CSE 477: Data Mining
Section: 02, Summer-2021

A Project Report
on
Apriori and FP-Growth

Submitted by:
Group Members:

Amit Roy	2017-3-60-021
Shekhor Chandra Saha	2017-3-60-025
Fariha Ibnat	2017-2-60-149
Sharika Tasnim Arshi	2017-3-60-084

Submitted to:
Jesan Ahammed Ovi
Lecturer, Dept. of CSE

Date of Submission:21-09-2021

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

EAST WEST UNIVERSITY

I. Dataset

Dataset analysis:

Chess dataset analysis:

The dataset was given in dat file format. A .dat file extension is a generic data file which stores specified information associated to the program that produced the file. It is suitable for classification mining. It has 3195 number of observations. There are no missing values. It has a total memory size of 923.7 KiB. To analyze the dataset properly we have used pandas profiling which is an open-source python module that can easily do analytic data analysis with a few lines of codes. In the report we can see more detailed information about the dataset's overview, variables, correlations, missing values, samples etc. We will shed a light to all of these below.

Overview:

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

For the chess dataset, we have above dataset statistics. If we analyze it, there are 37 number of variables, 3195 number of observations, no missing vales, no duplicate rows and no missing cells. The average record size in memory is 296.0B. for categorial variables the values is 37. Total size in memory for the dataset is 923.7 KiB.

Sample:

Pandas Profiling Report — Overview | Variables | Correlations | Missing values | Sample

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

In the first rows, there are first few (0-9) rows, total 9 rows and (1-74) columns, 37 columns. We can see the lowest value is 1 and highest value is 74 here.

Pandas Profiling Report — Overview | Variables | Correlations | Missing values | Sample

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
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
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
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
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
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
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
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
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
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
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

Last rows

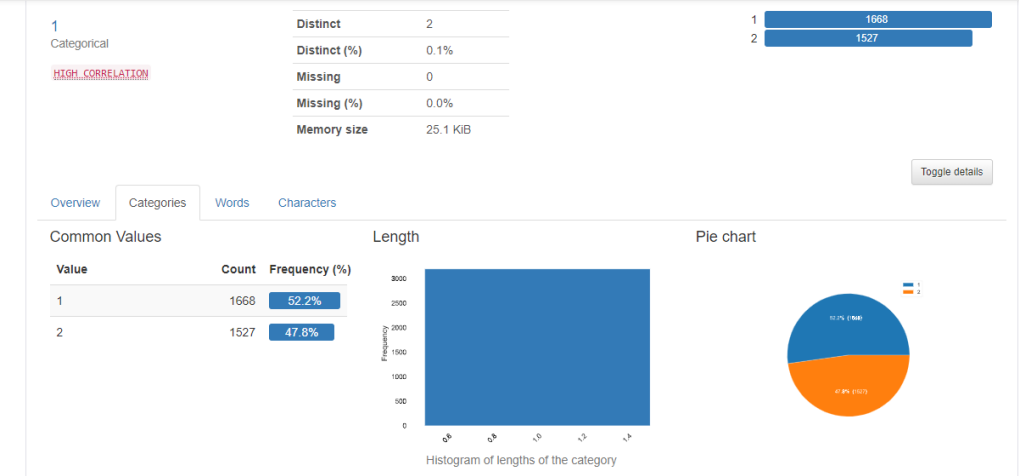
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	74
2	4	5	7	9	12	13	16	17	20	21	23	26	27	29	33	34	36	39	40	42	44	47	48	51	52	54	56	58	61	62	64	67	68	70	73	74
2	3	5	7	10	11	13	16	18	20	21	23	26	27	29	32	34	36	38	40	42	44	46	48	50	52	54	56	58	61	62	64	67	68	71	73	74
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	74
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	74
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	74
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	74
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	74
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	74
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	74
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	74

In the last rows there are last few rows from (3185-3195) total 10 rows and columns from (1-74) columns, total 37 columns. Lowest value here is 2 and highest value is 74.

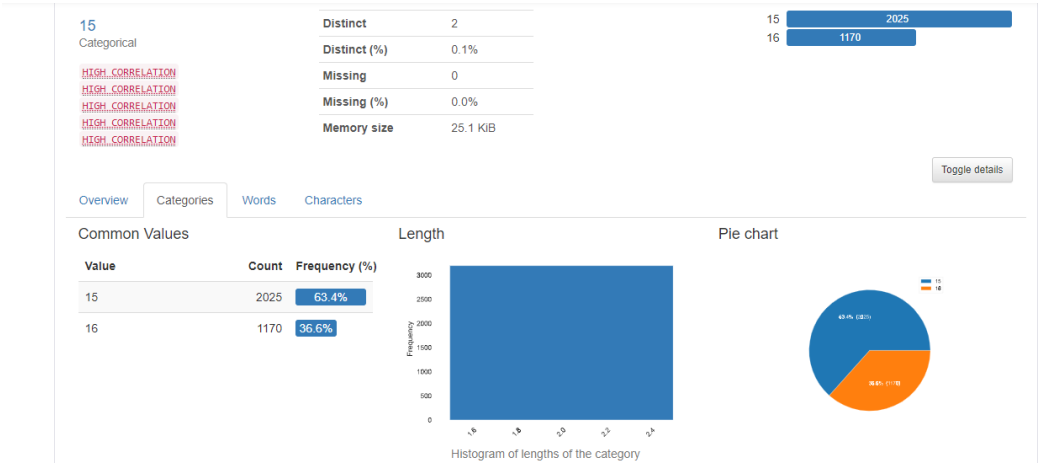
Variables:

There are 37 categorical variable types. Some of the categorical will be explained below:

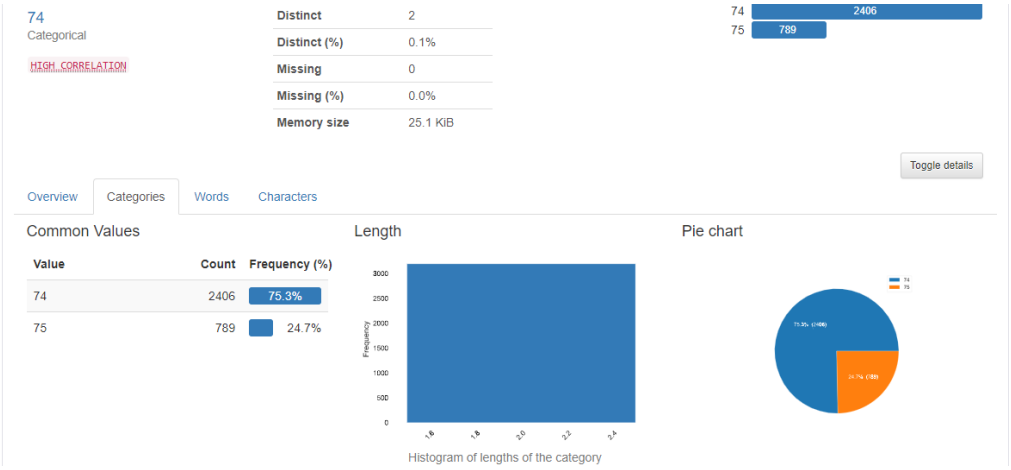
1) 1st categorical has two value which are 1 and 2. Value 1 has count of 1668 and frequency of 52.2%. On the other hand, value 2 has count of 1527 and frequency of 47.8%. This variable has a high correlation with 3 fields- 21, 44, 68



