Project Report:

Project Title: Implementing FP-GROWTH and APRIORI algorithm on Mushroom and Chess dataset

Course Title: Software Data Mining
Course Code: CSE477
Section: 02
Semester: SUMMER'21

SUBMITTED TO:

Jesan Ahammed Ovi,
Senior lecturer, CSE Department of Computer Science & Engineering,
East West University, Dhaka, Bangladesh.

SUBMITTED BY:

1. Ishrat Jaben Bushra (2018-1-60-099)
2.Rafina Afreen (2018-1-60-119)
3.Sumaita Tanjim Hridy(2018-1-60-251)
4.Maria Mehjabin Shenjuti (2018-1-60-244)
5.Nayma Alam (2018-1-60-180)

Contents:

I.Data Analysis

a.Dataset Analysis (Mushroom):

1.Mushroom Dataset file:

Mushroom dataset was in dat file. The difference between dat and text files are- text can be easily opened and viewed in a simple text editor but dat files need to convert to open and view, text files used for plain text and the other side dat file used for all types of data which is not plain text. If we want to use dat files we need to change it into another format. As our Mushroom dataset was in dat file so we need to convert it into a csv file.

2.Mushroom Dataset Data analysis:

The Mushroom file was in dat file and they were suitable for classification mining because the values of the attributes were categorized and the records had class attributes. It contained **8123** number of observations dataset with **23** number of attributes, no missing value and all are **int64**. For dataset analysis we used pandas profiling report library in python for generating dataset reports. This report is going to describe the overview, visualization, correlations, missing values and so on. We are discussing every part in detail below.

3.Overview:

In the Mushroom dataset we have got dataset statistics. It shows that there are all categorical variable types. The set of permitted values is usually fixed, and rarely changes for the categorical variables.

Dataset statistics

Number of variables	23
Number of observations	8123
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.4 MiB
Average record size in memory	184.0 B

Variable types

Categorical	14
Numeric	9

From this above analysis we can see that there are **23** numbers of variables, **8123**(considering 0 number row is a transaction .So with 0 row total 8123 transactions) Number of observations and there are no missing cells and there are no duplicate rows found in this dataset. Dataset total memory size shows **1.4 MIB**. In this mushroom dataset we have found **23** column-size and row-size **8123.**

4.Sample Mushroom's Data(For better visualization):

Here in the first (0 to 9) row the highest value is 115 and lowest value is 3.

First rows

	1	3	9	13	23	25	34	36	38	40	52	54	59	63	67	76	85	86	90	93	98	107	113
0	2	3	9	14	23	26	34	36	39	40	52	55	59	63	67	76	85	86	90	93	99	108	114
1	2	4	9	15	23	27	34	36	39	41	52	55	59	63	67	76	85	86	90	93	99	108	115
2	1	3	10	15	23	25	34	36	38	41	52	54	59	63	67	76	85	86	90	93	98	107	113
3	2	3	9	16	24	28	34	37	39	40	53	54	59	63	67	76	85	86	90	94	99	109	114
4	2	3	10	14	23	26	34	36	39	41	52	55	59	63	67	76	85	86	90	93	98	108	114
5	2	4	9	15	23	26	34	36	39	42	52	55	59	63	67	76	85	86	90	93	98	108	115
6	2	4	10	15	23	27	34	36	39	41	52	55	59	63	67	76	85	86	90	93	99	107	115
7	1	3	10	15	23	25	34	36	38	43	52	54	59	63	67	76	85	86	90	93	98	110	114
8	2	4	9	14	23	26	34	36	39	42	52	55	59	63	67	76	85	86	90	93	98	107	115
9	2	3	10	14	23	27	34	36	39	42	52	55	59	63	67	76	85	86	90	93	99	108	114

Here in the last (8113 to 8122) row the highest value is 119 and lowest value is 3.

Last rows

	1	3	9	13	23	25	34	36	38	40	52	54	59	63	67	76	85	86	90	93	98	107	113
8113	1	6	10	21	24	33	35	36	39	50	52	55	61	65	74	84	85	86	92	97	102	112	116
8114	2	3	9	13	24	28	35	36	39	50	52	58	59	63	73	83	85	88	90	93	104	110	119
8115	1	7	10	13	24	32	34	36	38	48	53	58	59	66	69	76	85	86	90	94	102	110	119
8116	1	7	9	17	24	31	34	36	38	48	53	58	61	63	69	76	85	86	90	94	102	110	116
8117	1	7	10	13	24	29	34	36	38	48	53	58	61	63	69	76	85	86	90	94	102	110	116
8118	2	7	9	13	24	28	35	36	39	50	52	58	59	63	73	83	85	88	90	93	106	112	119
8119	2	3	9	13	24	28	35	36	39	50	52	58	59	63	73	83	85	87	90	93	106	110	119
8120	2	6	9	13	24	28	35	36	39	41	52	58	59	63	73	83	85	88	90	93	106	112	119
8121	1	7	10	13	24	31	34	36	38	48	53	58	59	66	67	76	85	86	90	94	102	110	119
8122	2	3	9	13	24	28	35	36	39	50	52	58	59	63	73	83	85	88	90	93	104	112	119

5.Correlations:

i.Pearson's r:

In statistics, the Pearson correlation coefficient(r) is a measure of linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations; thus it is essentially a normalised measurement of the covariance, such that the result always has a value `between −1 and 1`.

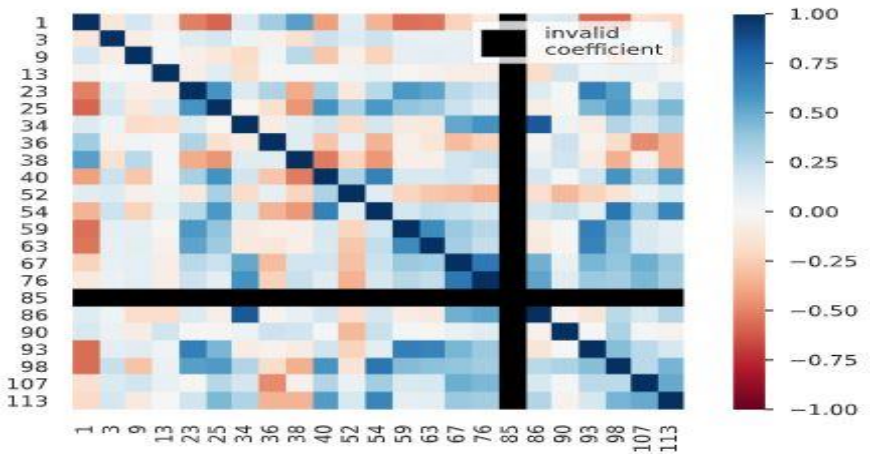
Here ,

-1 = total negative linear correlation,

0 = no linear correlation and

1 = total positive linear correlation.

