair}->{Opera Cake}: 0.5444/0.0490 {Lemon Cake}->{Hot Coffee}: 0.5417/0.0390
	Min_conf=0.4	{Chocolate Croissant}->{Chocolate Tart}: 0.6133/0.0460 {Vanilla Meringue}->{Marzipan Cookie}: 0.5889/0.0530

		{Strawberry Cake}->{Ganache Cookie}: 0.5631/0.0580 {Single Espresso}->{Hot Coffee}: 0.5595/0.0470 {Apple Pie}->{Almond Croissant}: 0.5570/0.0440 {Lemon Cake}->{Single Espresso}: 0.5556/0.0400 {Hot Coffee}->{Single Espresso}: 0.5529/0.0470 {Chocolate Tart}->{Chocolate Croissant}: 0.5476/0.0460 {Vanilla Eclair}->{Opera Cake}: 0.5444/0.0490 {Lemon Cake}->{Hot Coffee}: 0.5417/0.0390
	Min_conf=0.5	{Chocolate Croissant}->{Chocolate Tart}: 0.6133/0.0460 {Vanilla Meringue}->{Marzipan Cookie}: 0.5889/0.0530 {Strawberry Cake}->{Ganache Cookie}: 0.5631/0.0580 {Single Espresso}->{Hot Coffee}: 0.5595/0.0470 {Apple Pie}->{Almond Croissant}: 0.5570/0.0440 {Lemon Cake}->{Single Espresso}: 0.5556/0.0400 {Hot Coffee}->{Single Espresso}: 0.5529/0.0470 {Chocolate Tart}->{Chocolate Croissant}: 0.5476/0.0460 {Vanilla Eclair}->{Opera Cake}: 0.5444/0.0490 {Lemon Cake}->{Hot Coffee}: 0.5417/0.0390

Data sets		car.txt (1728 examples)
Method		Top 10 association rules(ex: rules: confidence/support)
Min_sup=0.02	Min_conf=0.3	{class_value_vgood}->{safty_high}: 1.0000/0.0376 {persons_2}->{class_value_unacc}: 1.0000/0.3333 {safty_low}->{class_value_unacc}: 1.0000/0.3333 {buying_vhigh}->{class_value_unacc}: 0.8333/0.2083 {maint_vhigh}->{class_value_unacc}: 0.8333/0.2083 {lug_boot_small}->{class_value_unacc}: 0.7812/0.2604 {doors_2}->{class_value_unacc}: 0.7546/0.1887 {buying_high}->{class_value_unacc}: 0.7500/0.1875 {maint_high}->{class_value_unacc}: 0.7269/0.1817 {doors_3}->{class_value_unacc}: 0.6944/0.1736
	Min_conf=0.4	{class_value_vgood}->{safty_high}: 1.0000/0.0376 {persons_2}->{class_value_unacc}: 1.0000/0.3333 {safty_low}->{class_value_unacc}: 1.0000/0.3333 {buying_vhigh}->{class_value_unacc}: 0.8333/0.2083 {maint_vhigh}->{class_value_unacc}: 0.8333/0.2083 {lug_boot_small}->{class_value_unacc}: 0.7812/0.2604 {doors_2}->{class_value_unacc}: 0.7546/0.1887 {buying_high}->{class_value_unacc}: 0.7500/0.1875 {maint_high}->{class_value_unacc}: 0.7269/0.1817 {doors_3}->{class_value_unacc}: 0.6944/0.1736
	Min_conf=0.5	{class_value_vgood}->{safty_high}: 1.0000/0.0376

		{persons_2}->{class_value_unacc}: 1.0000/0.3333 {safty_low}->{class_value_unacc}: 1.0000/0.3333 {buying_vhigh}->{class_value_unacc}: 0.8333/0.2083 {maint_vhigh}->{class_value_unacc}: 0.8333/0.2083 {lug_boot_small}->{class_value_unacc}: 0.7812/0.2604 {doors_2}->{class_value_unacc}: 0.7546/0.1887 {buying_high}->{class_value_unacc}: 0.7500/0.1875 {maint_high}->{class_value_unacc}: 0.7269/0.1817 {doors_3}->{class_value_unacc}: 0.6944/0.1736
Min_sup=0.03	Min_conf=0.3	{class_value_vgood}->{safty_high}: 1.0000/0.0376 {persons_2}->{class_value_unacc}: 1.0000/0.3333 {safty_low}->{class_value_unacc}: 1.0000/0.3333 {buying_vhigh}->{class_value_unacc}: 0.8333/0.2083 {maint_vhigh}->{class_value_unacc}: 0.8333/0.2083 {lug_boot_small}->{class_value_unacc}: 0.7812/0.2604 {doors_2}->{class_value_unacc}: 0.7546/0.1887 {buying_high}->{class_value_unacc}: 0.7500/0.1875 {maint_high}->{class_value_unacc}: 0.7269/0.1817 {doors_3}->{class_value_unacc}: 0.6944/0.1736
	Min_conf=0.4	{class_value_vgood}->{safty_high}: 1.0000/0.0376 {persons_2}->{class_value_unacc}: 1.0000/0.3333 {safty_low}->{class_value_unacc}: 1.0000/0.3333 {buying_vhigh}->{class_value_unacc}: 0.8333/0.2083 {maint_vhigh}->{class_value_unacc}: 0.8333/0.2083 {lug_boot_small}->{class_value_unacc}: 0.7812/0.2604 {doors_2}->{class_value_unacc}: 0.7546/0.1887 {buying_high}->{class_value_unacc}: 0.7500/0.1875 {maint_high}->{class_value_unacc}: 0.7269/0.1817 {doors_3}->{class_value_unacc}: 0.6944/0.1736
	Min_conf=0.5	{class_value_vgood}->{safty_high}: 1.0000/0.0376 {persons_2}->{class_value_unacc}: 1.0000/0.3333 {safty_low}->{class_value_unacc}: 1.0000/0.3333 {buying_vhigh}->{class_value_unacc}: 0.8333/0.2083 {maint_vhigh}->{class_value_unacc}: 0.8333/0.2083 {lug_boot_small}->{class_value_unacc}: 0.7812/0.2604 {doors_2}->{class_value_unacc}: 0.7546/0.1887 {buying_high}->{class_value_unacc}: 0.7500/0.1875 {maint_high}->{class_value_unacc}: 0.7269/0.1817 {doors_3}->{class_value_unacc}: 0.6944/0.1736
Min_sup=0.04	Min_conf=0.3	{persons_2}->{class_value_unacc}: 1.0000/0.3333 {safty_low}->{class_value_unacc}: 1.0000/0.3333 {buying_vhigh}->{class_value_unacc}: 0.8333/0.2083