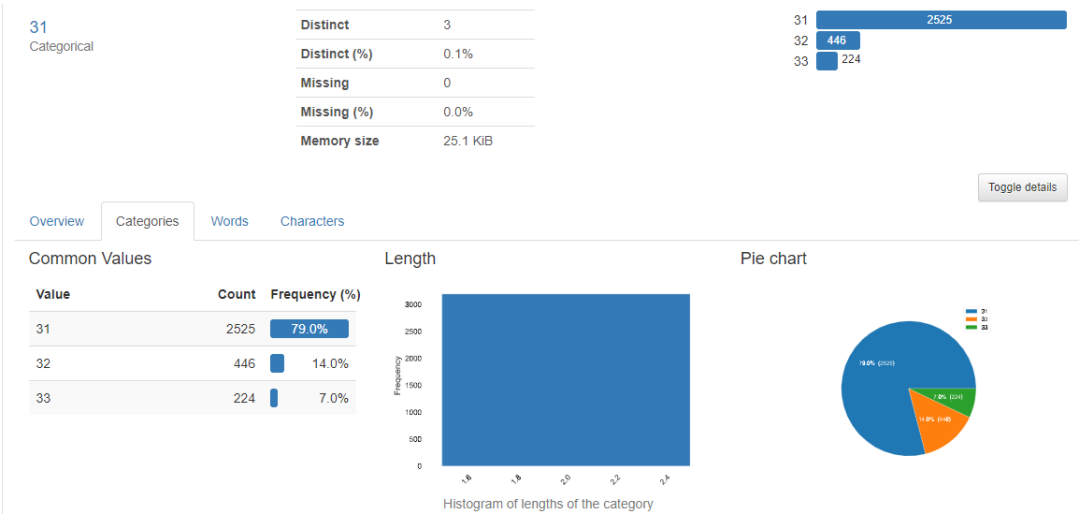
2) 15th categorical has two values which are 15 and 16. Value 15 has count of 2025 and frequency of 63.4%. On the other hand, value 16 has count of 1170 and frequency of 36.6%. This variable has a high correlation with 1 field – 17



3) 74th categorical has two values which are 74 and 75. Value 74 has count of 2406 and frequency of 75.3%. On the other hand, value 75 has count of 789 and frequency of 24.7%. This variable has a high correlation with 3 fields- 23, 54, 72



4) 31th categorical has three values which are 31, 32 and 33. Value 31 has count of 2525 and frequency of 79.0%. Value 32 has count of 446 and frequency of 14.0%. Value 33 has count of 224 and frequency of 7.0%. This variable has no c correlations with any other fields.

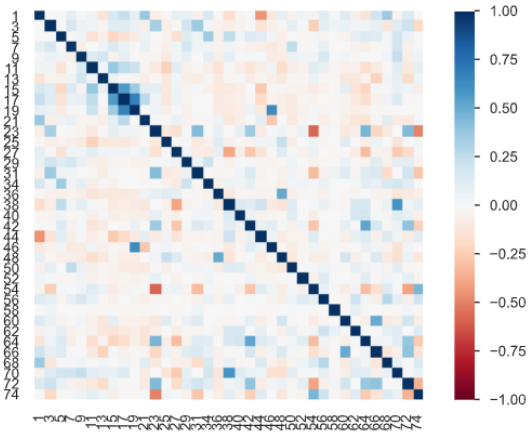


Correlation:

Correlations is used to define relationship between categorical variables. Correlation analysis is a measure of how two variables are related. Using pandas, we have found out correlations of all columns in the data frame.

Here is the detailed information about it which were generated in pandas profiling report for chess dataset:

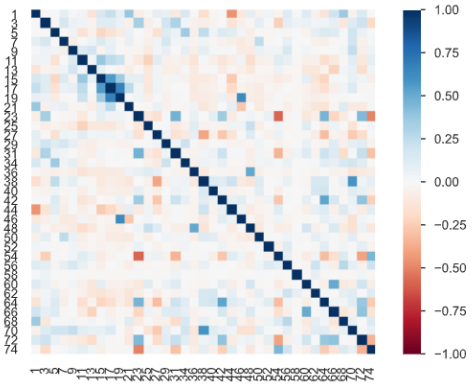
Pearson's r:



The Pearson's correlation coefficient (r) is a measure of linear correlation between two variables. Its value lies between -1 and +1, -1 indicating total negative linear correlation, 0 indicating no linear correlation and 1 indicating total positive linear correlation. Furthermore, 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.

Spearman's ρ:

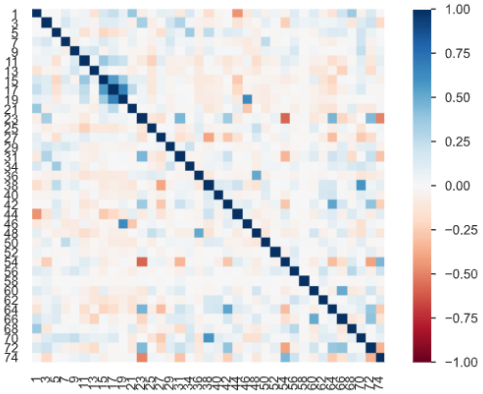


The Spearman's rank correlation coefficient (ρ) is a measure of monotonic correlation between two variables, and is therefore better in catching nonlinear monotonic correlations than Pearson's

r. Its value lies between -1 and +1, -1 indicating 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.

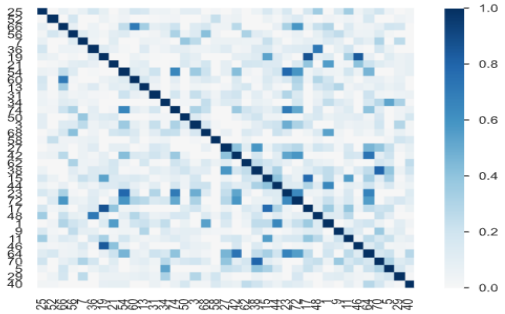
Kendall's T:



Similarly, to Spearman's rank correlation coefficient, the Kendall rank correlation coefficient (τ) measures ordinal association between two variables. Its value lies between -1 and +1, -1 indicating total negative correlation, 0 indicating no correlation and 1 indicating total positive correlation.

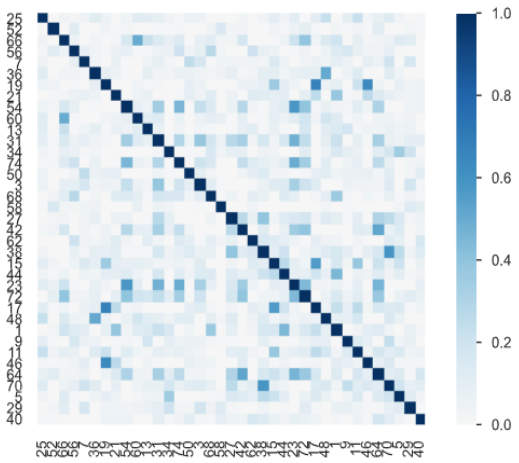
To calculate τ for two variables X and Y, one determines the number of concordant and discordant pairs of observations. τ is given by the number of concordant pairs minus the discordant pairs divided by the total number of pairs.