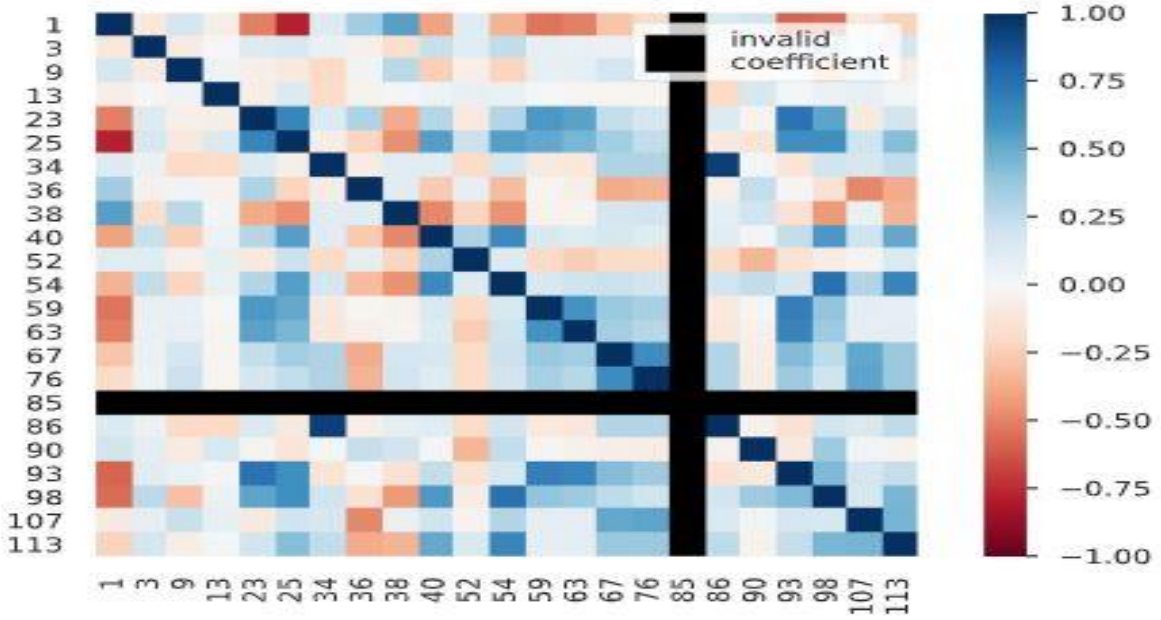
Also, r is invariant under separate changes in location and scale of the two variables, implying that for a linear function the angle to the x-axis does not affect r.
To calculate r for two variables X and Y, one divides the covariance of X and Y by the product of their standard deviations.



ii.Spearman's ρ:

Spearman's correlation coefficient, (ρ, also signified by rs) measures the strength and direction of the association between two ranked variables. You interpret the value of Spearman's rank correlation, ρ the same way you interpret Pearson's correlation, r. The values of ρ can go between –1 and +1.
Here,
-1 = total negative monotonic correlation,
0 indicating no monotonic correlation and
1 indicating total positive monotonic correlation.

To calculate ρ for two variables X and Y, one divides the covariance of the rank variables of X and Y by the product of their standard deviations.



iii.Kendall's τ:

The Kendall rank correlation coefficient (τ) rank-based correlation coefficients, are known as non-parametric correlation measures ordinal association between two variables. It's value lies between -1 and +1,

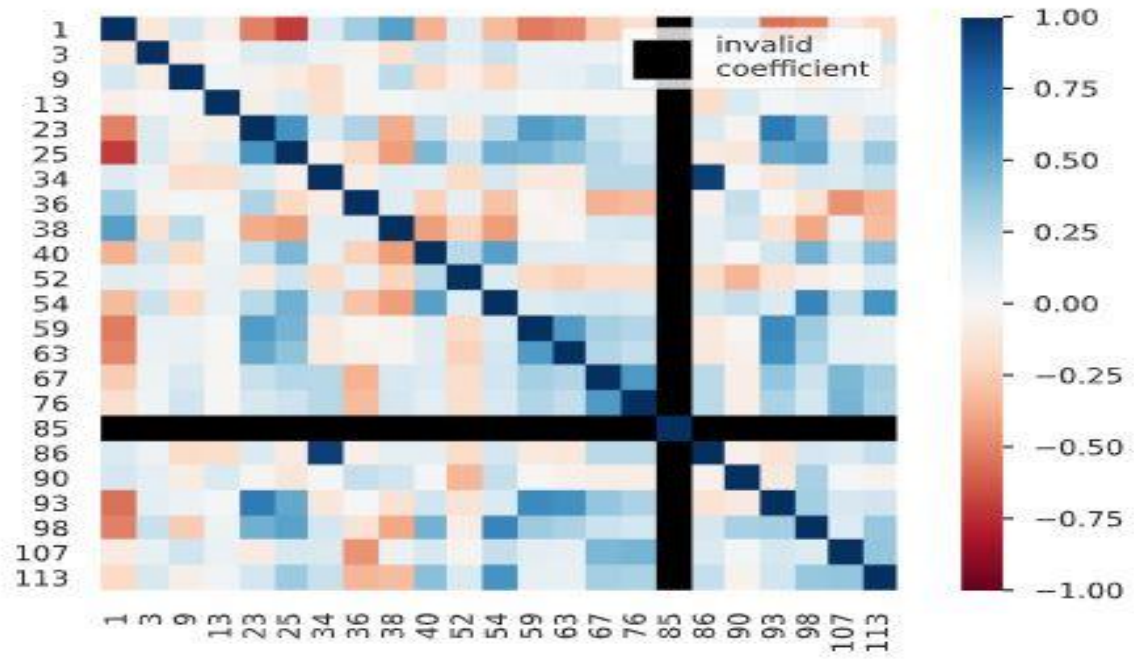
Here,

-1 = total negative correlation,

0 indicating no correlation and

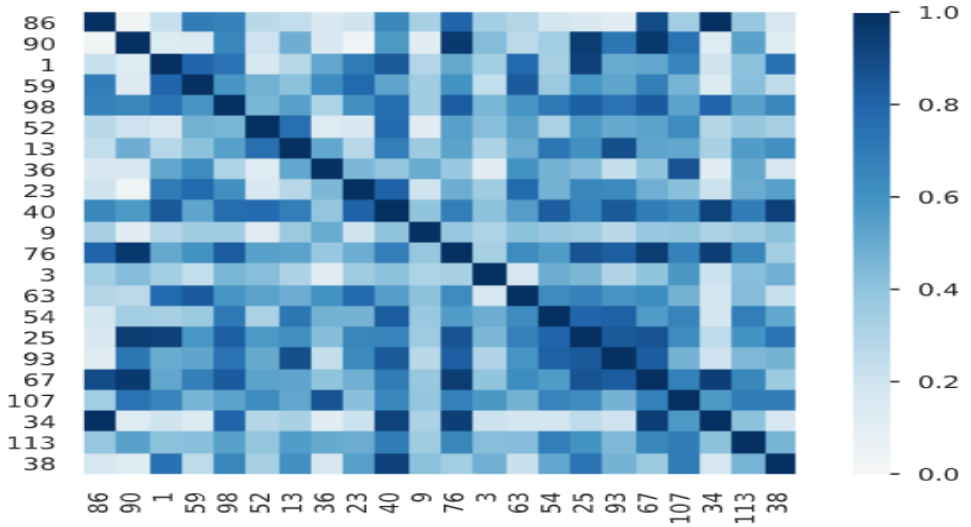
1 indicating total positive correlation.