		{maint_vhigh}->{class_value_unacc}: 0.8333/0.2083 {lug_boot_small}->{class_value_unacc}: 0.7812/0.2604 {doors_2}->{class_value_unacc}: 0.7546/0.1887 {buying_high}->{class_value_unacc}: 0.7500/0.1875 {maint_high}->{class_value_unacc}: 0.7269/0.1817 {doors_3}->{class_value_unacc}: 0.6944/0.1736 {lug_boot_med}->{class_value_unacc}: 0.6806/0.2269
	Min_conf=0.4	{persons_2}->{class_value_unacc}: 1.0000/0.3333 {safty_low}->{class_value_unacc}: 1.0000/0.3333 {buying_vhigh}->{class_value_unacc}: 0.8333/0.2083 {maint_vhigh}->{class_value_unacc}: 0.8333/0.2083 {lug_boot_small}->{class_value_unacc}: 0.7812/0.2604 {doors_2}->{class_value_unacc}: 0.7546/0.1887 {buying_high}->{class_value_unacc}: 0.7500/0.1875 {maint_high}->{class_value_unacc}: 0.7269/0.1817 {doors_3}->{class_value_unacc}: 0.6944/0.1736 {lug_boot_med}->{class_value_unacc}: 0.6806/0.2269
	Min_conf=0.5	{persons_2}->{class_value_unacc}: 1.0000/0.3333 {safty_low}->{class_value_unacc}: 1.0000/0.3333 {buying_vhigh}->{class_value_unacc}: 0.8333/0.2083 {maint_vhigh}->{class_value_unacc}: 0.8333/0.2083 {lug_boot_small}->{class_value_unacc}: 0.7812/0.2604 {doors_2}->{class_value_unacc}: 0.7546/0.1887 {buying_high}->{class_value_unacc}: 0.7500/0.1875 {maint_high}->{class_value_unacc}: 0.7269/0.1817 {doors_3}->{class_value_unacc}: 0.6944/0.1736 {lug_boot_med}->{class_value_unacc}: 0.6806/0.2269

Data sets		chess.txt (28056 examples)
Method		Top 10 association rules(ex: rules: confidence/support)
Min_sup=0.005	Min_conf=0.3	{sixteen}->{White_King_Rank_1}: 1.0000/0.0139 {White_King_Rank_4}->{White_King_File_d}: 1.0000/0.0613 {White_King_File_a}->{White_King_Rank_1}: 1.0000/0.0669 {four}->{White_King_File_d}: 0.9949/0.0070 {fifteen}->{White_King_Rank_1}: 0.9206/0.0711 {five}->{White_King_File_d}: 0.9172/0.0154 {six}->{White_King_File_d}: 0.8716/0.0184 {two}->{Black_King_Rank_1}: 0.8333/0.0073 {two}->{White_King_File_c}: 0.8252/0.0072 {four}->{Black_King_Rank_1}: 0.8081/0.0057
	Min_conf=0.4	{sixteen}->{White_King_Rank_1}: 1.0000/0.0139 {White_King_Rank_4}->{White_King_File_d}: 1.0000/0.0613