Phik (φ k):



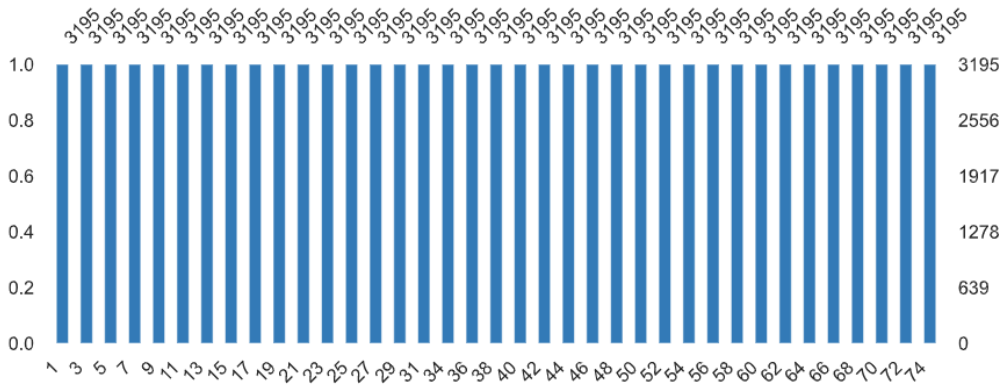
Phik (φk) is a new and practical correlation coefficient that works consistently between categorical, ordinal and interval variables, captures non-linear dependency and reverts to the Pearson correlation coefficient in case of a bivariate normal input distribution.

Cramer's V (φ c):



Cramér's V is an association measure for nominal random variables. The coefficient ranges from 0 to 1, with 0 indicating independence and 1 indicating perfect association. The empirical estimators used for Cramér's V have been proved to be biased, even for large samples. We use a bias-corrected measure that has been proposed by Bergsma in 2013.

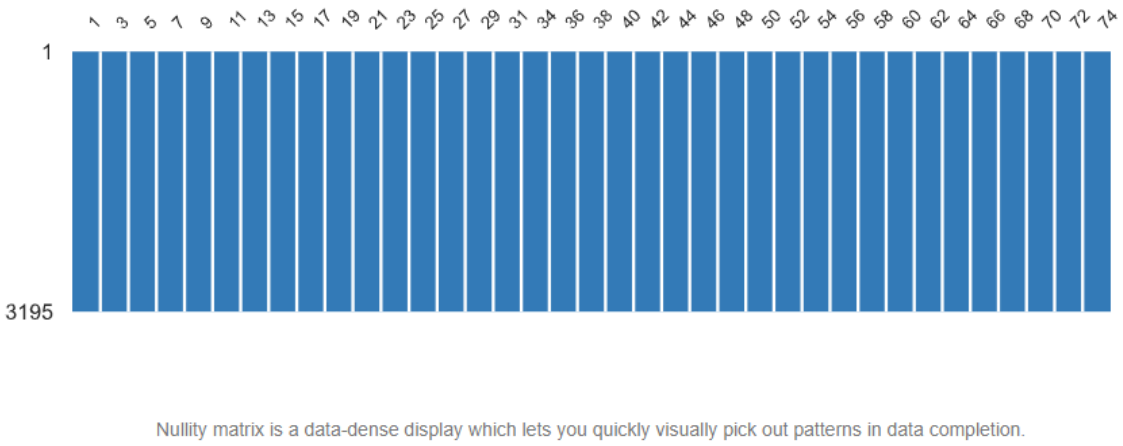
Missing values (count):



A simple visualization of nullity by column.

The above bar chart represents count of values per column ignoring missing values from the chess dataset. Each bar represents columns of the dataset. The height of the bar represents about the completeness of the column on the left, y-axis ranges from 0 to1. If the bar is less than 1.0 then we can say that there are missing values. On the right, index values are measured. In the above figure we can conclude that, this dataset has no missing values. Here bottom column represents sample of a column length and top column represents highest column length for that for that column.

Matrix:



Nullity matrix is a data-dense display which lets you quickly visually pick out patterns in data completion.

In the above figure we can see the plotting of matrix representing no missing values of chess dataset. Matrix plot is a useful tool that provides a color fill for each column. When values are missing it will be shaded in white. As we can see from the figure that there are no missing values so every column is shaded into blue.

Mushroom dataset analysis:

The dataset was given in dat file format. A .dat file extension is a generic data file which stores specified information associated to the program that produced the file. It is suitable for classification mining. It has 8123 number of observations. There are no missing values. It has a total memory size of 1.4 MiB. There are no duplicate rows. There are 14 categorical and 9 numeric variable types. To analyze the dataset properly we have used pandas profiling which is an open-source python module that can easily do analytic data analysis with a few lines of codes. In the report we can see more detailed information about the dataset's overview, variables, correlations, missing values, samples etc. We will shed a light to all these below.

Overview:

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

For the mushroom dataset, we have above dataset statistics. If we analyze it, there are 23 number of variables, 8123 number of observations, no missing vales, no duplicate rows and no missing cells. The average record size in memory is 184.0B. for categorial variables the values is 14. For numerical variable the value is 9. Total size in memory for the dataset is 1.4 MiB.

Sample:

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

In the first rows there are first few rows (0-9) rows total 9 rows and (1-113) columns total 23 columns. Lowest value here is 1 and highest value here is 115.

Last rows

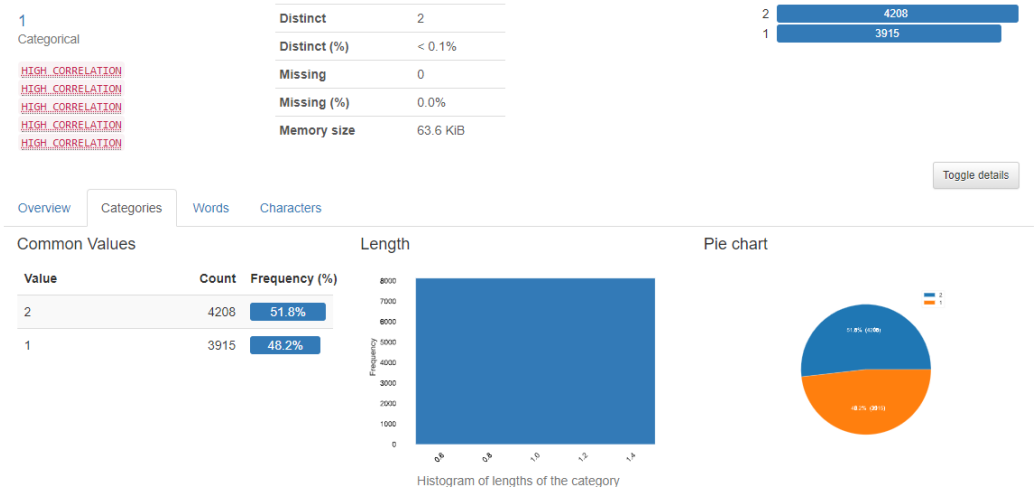
	1	3	9	13	23	25	34	36	38	40	52	54	59	63	67	76	85	86	90	93	98	107	113
8113	1	6	10	21	24	33	35	36	39	50	52	55	61	65	74	84	85	86	92	97	102	112	116
8114	2	3	9	13	24	28	35	36	39	50	52	58	59	63	73	83	85	88	90	93	104	110	119
8115	1	7	10	13	24	32	34	36	38	48	53	58	59	66	69	76	85	86	90	94	102	110	119
8116	1	7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

In last rows there are last few rows (8113-8122) rows total 9 rows and (1-113) columns total 23 columns. Lowest value here is 1 and highest value here is 119.