To calculate τ for two variables X and Y, one determines the number of concordant and discordant pairs of observations. τ is given by the number of concordant pairs minus the discordant pairs divided by the total number of pairs.



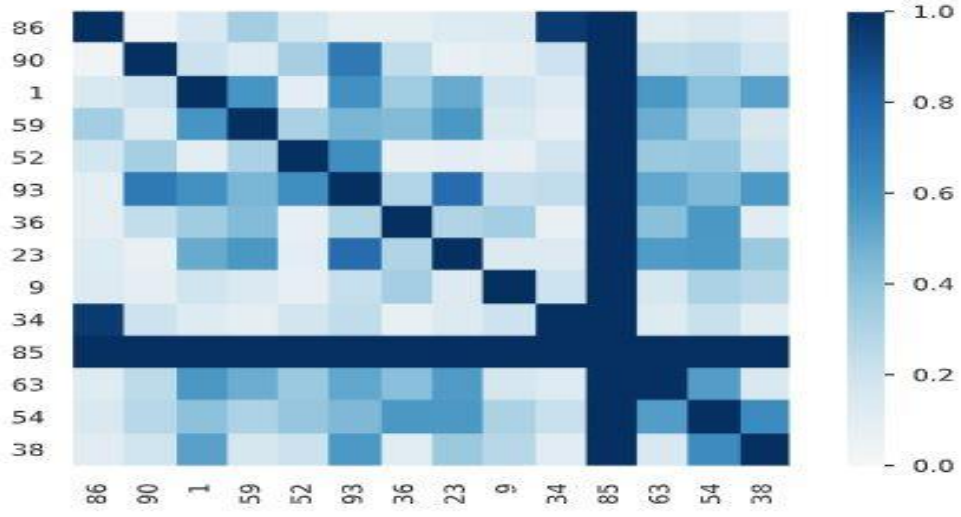
iv.Phik (φk):

Phik (ϕk) is a new and practical correlation coefficient that works consistently between categorical, ordinal and interval variables, captures non-linear dependency and reverts to the Pearson correlation coefficient in case of a bivariate normal input distribution.



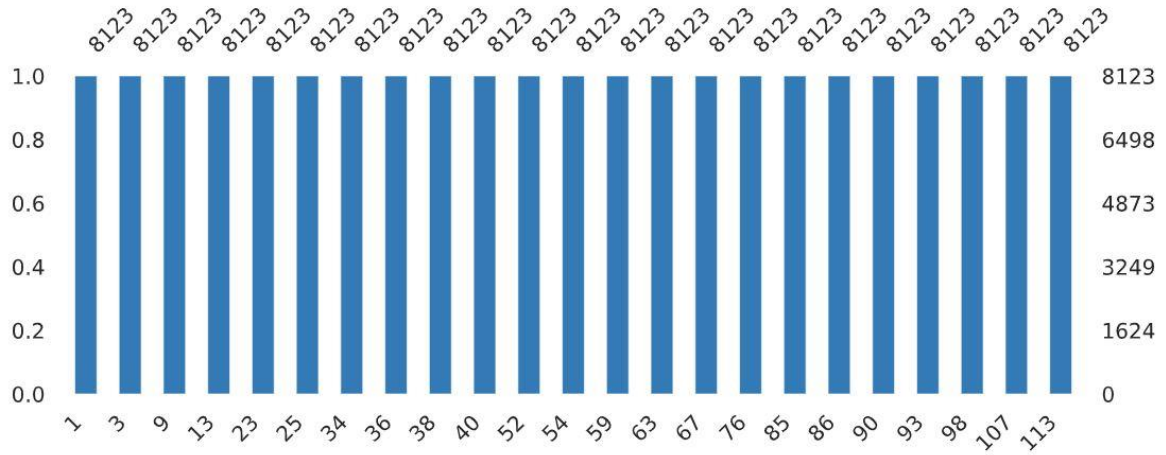
v.Cramér's V (φc):

In statistics, Cramér's V (sometimes referred to as Cramér's phi and denoted as φ_c) is a measure of association between two nominal variables, giving a value between 0 and +1 (inclusive). It is based on Pearson's chi-squared statistic and It measures how strongly two categorical fields are associated.

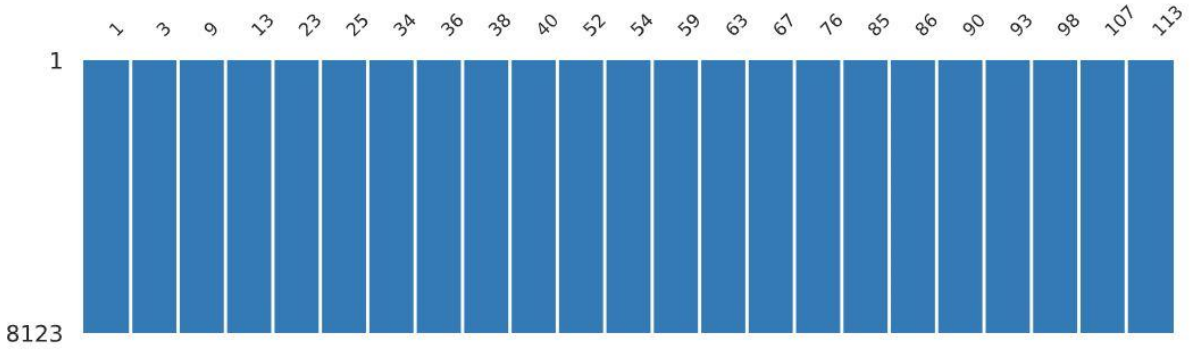


6.Missing values:

i. Count:



ii. Matrix :



In this dataset there are no missing values. The highest column-length is represented by the top columns and the sample of a column-length is represented by the bottom columns.

b.Dataset_Analysis(Chess):

1.Chess Dataset file:

Chess dataset was also in dat file, so here we also need to convert it into a csv file.