		{White_King_File_a}->{White_King_Rank_1}: 1.0000/0.0669 {four}->{White_King_File_d}: 0.9949/0.0070 {fifteen}->{White_King_Rank_1}: 0.9206/0.0711 {five}->{White_King_File_d}: 0.9172/0.0154 {six}->{White_King_File_d}: 0.8716/0.0184 {two}->{Black_King_Rank_1}: 0.8333/0.0073 {two}->{White_King_File_c}: 0.8252/0.0072 {four}->{Black_King_Rank_1}: 0.8081/0.0057
	Min_conf=0.5	{sixteen}->{White_King_Rank_1}: 1.0000/0.0139 {White_King_Rank_4}->{White_King_File_d}: 1.0000/0.0613 {White_King_File_a}->{White_King_Rank_1}: 1.0000/0.0669 {four}->{White_King_File_d}: 0.9949/0.0070 {fifteen}->{White_King_Rank_1}: 0.9206/0.0711 {five}->{White_King_File_d}: 0.9172/0.0154 {six}->{White_King_File_d}: 0.8716/0.0184 {two}->{Black_King_Rank_1}: 0.8333/0.0073 {two}->{White_King_File_c}: 0.8252/0.0072 {four}->{Black_King_Rank_1}: 0.8081/0.0057
Min_sup=0.01	Min_conf=0.3	{sixteen}->{White_King_Rank_1}: 1.0000/0.0139 {White_King_Rank_4}->{White_King_File_d}: 1.0000/0.0613 {White_King_File_a}->{White_King_Rank_1}: 1.0000/0.0669 {fifteen}->{White_King_Rank_1}: 0.9206/0.0711 {five}->{White_King_File_d}: 0.9172/0.0154 {six}->{White_King_File_d}: 0.8716/0.0184 {fourteen}->{White_King_Rank_1}: 0.7571/0.1229 {White_King_File_b}->{White_King_Rank_1}: 0.6764/0.1282 {seven}->{White_King_File_d}: 0.6764/0.0165 {White_King_Rank_3}->{White_King_File_d}: 0.6647/0.1215
	Min_conf=0.4	{sixteen}->{White_King_Rank_1}: 1.0000/0.0139 {White_King_Rank_4}->{White_King_File_d}: 1.0000/0.0613 {White_King_File_a}->{White_King_Rank_1}: 1.0000/0.0669 {fifteen}->{White_King_Rank_1}: 0.9206/0.0711 {five}->{White_King_File_d}: 0.9172/0.0154 {six}->{White_King_File_d}: 0.8716/0.0184 {fourteen}->{White_King_Rank_1}: 0.7571/0.1229 {White_King_File_b}->{White_King_Rank_1}: 0.6764/0.1282 {seven}->{White_King_File_d}: 0.6764/0.0165 {White_King_Rank_3}->{White_King_File_d}: 0.6647/0.1215
	Min_conf=0.5	{sixteen}->{White_King_Rank_1}: 1.0000/0.0139 {White_King_Rank_4}->{White_King_File_d}: 1.0000/0.0613 {White_King_File_a}->{White_King_Rank_1}: 1.0000/0.0669 {fifteen}->{White_King_Rank_1}: 0.9206/0.0711

		{five}->{White_King_File_d}: 0.9172/0.0154 {six}->{White_King_File_d}: 0.8716/0.0184 {fourteen}->{White_King_Rank_1}: 0.7571/0.1229 {White_King_File_b}->{White_King_Rank_1}: 0.6764/0.1282 {seven}->{White_King_File_d}: 0.6764/0.0165 {White_King_Rank_3}->{White_King_File_d}: 0.6647/0.1215
Min_sup=0.02	Min_conf=0.3	{White_King_Rank_4}->{White_King_File_d}: 1.0000/0.0613 {White_King_File_a}->{White_King_Rank_1}: 1.0000/0.0669 {fifteen}->{White_King_Rank_1}: 0.9206/0.0711 {fourteen}->{White_King_Rank_1}: 0.7571/0.1229 {White_King_File_b}->{White_King_Rank_1}: 0.6764/0.1282 {White_King_Rank_3}->{White_King_File_d}: 0.6647/0.1215 {eight}->{White_King_File_d}: 0.6315/0.0323 {Black_King_File_b}->{White_King_File_d}: 0.5867/0.0565 {nine}->{White_King_File_d}: 0.5537/0.0338 {Black_King_File_c}->{White_King_Rank_1}: 0.5331/0.0454
	Min_conf=0.4	{White_King_Rank_4}->{White_King_File_d}: 1.0000/0.0613 {White_King_File_a}->{White_King_Rank_1}: 1.0000/0.0669 {fifteen}->{White_King_Rank_1}: 0.9206/0.0711 {fourteen}->{White_King_Rank_1}: 0.7571/0.1229 {White_King_File_b}->{White_King_Rank_1}: 0.6764/0.1282 {White_King_Rank_3}->{White_King_File_d}: 0.6647/0.1215 {eight}->{White_King_File_d}: 0.6315/0.0323 {Black_King_File_b}->{White_King_File_d}: 0.5867/0.0565 {nine}->{White_King_File_d}: 0.5537/0.0338 {Black_King_File_c}->{White_King_Rank_1}: 0.5331/0.0454
	Min_conf=0.5	{White_King_Rank_4}->{White_King_File_d}: 1.0000/0.0613 {White_King_File_a}->{White_King_Rank_1}: 1.0000/0.0669 {fifteen}->{White_King_Rank_1}: 0.9206/0.0711 {fourteen}->{White_King_Rank_1}: 0.7571/0.1229 {White_King_File_b}->{White_King_Rank_1}: 0.6764/0.1282 {White_King_Rank_3}->{White_King_File_d}: 0.6647/0.1215 {eight}->{White_King_File_d}: 0.6315/0.0323 {Black_King_File_b}->{White_King_File_d}: 0.5867/0.0565 {nine}->{White_King_File_d}: 0.5537/0.0338 {Black_King_File_c}->{White_King_Rank_1}: 0.5331/0.0454

At first I set the same threshold as the previous question and found out that some combination of minimum support and minimum confidence do not generate enough association rule for identifying the top 10. Therefore, I decline the threshold and get the result above. Also, I found out that different minimum confidence threshold in same minimum support threshold does not change the top ten rules. The only thing which can change the top ten rules is to provide different minimum support threshold.

(f) Following result can get by running “q3_f.py”. The output file shows the top ten association rules which is generated by measuring the lift and the table below is for the comparison between the relationship of confidence and lift. Also, I consider the situation that several association rules have same lift and I decide to use their support to generate the ranking. Therefore, it means that the top ten association rules is based on considering lift first, then their support second.