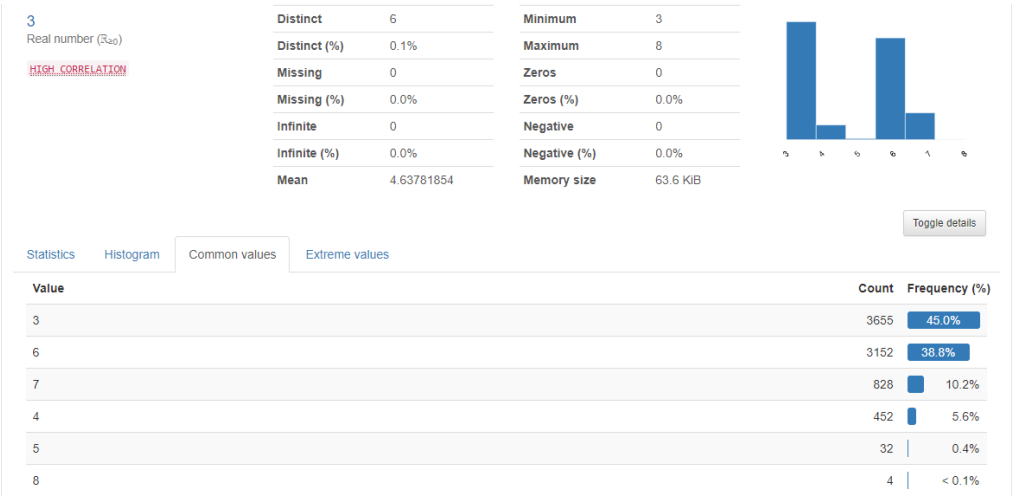
Variables:

There are total 23 variable. Among them 14 are categorical and 9 are numeric. Some of them will be explained below:

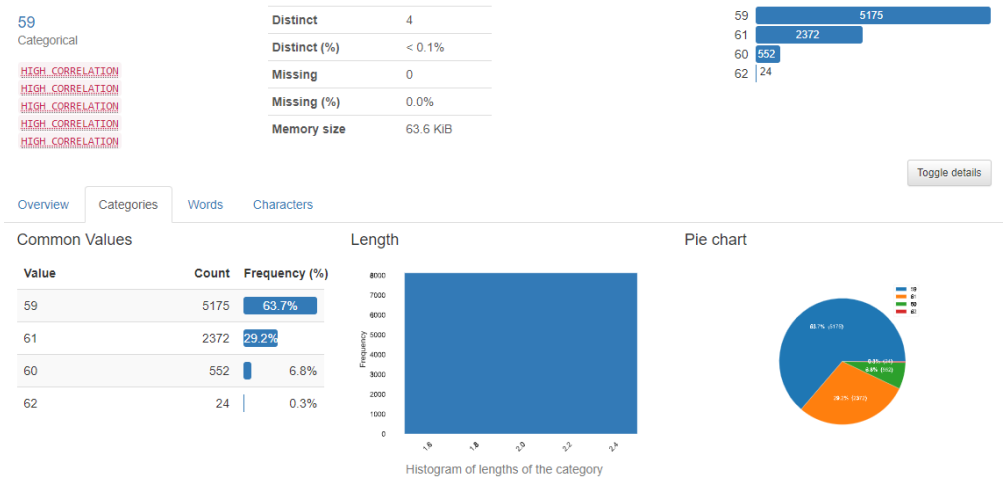
1) 1st categorical has two values which are 1 and 2. Value 1 has count of 3915 and frequency of 48.2%. Value 2 has count of 4208 and frequency of 51.8%. This variable has a high correlation with 7 fields- 23,25,38,59,63,93,98



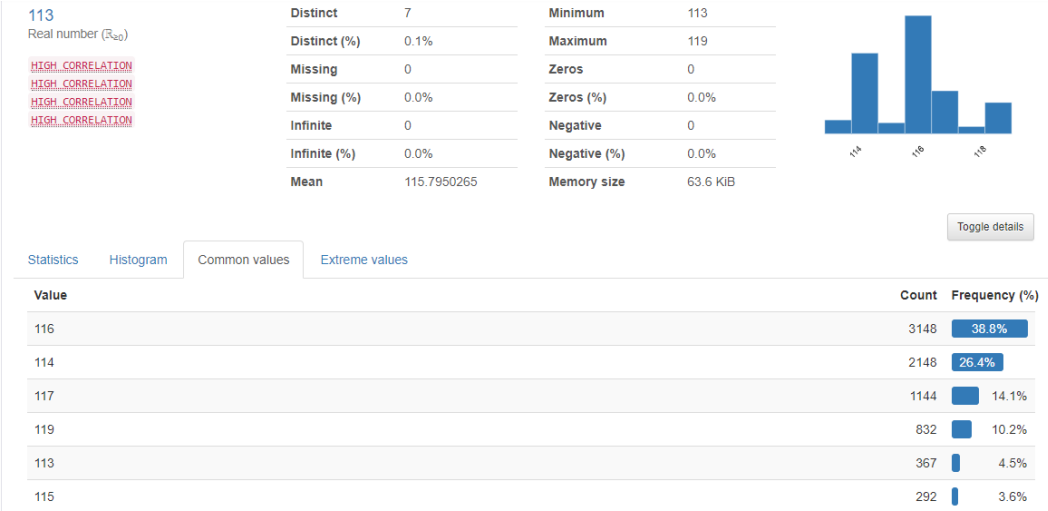
2) 3rd variable is a real number. This variable has a high correlation with 1 field- 107.



3) 59th categorical has four values which are 59, 60, 61 and 62. Value 59 has count of 5175 and frequency of 63.7%. Value 62 has count of 24 and frequency of 0.3%. This variable has highest correlation with 11 fields- 36, 86, 40, 76, 93, 67, 25, 63, 23, 1, and 98.



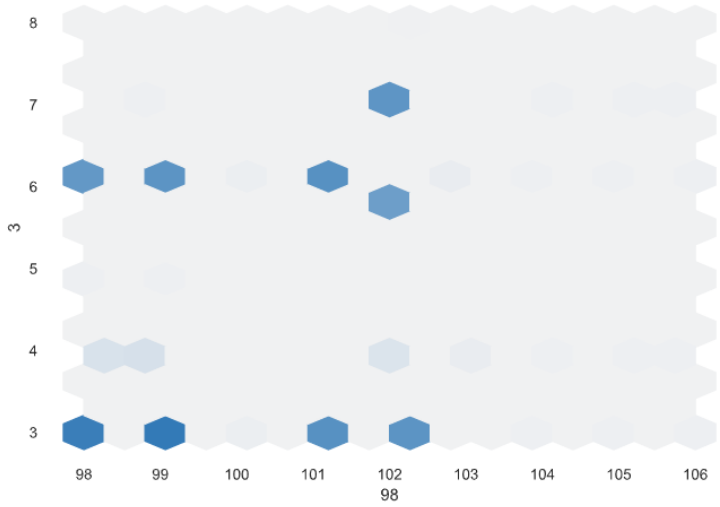
4) 113th variable is a real number. This variable has highest correlation with 10 fields- 13, 36, 54, 40, 107, 76, 67, 25, 90 and 98.