2.Chess Dataset Data analysis:

The Chess file was in dat file and they were suitable for classification mining because the values of the attributes were categorized and the records had class attributes. It contained **3195** number of observations dataset with **37** number of attributes, no missing value and all are **int64**. For dataset analysis we used pandas profiling report library in python for generating dataset reports. This report is going to describe the overview,visualization, correlations, missing values and so on. We are discussing every part in detail below.

3.Overview:

In the Chess dataset we have got dataset statistics. It shows that there are all categorical variable types. The set of permitted values is usually fixed, and rarely changes for the categorical variables.

Dataset statistics

Number of variables	37
Number of observations	3195
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	923.7 KiB
Average record size in memory	296.0 B

Variable types

Categorical	37

From this above analysis we can see that there are **37** numbers of variables, **3195**(considering 0 number row is a transaction .So with 0 row total 8123 transactions) Number of observations and there are no missing cells and there are no duplicate rows found in this dataset. Dataset total memory size shows **923.7 KiB**. In this chess dataset we have found **37** column-size and row-size **3195.**

4.Sample Chess's Data(For better visualization):

Here in the first (0 to 9) row the highest value is 74 and lowest value is 1.

First rows

	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	74
0	1	3	5	7	9	12	13	15	17	19	21	23	25	27	29	31	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	74
1	1	3	5	7	9	12	13	16	17	19	21	23	25	27	29	31	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	74
2	1	3	5	7	9	11	13	15	17	20	21	23	25	27	29	31	34	36	38	40	42	44	47	48	50	52	54	56	58	60	62	64	66	68	70	72	74
3	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	34	36	38	40	42	44	46	48	51	52	54	56	58	60	62	64	66	68	70	72	74
4	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	34	36	38	40	42	44	46	48	51	52	54	56	58	60	63	64	66	68	70	72	74
5	1	3	5	7	9	11	13	15	17	20	21	23	25	27	29	31	34	36	38	40	42	44	47	48	51	52	54	56	58	60	62	64	66	68	70	72	74
6	1	3	5	7	9	12	13	15	17	19	21	24	25	27	29	31	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	74
7	1	3	5	7	9	11	13	15	17	19	21	24	25	27	29	31	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	65	66	68	70	72	74
8	1	3	5	7	9	11	13	16	17	19	21	24	25	27	29	31	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	74
9	1	3	5	7	9	12	13	16	17	19	21	24	25	27	29	31	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	74

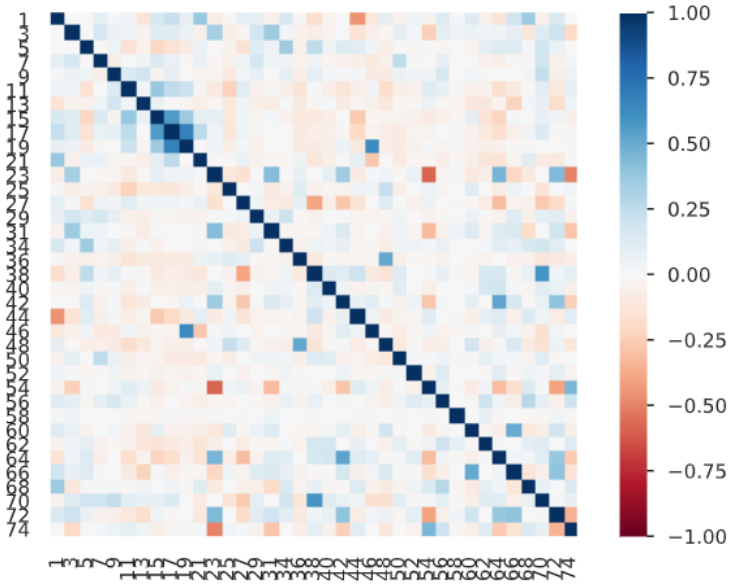
Here in the last (3185 to 3194) row the highest value is 74 and lowest value is 2.

Last rows

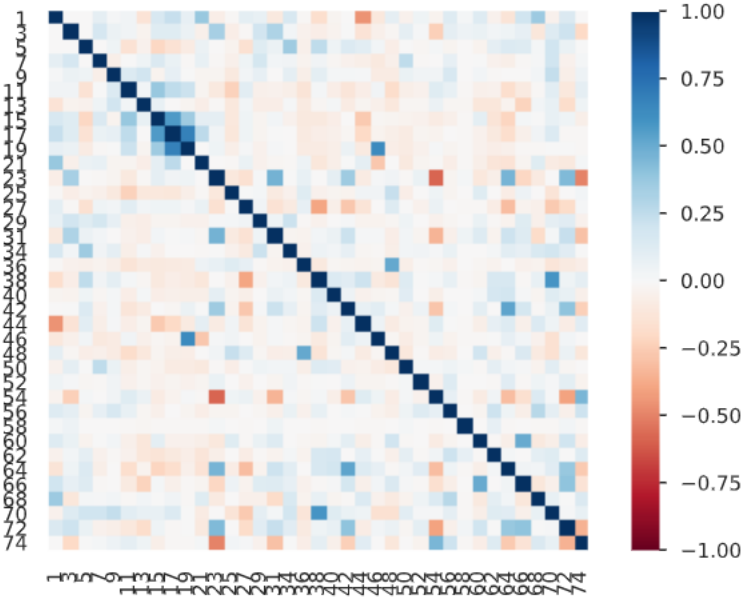
	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72
3185	2	4	5	7	9	12	13	16	17	20	21	23	26	27	29	33	34	36	39	40	42	44	47	48	51	52	54	56	58	61	62	64	67	68	70	73
3186	2	3	5	7	10	11	13	16	18	20	21	23	26	27	29	32	34	36	38	40	42	44	46	48	50	52	54	56	58	61	62	64	67	68	71	73
3187	2	3	5	7	10	11	13	16	18	20	21	23	26	27	29	32	34	36	38	40	42	44	47	48	50	52	54	56	58	61	62	64	67	68	70	73
3188	2	3	5	7	10	11	13	16	18	20	21	23	26	27	29	32	34	36	39	40	42	44	46	48	51	52	54	56	58	61	62	64	67	68	71	73
3189	2	3	5	7	9	11	13	16	17	19	21	23	26	27	29	31	34	36	38	40	43	44	46	49	50	52	54	56	58	61	62	64	67	68	70	73
3190	2	4	5	7	9	11	13	16	17	19	21	23	26	27	29	33	34	36	38	40	42	44	46	49	51	52	54	56	58	61	62	64	67	68	70	73
3191	2	4	5	7	9	11	13	16	17	19	21	23	26	27	29	33	34	36	38	40	42	44	46	49	50	52	54	56	58	61	62	64	67	68	70	73
3192	2	4	5	7	9	11	13	16	17	19	21	23	26	27	29	31	34	36	38	40	42	44	46	49	50	52	54	56	58	61	62	64	67	68	70	73
3193	2	4	5	8	9	11	13	16	17	19	21	23	26	27	30	33	35	36	38	40	42	44	46	48	51	52	54	56	58	61	62	64	67	68	71	73
3194	2	4	5	8	9	11	13	16	17	19	21	23	26	27	30	31	35	36	38	40	42	44	46	48	51	52	54	56	58	61	62	64	67	68	71	73