Data sets		item_1000.txt (1000 examples)
Method		Top 10 association rules (ex: rules: lift/support)
Min_sup=0.01	Min_conf=0.3	{Chocolate Tart}->{Chocolate Croissant}: 7.3016/0.0460
		{Chocolate Croissant}->{Chocolate Tart}: 7.3016/0.0460
		{Blueberry Danish}->{Raspberry Cookie}: 7.1166/0.0310
		{Raspberry Cookie}->{Blueberry Danish}: 7.1166/0.0310
		{Chocolate Coffee}->{Blueberry Tart}: 6.8892/0.0260
		{Blueberry Tart}->{Chocolate Coffee}: 6.8892/0.0260
		{Casino Cake}->{Chocolate Croissant}: 6.6667/0.0390
		{Chocolate Croissant}->{Casino Cake}: 6.6667/0.0390
		{Lemon Cake}->{Single Espresso}: 6.6138/0.0400
		{Single Espresso}->{Lemon Cake}: 6.6138/0.0400
	Min_conf=0.4	{Chocolate Tart}->{Chocolate Croissant}: 7.3016/0.0460
		{Chocolate Croissant}->{Chocolate Tart}: 7.3016/0.0460
		{Blueberry Danish}->{Raspberry Cookie}: 7.1166/0.0310
		{Raspberry Cookie}->{Blueberry Danish}: 7.1166/0.0310
		{Blueberry Tart}->{Chocolate Coffee}: 6.8892/0.0260
		{Casino Cake}->{Chocolate Croissant}: 6.6667/0.0390
		{Chocolate Croissant}->{Casino Cake}: 6.6667/0.0390
		{Lemon Cake}->{Single Espresso}: 6.6138/0.0400
		{Single Espresso}->{Lemon Cake}: 6.6138/0.0400
		{Hot Coffee}->{Single Espresso}: 6.5826/0.0470
	Min_conf=0.5	{Chocolate Tart}->{Chocolate Croissant}: 7.3016/0.0460
		{Chocolate Croissant}->{Chocolate Tart}: 7.3016/0.0460
		{Blueberry Tart}->{Chocolate Coffee}: 6.8892/0.0260
		{Casino Cake}->{Chocolate Croissant}: 6.6667/0.0390
		{Chocolate Croissant}->{Casino Cake}: 6.6667/0.0390
		{Lemon Cake}->{Single Espresso}: 6.6138/0.0400
		{Hot Coffee}->{Single Espresso}: 6.5826/0.0470
		{Single Espresso}->{Hot Coffee}: 6.5826/0.0470
		{Apple Croissant}->{Blackberry Tart}: 6.4977/0.0400
		{Lemon Cake}->{Hot Coffee}: 6.3725/0.0390
Min_sup=0.02	Min_conf=0.3	{Chocolate Tart}->{Chocolate Croissant}: 7.3016/0.0460
		{Chocolate Croissant}->{Chocolate Tart}: 7.3016/0.0460

		{Blueberry Danish}->{Raspberry Cookie}: 7.1166/0.0310 {Raspberry Cookie}->{Blueberry Danish}: 7.1166/0.0310 {Chocolate Coffee}->{Blueberry Tart}: 6.8892/0.0260 {Blueberry Tart}->{Chocolate Coffee}: 6.8892/0.0260 {Casino Cake}->{Chocolate Croissant}: 6.6667/0.0390 {Chocolate Croissant}->{Casino Cake}: 6.6667/0.0390 {Lemon Cake}->{Single Espresso}: 6.6138/0.0400 {Single Espresso}->{Lemon Cake}: 6.6138/0.0400
	Min_conf=0.4	{Chocolate Tart}->{Chocolate Croissant}: 7.3016/0.0460 {Chocolate Croissant}->{Chocolate Tart}: 7.3016/0.0460 {Blueberry Danish}->{Raspberry Cookie}: 7.1166/0.0310 {Raspberry Cookie}->{Blueberry Danish}: 7.1166/0.0310 {Blueberry Tart}->{Chocolate Coffee}: 6.8892/0.0260 {Casino Cake}->{Chocolate Croissant}: 6.6667/0.0390 {Chocolate Croissant}->{Casino Cake}: 6.6667/0.0390 {Lemon Cake}->{Single Espresso}: 6.6138/0.0400 {Single Espresso}->{Lemon Cake}: 6.6138/0.0400 {Hot Coffee}->{Single Espresso}: 6.5826/0.0470
	Min_conf=0.5	{Chocolate Tart}->{Chocolate Croissant}: 7.3016/0.0460 {Chocolate Croissant}->{Chocolate Tart}: 7.3016/0.0460 {Blueberry Tart}->{Chocolate Coffee}: 6.8892/0.0260 {Casino Cake}->{Chocolate Croissant}: 6.6667/0.0390 {Chocolate Croissant}->{Casino Cake}: 6.6667/0.0390 {Lemon Cake}->{Single Espresso}: 6.6138/0.0400 {Hot Coffee}->{Single Espresso}: 6.5826/0.0470 {Single Espresso}->{Hot Coffee}: 6.5826/0.0470 {Apple Croissant}->{Blackberry Tart}: 6.4977/0.0400 {Lemon Cake}->{Hot Coffee}: 6.3725/0.0390
Min_sup=0.03	Min_conf=0.3	{Chocolate Tart}->{Chocolate Croissant}: 7.3016/0.0460 {Chocolate Croissant}->{Chocolate Tart}: 7.3016/0.0460 {Blueberry Danish}->{Raspberry Cookie}: 7.1166/0.0310 {Raspberry Cookie}->{Blueberry Danish}: 7.1166/0.0310 {Casino Cake}->{Chocolate Croissant}: 6.6667/0.0390 {Chocolate Croissant}->{Casino Cake}: 6.6667/0.0390 {Lemon Cake}->{Single Espresso}: 6.6138/0.0400 {Single Espresso}->{Lemon Cake}: 6.6138/0.0400 {Hot Coffee}->{Single Espresso}: 6.5826/0.0470 {Single Espresso}->{Hot Coffee}: 6.5826/0.0470
	Min_conf=0.4	{Chocolate Tart}->{Chocolate Croissant}: 7.3016/0.0460 {Chocolate Croissant}->{Chocolate Tart}: 7.3016/0.0460 {Blueberry Danish}->{Raspberry Cookie}: 7.1166/0.0310 {Raspberry Cookie}->{Blueberry Danish}: 7.1166/0.0310