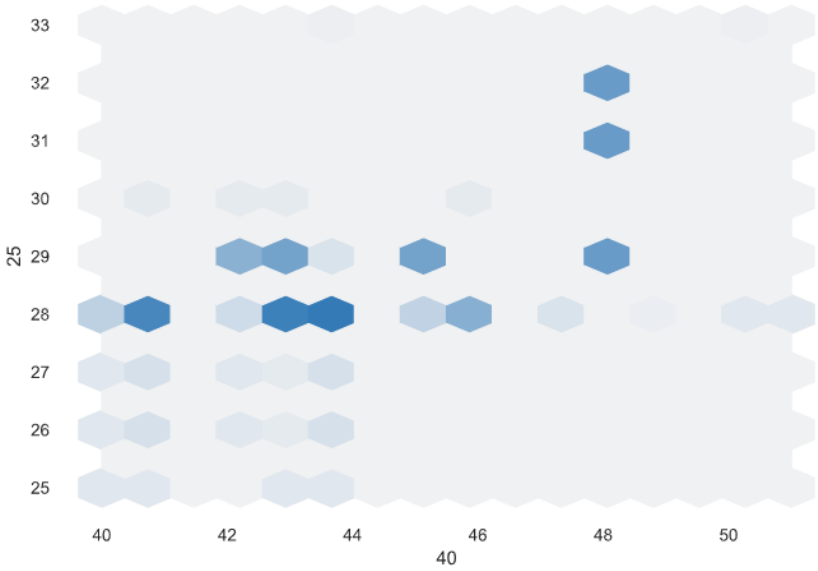
Interactions:

This is the relationship between two variables in the dataset as scatter plots. Here pandas generate interaction plots for every pair of variables.

1) Here for variables 98 and 3 we have got below plot:

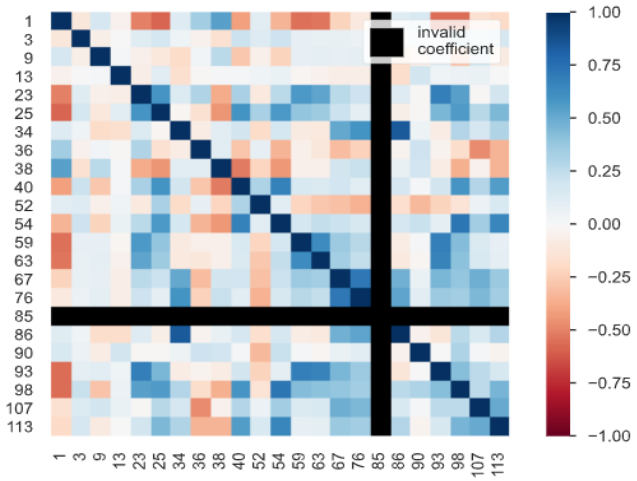


2) Here for variables 40 and 25 we have got below plot:



Correlations:

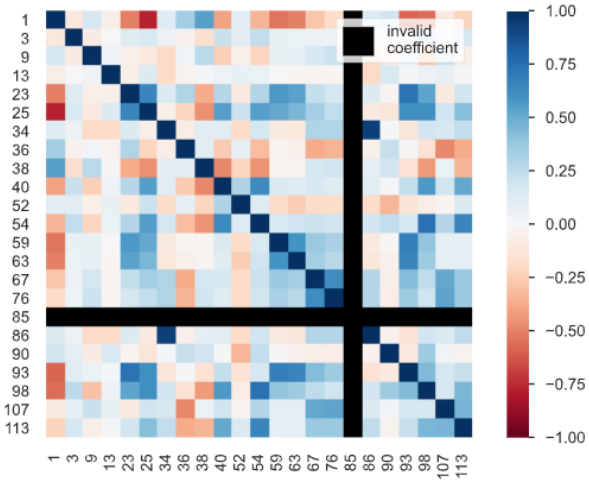
Pearson's r:



The Pearson's correlation coefficient (r) is a measure of linear correlation between two variables. Its value lies between -1 and +1, -1 indicating total negative linear correlation, 0 indicating no linear correlation and 1 indicating total positive linear correlation. Furthermore, 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.

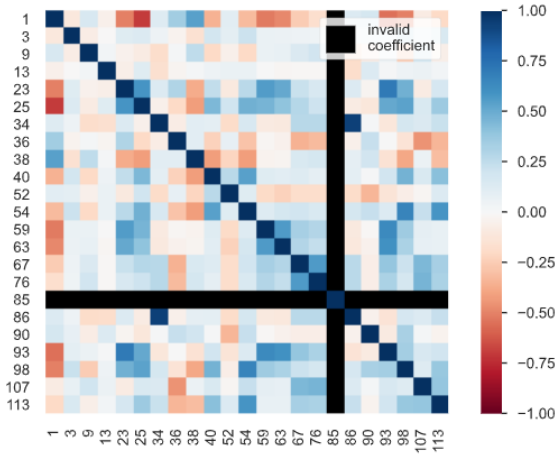
Spearman's ρ:



The Spearman's rank correlation coefficient (ρ) is a measure of monotonic correlation between two variables, and is therefore better in catching nonlinear monotonic correlations than Pearson's r. Its value lies between -1 and +1, -1 indicating total negative monotonic correlation, 0 indicating no monotonic correlation and 1 indicating total positive monotonic correlation.

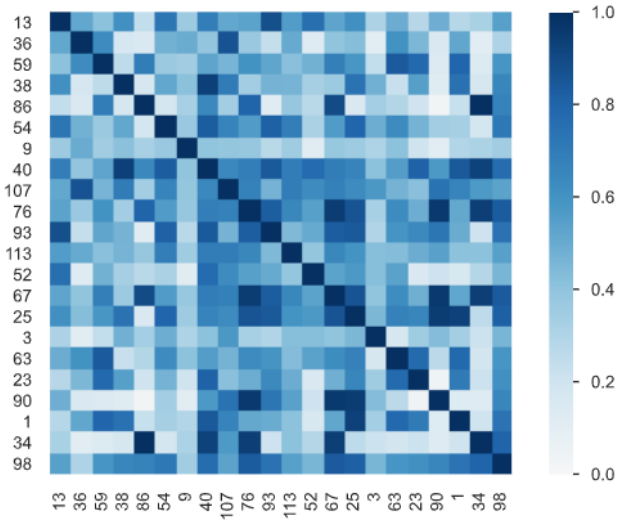
To calculate ρ for two variables X and Y, one divides the covariance of the rank variables of X and Y by the product of their standard deviations.