5.Correlations:

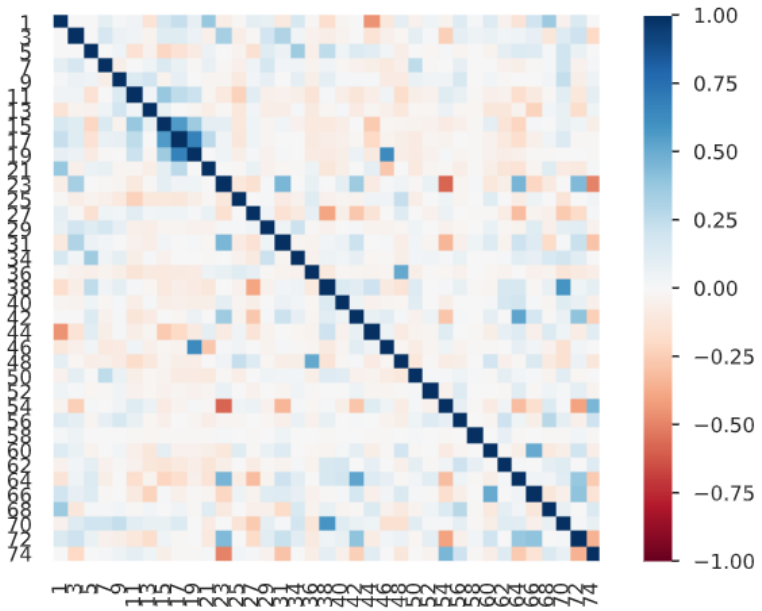
i.Pearson's r:



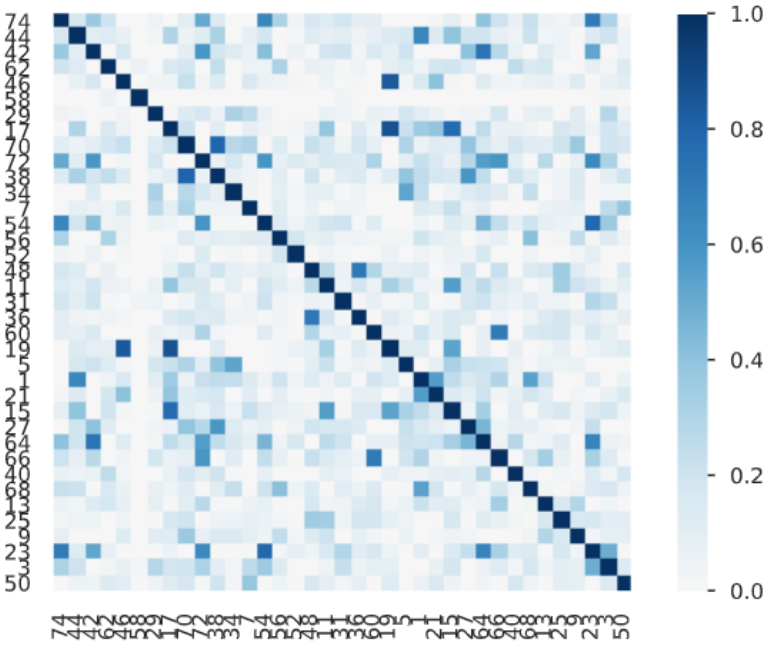
ii.Spearman's ρ:



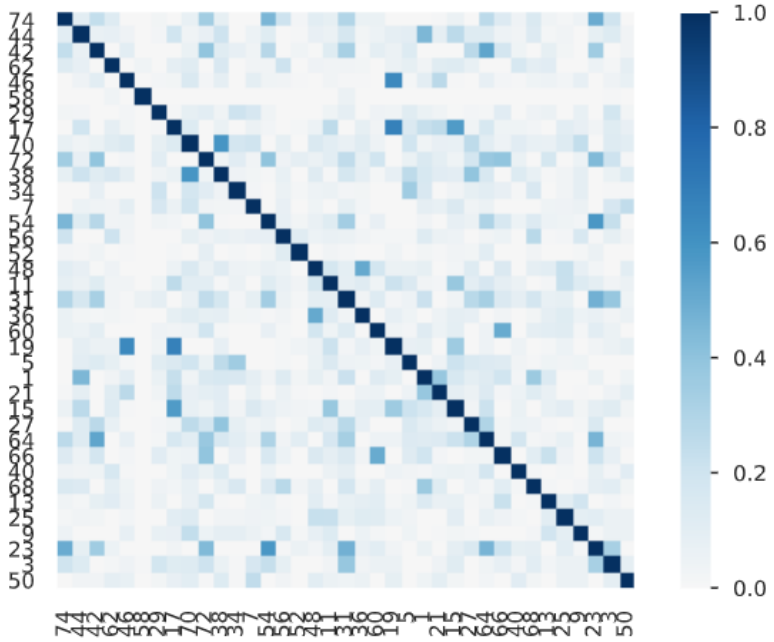
iii.Kendall's τ:



iv.Phik (φk):

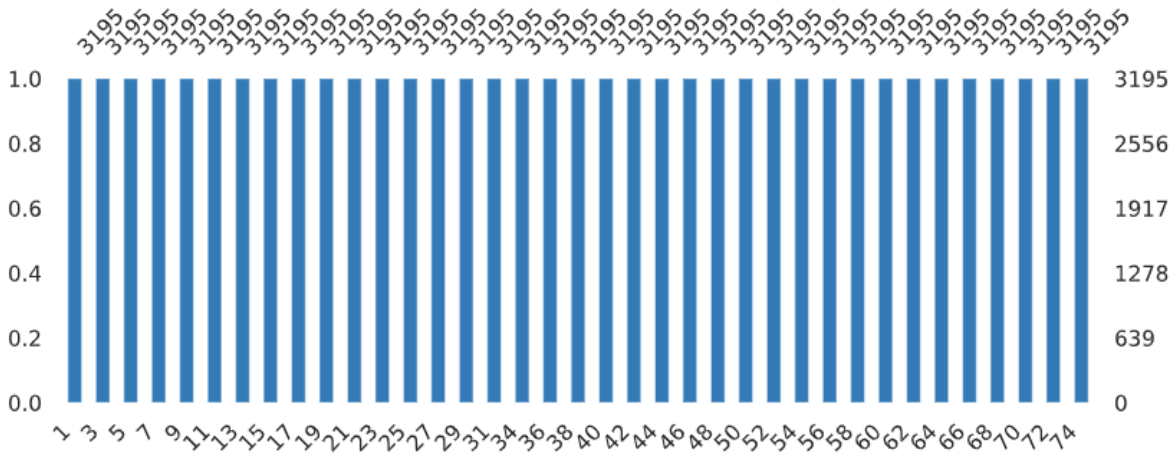


v.Cramér's V (φc):