		{Casino Cake}->{Chocolate Croissant}: 6.6667/0.0390 {Chocolate Croissant}->{Casino Cake}: 6.6667/0.0390 {Lemon Cake}->{Single Espresso}: 6.6138/0.0400 {Single Espresso}->{Lemon Cake}: 6.6138/0.0400 {Hot Coffee}->{Single Espresso}: 6.5826/0.0470 {Single Espresso}->{Hot Coffee}: 6.5826/0.0470
	Min_conf=0.5	{Chocolate Tart}->{Chocolate Croissant}: 7.3016/0.0460 {Chocolate Croissant}->{Chocolate Tart}: 7.3016/0.0460 {Casino Cake}->{Chocolate Croissant}: 6.6667/0.0390 {Chocolate Croissant}->{Casino Cake}: 6.6667/0.0390 {Lemon Cake}->{Single Espresso}: 6.6138/0.0400 {Hot Coffee}->{Single Espresso}: 6.5826/0.0470 {Single Espresso}->{Hot Coffee}: 6.5826/0.0470 {Apple Croissant}->{Blackberry Tart}: 6.4977/0.0400 {Lemon Cake}->{Hot Coffee}: 6.3725/0.0390 {Casino Cake}->{Chocolate Tart}: 6.2576/0.0410

Data sets		car.txt (1728 examples)
Method		Top 10 association rules (ex: rules: lift/support)
Min_sup=0.02	Min_conf=0.3	{class_value_vgood}->{safty_high}: 3.0000/0.0376 {class_value_good}->{buying_low}: 2.6667/0.0266 {class_value_good}->{maint_low}: 2.6667/0.0266 {class_value_vgood}->{buying_low}: 2.4000/0.0226 {class_value_vgood}->{lug_boot_big}: 1.8462/0.0231 {class_value_good}->{safty_med}: 1.6957/0.0226 {class_value_vgood}->{persons_more}: 1.6154/0.0203 {safty_high}->{class_value_acc}: 1.5938/0.1181 {class_value_acc}->{safty_high}: 1.5938/0.1181 {class_value_good}->{persons_4}: 1.5652/0.0208
	Min_conf=0.4	{class_value_vgood}->{safty_high}: 3.0000/0.0376 {class_value_good}->{buying_low}: 2.6667/0.0266 {class_value_good}->{maint_low}: 2.6667/0.0266 {class_value_vgood}->{buying_low}: 2.4000/0.0226 {class_value_vgood}->{lug_boot_big}: 1.8462/0.0231 {class_value_good}->{safty_med}: 1.6957/0.0226 {class_value_vgood}->{persons_more}: 1.6154/0.0203 {class_value_acc}->{safty_high}: 1.5938/0.1181 {class_value_good}->{persons_4}: 1.5652/0.0208 {class_value_acc}->{persons_4}: 1.5469/0.1146
	Min_conf=0.5	{class_value_vgood}->{safty_high}: 3.0000/0.0376 {class_value_good}->{buying_low}: 2.6667/0.0266 {class_value_good}->{maint_low}: 2.6667/0.0266

		{class_value_vgood}->{buying_low}: 2.4000/0.0226 {class_value_vgood}->{lug_boot_big}: 1.8462/0.0231 {class_value_good}->{safty_med}: 1.6957/0.0226 {class_value_vgood}->{persons_more}: 1.6154/0.0203 {class_value_acc}->{safty_high}: 1.5938/0.1181 {class_value_good}->{persons_4}: 1.5652/0.0208 {class_value_acc}->{persons_4}: 1.5469/0.1146
Min_sup=0.03	Min_conf=0.3	{class_value_vgood}->{safty_high}: 3.0000/0.0376 {safty_high}->{class_value_acc}: 1.5938/0.1181 {class_value_acc}->{safty_high}: 1.5938/0.1181 {class_value_acc}->{persons_4}: 1.5469/0.1146 {persons_4}->{class_value_acc}: 1.5469/0.1146 {persons_more}->{class_value_acc}: 1.4531/0.1076 {class_value_acc}->{persons_more}: 1.4531/0.1076 {class_value_unacc}->{persons_2}: 1.4281/0.3333 {class_value_unacc}->{safty_low}: 1.4281/0.3333 {persons_2}->{class_value_unacc}: 1.4281/0.3333
	Min_conf=0.4	{class_value_vgood}->{safty_high}: 3.0000/0.0376 {class_value_acc}->{safty_high}: 1.5938/0.1181 {class_value_acc}->{persons_4}: 1.5469/0.1146 {class_value_acc}->{persons_more}: 1.4531/0.1076 {class_value_unacc}->{persons_2}: 1.4281/0.3333 {class_value_unacc}->{safty_low}: 1.4281/0.3333 {persons_2}->{class_value_unacc}: 1.4281/0.3333 {safty_low}->{class_value_unacc}: 1.4281/0.3333 {class_value_acc}->{safty_med}: 1.4062/0.1042 {buying_vhigh}->{class_value_unacc}: 1.1901/0.2083
	Min_conf=0.5	{class_value_vgood}->{safty_high}: 3.0000/0.0376 {class_value_acc}->{safty_high}: 1.5938/0.1181 {class_value_acc}->{persons_4}: 1.5469/0.1146 {persons_2}->{class_value_unacc}: 1.4281/0.3333 {safty_low}->{class_value_unacc}: 1.4281/0.3333 {buying_vhigh}->{class_value_unacc}: 1.1901/0.2083 {maint_vhigh}->{class_value_unacc}: 1.1901/0.2083 {lug_boot_small}->{class_value_unacc}: 1.1157/0.2604 {doors_2}->{class_value_unacc}: 1.0777/0.1887 {buying_high}->{class_value_unacc}: 1.0711/0.1875
Min_sup=0.04	Min_conf=0.3	{safty_high}->{class_value_acc}: 1.5938/0.1181 {class_value_acc}->{safty_high}: 1.5938/0.1181 {class_value_acc}->{persons_4}: 1.5469/0.1146 {persons_4}->{class_value_acc}: 1.5469/0.1146 {persons_more}->{class_value_acc}: 1.4531/0.1076