Kendall's T:



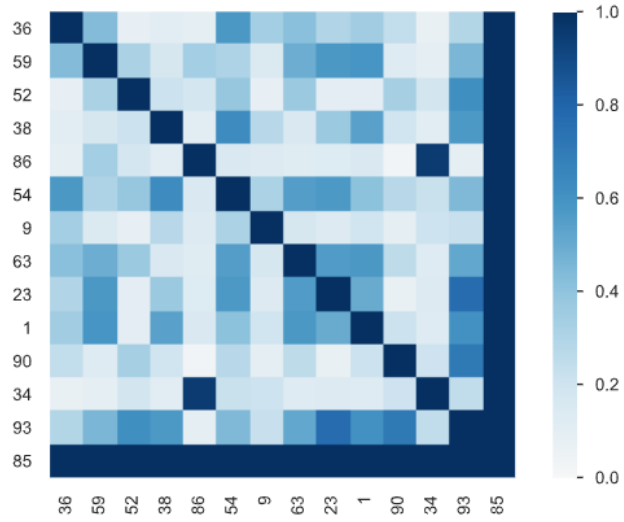
Similarly, to Spearman's rank correlation coefficient, the Kendall rank correlation coefficient (τ) measures ordinal association between two variables. Its value lies between -1 and +1, -1 indicating total negative correlation, 0 indicating no correlation and 1 indicating total positive correlation. 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.

Phik (φ k):



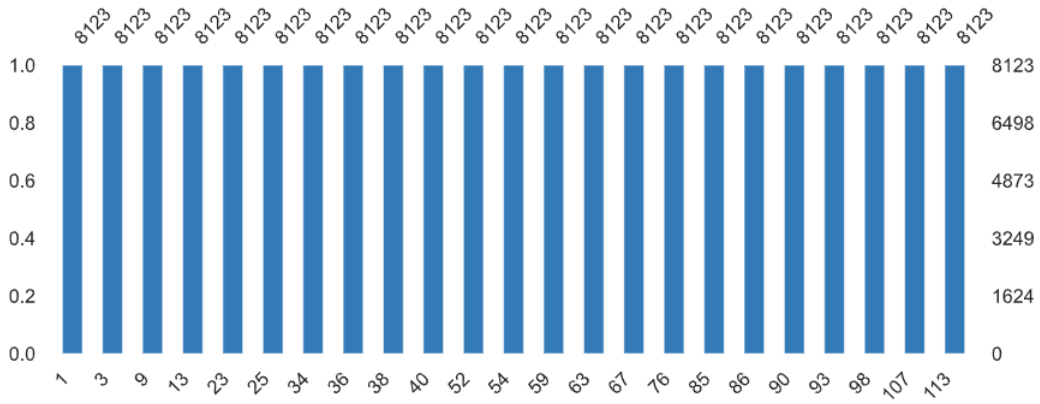
Phik (φk) is a new and practical correlation coefficient that works consistently between categorical, ordinal and interval variables, captures non-linear dependency and reverts to the Pearson correlation coefficient in case of a bivariate normal input distribution.

Cramer's V (φ c):



Cramér's V is an association measure for nominal random variables. The coefficient ranges from 0 to 1, with 0 indicating independence and 1 indicating perfect association. The empirical estimators used for Cramér's V have been proved to be biased, even for large samples. We use a bias-corrected measure that has been proposed by Bergsma in 2013.

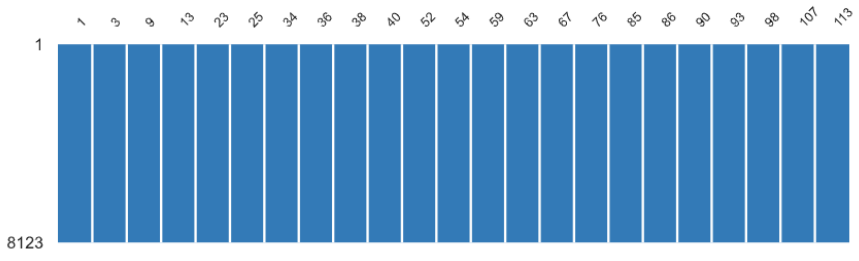
Missing values (counts):



A simple visualization of nullity by column.

The above bar chart represents count of values per column ignoring missing values from the mushroom dataset. Each bar represents columns of the dataset. The height of the bar represents about the completeness of the column on the left, y-axis ranges from 0 to1. If the bar is less than 1.0 then we can say that there are missing values. On the right, index values are measured. In the above figure we can conclude that, this dataset has no missing values. Here bottom column represents sample of a column length and top column represents highest column length for that for that column.

Matrix:



Nullity matrix is a data-dense display which lets you quickly visually pick out patterns in data completion.

In the above figure we can see the plotting of matrix representing no missing values of mushroom dataset. Matrix plot is a useful tool that provides a color fill for each column. When values are missing it will be shaded in white. As we can see from the figure that there are no missing values, so every column is shaded into blue.

II. Result

Result analysis:

In this section we will discuss the output after applying both apriori and fp-growth algorithm in both chess and mushroom dataset. The performance evaluation was based on changes of threshold vs time. Here on y-axis, there is threshold that is minimum support threshold. It is usually applied to mine frequent patterns from the dataset. A pattern must satisfy minimum support threshold to become frequent. Here on x-axis, there is execution time in seconds. We will discuss how each algorithm performs on given datasets.

Mushroom dataset:

There are 23 number of variables, 8123 number of observations, no missing vales, no duplicate rows, and no missing cells. The average record size in memory is 184.0B. Total size in memory for the dataset is 1.4 MiB. We have applied apriori and fp-growth alorithm on this dataset and got below result.

Apriori :

Threshold	Time (s)
1	0.06092262
0.9	0.05368447
0.8	0.07416153
0.7	0.1430566
0.6	0.157027
0.5	0.5739486
0.4	3.38636
0.3	60.58499

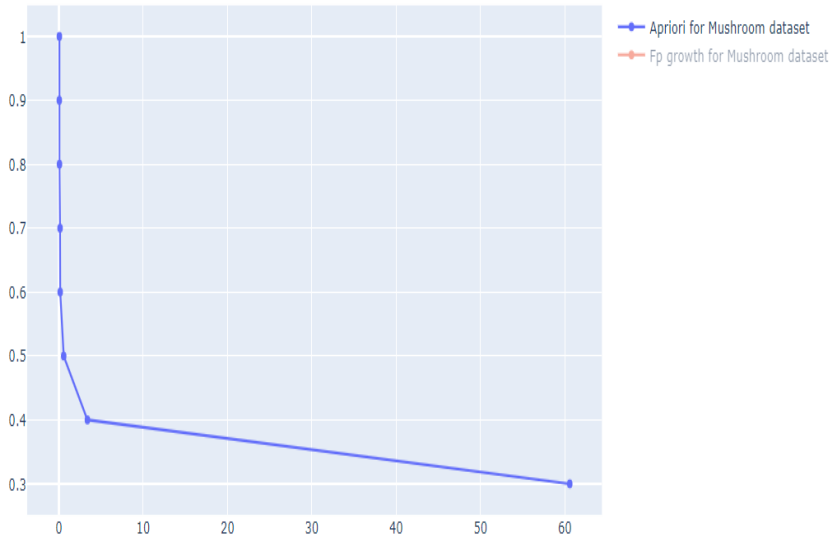


Fig 1: threshold vs time for apriori on mushroom dataset

When we applied apriori algorithm on mushroom dataset, minimum support threshold was set to 1. For that we got no frequent pattern. When the support of an item is higher there will be more shared branches and tree size will get smaller. Eventually there will be lesser cost to run the algorithm. As we can see in fig1, we have taken 8 points. So, when threshold is decreased to 0.9, we got some frequent patterns, and the execution time was 0.05368447. Then for threshold 0.8, 0.7, 0.6, and 0.5 and got execution time of 0.07416153, 0.1430566, 0.157027, and 0.5739486. From the graph we can see a vertical line for these 6 points. However, when threshold decreases to a point of 0.4, the curve does not remain vertical. Now time is 3.38636. By the decrease of threshold, there are a greater number of frequent patterns. We know apriori algorithm scans the dataset repeatedly for each iteration. Therefore, more execution time is needed. So that is why for threshold 0.3 time is needed 60.58499. So, we can see a downward slope till time 60 on the graph.