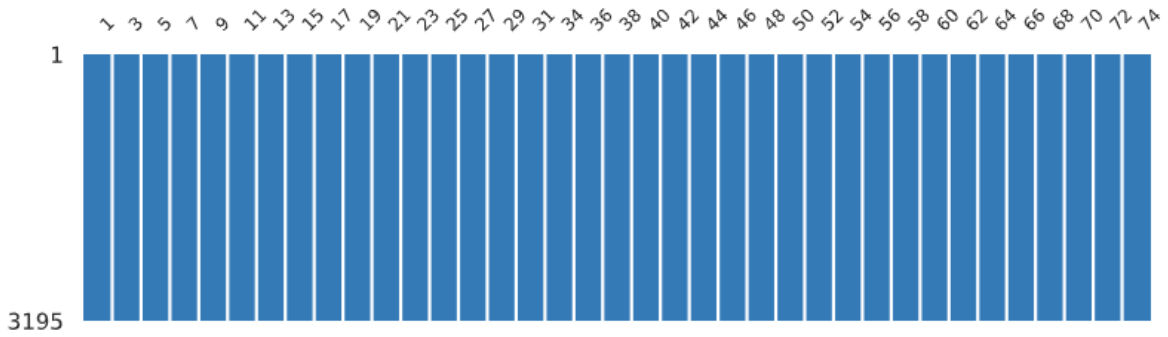


6.Missing values:

i. Count:



ii. Matrix :



In this dataset there are no missing values. The highest column-length is represented by the top columns and the sample of a column-length is represented by the bottom columns.

II.Result Analysis

a.Mushroom Dataset Result Analysis :

1.FP-GROWTH ALGORITHM:

We have implemented FP-Growth algorithm on mushroom dataset .Here, minimum support threshold is threshold ,unit for time is second.

Threshold	Time(s)
1.0	0.08
0.9	0.09
0.8	0.095
0.7	0.11
0.6	0.11
0.5	0.1225
0.4	0.16

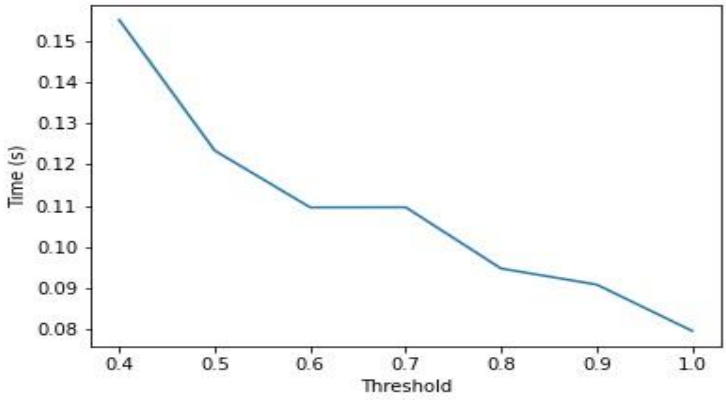


Figure1: Minimum support threshold vs execution time of Mushroom dataset

From the figure of Threshold vs Time we can observe that as Threshold increases the execution time decreases .Here,at threshold 1 we found 1 frequent pattern and execution time 0.08 sec which is the lowest time .For threshold 0.4 we found 412 frequent patterns and execution time is 0.16 sec which the highest .Also, for threshold 0.5,0.6,0.7,0.8,0.9 we get execution times respectively,0.1225sec,0.11sec, 0.11sec,0.095sec,0.09sec .For threshold 0.6 and 0.7 the execution time did not change and when threshold becomes 0.5 to 0.4 the execution time difference between them is drastic.The time difference between other thresholds has slowly changed.Another point here is as thresholds decrease number of frequent patterns gradually increases. For threshold 1, 0.9,0.8,0.7,0.6,0.5,0.4 the frequent patterns are 1,8,14,8,20,102,412.The performance has decreasing because the lower the threshold is the more execution time is needed .Threshold 0.5 and 0.4 generates most patterns.Here,in fp growth algorithm running cost will be less ,the more the tree is small.

2.APRIORI ALGORITHM:

We have implemented Apriori algorithm on mushroom dataset .Here, minimum support threshold is threshold ,unit for time is second.

Threshold	**Time(s)**
1	0.18
0.9	0.15
0.8	0.18
0.7	0.2
0.6	0.25
0.5	0.75
0.4	4.2

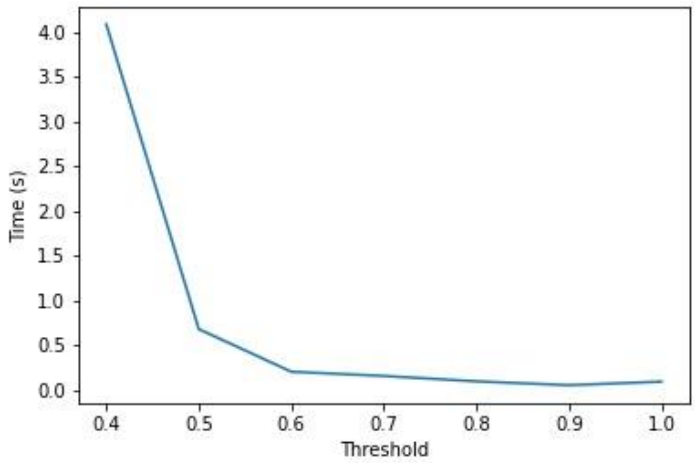


Figure2: Minimum support threshold vs execution time of Mushroom dataset

From the figure of Threshold vs Time we can observe that as Threshold increases the execution time decreases .Here,at threshold 1 we found 1 frequent pattern and execution time 0.18 sec which is the lowest time .For threshold 0.4 we found 412 frequent patterns and execution time is 4.2 sec which the highest .Also, for threshold 0.5,0.6,0.7,0.8,0.9 we get execution times respectively,0.75sec,0.25sec,0.2sec,0.18sec,0.15sec.For threshold 0.5 to 0.4 the execution time difference between them is drastic and there is a huge time gap.For threshold 0.9 to 1.0 execution time is 0.03sec greater .Although the time difference between other thresholds has slowly changed.The performance has decreasing because the lower the threshold is the more execution time is needed .Another point here is as thresholds decrease number of frequent patterns gradually increases. For threshold 1, 0.9,0.8,0.7,0.6,0.5,0.4 the frequent patterns are 1,8,14,8,20,102,412.Threshold 0.5 and 0.4 generates most patterns.As apriori algorithm scans the dataset at every step it takes a lot of time and uses more storage.