		{class_value_acc}->{persons_more}: 1.4531/0.1076 {class_value_unacc}->{persons_2}: 1.4281/0.3333 {class_value_unacc}->{safty_low}: 1.4281/0.3333 {persons_2}->{class_value_unacc}: 1.4281/0.3333 {safty_low}->{class_value_unacc}: 1.4281/0.3333
	Min_conf=0.4	{class_value_acc}->{safty_high}: 1.5938/0.1181 {class_value_acc}->{persons_4}: 1.5469/0.1146 {class_value_acc}->{persons_more}: 1.4531/0.1076 {class_value_unacc}->{persons_2}: 1.4281/0.3333 {class_value_unacc}->{safty_low}: 1.4281/0.3333 {persons_2}->{class_value_unacc}: 1.4281/0.3333 {safty_low}->{class_value_unacc}: 1.4281/0.3333 {class_value_acc}->{safty_med}: 1.4062/0.1042 {buying_vhigh}->{class_value_unacc}: 1.1901/0.2083 {maint_vhigh}->{class_value_unacc}: 1.1901/0.2083
	Min_conf=0.5	{class_value_acc}->{safty_high}: 1.5938/0.1181 {class_value_acc}->{persons_4}: 1.5469/0.1146 {persons_2}->{class_value_unacc}: 1.4281/0.3333 {safty_low}->{class_value_unacc}: 1.4281/0.3333 {buying_vhigh}->{class_value_unacc}: 1.1901/0.2083 {maint_vhigh}->{class_value_unacc}: 1.1901/0.2083 {lug_boot_small}->{class_value_unacc}: 1.1157/0.2604 {doors_2}->{class_value_unacc}: 1.0777/0.1887 {buying_high}->{class_value_unacc}: 1.0711/0.1875 {maint_high}->{class_value_unacc}: 1.0380/0.1817

Data sets		chess.txt (28056 examples)
Method		Top 10 association rules (ex: rules: lift/support)
Min_sup=0.005	Min_conf=0.3	{sixteen}->{White_King_File_a}: 10.6490/0.0099 {two}->{Black_King_Rank_1}: 6.3810/0.0073 {four}->{Black_King_Rank_1}: 6.1876/0.0057 {five}->{Black_King_Rank_1}: 4.9910/0.0109 {six}->{Black_King_Rank_1}: 4.6564/0.0128 {five}->{Black_King_File_a}: 3.3251/0.0058 {eight}->{Black_King_Rank_1}: 3.1580/0.0211 {two}->{White_King_Rank_3}: 3.1569/0.0051 {eight}->{Black_King_Rank_2}: 3.1120/0.0181 {seven}->{Black_King_Rank_1}: 3.0943/0.0098
	Min_conf=0.4	{sixteen}->{White_King_File_a}: 10.6490/0.0099 {two}->{Black_King_Rank_1}: 6.3810/0.0073 {four}->{Black_King_Rank_1}: 6.1876/0.0057 {five}->{Black_King_Rank_1}: 4.9910/0.0109

		{six}->{Black_King_Rank_1}: 4.6564/0.0128 {eight}->{Black_King_Rank_1}: 3.1580/0.0211 {two}->{White_King_Rank_3}: 3.1569/0.0051 {seven}->{Black_King_Rank_1}: 3.0943/0.0098 {two}->{White_King_File_c}: 2.6532/0.0072 {five}->{White_King_Rank_3}: 2.3455/0.0072
	Min_conf=0.5	{sixteen}->{White_King_File_a}: 10.6490/0.0099 {two}->{Black_King_Rank_1}: 6.3810/0.0073 {four}->{Black_King_Rank_1}: 6.1876/0.0057 {five}->{Black_King_Rank_1}: 4.9910/0.0109 {six}->{Black_King_Rank_1}: 4.6564/0.0128 {two}->{White_King_Rank_3}: 3.1569/0.0051 {two}->{White_King_File_c}: 2.6532/0.0072 {White_King_Rank_4}->{White_King_File_d}: 2.3118/0.0613 {four}->{White_King_File_d}: 2.3001/0.0070 {sixteen}->{White_King_Rank_1}: 2.2151/0.0139
Min_sup=0.01	Min_conf=0.3	{five}->{Black_King_Rank_1}: 4.9910/0.0109 {six}->{Black_King_Rank_1}: 4.6564/0.0128 {eight}->{Black_King_Rank_1}: 3.1580/0.0211 {eight}->{Black_King_Rank_2}: 3.1120/0.0181 {White_King_Rank_4}->{White_King_File_d}: 2.3118/0.0613 {sixteen}->{White_King_Rank_1}: 2.2151/0.0139 {White_King_File_a}->{White_King_Rank_1}: 2.2151/0.0669 {nine}->{Black_King_File_h}: 2.1533/0.0227 {five}->{White_King_File_d}: 2.1204/0.0154 {fifteen}->{White_King_File_b}: 2.1125/0.0309
	Min_conf=0.4	{five}->{Black_King_Rank_1}: 4.9910/0.0109 {six}->{Black_King_Rank_1}: 4.6564/0.0128 {eight}->{Black_King_Rank_1}: 3.1580/0.0211 {White_King_Rank_4}->{White_King_File_d}: 2.3118/0.0613 {sixteen}->{White_King_Rank_1}: 2.2151/0.0139 {White_King_File_a}->{White_King_Rank_1}: 2.2151/0.0669 {five}->{White_King_File_d}: 2.1204/0.0154 {fifteen}->{White_King_File_b}: 2.1125/0.0309 {fifteen}->{White_King_Rank_1}: 2.0392/0.0711 {six}->{White_King_File_d}: 2.0150/0.0184
	Min_conf=0.5	{five}->{Black_King_Rank_1}: 4.9910/0.0109 {six}->{Black_King_Rank_1}: 4.6564/0.0128 {White_King_Rank_4}->{White_King_File_d}: 2.3118/0.0613 {sixteen}->{White_King_Rank_1}: 2.2151/0.0139 {White_King_File_a}->{White_King_Rank_1}: 2.2151/0.0669 {five}->{White_King_File_d}: 2.1204/0.0154