Fp-growth:

Threshold	Time (s)
1	0.07779169
0.9	0.0987377
0.8	0.0997336
0.7	0.1276586
0.6	0.1176867
0.5	0.1276624
0.4	0.1555843
0.3	0.2862358

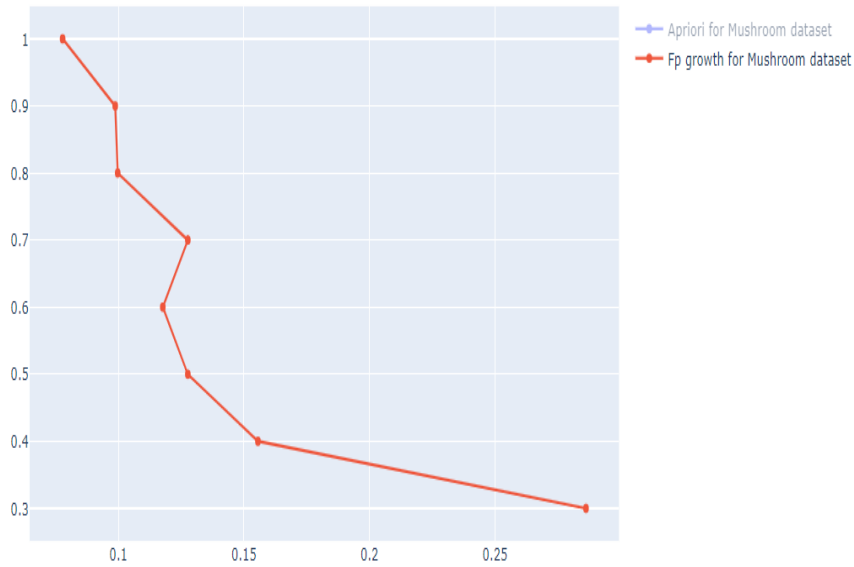


Fig 2: threshold vs time for fp-growth on mushroom dataset

When we applied fp-growth algorithm on mushroom dataset, minimum support threshold was set to 1. For that we got no frequent pattern. Execution time is 0.07779169. Again, for decreased

threshold we got more frequent patterns. As we can see in fig2, there are total 8 points. Then again, we have decreased threshold 0.9, 0.8, 0.7, 0.6, and 0.5 and got execution time of 0.0987377, 0.0997336, 0.1276586, 0.1176867, and 0.1276624. From the graph we can see a zigzag line up to (0.4, 0.15). For these 5 points, execution time fluctuates. For threshold 0.4 execution time is 0.1555843. From this point we get a downward slope showing more execution time till next point. For threshold 0.3 execution time is 0.2862358. After that output is shown. We have got the output quite faster. We know fp-growth algorithm scans the dataset 2 times. For this reason, efficiency of it decreases. However, for mushroom dataset with 8123 number of rows we get the output within approximately 0.287s.

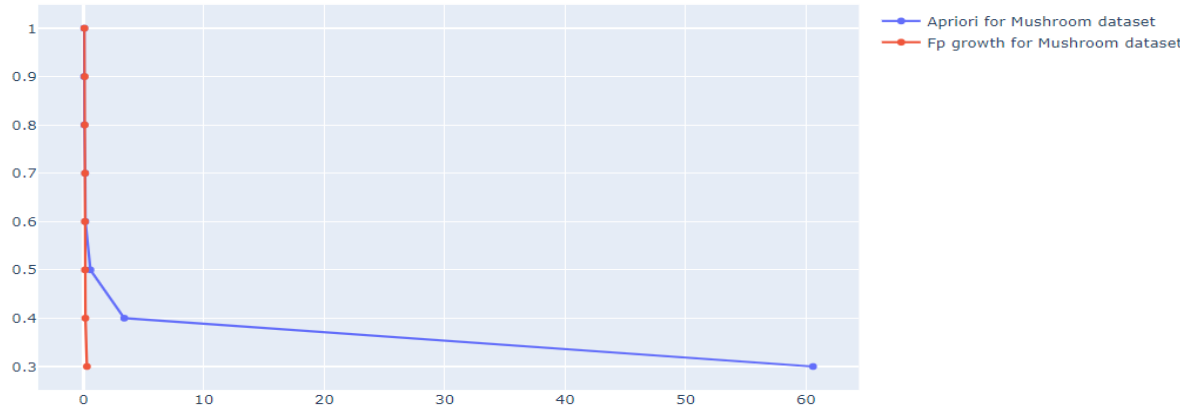


Fig 3: Comparison between apriori and fp-growth for mushroom dataset

Apriori vs Fp-growth:

After comparing both algorithms, it can be concluded that fp-growth has a better performance on this dataset based on pattern generation, execution time, scanning, memory size, storage, and available data. From fig3 we can see that fp-growth maintains a vertical line till end time which is 0.287s. It generates the result quite faster. Although fp-growth scans dataset twice and has a decreased performance because of it, it still gives better result than apriori algorithm. Initially apriori was giving good result and less execution time. However, apriori algorithm scans the whole dataset again and again. It needs more time for calculation. For result when the threshold is 0.4 apriori takes 3.3863660 seconds while fp-growth takes 0.1555843 seconds. Then apriori stats to maintain a downward slope which means performance is decreasing. For threshold 0.3

the execution time for apriori and fp-growth is 60.58499 and 0.2862358 respectively. This is a huge turning point. As it is clearly seen that apriori is taking huge time compared to fp-growth. Performance is degrading after that point, and we are getting a downward slope for apriori. On the other hand, fp-growth is still maintaining a vertical slope proving better performance instead of decreasing threshold and more scanning. So, we have found fp-growth performing better and faster on mushroom dataset.

Chess dataset:

There are 37 number of variables, 3195 number of observations, no missing values, no duplicate values, and a total memory size of 923.7 KiB. We have applied apriori and fp-growth alorithm on this dataset and got below result.