3.Performance Comparison between Apriori and Fp-Growth for Mushroom Dataset:

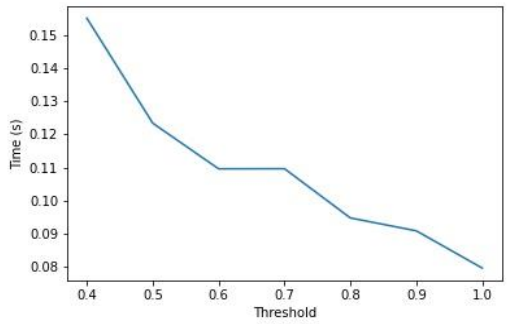
Fp-growth algorithm is comparatively better than apriori algorithm .As, it is faster and with increase of number of itemsets runtime increases,it needs two scans for the whole dataset,uses less space as no candidates are generated,and frequent patterns are generated with conditional trees from the fp-growth tree.

Whereas, Apriori algorithm is slow and with increase of itemsets the runtime increases exponentially.It scans dataset in every step repeatedly ,uses a lot of memory space as candidates are generated by self-joining mechanism ,and frequent patterns are generated by sorting which candidates have support $\geq$ minimum support.

After applying both algorithms on mushroom dataset we have seen fp-growth algorithm works faster and uses less space.From , the table below it can be seen that Fp growth takes 0.08 sec

for total execution and Apriori takes 0.24 sec for total execution .So,apriori takes 0.16 sec more than fp growth.

Threshold	Execution time(s) of FP-Growth	Execution time(s) of Apriori
1	0.08	0.18
0.9	0.9	0.15
0.8	0.95	0.18
0.7	0.11	0.2
0.6	0.11	0.25
0.5	0.1225	0.75
0.4	0.16	0.42



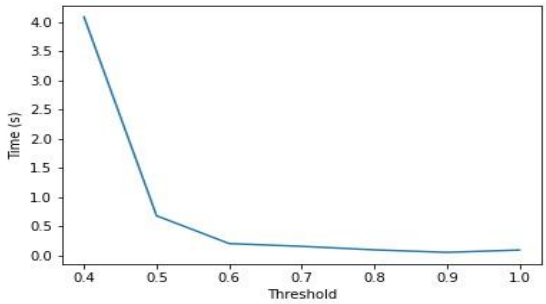


Fig1: FP-Growth Fig2: Apriori

We can see here that when threshold is 1 Fp growth algorithm takes 0.08 sec where apriori algorithm takes 0.18 sec which is 0.1 sec more than fp growth. For other thresholds 0.9,0.8,0.7,0.6,0.5,0.4 fp growth execution time are 0.9,0.95,0.11,0.11,0.1225,0.16 and apriori time are 0.18,0.15,0.18,0.2,0.25,0.75,0.42.Here, the time for apriori has increased exponentially.Also we know apriori algorithm works best on small dataset and mushroom dataset has small transections so apriori algorithm could be used properly even if it took more time than fp growth.So, overall fp growth is more suitable for mushroom dataset.

b.Chess Dataset Result Analysis:

1.FP-GROWTH ALGORITHM:

We have implemented FP-Growth algorithm on chess dataset .Here, minimum support threshold is threshold ,unit for time is second.

Threshold	Time(s)
1	2
0.9	2
0.8	3
0.7	5
0.6	12
0.5	35
0.4	155

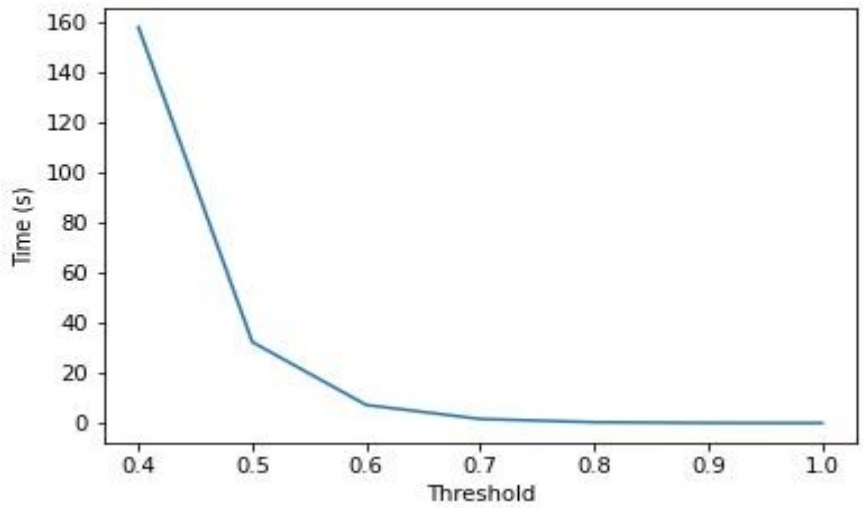


Figure3: Minimum support threshold vs execution time of Chess dataset

From this figure, we can make a decision that when execution time decreases, the minimum support threshold is increasing. At the highest execution time threshold decreases drastically but after some time the threshold changes are quite negligible. When the minimum support threshold is 1, execution time is 2 seconds and did not find any frequent patterns for threshold 1. Here, in the FP growth algorithm running cost will be less, the more the tree is small. Smaller decreases to threshold=0.9, execution time= 2 seconds, and frequency patterns= 622. For threshold=0.8 , execution time= 3 and frequency patterns= 7605, for threshold- 0.7, execution time=5 and we got 40507 frequency patterns, for threshold=0.6, execution time=12 and got 206210 frequency pattern, for threshold=0.5, execution time=35 and got 1017988 frequency pattern and last for threshold = 0.4 , execution time= 155 and frequency pattern we got 5166770 .

2.APRIORI ALGORITHM:

We have implemented Apriori algorithm on chess dataset .Here, minimum support threshold is threshold ,unit for time is second.