		{fifteen}->{White_King_Rank_1}: 2.0392/0.0711
		{six}->{White_King_File_d}: 2.0150/0.0184
		{fourteen}->{White_King_Rank_1}: 1.6770/0.1229
		{seven}->{White_King_File_d}: 1.5638/0.0165
Min_sup=0.02	Min_conf=0.3	{eight}->{Black_King_Rank_1}: 3.1580/0.0211
		{White_King_Rank_4}->{White_King_File_d}: 2.3118/0.0613
		{White_King_File_a}->{White_King_Rank_1}: 2.2151/0.0669
		{nine}->{Black_King_File_h}: 2.1533/0.0227
		{fifteen}->{White_King_File_b}: 2.1125/0.0309
		{fifteen}->{White_King_Rank_1}: 2.0392/0.0711
		{fourteen}->{White_King_Rank_1}: 1.6770/0.1229
		{thirteen}->{White_King_Rank_2}: 1.5956/0.0726
		{White_King_Rank_3}->{White_King_File_d}: 1.5367/0.1215
		{White_King_File_b}->{White_King_Rank_1}: 1.4984/0.1282
	Min_conf=0.4	{eight}->{Black_King_Rank_1}: 3.1580/0.0211
		{White_King_Rank_4}->{White_King_File_d}: 2.3118/0.0613
		{White_King_File_a}->{White_King_Rank_1}: 2.2151/0.0669
		{fifteen}->{White_King_File_b}: 2.1125/0.0309
		{fifteen}->{White_King_Rank_1}: 2.0392/0.0711
		{fourteen}->{White_King_Rank_1}: 1.6770/0.1229
		{thirteen}->{White_King_Rank_2}: 1.5956/0.0726
		{White_King_Rank_3}->{White_King_File_d}: 1.5367/0.1215
		{White_King_File_b}->{White_King_Rank_1}: 1.4984/0.1282
		{eight}->{White_King_File_d}: 1.4600/0.0323
	Min_conf=0.5	{White_King_Rank_4}->{White_King_File_d}: 2.3118/0.0613
		{White_King_File_a}->{White_King_Rank_1}: 2.2151/0.0669
		{fifteen}->{White_King_Rank_1}: 2.0392/0.0711
		{fourteen}->{White_King_Rank_1}: 1.6770/0.1229
		{White_King_Rank_3}->{White_King_File_d}: 1.5367/0.1215
		{White_King_File_b}->{White_King_Rank_1}: 1.4984/0.1282
		{eight}->{White_King_File_d}: 1.4600/0.0323
		{Black_King_File_b}->{White_King_File_d}: 1.3563/0.0565
		{nine}->{White_King_File_d}: 1.2801/0.0338
		{eleven}->{White_King_File_d}: 1.2239/0.0539

As the conclusion in question (e), the top ten which using the minimum confidence for ranking does not change, but using the lift for ranking will change. Because lift is a kind of performance (quality) for estimating the confidence along with its association rules which means high confidence but low lift will not have value to use the rule. Therefore, any rule with a lift which is less than 1 does not indicate a real cross-selling opportunity, no matter how high its support and confidence is, because it actually offers less ability to predict a purchase than does a random chance. For my result, the lifts are all above 1 which means the association rules does not happen randomly.

4.

(a)

The author in this paper talks about the purpose of researching the study which is because it is hard to define clusters and unify the diverse approaches of clustering till now. Although there are a lot of axiomatic approaches for clustering, he still think they are not compactly enough to formalize clustering. Therefore, he proposes an axiomatic framework called impossibility theorem to solve the difficulty in formalizing clustering.

First, in impossibility theorem, there are three natural properties which are scale-invariance, richness and consistency. For the first property, scale-invariance, it tells us that for every distance function d and every non-negative number α, we can get f(d) = f(αd) which means the result of a clustering function won't be change by adding the scaling in the input. For the second property, richness, it tells us that for the data set S, the result of applying clustering function should be rich which means every partition in S are able to be obtained. For the third property, consistency, it shows us that the clustering function has such a behavior that when we decrease the distance between any pair of points within the cluster and increase the distance between any pair of points in different cluster, the outcome will be the same and this behavior is called consistency. However, here the author claims a significant theorem that there is no clustering function that can satisfy these three properties in the same time. He gives us a classical example to show this feature which is the single-linkage. He says that single linkage can satisfy either of the two properties if we consider the case respectively, but cannot meet three properties simultaneously. For example, for the k-cluster stopping condition, since it does not have to generate all possible partition from the data set, it only satisfies scale-invariance and consistency. For the scale-α stopping condition, it only satisfies scale-invariance and richness. For the distance-r stopping condition, since it builds in a fundamental length scale, it only satisfies richness and consistency. Therefore, he states that the three properties, scale-invariance, richness and consistency cannot be met in the same time.