Apriori:

Threshold	Time (s)
1	0.03070951
0.9	2.201063
0.8	175.8379
0.7	3335.575

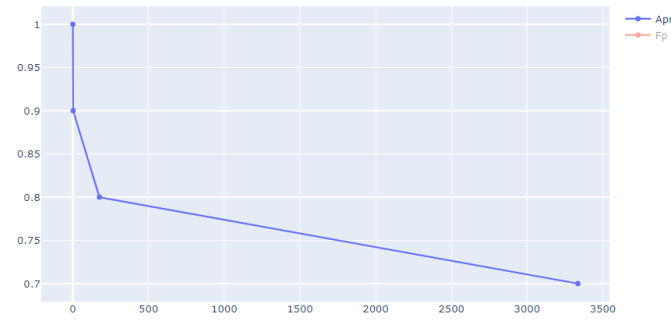


Fig 4: threshold vs time for apriori on chess dataset

When we applied apriori algorithm on chess dataset, minimum support threshold was set to 1. For that we got no frequent pattern. Execution time was 0.03070951. When the support of an item is higher there will be more shared branches and tree size will get smaller. Eventually there will be lesser cost to run the algorithm. We have taken 4 points which can be seen in fig4. So, when threshold is decreased to 0.9, we got some frequent patterns, and the execution time was 2.201063. Till this point the slope remains vertical. From 0.9 to 0.8 the slope is getting downwards and moving towards right. Execution time for 0.8 threshold is 175.8379. However, when threshold decreases to a point of 0.7, the slope takes a drastic downward turn. Now the time it takes to show output is 3335.575s. This is huge. By the decrease of threshold, there are a

greater number of frequent patterns. We know apriori algorithm scans the dataset repeatedly for each iteration. Therefore, more execution time is needed. But after 0.7 threshold there is a memory error and crashing problem. The algorithm is taking vast memory for frequent scanning. That is why only 4 points has been shown in the curve.

Fp-growth:

Threshold	Time (s)
1	0.02393413
0.9	0.05880523
0.8	0.2082586
0.7	0.9165969
0.6	12.23962
0.5	25.0869
0.4	485.4707

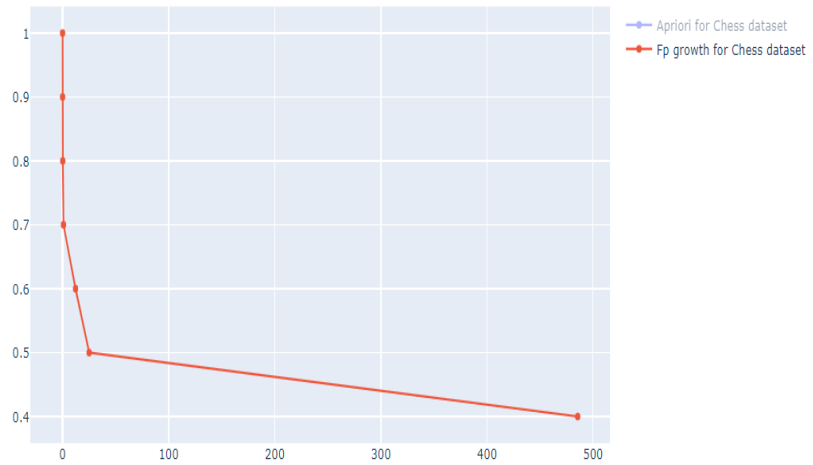


Fig 5: threshold vs time for fp-growth on chess dataset

When we applied fp-growth algorithm on chess dataset, minimum support threshold was set to 1. For that we got no frequent pattern. Execution time is0.02393413. Again, for decreased threshold we got more frequent patterns. There is total 8 points which is shown in fig5. Then again, we have decreased threshold 0.9, 0.8, 0.7, 0.6, and 0.5 and got execution time of 0.05880523, 0.2082586, 0.9165969, 12.23962, and 25.0869. From the graph we can see from threshold 1 to 0.7, the slope is vertical and execution time is good. After these 4 points, execution time fluctuates. From threshold 0.7 to 0.5, the sloping is downward and moving toward right indicating higher execution time. Reason for this might be, fp-growth scans the dataset twice. For lower threshold its performance degrades. From the threshold point 0.5 we get a downward slope showing more execution time till next point. For threshold 0.4 execution time is 485.4707s. Then the output is shown. We have got the output quite slower this time for chess dataset. Chess

dataset has 3195 number of rows. There are two types of variables. Transactions like these have made an impact in taking longer time.

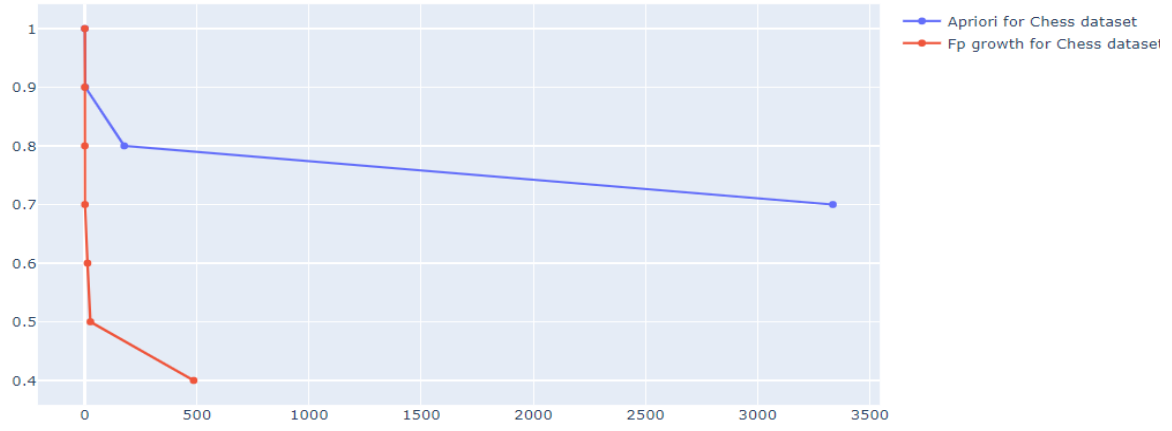


Fig 6: Comparison between apriori and fp-growth for chess dataset

Apriori vs Fp-growth:

After comparing both algorithms, it can be concluded that fp-growth has a slight better performance on chess dataset based on pattern generation, execution time, scanning, memory size, storage, and available data. From fig6 we can see that fp-growth maintains a vertical line till threshold 0.5. After that it takes slightly longer time and ends at 485.4707s. It generates the result quite faster than apriori. However, performance deterioration has been noticed. Although fp-growth scans dataset twice and has a decreased performance because of it, it still gives better result than apriori algorithm. We know apriori algorithm scans the whole dataset again and again. It needs more time for calculation. For this, after the threshold is 0.8 apriori slope takes a drastic downward turn. The longer execution time is clearly visible on the graph. Total execution time is 3335.575s. While fp- growths' is 485.4707s. Not only has that apriori given memory error after threshold 0.7. Performance is degrading after some point for both algorithms. The types of transactions on chess dataset might have made an impact on both datasets. Considering all of these, we have found fp-growth performing better and faster on chess dataset.

Overall Comparison:

Apriori Algorithm	Fp Growth Algorithm
Slower speed. Because run time increase exponentially by increasing number of item set.	Faster speed. Because run time increase linearly by increasing number of item set.
Need large amount of memory. Because all the candidates from self-joining are stored in the memory.	Need small amount of memory. Because the compact version of dataset is storing.
Use self-joining for candidate generation.	No candidate generation.
Scan full dataset repeatedly.	Only require two scans.
Pattern selected from the candidates whose support is higher than minimum support.	Pattern growths achieve by mining conditional