Threshold	Time(s)
1	0
0.9	10
0.8	180

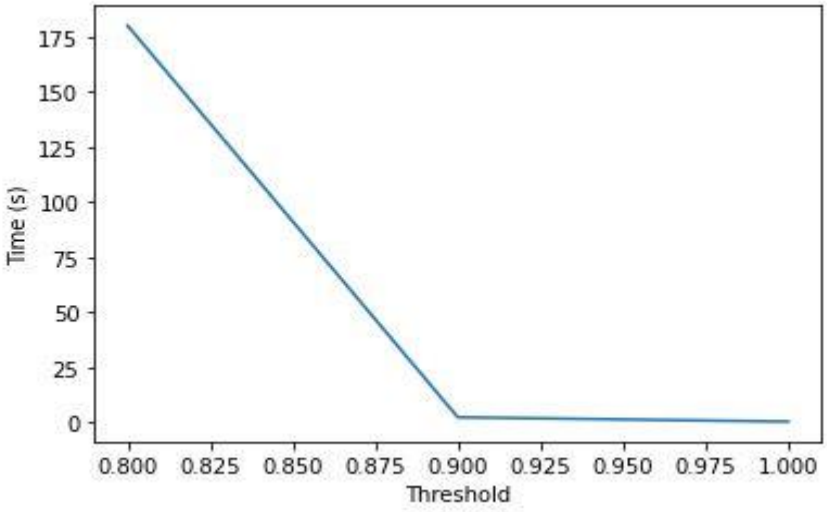


Figure 4: Minimum Support threshold vs Execution Time(s) of Chess dataset

From this above graph, we can see that when execution time is decreasing, the minimum support threshold is increasing. When minimum support threshold = 1 then executing time= 0 and there are no frequency patterns we got. When the minimum support threshold decreased to 0.9 then execution time= 10, which drastically change we can view in the graph and its frequency patterns= 622. And for 0.8 minimum support threshold execution time = 180 and frequency patterns=7605. But we cannot run for threshold 0.7,0.6, 0.5, 0.4 because of memory crash where we got frequent itemset for threshold 0.7, 0.6,0.5, 0.4 from FP- growth of chess .The reason is from threshold 0.6 - 0.4 not working is Apriori scans every step it took more time and storage eventually RAM was full and a memory crash occurred. Also, apriori could not work properly on chess dataset because it has bigger transections in the dataset.That's why no frequency patterns were generated further.

3.Performance comparison of FP-Growth and Apriori for Chess dataset:

Fp-growth algorithm is comparatively better than the apriori algorithm. As it is faster and with the increased number of itemsets runtime increases, it needs two scans for the whole dataset, uses less space as no candidates are generated, and frequent patterns are generated with conditional trees from the FP-Growth tree.

Whereas the Apriori algorithm is slow and with the increase of itemsets the runtime increases exponentially. It scans the dataset in every step repeatedly, uses a lot of memory space as candidates are generated by a self-joining mechanism, and frequent patterns are generated by sorting which candidates have support ≥ minimum support.

After applying both algorithms to the Chess dataset we have seen the FP-Growth algorithm works faster and uses less space. From, the table below it can be seen that Fp growth takes 153 sec for total execution and Apriori takes 180 sec for total execution. So, apriori takes 27 sec more than FP-Growth.

Threshold	**Execution time(s) of FP-Growth**	**Execution time(s) of Apriori**
1	2	0
0.9	2	10
0.8	3	180
0.7	5	x
0.6	12	x
0.5	35	x
0.4	155	x

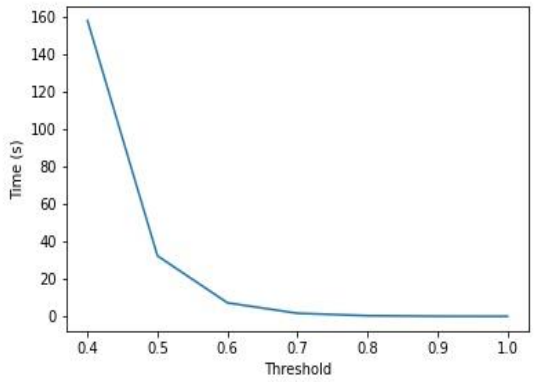


Fig1: FP-Growth

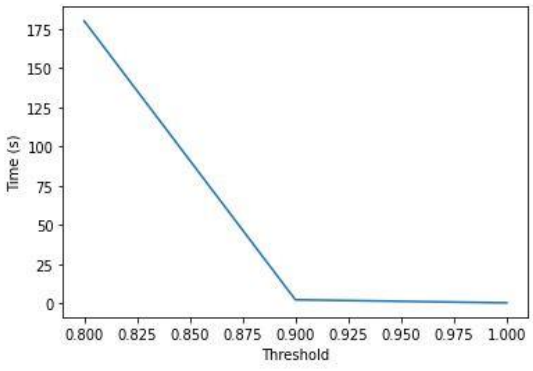


Fig2: Apriori

We can see here that when the threshold is 1 Fp growth algorithm takes 155 sec where the Apriori algorithm takes 180 sec which is 27 sec more than FP-Growth. For other thresholds, 0.9,0.8 FP-Growth execution time are 35,12 and apriori time are 10,0 which are 25,12 respectively more than FP-Growth. For threshold 0.7,0.6,0.5,0.4 Apriori offer no execution time it took more time and storage eventually RAM was full and a memory crash occurred. That's why no frequency patterns were generated further. So no comparison is possible.As,apriori works best on smaller dataset and it scans the dataset repeatedly it is not suitable for chess dataset.Apriori wastes time and storage for chess dataset. So, it can be said that only fp-growth algorithm is applicable for chess dataset and mining.

c.Effect on Mushroom and Chess Dataset for applying FP-Growth and Apriori:

Fp growth algorithm effect on both datasets:

FP Growth	
Mushroom Dataset	**Chess Dataset**
● Mushroom dataset is comparatively smaller transections than chess dataset so fp growth worked faster. It has 23 variables (column)	● Chess dataset is took more time as it is a big dataset . It has 37 variables (column)
● For 1-0.4 threshold execution time is 0.16-0.08second	● For 1-0.4 threshold execution time is 2-155second

Apriori algorithm effect on both datasets:

Apriori	
Mushroom Dataset	**Chess Dataset**
● Mushroom dataset is comparatively smaller transections than chess dataset so apriori worked faster as,apriori works best on smaller dataset	● Chess dataset did not work better for the apriori algorithm because it is a big dataset .
● For threshold 1-0.4 execution time is 0.18-0.42second	● For threshold 1-0.8 execution time is 180-0second ,other threshold could not work
● We did not face RAM crash problem.	● We faced RAM crash problem

	because of storage full from 0.6 threshold
• Threshold vs time graph changed slowly and exponentially	• Threshold vs time graph changed drastically and exponentially

From observing both datasets we can see that the mushroom dataset works for both algorithm, whereas chess dataset only works better for fp growth algorithm.