Second, in order to strengthen the impossibility theorem, he uses antichains of partitions to prove his statement. Here notice that the collection of partition is an antichain means that the partition does not contain in the refinement of a partition Γ'and Γ itself. Then, he proposes the theorem that Range(f) is an antichain, if the clustering function f satisfies scale-invariance and consistency, because there is a contradiction of Γ'≠ Γ. The other theorem he claims is that for every antichain of partitions, it can satisfy scale-invariance and consistency if Range(f) is equal to the antichain. He uses a variant of the sum-of-pairs clustering function to prove the theorem.

Third, the author talks about centroid-based clustering and consistency and says that for the general centroid-based clustering such as k-means clustering and k-median clustering, all of them do not satisfy the consistency property. Therefore, he sets the above observation by using two sets X and Y and dividing X into two equal sets for proof.

Finally, he try to figure out a way to let the clustering function can fit the three properties in the same time by relaxing the property. Therefore, he proposes a refinement-consistency which is the relaxation of consistency. However, he finds out that as the theorem he proved, although relaxing the property, there is still no clustering can meet all of the properties. He also uses near-richness property to challenge the theorem and he find out that although near-richness can satisfy scale-invariance and refinement-consistency in the same time, it is not convincing to break the theorem

in the general case. Therefore, he conclude that the impossibility theorem can be fit in the general case of clustering and give the axiomatic structure for clustering.
For this paper, I think the impossibility theorem clearly presents the features of clustering. The three properties give us a very good picture for formulizing clustering. Also, the examples clearly shows the theorem which says that only any two of the three properties can satisfy the clustering function in the same time. However, I don't think the theorem can fit in any case. For some specific case might have the contradiction with the theorem just as the second paper mentions. Also, it is not quite clear for the author to mention about the relaxation of the property part since the author does not give us the real example to show his inference. Therefore, it is a good theorem but is not intact at all.

(b)

Firstly, the author talk about the purpose of writing this paper is to challenge the concepts what Kleinberg proposed in his research paper: An Impossibility Theorem for Clustering. They note that the result of the impossibility theorem is not a natural feature of clustering. Therefore, they propose a new clustering axiom, Clustering-quality measure (CQM), to against the ideas in the impossibility theorem. The most important point in CQM is its output which is a non-negative real number can represent the quality (how strong or conclusive the clustering is) by giving a data set. Also, the authors claim that CQM can break the impossibility theorem by holding the three properties, scale-invariance, richness and consistency, in the same time. Moreover, there are other advantages of the CQM. One is that it is very helpful for selecting the clustering model since it can compare different clustering over the same data set. Another is that it can be completely computing in polynomial time. The other is that it can be applied in any specific generative model. The author also say that CQM can reach the requirements of relevance and consistency. In order to achieve the requirements above, the author say that the framework of clustering-quality is more flexible and richer than clustering function which is used in explaining the impossibility theorem. Therefore, by applying the CQM can center-based clustering and linkage-based clustering meet the requirements above which is impossible for impossibility theorem to achieve it.
Second, the author revises the axiom from impossibility theorem and add some new lemma to support CQM. They use a 6-clustering figure and modify the distance by obeying the consistency property to overturn the concept in the impossibility theorem which is that there is no clustering function can satisfy scale-invariance, richness and consistency in the same time. They note that after modification, the quality of the 3-clustering improve and it is the main reason that they point out the weakness of the consistency axiom for clustering function. Therefore, they redefine the axioms of from the impossibility theorem and set them for the axioms of CQM. For the two properties, scale-invariance and richness, in the impossibility theorem stay the same, but they revise the consistency property to that by changing the distance, it won't hurt the quality of the given partition, but it may improve the partition.
Third, the author add relative point margin in the definition of CQM and point out that the smaller values of relative margin stands for the better cluster. Also, the relative margin can satisfy scale-invariance, consistency and richness simultaneously. Here the author mention about soundness and completeness of axioms and claims that CQM can fit those requirements. Moreover, they also

include isomorphism invariance in their definition and clarify that CQR can satisfy all definitions above.
Finally, the author introduce linkage-based clustering and center-based clustering to prove the strength of CQM. In linkage-based clustering, they add the definition of weakest link between points and the weakest link of a clustering. For the center-based clustering, they note the additive margin which is similar to the relative margin but instead of looking the ratios, here it evaluates the difference. Here notice that the higher value of additive margin stands for the better cluster. Also, the author say that the variants of quality measures do not influence the satisfaction to the axioms and the number of clusters is important since it may cause the failure of satisfying the axioms. They use refinement and coarsening preference to stand their statement. Thus, they mention that fix the number of clusters is the best way to applying the CQM.
For this paper, I think the author does not successfully challenge the impossibility theorem, although they propose lots of evidence that their CQR is more useful and critical than the impossibility theorem by revising the definition. Because in the end of previous paper, Kleinberg states that by relaxing the property may cause all the properties satisfy the theorem. Also, in their axiom of CQM, the author does not against the whole picture of the impossibility theorem, instead they only say that their definition of the consistency can improve the clustering but does not change the quality. Also, the author use a lot more definition to strength the CQM which is just add some supplements for the impossibility theorem. Therefore, I only agree that CQM can hold the three properties in the same time but does not agree that this measurement give us a new axiom of formulizing clustering. However, the good thing in the CQM is that it is very clearly to show the quality of the measurement is by using a non-negative real number. By adding this feature, the function can be more useful and handful in a lot of clustering works.